Adil M. Bagirov · Manlio Gaudioso
Napsu Karmitsa · Marko M. Mäkelä
Sona Taheri *Editors*

Numerical Nonsmooth Optimization

State of the Art Algorithms

Springer

Numerical Nonsmooth Optimization

Adil M. Bagirov • Manlio Gaudioso •
Napsu Karmitsa • Marko M. Mäkelä • Sona Taheri
Editors

Numerical Nonsmooth Optimization

State of the Art Algorithms

Springer

Editors
Adil M. Bagirov
School of Science, Engineering and Information Technology
Federation University Australia
Ballarat
Victoria, Australia

Manlio Gaudioso
Department of Informatics, Modeling, Electronics and System Engineering
University of Calabria
Rende (CS), Italy

Napsu Karmitsa
Department of Mathematics and Statistics
University of Turku
Turku, Finland

Marko M. Mäkelä
Department of Mathematics and Statistics
University of Turku
Turku, Finland

Sona Taheri
School of Science, Engineering and Information Technology
Federation University Australia
Ballarat
Victoria, Australia

ISBN 978-3-030-34909-7 ISBN 978-3-030-34910-3 (eBook)
https://doi.org/10.1007/978-3-030-34910-3

© Springer Nature Switzerland AG 2020
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Nonsmooth optimization (NSO) refers to the general problem of minimizing (or maximizing) functions that are typically not differentiable at their minimizers (maximizers). This kind of functions can be found in many applications, for instance, in image denoising, optimal control, data mining, economics, computational chemistry, mechanics, engineering, biology, and physics. Since the classical optimization theory presumes certain differentiability and strong regularity assumptions for the functions to be optimized, it cannot be directly utilized, nor can the methods developed for smooth problems.

The aim of this book is to give a survey of different numerical methods for solving NSO problems and to summarize the most recent developments in the field. The book covers both traditional methods and the methods developed to utilize special structures of the NSO problems.

The book opens with an introductory chapter where the main notations, notions, and concepts—used throughout the book—are described. In Chap. 2: "Advances in Low-Memory Subgradient Optimization" by P.E. Dvurechensky, A.V. Gasnikov, E.A. Nurminski, and F.S. Stonyaking, the authors give some survey on the subgradient method from historical perspective, present results on complexity of the method for smooth and nonsmooth convex and quasiconvex problems, and discuss recent advances in low-memory subgradient methods. Some applications of the subgradient method are also discussed. Chapter 3: "Standard Bundle Methods: Untrusted Models and Duality" by A. Frangioni provides the review of the basic of the bundle methods at three different axes: form of the stabilization, form of the model, and approximate evaluation of the function.

In Chap. 4: "A Second-Order Bundle Algorithm for Nonsmooth, Nonconvex Optimization Problems," the authors, H. Schichl and H. Fendl, extend the SQP-approach of the well-known bundle-Newton method for solving nonsmooth unconstrained minimization and the second-order bundle method to the general nonlinearly constrained case. Large-scale NSO methods, such as the limited memory bundle, the diagonal bundle, and the splitting metrics diagonal bundle methods, are discussed in Chap. 5: "Limited Memory Bundle Method and Its Variations for Large-Scale Nonsmooth Optimization" by N. Karmitsa.

The gradient sampling method is presented in Chap. 6: "Gradient Sampling Methods for Nonsmooth Optimization" by J.V. Burke, F.E. Curtis, A.S. Lewis, M.L. Overton, and L.E.A. Simões. The authors provide an overview of various enhancements that have been proposed to improve practical performance, as well as an overview of several extensions of the GS method to solve constrained problems.

Chapter 7: "Local Search for Nonsmooth DC Optimization with DC Equality and Inequality Constraints" by A.S. Strekalovsky addresses the NSO problems with the objective, equality, and inequality constraints given by DC functions. Using the exact penalty function, this problem is reduced to an unconstrained DC minimization problem and a local search method is designed to solve it. In Chap. 8: "Bundle Methods for Nonsmooth DC Optimization" by K. Joki and A.M. Bagirov provides a survey of bundle methods for solving nonsmooth DC optimization problems including their comparison using numerical results.

The main ideas and concepts related to the $\mathcal{VU}$-decomposition approach are discussed in Chap. 9: "Beyond First Order: $\mathcal{VU}$-Decomposition Methods" by Sh. Liu and C. Sagastizábal. It is shown that nonsmoothness often appears in a structured manner, and this fact can be exploited to design algorithms having superlinear convergence. In Chap. 10: "Beyond the Oracle: Opportunities of Piecewise Differentiation" by A. Griewank and A. Walther using the abs-linearization of a piecewise smooth objective in abs-normal form it is shown how directionally active generalized gradients can be calculated.

Numerical methods for solving generalized minimax problems are studied in Chap. 11: "Numerical Solution of Generalized Minimax Problems" by L. Lukšan, C. Matonoha, and J. Vlćek. Such problems include nonsmooth functions which are compositions of special smooth convex functions with maxima of smooth functions and in particular, functions which are represented as the sums of maxima of smooth functions. NSO problems with the objective and/or constraint functions that are assessed through "noisy" oracles are discussed in Chap. 12: "Bundle Methods for Inexact Data" by W. de Oliveira and M. Solodov. The approaches for solving such problems are demonstrated using different types of optimization problems.

In Chap. 13: "New Multiobjective Proximal Bundle Method with Scaled Improvement Function" by M.M. Mäkelä and O. Montonen, the authors describe how to use the improvement functions both for constraint handling and scalarization of multiple objectives. Since the standard improvement functions are sensitive to scaling their scaled versions are investigated. Application of the double bundle method for solving multiobjective optimization problems is studied in Chap. 14: "Multiobjective Double Bundle Method for DC Optimization" by O. Montonen and K. Joki. The improvement function is used to deal with the constrained multiobjective DC optimization problems. It is proved that under the mild assumptions the method converges to a weakly Pareto stationary point.

Chapter 15: "Mixed-Integer Linear Programming: Primal–Dual Relations and Dual Subgradient and Cutting-Plane Methods" by A.-B. Strömberg, T. Larsson, and M. Patriksson presents theory and methodology for solving mixed binary linear optimization problems by means of Lagrangian duals, subgradient methods, a cutting-plane model, and recovery of primal solutions. In Chap. 16: "On

Mixed Integer Nonsmooth Optimization" by V.-P. Eronen, T. Westerlund, and M.M. Mäkelä deterministic methods for solving convex mixed-integer NSO problems is reviewed. These methods include branch and bound, outer approximation, extended cutting-plane, extended supporting hyperplane and extended level bundle methods. Lagrangian relaxation methodology for solving integer programming problems and its various applications such as assignment problems, network optimization, wireless sensor networks, and machine learning are discussed in Chap. 17: "A View of Lagrangian Relaxation and Its Applications" by M. Gaudioso.

In Chap. 18: "Discrete Gradient Methods" by A.M. Bagirov, S. Taheri, and N. Karmitsa, two different semi-derivative-free methods are described. The discrete gradients are used to approximate subdifferentials of a class of nonsmooth functions. Some illustrative examples to demonstrate the performance of these methods are also given. Finally, Chap. 19: "Model-Based Methods in Derivative-Free Nonsmooth Optimization" by Ch. Audet and W. Hare presents a survey on the progress of model-based nonsmooth derivative-free optimization. Methods for constructing models of smooth functions, their accuracy, and frameworks for model-based nonsmooth derivative-free optimization are also discussed.

We hope that this book will demonstrate the significant progress that has occurred in numerical NSO in recent years and that the developments reported will motivate further research in NSO and its diverse applications.

Acknowledgements As editors, we would like to warmly thank all authors who accepted the invitation to contribute to this book and all reviewers who provided constructive comments.

This work was supported by the University of Turku (Finland), Federation University Australia, Universitá della Calabria (Italy), the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project No. DP190100580), and by the Academy of Finland (Project No. 289500 and 319274).

We wish to acknowledge gratefully the Springer International Publisher and in particular, Christian Rauscher, Senior Editor, Business, Operations Research, and Information Systems, for the strong support during the preparation of this book.

Ballarat, VIC, Australia — Adil M. Bagirov
Rende (CS), Italy — Manlio Gaudioso
Turku, Finland — Napsu Karmitsa
Turku, Finland — Marko M. Mäkelä
Ballarat, VIC, Australia — Sona Taheri
June 2019

Contents

Contributors

Charles Audet GERAD and Département de mathématiques et génie industriel, École Polytechnique de Montréal, Montréal, QC, Canada

Adil M. Bagirov School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, VIC, Australia

James V. Burke Department of Mathematics, University of Washington, Seattle, WA, USA

Frank E. Curtis Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

Welington de Oliveira Centre de Mathématiques Appliquées (CMA), MINES ParisTech, Paris, France

Pavel E. Dvurechensky Applied Analysis and Stochastic, Weierstrass Institute, Berlin, Germany

Ville-Pekka Eronen Department of Mathematics and Statistics, University of Turku, Turku, Finland

Hannes Fendl Faculty of Mathematics, University of Vienna, Wien, Austria

Antonio Frangioni Dipartimento di Informatica, Universitá di Pisa, Pisa, Italy

Alexander V. Gasnikov Moscow Institute of Physics and Technology, Moscow, Russia

Manlio Gaudioso Dipartimento di Elettronica e Sistemistica, Universitá della Calabria, Arcavacata, Italy

Andreas Griewank Department of Mathematics, Humboldt University zu Berlin, Berlin, Germany

Warren Hare Mathematics, University of British Columbia, Vancouver, BC, Canada

Kaisa Joki Department of Mathematics and Statistics, University of Turku, Turku, Finland

Napsu Karmitsa Department of Mathematics and Statistics, University of Turku, Turku, Finland

Torbjörn Larsson Department of Mathematics, Linköping University, Linköping, Sweden

Adrian S. Lewis School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Shuai Liu IMECC-UNICAMP, Cidade Universitária, Rio de Janeiro, Brazil

Ladislav Lukšan Institute of Computer Science, The Czech Academy of Sciences, Prague 1, Czech Republic

Marko M. Mäkelä Department of Mathematics and Statistics, University of Turku, Turku, Finland

Ctirad Matonoha Institute of Computer Science, The Czech Academy of Sciences, Prague 1, Czech Republic

Outi Montonen Department of Mathematics and Statistics, University of Turku, Turku, Finland

Evgeni A. Nurminski School of Natural Sciences, Far Eastern Federal University, Vladivostok, Russia

Michael L. Overton Courant Institute of Mathematical Sciences, New York University, New York, NY, USA

Michael Patriksson Chalmers University of Technology and the University of Gothenburg, Göteborg, Sweden

Claudia Sagastizábal IMECC-UNICAMP, Cidade Universitária, Rio de Janeiro, Brazil

Hermann Schichl Faculty of Mathematics, University of Vienna, Wien, Austria

Lucas E. A. Simões Department of Applied Mathematics, University of Campinas, Campinas, Brazil

Mikhail Solodov IMPA – Instituto de Matemática Pura e Aplicada, Rio de Janeiro, Brazil

Fedor S. Stonyakin Algebra and Functional Analysis Department, V.I. Vernadsky Crimean Federal University, Simferopol, Russia

Alexander S. Strekalovsky Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, Novosibirsk, Russia

Ann-Brith Strömberg Chalmers University of Technology and the University of Gothenburg, Göteborg, Sweden

Sona Taheri School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, VIC, Australia

Jan Vlček Institute of Computer Science, The Czech Academy of Sciences, Prague 1, Czech Republic

Andrea Walther Department of Mathematics, Humboldt University zu Berlin, Berlin, Germany

Tapio Westerlund Department of Mathematics and Statistics, University of Turku, Turku, Finland

Acronyms and Symbols

$\mathbb{R}^n$	n-Dimensional Euclidean space
$\mathbb{N}$	Set of natural numbers
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	(Column) vectors
$\boldsymbol{x}^T$	Transposed vector
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y}$	Inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$
$\|\boldsymbol{x}\|$	Euclidean norm of $\boldsymbol{x}$ in $\mathbb{R}^n$, $\|\boldsymbol{x}\| = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{\frac{1}{2}}$
$\|\boldsymbol{x}\|_1$	L_1-Norm of $\boldsymbol{x}$ in $\mathbb{R}^n$
$\|\boldsymbol{x}\|_\infty$	L_∞-Norm of $\boldsymbol{x}$ in $\mathbb{R}^n$
x_i	i-th component of vector $\boldsymbol{x}$
$\{\boldsymbol{x}_k\}, \{\boldsymbol{x}^k\}$	Sequence of vectors
$\boldsymbol{a} \prec \boldsymbol{b}$	Partial order of vectors, $\boldsymbol{a} \prec \boldsymbol{b} \iff a_i b_i \leq b_i^2$ for all $i \in \mathbb{R}^n$
$\mathbf{0}$	Zero vector or matrix
$\mathbf{1}$	Vector of ones
$a, b, c, \alpha, \varepsilon, \lambda$	Scalars
$t \downarrow 0$	$t \to 0_+$
A, B	Matrices
$(A)_{ij}$	Element of matrix A in row i of column j
A^T	Transposed matrix
A^{-1}	Inverse of matrix A
$\|A\|_{m \times n}$	Matrix norm $\|A\|_{m \times n} = \left(\sum_{i=1}^m \|A_i\|^2\right)^{\frac{1}{2}}$
$A \succ 0$	A is a positive definite matrix
I	Identity matrix
$\boldsymbol{e}_i$	i-th column of the identity matrix
$\det A$	Determinant of matrix A
$\operatorname{tr} A$	Trace of matrix A
$\operatorname{diag}[\theta_1, \ldots, \theta_n]$	Diagonal matrix with diagonal elements $\theta_1, \ldots, \theta_n$
$A^\dagger$	Moore–Penrose pseudoinverse of matrix A
$B(\boldsymbol{x}; r)$	Open ball with radius r and central point $\boldsymbol{x}$
$\bar{B}(\boldsymbol{x}; r)$	Closed ball with radius r and central point $\boldsymbol{x}$

S_1	Sphere of the unit ball
(a, b)	Open interval
$[a, b]$	Closed interval
$[a, b)$, $(a, b]$	Half-open intervals
$H(\boldsymbol{p}, \alpha)$	Hyperplane
$H^+(\boldsymbol{p}, \alpha)$, $H^-(\boldsymbol{p}, \alpha)$	Half-spaces
S, U	Sets
cl S	Closure of set S
int S	Interior of set S
bd S	Boundary of set S
aff S	Affine hull of set $S \subset \mathbb{R}^n$
ri S	Relative interior of set S
conv S	Convex hull of set S
$\cap_{i=1}^m S_i$	Intersection of sets S_i, $i = 1, \ldots, m$
$\boldsymbol{x}_{\mathcal{S}}$	Projection of point $\boldsymbol{x} \in \mathbb{R}^n$ onto a subspace $\mathcal{S} \subset \mathbb{R}^n$
$\mathcal{S}^{\perp}$	Orthogonal complement of the linear subspace $\mathcal{S}$
$P_S(C)$	Projection of the points in set C onto set S
$\|\cdot\|_{\mathcal{S}}$	Euclidean norm of the linear subspace $\mathcal{S}$ induced by $\mathbb{R}^n$
$\langle \cdot, \cdot \rangle_{\mathcal{S}}$	Inner product of the linear subspace $\mathcal{S}$ induced by $\mathbb{R}^n$
cone S	Conic hull of set S
$T_S(\boldsymbol{x})$	Tangent cone of set S at $\boldsymbol{x} \in S$
$N_S(\boldsymbol{x})$	Normal cone of set S at $\boldsymbol{x} \in S$
$K_S(\boldsymbol{x})$	Contingent cone of set S at $\boldsymbol{x} \in S$
short S	The smallest element in set S with respect to the Euclidean norm
$\text{lev}_\alpha f$	Level set of f with parameter α
epi f	Epigraph of f
$\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$	Sets of indices
$\|\mathcal{I}\|$	Number of elements in set $\mathcal{I}$
$\nabla f(\boldsymbol{x})$	Gradient of function f at $\boldsymbol{x}$
$\frac{\partial}{\partial x_i} f(\boldsymbol{x})$	Partial derivative of function f with respect to x_i
$\nabla^2 f(\boldsymbol{x})$	Hessian matrix of function f at $\boldsymbol{x}$
$C^m(\mathbb{R}^n)$	The space of functions $f : \mathbb{R}^n \to \mathbb{R}$ with continuous partial derivatives up to order m
$C^{m+}(\mathbb{R}^n)$	The space of functions $f : \mathbb{R}^n \to \mathbb{R}$ with LLC partial derivatives up to order m
$f'(\boldsymbol{x}; \boldsymbol{d})$	Directional derivative of function f at $\boldsymbol{x}$ in the direction $\boldsymbol{d}$
$f'_\varepsilon(\boldsymbol{x}; \boldsymbol{d})$	ε-Directional derivative of function f at $\boldsymbol{x}$ in the direction $\boldsymbol{d}$
$f^\circ(\boldsymbol{x}; \boldsymbol{d})$	Generalized directional derivative of function f at $\boldsymbol{x}$ in the direction $\boldsymbol{d}$
dist$(\boldsymbol{x}, S)$	Distance of the point $\boldsymbol{x}$ to the set S
$\partial_c f(\boldsymbol{x})$	Subdifferential of convex function f at $\boldsymbol{x}$
$\partial f(\boldsymbol{x})$	Subdifferential of function f at $\boldsymbol{x}$

$\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$	Subgradient of function f at $\boldsymbol{x}$
$\partial_\varepsilon f(\boldsymbol{x})$	ε-Subdifferential of convex function f at $\boldsymbol{x}$
$\partial_\varepsilon^G f(\boldsymbol{x})$	Goldstein ε-subdifferential of function f at $\boldsymbol{x}$
Ω_f	A set in $\mathbb{R}^n$ where function f is not differentiable
$\hat{f}_k(\boldsymbol{x})$	Piecewise linear cutting-plane model of function f at $\boldsymbol{x}$
$\nabla \boldsymbol{h}(\boldsymbol{x})$	Jacobian matrix of function $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ at $\boldsymbol{x}$
$\partial \boldsymbol{h}(\boldsymbol{x})$	Generalized Jacobian matrix of function $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ at $\boldsymbol{x}$
dom	Domain of a function
sign	Signum function
AD	Algorithmic differentiation
BM	Bundle method
COP	Convex optimization problem
CPM	Cutting-plane method
DC	Difference of convex functions
DFO	Derivative-free optimization
FOM	First-order minimality
GS	Gradient sampling
LD	Lagrangian dual
LIKQ	Linear independent kink qualification
LLC	Locally Lipschitz continuous
LP	Linear programming problem
MILP	Mixed integer linear programming
MINLP	Mixed integer nonlinear programming
MIQP	Mixed integer quadratic programming
MPEC	Mathematical programming with equilibrium constraints
NSO	Nonsmooth optimization
QP	Quadratic programming problem
QCQP	Quadratically constrained quadratic problem
SDP	Semidefinite programming
SQP	Sequential quadratic programming

Chapter 1
Introduction

Adil M. Bagirov, Manlio Gaudioso, Napsu Karmitsa, Marko M. Mäkelä, and Sona Taheri

Nonsmooth optimization (NSO) is among the most challenging tasks in the field of mathematical programming. It addresses optimization problems where objective and/or constraint functions have discontinuous gradients. NSO problems arise in many real life applications. Moreover, some smooth optimization techniques like different decomposition methods, dual formulations and exact penalty methods may lead us to solve NSO problems being either smaller in dimension or simpler in structure. In addition, some optimization problems may be analytically smooth but numerically nonsmooth. This is the case, for instance, with noisy input data and so-called stiff problems, which are numerically unstable and behave like nonsmooth problems.

The history of NSO methods dates back to the 1960s when the very first NSO method—the subgradient method was developed. In the 1960s and early 1970s, NSO was mainly applied to solve minimax problems as well as large linear problems using their decompositions. The most important developments in NSO start with the introduction of the bundle methods in mid-1970s being inspired by the classical cutting-plane method. In its original form, the bundle method was introduced to

A. M. Bagirov · S. Taheri
School of Science, Engineering and Information Technology, Federation University, Ballarat, VIC, Australia
e-mail: a.bagirov@federation.edu.au; s.taheri@federation.edu.au

M. Gaudioso
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, Rende, Italy
e-mail: manlio.gaudioso@unical.it

N. Karmitsa · M. M. Mäkelä (✉)
University of Turku, Department of Mathematics and Statistics, Turku, Finland
e-mail: napsu@karmitsa.fi; makela@utu.fi

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_1

solve nonsmooth convex problems, but the usage of the Clarke subdifferential allowed to extend bundle methods also to solve nonconvex problems in the 1980s.

Since the early 1990s, a significant progress has been made in numerical NSO. Especially different versions of bundle methods including the proximal bundle, trust-region, level bundle, variable metric, bundle-Newton and spectral methods have been introduced. In addition, outside of the traditional subgradient and bundle framework, some new derivative-free and gradient sampling techniques have gained ground. Recently, methods exploiting some special structure of the problems, like DC or VU-decompositions have been a subject of the growing interest. Furthermore, NSO has extended its territory outside the single objective continuous optimization to the fields of multiobjective and discrete optimization.

Next we will give some notations and definitions to be used in the forthcoming parts, as well as short overviews of the basic subgradient and bundle methods.

1.1 Notations

Throughout this book we use the following notations. All the vectors $\boldsymbol{x}$ are considered as column vectors and, correspondingly, all the transposed vectors $\boldsymbol{x}^T$ are considered as row vectors. We denote either by $\boldsymbol{x}^T\boldsymbol{y}$ or by $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ the usual inner product and by $\|\boldsymbol{x}\|$ the norm in the n-dimensional real Euclidean space $\mathbb{R}^n$. In other words,

$$\boldsymbol{x}^T\boldsymbol{y} = \langle \boldsymbol{x}, \boldsymbol{y}\rangle = \sum_{i=1}^{n} x_i y_i \qquad \text{and} \qquad \|\boldsymbol{x}\| = \left(\boldsymbol{x}^T\boldsymbol{x}\right)^{\frac{1}{2}} = \langle \boldsymbol{x}, \boldsymbol{x}\rangle^{\frac{1}{2}},$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $x_i, y_i \in \mathbb{R}$ are the i-th components of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

We denote by $[\boldsymbol{x}, \boldsymbol{y}]$ the closed line-segment joining $\boldsymbol{x}$ and $\boldsymbol{y}$, that is,

$$[\boldsymbol{x}, \boldsymbol{y}] = \left\{ z \in \mathbb{R}^n \mid z = \lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \ \text{ for } \ 0 \le \lambda \le 1 \right\},$$

and by $(\boldsymbol{x}, \boldsymbol{y})$ the corresponding open line-segment.

An open ball and closed ball with the center $\boldsymbol{x} \in \mathbb{R}^n$ and the radius $r > 0$ are denoted by $B(\boldsymbol{x}; r)$ and $\bar{B}(\boldsymbol{x}; r)$, respectively:

$$B(\boldsymbol{x}; r) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| < r\} \quad \text{and}$$

$$\bar{B}(\boldsymbol{x}; r) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| \le r\}.$$

We also denote by S_1 the sphere of the unit ball as follows:

$$S_1 = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y}\| = 1\}.$$

The closure, interior, boundary, and convex hull of the set $S \subset \mathbb{R}^n$ are denoted by cl S, int S, bd S, and conv S, respectively.

1.2 Definitions and Basic Results

We consider an optimization problem of the form

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \tag{1.1}$$

where the *objective function* f and the *feasible region* $S \subset \mathbb{R}^n$ will meet different assumptions and different structures depending on a problem.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}), \text{ for all } \boldsymbol{x}, \boldsymbol{y} \in R^n \text{ and } \lambda \in [0,1].$$

Most commonly we are operating with Lipschitz functions.

Definition 1.1 A function $f : \mathbb{R}^n \to \mathbb{R}$ is *locally Lipschitz continuous (LLC) at a point* $\boldsymbol{x} \in \mathbb{R}^n$ if there exists a Lipschitz constant $K > 0$ and $\varepsilon > 0$ such that

$$|f(\boldsymbol{y}) - f(\boldsymbol{z})| \leq K \|\boldsymbol{y} - \boldsymbol{z}\| \quad \text{for all } \boldsymbol{y}, \boldsymbol{z} \in B(\boldsymbol{x}; \varepsilon).$$

Definition 1.2 A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *LLC on a set* $U \subseteq \mathbb{R}^n$ if it is LLC at every point belonging to the set U. Note that, if $U = \mathbb{R}^n$ the function is called *LLC*.

A convex function $f : \mathbb{R}^n \to \mathbb{R}$ is always LLC.

Definition 1.3 A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *Lipschitz continuous on a set* $U \subseteq \mathbb{R}^n$ if there exists a Lipschitz constant $K > 0$ such that

$$|f(\boldsymbol{y}) - f(\boldsymbol{z})| \leq K \|\boldsymbol{y} - \boldsymbol{z}\| \quad \text{for all } \boldsymbol{y}, \boldsymbol{z} \in U.$$

If $U = \mathbb{R}^n$, then f is said to be *Lipschitz continuous*.

Definition 1.4 The limit

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{t \downarrow 0} \frac{f(\boldsymbol{x} + t\boldsymbol{d}) - f(\boldsymbol{x})}{t}$$

(if it exists) is called the *directional derivative* of f at $\boldsymbol{x} \in \mathbb{R}^n$ in the direction $\boldsymbol{d} \in \mathbb{R}^n$.

Definition 1.5 (Clarke) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a LLC function at $\boldsymbol{x} \in \mathbb{R}^n$. The *generalized directional derivative* of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d} \in \mathbb{R}^n$ is defined by

$$f^\circ(\boldsymbol{x}; \boldsymbol{d}) = \limsup_{\substack{\boldsymbol{y} \to \boldsymbol{x} \\ t \downarrow 0}} \frac{f(\boldsymbol{y} + t\boldsymbol{d}) - f(\boldsymbol{y})}{t}.$$

Definition 1.6 A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *subdifferentially regular* at $\boldsymbol{x} \in \mathbb{R}^n$ if it is LLC at $\boldsymbol{x}$, the directional derivative $f'(\boldsymbol{x}; \boldsymbol{d})$ exists in every direction $\boldsymbol{d} \in \mathbb{R}^n$ and we have

$$f^\circ(\boldsymbol{x}; \boldsymbol{d}) = f'(\boldsymbol{x}; \boldsymbol{d}), \text{ for all } \boldsymbol{d} \in \mathbb{R}^n.$$

Convex and continuously differentiable functions are examples of subdifferentially regular functions.

Definition 1.7 The *subdifferential of a convex function* $f : \mathbb{R}^n \to \mathbb{R}$ at $\boldsymbol{x} \in \mathbb{R}^n$ is the set $\partial_c f(\boldsymbol{x})$ of vectors $\boldsymbol{\xi} \in \mathbb{R}^n$ such that

$$\partial_c f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\xi}^T(\boldsymbol{y} - \boldsymbol{x}) \text{ for all } \boldsymbol{y} \in \mathbb{R}^n \right\}.$$

Each vector $\boldsymbol{\xi} \in \partial_c f(\boldsymbol{x})$ is called a *subgradient* of f at $\boldsymbol{x}$.

Definition 1.8 (Clarke) Let $f : \mathbb{R}^n \to \mathbb{R}$ be LLC at $\boldsymbol{x} \in \mathbb{R}^n$. Then the *subdifferential* of f at $\boldsymbol{x}$ is the set $\partial f(\boldsymbol{x})$ defined as

$$\partial f(\boldsymbol{x}) = \{ \boldsymbol{\xi} \in \mathbb{R}^n \mid f^\circ(\boldsymbol{x}; \boldsymbol{d}) \geq \boldsymbol{\xi}^T \boldsymbol{d} \text{ for all } \boldsymbol{d} \in \mathbb{R}^n \}.$$

The subdifferential $\partial f(\boldsymbol{x})$ is a nonempty, convex and compact set.

Theorem 1.1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be LLC at $\boldsymbol{x} \in \mathbb{R}^n$. Then*

$$f^\circ(\boldsymbol{x}; \boldsymbol{d}) = \max_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x})} \boldsymbol{\xi}^T \boldsymbol{d} \quad \textit{for all } \boldsymbol{d} \in \mathbb{R}^n.$$

Theorem 1.2 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be LLC at $\boldsymbol{x} \in \mathbb{R}^n$. Then*

$$\partial f(\boldsymbol{x}) = \operatorname{conv} \{ \boldsymbol{\xi} \in \mathbb{R}^n \mid \textit{there exists } \{\boldsymbol{x}_i\} \subset \mathbb{R}^n \setminus \Omega_f \textit{ such that } \boldsymbol{x}_i \to \boldsymbol{x} \textit{ and } \nabla f(\boldsymbol{x}_i) \to \boldsymbol{\xi} \}.$$

Here, Ω_f denotes the set of points in which f fails to be differentiable.

Note that if f is convex, then $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$ and if f is continuously differentiable, then $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.

Definition 1.9 For any $\varepsilon \geq 0$, the *ε-subdifferential of a convex function* $f : \mathbb{R}^n \to \mathbb{R}$ at $\boldsymbol{x} \in \mathbb{R}^n$ is the set

$$\partial_\varepsilon f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\xi}^T(\boldsymbol{y} - \boldsymbol{x}) - \varepsilon \text{ for all } \boldsymbol{y} \in \mathbb{R}^n \right\}.$$

Theorem 1.3 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function with a Lipschitz constant $K > 0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$. Then for any $\varepsilon \geq 0$, we have*

$$\partial_c f(\boldsymbol{y}) \subset \partial_\varepsilon f(\boldsymbol{x}) \quad \textit{for all } \boldsymbol{y} \in B\left(\boldsymbol{x}; \tfrac{\varepsilon}{2K}\right).$$

Definition 1.10 Let a function $f : \mathbb{R}^n \to \mathbb{R}$ be LLC at $\boldsymbol{x} \in \mathbb{R}^n$ and let $\varepsilon \geq 0$. Then the *Goldstein ε-subdifferential* of f is the set

$$\partial_\varepsilon^G f(\boldsymbol{x}) = \operatorname{cl} \operatorname{conv} \{ \partial f(\boldsymbol{y}) \mid \boldsymbol{y} \in B(\boldsymbol{x}; \varepsilon) \}.$$

Note that if f is convex, then for all $\varepsilon \geq 0$ we have

$$\partial_\varepsilon^G f(\boldsymbol{x}) \subseteq \partial_{2K\varepsilon} f(\boldsymbol{x}), \tag{1.2}$$

where $K > 0$ is the Lipschitz constant of f at $\boldsymbol{x}$.

Definition 1.11 A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *semismooth* or *weakly semismooth* at $\boldsymbol{x} \in \mathbb{R}^n$ if the limit

$$\lim_{\substack{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}+h\boldsymbol{d}'), \\ \boldsymbol{d}' \to \boldsymbol{d},\, h \downarrow 0}} \boldsymbol{\xi}^T \boldsymbol{d}', \tag{1.3}$$

or

$$\lim_{\substack{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}+h\boldsymbol{d}), \\ h \downarrow 0}} \boldsymbol{\xi}^T \boldsymbol{d} \tag{1.4}$$

exists for every $\boldsymbol{d} \in \mathbb{R}^n$, respectively.

Evidently, semismoothness implies weak semismoothness. The class of semismooth functions is fairly wide and it contains, for instance, convex, concave, max- and min-type functions. The weakly semismooth function f is directionally differentiable and

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{\substack{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}+h\boldsymbol{d}), \\ h \downarrow 0}} \boldsymbol{\xi}^T \boldsymbol{d}. \tag{1.5}$$

Next we give two classical but useful results in NSO.

Theorem 1.4 (Weierstrass) *If $S \subset \mathbb{R}^n$ is a nonempty compact set and $f : \mathbb{R}^n \to \mathbb{R}$ is continuous, then f attains its minimum and maximum over S.*

Theorem 1.5 (Rademacher) *Let $S \subset \mathbb{R}^n$ be an open set. A function $f : S \to \mathbb{R}$ that is LLC on S is differentiable almost everywhere on S.*

Definition 1.12 The *contingent cone* of a set $S \subset \mathbb{R}^n$ at a point $\boldsymbol{x}$ is given by

$$K_S(\boldsymbol{x}) = \left\{ \boldsymbol{d} \in \mathbb{R}^n \mid \text{there exist } t_i \downarrow 0 \text{ and } \boldsymbol{d}_i \to \boldsymbol{d} \text{ with } \boldsymbol{x} + t_i \boldsymbol{d}_i \in S \right\},$$

and the *tangent cone* by

$$T_S(\boldsymbol{x}) = \{\boldsymbol{d} \in \mathbb{R}^n \mid \text{for all } t_i \downarrow 0 \text{ and } \boldsymbol{x}_i \to \boldsymbol{x} \text{ with } \boldsymbol{x}_i \in S \\ \text{there exist } \boldsymbol{d}_i \to \boldsymbol{d} \text{ with } \boldsymbol{x}_i + t_i \boldsymbol{d}_i \in S\}.$$

Note that $T_S(\boldsymbol{x}) \subseteq K_S(\boldsymbol{x})$ and if S is convex, then we have $T_S(\boldsymbol{x}) = K_S(\boldsymbol{x})$.

Definition 1.13 The *polar cone* of a set $S \subset \mathbb{R}^n$ is given by the formula

$$S^{\leq} = \{\boldsymbol{d} \in \mathbb{R}^n \mid \boldsymbol{s}^T \boldsymbol{d} \leq 0 \text{ for all } \boldsymbol{s} \in S\}.$$

The *normal cone* of a set $S \subset \mathbb{R}^n$ at a point $\boldsymbol{x}$ is defined as the polar cone of the tangent cone, that is

$$N_S(\boldsymbol{x}) = T_S(\boldsymbol{x})^{\leq} = \{\boldsymbol{z} \in \mathbb{R}^n \mid \boldsymbol{z}^T \boldsymbol{d} \leq 0 \text{ for all } \boldsymbol{d} \in T_S(\boldsymbol{x})\}.$$

Finally, we formulate necessary and sufficient optimality conditions for the problem (1.1).

Theorem 1.6 *Let $\boldsymbol{x}^*$ be the local optimum of the problem (1.1), where $f : \mathbb{R}^n \to \mathbb{R}$ is LLC at $\boldsymbol{x}^* \in S$. Then*

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*) + N_S(\boldsymbol{x}^*). \tag{1.6}$$

If, in addition, f is a convex function and S is a convex set, then the condition (1.6) implies that $\boldsymbol{x}^*$ is a global optimum of (1.1).

In this book, *unconstrained optimization* problems are considered in most of the chapters, in other words $S = \mathbb{R}^n$ in the problem (1.1). Since $N_{\mathbb{R}^n}(\boldsymbol{x}_k) = \{\mathbf{0}\}$ the optimality condition (1.6) reduces to

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*). \tag{1.7}$$

A point $\boldsymbol{x}^*$ is called *Clarke stationary* if it satisfies the optimality condition (1.7).

1.3 Nonsmooth vs. Smooth Optimization

Next we consider the solution process of the problem (1.1). For simplicity we assume that $S = \mathbb{R}^n$, thus we have an unconstrained optimization problem. Most iterative optimization methods are so-called descent methods in the sense that from the current iteration point $\boldsymbol{x}_k$ their aim is to find a *descent direction* $\boldsymbol{d}_k \in S_1$ such that for some $\delta > 0$, we have

$$f(\boldsymbol{x}_k + t\boldsymbol{d}_k) < f(\boldsymbol{x}_k) \quad \text{for all } t \in (0, \delta]. \tag{1.8}$$

Suppose that $\boldsymbol{x}_k$ is not yet an optimal solution of (1.1). It follows from Theorem 1.6 that $\mathbf{0} \notin \partial f(\boldsymbol{x}_k)$ and thus, due to Definition 1.8, there must be at least one $\boldsymbol{d}_k \in S_1$ such that $f^\circ(\boldsymbol{x}_k; \boldsymbol{d}_k) < 0$. This implies that the condition (1.8) is valid and thus, there exists a descent direction.

A *greedy algorithm* tries to find the *steepest descent* direction by solving the problem

$$\min_{\boldsymbol{d} \in S_1} f^\circ(\boldsymbol{x}_k; \boldsymbol{d}) \tag{1.9}$$

or, due to Theorem 1.1, equivalently

$$\min_{\boldsymbol{d} \in S_1} \max_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k)} \boldsymbol{\xi}^T \boldsymbol{d}.$$

Since S_1 and $\partial f(\boldsymbol{x}_k)$ are compact sets and $\partial f(\boldsymbol{x}_k)$ is convex "von Neumann's minimax" Theorem allows us to change the optimization order and we get

$$\max_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k)} \min_{\boldsymbol{d} \in S_1} \boldsymbol{\xi}^T \boldsymbol{d} = \max_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k)} \boldsymbol{\xi}^T (-\boldsymbol{\xi} / \|\boldsymbol{\xi}\|) = - \min_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k)} \|\boldsymbol{\xi}\|.$$

Thus, the steepest descent direction for a LLC function f at $\boldsymbol{x}_k$ is derived by

$$\boldsymbol{d}_k = - \operatorname*{argmin}_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k)} \|\boldsymbol{\xi}\|. \tag{1.10}$$

Notice that if f is smooth (i.e. continuously differentiable), we have $\partial f(\boldsymbol{x}_k) = \{\nabla f(\boldsymbol{x}_k)\}$ and (1.10) reduces to the standard steepest descent direction

$$\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k).$$

Since we have found a descent search direction we can perform a *line search* in order to specify the optimal step size by

$$t_k = \operatorname*{argmin}_{t > 0} f(\boldsymbol{x}_k + t \boldsymbol{d}_k). \tag{1.11}$$

Now we are ready to update the current iteration point by

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + t_k \boldsymbol{d}_k$$

and according to (1.9) and (1.11) we obtain a descent step, in other words $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$. Furthermore, the iteration can be stopped if $\boldsymbol{d}_k = \mathbf{0}$. Namely, in that case (1.10) implies that $\mathbf{0} \in \partial f(\boldsymbol{x}_k)$ and due to Theorem 1.6 this is the necessary optimality condition for $\boldsymbol{x}_k$ to be a local optimum for the problem (1.1). In addition, if f is convex we have found a global optimum. Thus, in practice we can stop our algorithm whenever

$$\|\boldsymbol{d}_k\| < \varepsilon_s$$

for a predefined optimality tolerance $\varepsilon_s > 0$. Note that in the smooth case this corresponds to the classical stopping criterion

$$\|\nabla f(\boldsymbol{x}_k)\| < \varepsilon_s.$$

Now we summarize our simple algorithm.

Algorithm 1.1: Generalized steepest descent method

Data: A final optimality tolerance $\varepsilon_s > 0$.
Step 1. *(Initialization)* Select a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$ and set $k := 1$.
Step 2. *(Direction finding)* Calculate the search direction $\boldsymbol{d}_k$ by (1.10).
Step 3. *(Stopping criterion)* If $\|\boldsymbol{d}_k\| < \varepsilon_s$, then **stop**.
Step 4. *(Line search)* Calculate the step size $t_k > 0$ by (1.11).
Step 5. *(Updating)* Set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + t_k \boldsymbol{d}_k$ and $k := k + 1$. Go to Step 2.

The next example shows how this algorithm works in practice.

Example 1.1 Suppose, that we want to minimize the convex function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(\boldsymbol{x}) = \max\{|x_1|, |x_2|\} = \max\{x_1, -x_1, x_2, -x_2\}$$

from the starting point $\boldsymbol{x}_1 = (1, 1)^T$. By Theorem 1.2 we have

$$\partial f(\boldsymbol{x}_1) = \operatorname{conv}\{(1, 0)^T, (0, 1)^T\}$$

and thus in Step 2 of Algorithm 1.1 we get

$$\boldsymbol{d}_1 = -\operatorname*{argmin}_{\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_1)} \|\boldsymbol{\xi}\| = (-\tfrac{1}{2}, -\tfrac{1}{2})^T.$$

Note that

$$\begin{aligned} f^\circ(\boldsymbol{x}_1; \boldsymbol{d}_1) = f'(\boldsymbol{x}_1; \boldsymbol{d}_1) &= \lim_{t \downarrow 0} \frac{f(\boldsymbol{x}_1 + t\boldsymbol{d}_1) - f(\boldsymbol{x}_1)}{t} \\ &= \lim_{t \downarrow 0} \frac{f((1 - \frac{1}{2}t, 1 - \frac{1}{2}t)^T) - f((1, 1)^T)}{t} \\ &= \lim_{t \downarrow 0} \frac{|1 - \frac{1}{2}t| - 1}{t} = -\tfrac{1}{2} < 0, \end{aligned}$$

meaning that $\boldsymbol{d}_1$ is a descent direction. In Step 4 we specify the step size by

$$t_1 = \underset{t>0}{\operatorname{argmin}} f(\boldsymbol{x}_1 + t\boldsymbol{d}_1) = \underset{t>0}{\operatorname{argmin}} |1 - \tfrac{1}{2}t| = 2$$

and thus in Step 5 we get

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + t_1\boldsymbol{d}_1 = (1,1)^T + 2(-\tfrac{1}{2}, -\tfrac{1}{2})^T = (0,0)^T.$$

Now we have

$$\partial f(\boldsymbol{x}_2) = \operatorname{conv}\{(1,0)^T, (-1,0)^T, (0,1)^T, (0,-1)^T\}$$

and we get

$$\boldsymbol{d}_2 = -\underset{\boldsymbol{\xi}\in\partial f(\boldsymbol{x}_2)}{\operatorname{argmin}} \|\boldsymbol{\xi}\| = (0,0)^T.$$

Then in Step 3 the stopping criterion

$$\|\boldsymbol{d}_2\| = 0 < \varepsilon_s$$

is valid for any optimality tolerance $\varepsilon_s > 0$. Since $\mathbf{0} \in \partial f(\boldsymbol{x}_2)$ and f is convex, Theorem (1.6) implies that $\boldsymbol{x}_2 = (0,0)^T$ is a global minimum.

To conclude, via subgradients and subdifferentials, we can completely generalize the classical optimization theory. Thus, in principle, we do not need special NSO methods since we could use Algorithm 1.1, and for LLC functions the gradient could be replaced by the minimum norm subgradient. However, this necessitates the knowledge of the whole subdifferetial $\partial f(\boldsymbol{x})$ at every point $\boldsymbol{x} \in \mathbb{R}^n$ which is a big requirement in practice and in most cases we have to content with the fact that we can only calculate

- one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential $\partial f(\boldsymbol{x})$. (1.12)

This impairment involves several drawbacks to the above procedure:

1. The opposite direction of an arbitrary subgradient is not necessarily a descent direction. For instance, if in Example 1.1 we choose $\boldsymbol{\xi}_1 = (1,0)^T \in \partial f(\boldsymbol{x}_1)$ we

have $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$ and

$$f^\circ(\boldsymbol{x}_1; \boldsymbol{d}_1) = \lim_{t \downarrow 0} \frac{f(\boldsymbol{x}_1 + t\boldsymbol{d}_1) - f(\boldsymbol{x}_1)}{t}$$
$$= \lim_{t \downarrow 0} \frac{f((1-t, 1)^T) - f((1, 1)^T)}{t}$$
$$= \lim_{t \downarrow 0} \frac{1-1}{t} = 0 \not< 0,$$

meaning that $\boldsymbol{d}_1$ is not a descent direction.

2. The line search operation of Step 4 becomes unrealistic since the search direction is not necessarily a descent one.
3. The stopping criterion of Step 3 is not applicable anymore since the norm of an arbitrary subgradient $\|\boldsymbol{\xi}_k\|$ need not to be small even if $\mathbf{0} \in \partial f(\boldsymbol{x}_k)$.

1.4 Subgradient Methods

The idea behind the *subgradient methods* is to follow Algorithm 1.1 by taking into account the restriction (1.12) and its consequences 1–3 listed above. Since we have only one arbitrary subgradient $\boldsymbol{\xi}_k \in \partial f(\boldsymbol{x}_k)$ instead of the whole subdifferential $\partial f(\boldsymbol{x}_k)$ the search direction problem (1.10) reduces to

$$\boldsymbol{d}_k = -\boldsymbol{\xi}_k,$$

leading to the scheme

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + t_k \boldsymbol{d}_k, \tag{1.13}$$

where the step size t_k has to be chosen a priori due to the lack of implementable line search operation. The following lemma gives a hint how to choose the step size.

Lemma 1.1 *Suppose that $\boldsymbol{x}^*$ is an optimal solution of the problem (1.1), where $S = \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is convex. If $\boldsymbol{x}_k$ is not yet optimal, we have*

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| < \|\boldsymbol{x}_k - \boldsymbol{x}^*\|,$$

whenever

$$0 < t_k < 2[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)]/\|\boldsymbol{\xi}_k\|^2.$$

Although we can not guarantee anymore that $\boldsymbol{x}_{k+1}$ is better than $\boldsymbol{x}_k$ when comparing the objective function values, it is closer to the optimal point $\boldsymbol{x}^*$ if the step size is positive and small enough. This advices us to choose t_k such that

$$t_k > 0 \qquad \text{and} \qquad \lim_{k\to\infty} t_k = 0. \tag{1.14}$$

On the other hand, due to the triangle inequality we have

$$\|\boldsymbol{x}_1 - \boldsymbol{x}_{k+1}\| \leq \sum_{j=1}^{k} t_j.$$

Thus, in order to guarantee the global convergence of the method the sequence of the step sizes should diverge, in other words

$$\sum_{j=1}^{\infty} t_j = \infty. \tag{1.15}$$

Furthermore, in order to improve the convergence properties, the sequence of the squared step sizes should convergent, that is

$$\sum_{j=1}^{\infty} t_j^2 < \infty. \tag{1.16}$$

Due to the fact that we do not have any reliable (sub)gradient-based stopping criterion, the number of iterations $k_{\max}$ has also to be fixed a priori.

Now we are ready to present the basic subgradient algorithm.

Algorithm 1.2: Subgradient method

Data: A final optimality tolerance $\varepsilon_s > 0$, maximum number of iterations $k_{\max} > 0$ and the sequence of step sizes t_k satisfying the requirements (1.14)–(1.16).

Step 1. *(Initialization)* Select a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$, set $\boldsymbol{x}_{\text{best}} = \boldsymbol{x}_1$, calculate $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$ and set $k := 1$.

Step 2. *(Direction finding)* Set the search direction $\boldsymbol{d}_k = -\boldsymbol{\xi}_k$.

Step 3. *(Stopping criterion)* If $\|\boldsymbol{d}_k\| < \varepsilon_s$ or $k + 1 > k_{\max}$, then **stop**.

Step 4. *(Updating)* Set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + t_k \boldsymbol{d}_k$, calculate $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{x}_{k+1})$ and set $k := k + 1$. If $f(\boldsymbol{x}_k) < f(\boldsymbol{x}_{\text{best}})$, set $\boldsymbol{x}_{\text{best}} = \boldsymbol{x}_k$. Go to Step 2.

Let us next apply the subgradient method to the same problem as in Example 1.1.

Example 1.2 As in Example 1.1 consider the minimization of the convex function $f : \mathbb{R}^2 \to \mathbb{R}$

$$f(\boldsymbol{x}) = \max\{|x_1|, |x_2|\} = \max\{x_1, -x_1, x_2, -x_2\}$$

from the starting point $\boldsymbol{x}_1 = (1, 1)^T$. If we choose $\boldsymbol{\xi}_1 = (1, 0)^T \in \partial f(\boldsymbol{x}_1) = \text{conv}\{(1, 0)^T, (0, 1)^T\}$, then $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$ is not a descent direction as showed above. Since $f(\boldsymbol{x}_1) = 1$, $\|\boldsymbol{\xi}_1\| = 1$, $\boldsymbol{x}^* = (0, 0)^T$ and $f(\boldsymbol{x}^*) = 0$ we have

$$2[f(\boldsymbol{x}_1) - f(\boldsymbol{x}^*)]/\|\boldsymbol{\xi}_1\|^2 = 2, \tag{1.17}$$

meaning by Lemma 1.1 that if we choose the step size t_1 such that $0 < t_1 < 2$ we have

$$\begin{aligned}\|\boldsymbol{x}_2 - \boldsymbol{x}^*\| &= \|\boldsymbol{x}_1 + t_1\boldsymbol{d}_1\| \\ &= \|(1 - t_1, 1)^T\| = \sqrt{(1 - t_1)^2 + 1} < \sqrt{2} = \|\boldsymbol{x}_1 - \boldsymbol{x}^*\|.\end{aligned}$$

This is still a rather wide range and the calculation of the upper limit (1.17) necessitates the knowledge of the optimal value $f(\boldsymbol{x}^*)$. Furthermore, the choice of the step size clearly affects the operation of the method being one of its weaknesses. The best choice (which the method does not know!) would be $t_1 = 1$ leading to the next iteration point $\boldsymbol{x}_2 = (0, 1)^T$. Then $\partial f(\boldsymbol{x}_2) = (0, 1)^T$ and thus the next search direction $\boldsymbol{d}_2 = (0, -1)^T$ points into the origin. The range for the step size remains to be $0 < t_2 < 2$ and the feasible choice $t_2 = 1$ would give $\boldsymbol{x}_3 = (0, 0)^T = \boldsymbol{x}^*$.

What comes to the convergence properties of the basic subgradient algorithm, if f is convex and the step sizes satisfy the conditions (1.14) and (1.15) we can prove that

$$\lim_{k\to\infty} f(\boldsymbol{x}_k) = f(\boldsymbol{x}^*),$$

meaning that the objective values of the iterates converge to the optimal objective value of (1.1). If, in addition, the condition (1.16) is fulfilled it holds that

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}^*,$$

in other words, the subgradient method is globally convergent.

Due to their simple structure subgradient methods are very popular and widely used in NSO. They are applicable especially for large scale problems because of their low storage requirements. Nevertheless, the convergence speed of the standard

subgradient method may be slow (not even linear). To overcome this drawback some ideas from the conjugate gradient and the variable metric methods have been adopted to the subgradient context, for instance in Shor's space dilation methods.

1.5 Bundle Methods

The basic idea of bundle methods is to *approximate* the whole subdifferential of the objective function instead of using only one arbitrary subgradient at each point. In practice, this is done by gathering subgradients from the previous iterations into a bundle. Suppose that at the k-th iteration of the algorithm we have the current iteration point $\boldsymbol{x}_k$ and some trial points $\boldsymbol{y}_j \in \mathbb{R}^n$ (from past iterations) and subgradients $\boldsymbol{\xi}_j \in \partial f(\boldsymbol{y}_j)$ for $j \in \mathcal{J}_k$, where the index set $\mathcal{J}_k$ is a nonempty subset of $\{1, \ldots, k\}$. Then at every trial point $\boldsymbol{y}_j$ we can linearize the objective function f by

$$\bar{f}_j(\boldsymbol{x}) = f(\boldsymbol{y}_j) + \boldsymbol{\xi}_j^T(\boldsymbol{x} - \boldsymbol{y}_j).$$

Then the *linearization error*

$$\alpha_j^k = f(\boldsymbol{x}_k) - \bar{f}_j(\boldsymbol{x}_k)$$

tells, how well $\bar{f}_j$ approximates f at the current iteration point $\boldsymbol{x}_k$. Note that the convexity of f implies that for all $\boldsymbol{x} \in \mathbb{R}^n$ and $j \in \mathcal{J}_k$ we have

$$f(\boldsymbol{x}) \geq \bar{f}_j(\boldsymbol{x}) \quad \text{and} \quad \alpha_j^k \geq 0. \tag{1.18}$$

Then for some $\varepsilon_k > 0$ the *bundle*

$$\mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid \boldsymbol{\xi} = \sum_{j \in \mathcal{J}_k} \lambda_j \boldsymbol{\xi}_j, \sum_{j \in \mathcal{J}_k} \lambda_j = 1, \lambda_j \geq 0 \text{ and } \sum_{j \in \mathcal{J}_k} \lambda_j \alpha_j^k \leq \varepsilon_k \right\}$$

is a convex hull of $\boldsymbol{\xi}_j \in \partial f(\boldsymbol{y}_j)$ emphasizing good subgradients, in other words, those having small linearization errors α_j^k. Thus we have

$$\mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k) \subseteq \operatorname{cl}\operatorname{conv}\{\boldsymbol{\xi}_j \mid j \in \mathcal{J}_k\}. \tag{1.19}$$

Lemma 1.2 *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex and $\varepsilon_k \geq 0$, then $\mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k)$ is a convex and compact set such that*

$$\mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k) \subseteq \partial_{\varepsilon_k} f(\boldsymbol{x}_k). \tag{1.20}$$

Due to (1.20) and the fact that $\partial_{\varepsilon_k} f(\boldsymbol{x}_k)$ converges to $\partial_0 f(\boldsymbol{x}_k) = \partial f(\boldsymbol{x}_k)$ whenever $\varepsilon_k \downarrow 0$, we replace the subdifferental by the bundle in the search direction problem (1.10) and

$$\boldsymbol{d}_k = - \operatorname*{argmin}_{\boldsymbol{\xi} \in \mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k)} \|\boldsymbol{\xi}\|. \tag{1.21}$$

Taking into account the structure of the bundle we can rewrite this problem as

$$\begin{cases} \text{minimize} & \frac{1}{2} \| \sum\limits_{j \in \mathcal{J}_k} \lambda_j \boldsymbol{\xi}_j \|^2 \\ \text{subject to} & \sum\limits_{j \in \mathcal{J}_k} \lambda_j \alpha_j^k \leq \varepsilon_k, \\ & \sum\limits_{j \in \mathcal{J}_k} \lambda_j = 1, \\ & \lambda_j \geq 0 \quad \text{for all } j \in \mathcal{J}_k. \end{cases} \tag{1.22}$$

Applying Lagrangian relaxation and dualization, the problem (1.22) is equivalent to

$$\begin{cases} \text{minimize} & v + \frac{1}{2} u_k \|\boldsymbol{d}\|^2 \\ \text{subject to} & -\alpha_j^k + \boldsymbol{\xi}_j^T \boldsymbol{d} \leq v \quad \text{for all } j \in \mathcal{J}_k, \end{cases} \tag{1.23}$$

where u_k is a Lagrange multiplier of the first constraint of the problem (1.22). Note that by defining the so-called *cutting-plane model*

$$\hat{f}_k(\boldsymbol{x}) = \max_{j \in \mathcal{J}_k} \bar{f}_j(\boldsymbol{x}), \tag{1.24}$$

the problem (1.23) can be written in the form

$$\boldsymbol{d}_k = \operatorname*{argmin}_{\boldsymbol{d} \in \mathbb{R}^n} \{ \hat{f}(\boldsymbol{x}_k + \boldsymbol{d}) + \tfrac{1}{2} u_k \|\boldsymbol{d}\|^2 \}. \tag{1.25}$$

If f is convex, it follows from (1.18) and (1.24) that the cutting-plane model $\hat{f}_k$ underestimates f everywhere. However, if f is nonconvex, then the cutting-plane model is not guaranteed to be an underestimate of the objective even locally and the linearization error may be negative.

One possibility is to replace the linearization error α_j^k by so-called *subgradient locality measure* defined by

$$\beta_j^k = \max \{ |\alpha_j^k| , \gamma \|\boldsymbol{x}_k - \boldsymbol{y}_j\|^2 \}, \tag{1.26}$$

where $\gamma \geq 0$. If f is convex, we can set $\gamma = 0$ and the subgradient locality measure reverts to the linearization error. Then, again (1.19) is valid and instead of Lemma 1.2 we get the following result.

Lemma 1.3 *If $f : \mathbb{R}^n \to \mathbb{R}$ is LLC and $\varepsilon_k \geq 0$, then $\mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k)$ is a convex and compact set such that*

$$\mathcal{B}_{\varepsilon_k} f(\boldsymbol{x}_k) \subseteq \partial^G_{\delta_k} f(\boldsymbol{x}_k), \tag{1.27}$$

where

$$\delta_k = \sqrt{\frac{\varepsilon_k}{\gamma \lambda_k}} \qquad \text{and} \qquad \lambda_k = \operatorname*{argmin}_{\lambda_j > 0} \lambda_j .$$

Since the objective function in the problem (1.23) is strictly convex it is easy to prove that if $\boldsymbol{d}_k \neq \boldsymbol{0}$, then $\boldsymbol{d}_k$ is a descent direction to the model $\hat{f}_k$. However, it is not necessarily descent for the original objective f or, at least, the decrease in function values may not be sufficient. For this reason, we have to use the following line search procedure.

Assume that $m_L \in (0, \frac{1}{2})$, $m_R \in (m_L, 1)$ and $\bar{t} \in (0, 1]$ are some fixed line search parameters. We first search for the largest number $t_k^L \in [0, 1]$ such that $t_k^L \geq \bar{t}$ and

$$f(\boldsymbol{x}_k + t_k^L \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + m_L t_k^L v_k, \tag{1.28}$$

where v_k is the predicted amount of descent

$$v_k = \hat{f}_k(\boldsymbol{x}_k + \boldsymbol{d}_k) - f(\boldsymbol{x}_k) < 0. \tag{1.29}$$

If such a parameter exists we take a *long serious step*

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k^L \boldsymbol{d}_k \qquad \text{and} \qquad \boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1}.$$

Otherwise, if (1.28) holds but $0 < t_k^L < \bar{t}$, we take a *short serious step*

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k^L \boldsymbol{d}_k \qquad \text{and} \qquad \boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_k^R \boldsymbol{d}_k$$

and, if $t_k^L = 0$, we take a *null step*

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k \qquad \text{and} \qquad \boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_k^R \boldsymbol{d}_k,$$

where $t_k^R > t_L^k$ is such that

$$-\beta_{k+1}^{k+1} + \boldsymbol{\xi}_{k+1}^T \boldsymbol{d}_k \geq m_R v_k. \tag{1.30}$$

In short serious steps and null steps, there exists discontinuity in the gradient[1] of f. Then the requirement (1.30) ensures that $\boldsymbol{x}_k$ and $\boldsymbol{y}_{k+1}$ lie on the opposite sides of this discontinuity and the new subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ will force a remarkable modification of the next search direction finding problem.

[1] Existing almost everywhere due to Rademacher's Theorem 1.5.

The iteration is terminated if we can not find any descent direction to the model $\hat{f}_k$, in other words if $\boldsymbol{d}_k \approx \mathbf{0}$ and the model is a good enough approximation of the original objective meaning that $\hat{f}_k(\boldsymbol{x}_k) \approx f(\boldsymbol{x}_k)$. Thus due to (1.29) we can stop our algorithm whenever

$$\|\boldsymbol{d}_k\| + |v_k| < \varepsilon_s.$$

Now we can summarize our standard bundle method.

Algorithm 1.3: Proximal bundle method

Data: A final optimality tolerance $\varepsilon_s > 0$, an initial weight $u_1 > 0$, line search parameters $m_L \in (0, \frac{1}{2})$, $m_R \in (m_L, 1)$ and $\bar{t} \in (0, 1]$, and distance measure parameter $\gamma \geq 0$.

Step 1. *(Initialization)* Select a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$, set $\boldsymbol{y}_1 := \boldsymbol{x}_1$ and $\mathcal{J}_1 = \{1\}$, calculate $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{y}_1)$ and set $k := 1$.

Step 2. *(Direction finding)* Calculate the search direction $\boldsymbol{d}_k$ by (1.25).

Step 3. *(Stopping criterion)* If $\|\boldsymbol{d}_k\| + |v_k| < \varepsilon_s$, then **stop**.

Step 4. *(Line search)* Find the step sizes $t_k^L \in [0, 1]$ and $t_k^R \in [t_k^L, 1]$ fulfilling (1.28) or (1.30) to take either a *null step* ($t_k^L = 0$) or a *serious step* ($t_k^L > 0$).

Step 5. *(Updating)* Set

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k^L \boldsymbol{d}_k \quad \text{and} \quad \boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_k^R \boldsymbol{d}_k,$$

calculate $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, choose $\mathcal{J}_{k+1} \subseteq \{1, \ldots, k+1\}$, update the weight u_{k+1} and set $k := k + 1$. Go to Step 2.

If the objective function is LLC and the weakly semismoothness assumption (Definition 1.11) is valid, Algorithm 1.3 can be proved to converge globally to a Clarke stationary point $\boldsymbol{x}^*$ meaning that $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Due to Theorem 1.6 this is the necessary optimality condition for $\boldsymbol{x}^*$ to be a local optimum for the problem (1.1). Moreover, if f is convex we have found a global optimum.

1.6 Forthcoming Chapters

The aim of this book is to survey different class of numerical methods developed for NSO and to give an overview to the most recent developments in the area. The book contains four parts, where the first one considers general methods. The next two parts are devoted to methods exploiting a special structure of a problem and the methods for special problems, respectively. The last part consists of the latest advancements in derivative-free NSO.

Part I
General Methods

Chapter 2
Advances in Low-Memory Subgradient Optimization

Pavel E. Dvurechensky, Alexander V. Gasnikov, Evgeni A. Nurminski, and Fedor S. Stonyakin

Abstract This chapter is devoted to the blackbox subgradient algorithms with the minimal requirements for the storage of auxiliary results, which are necessary to execute these algorithms. To provide historical perspective this survey starts with the original result of Shor which opened this field with the application to the classical transportation problem. The theoretical complexity bounds for smooth and nonsmooth convex and quasiconvex optimization problems are briefly exposed in what follows to introduce the relevant fundamentals of nonsmooth optimization. Special attention in this section is given to the adaptive step size policy which aims to attain lowest complexity bounds. Nondifferentiability of objective function in convex optimization significantly slows down the rate of convergence in subgradient optimization compared to the smooth case, but there are different modern techniques that allow to solve nonsmooth convex optimization problems faster than dictate theoretical lower complexity bounds. In this work the particular attention is given to Nesterov smoothing technique, Nesterov universal approach, and Legendre (saddle point) representation approach. The new results on universal mirror prox algorithms represent the original parts of the survey. To demonstrate application of nonsmooth

P. E. Dvurechensky
Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany

Institute for Information Transmission Problems RAS, Moscow, Russia
e-mail: pavel.dvurechensky@wias-berlin.de

A. V. Gasnikov
Moscow Institute of Physics and Technology, Dolgoprudny, Moscow District, Russia

Institute for Information Transmission Problems RAS, Moscow, Russia
e-mail: gasnikov@yandex.ru

E. A. Nurminski (✉)
Far Eastern Federal University, Vladivostok, Russia
e-mail: nurminskiy.ea@dvfu.ru

F. S. Stonyakin
V.I. Vernadsky Crimean Federal University, Simferopol, Republic of Crimea

Moscow Institute of Physics and Technology, Dolgoprudny, Moscow District, Russia
e-mail: fedyor@mail.ru

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_2

convex optimization algorithms to solution of huge-scale extremal problems we consider convex optimization problems with nonsmooth functional constraints and propose two adaptive mirror descent methods. The first method is of primal-dual variety and proved to be optimal in terms of lower oracle bounds for the class of Lipschitz continuous convex objectives and constraints. The advantages of application of this method to the sparse truss topology design problem are discussed in essential details. The second method can be used for solution of convex and quasiconvex optimization problems and it is optimal in terms of complexity bounds. The conclusion part of the survey contains the important references that characterize recent developments of nonsmooth convex optimization.

2.1 Introduction

We consider a finite-dimensional nondifferentiable *convex optimization problem* (COP)

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = f_\star = f(\boldsymbol{x}^\star) \\ \text{subject to} & \boldsymbol{x} \in E,\ \boldsymbol{x}^\star \in X_\star, \end{cases} \tag{COP}$$

where E denotes a finite-dimensional space of primal variables and $f : E \to \mathbb{R}$ is a finite convex function, not necessarily differentiable.

One of the main goals in the development of *nonsmooth optimization* (NSO) is to cope with high dimensional problems by decomposition, duality or Lagrangian relaxation which greatly reduces the number of variables at the cost of worsening differentiability of objective or constraints. Small or medium dimensionality of resulting nonsmooth problems allows to use bundle-type algorithms to achieve higher rates of convergence and obtain higher accuracy, which of course comes at the cost of additional memory requirements, typically of the order of n^2, where n is the number of variables of the nonsmooth problem.

However, with the rapid development of more and more sophisticated models in industry, economy, finance, et all such memory requirements are becoming too hard to satisfy. It raised the interest in subgradient-based low-memory algorithms and later developments in this area significantly improved over their early variants still preserving $O(n)$ memory requirements.

These algorithms are considered as *blackbox* methods which use subgradients oracles only to provide information about objective function f in (COP). For a given point $\boldsymbol{x}$ the subgradient oracle returns value of objective function at that point $f(\boldsymbol{x})$ and subgradient $\boldsymbol{g} \in \partial f(\boldsymbol{x})$. We do not make any assumption about the choice of $\boldsymbol{g}$ from $\partial f(\boldsymbol{x})$. As we are interested in computational issues related to solving (COP) mainly we assume that this problem is solvable and has nonempty and bounded set of solutions $X_\star$.

This problem enjoys a considerable popularity due to its important theoretical properties and numerous applications in large-scale structured optimization, discrete optimization, exact penalization in constrained optimization, and others. NSO theory made it possible to solve in an efficient way classical discrete min-max problems [21], L_1-approximation and others, at the same time opening new approaches in bilevel, monotropic programming, two-stage stochastic optimization, to name a few.

To provide historical perspective this survey starts with the original result of Shor (see [71] for the overview and references to earliest works) which opened this field with the application to the classical n suppliers—m customers transportation problem. By using Lagrange relaxation it is possible to switch from nm flow variables to $n + m$ dual variables but with nondifferentiable objective and simple subgradient oracle. The pioneering work by Shor gave the first subgradient algorithm for approximate solution of (COP) but without estimates of the rate of convergence or, equivalently, complexity bounds.

The theoretical complexity bounds for smooth and nonsmooth convex and quasiconvex optimization problems are briefly exposed in what follows to introduce the relevant fundamentals of NSO. Special attention in this section is given to the adaptive step size policy which aims to attain lowest complexity bounds. Nondifferentiability of objective function in convex optimization greatly slows down the rate of convergence in blackbox subgradient optimization compared to the smooth case. However there are different modern techniques that allow to solve certain nonsmooth convex optimization problems faster then dictate theoretical lower complexity bounds. In this work the particular attention is given to Nesterov smoothing technique, applicable to convex functions with special structure. The equivalent problem was considered by Nemirovski and his mirror prox method allowed to attain about the same complexity bound. We also considered Nesterov universal approach, and Legendre (saddle point) representation approach.

New results on universal mirror prox algorithms represent the original parts of the survey. To demonstrate application of nonsmooth convex optimization algorithms to solution of huge-scale extremal problems we consider convex optimization problems with nonsmooth functional constraints and propose two adaptive mirror descent methods. The first method is of primal-dual variety and proved to be optimal in terms of lower oracle bounds for the class of Lipschitz continuous convex objectives and constraints. The advantages of application of this method to the sparse truss topology design problem are discussed in essential details. The second method can be used for solution of convex and quasiconvex optimization problems and is optimal in terms of complexity bounds. The conclusion part of the survey contains the important references that characterize recent developments of nonsmooth convex optimization.

2.2 Example Application: Transportation Problem and First Subgradient Algorithm

From utilitarian point of view the development of nonsmooth (convex) optimization started with the classical transportation problem

$$\begin{cases} \text{minimize} & \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} c_{ij}x_{ij} \\ \text{subject to} & \sum\limits_{i=1}^{m} x_{ij} = a_j,\ j = 1,2,\ldots,n, \\ & \sum\limits_{j=1}^{n} x_{ij} = b_i,\ i = 1,2,\ldots,m, \\ & x_{ij} \geq 0,\ i = 1,2,\ldots,m,\ j = 1,2,\ldots,n, \end{cases} \tag{2.1}$$

which is widely used in many applications.

By dualizing this problem with respect to balancing constrains we can convert (2.1) into the dual problem of the kind

$$\max\ \Phi(\boldsymbol{u}, \boldsymbol{v}), \tag{2.2}$$

where $\boldsymbol{u} = (u_i, i = 1,2,\ldots,m)$; $\boldsymbol{v} = (v_j, j = 1,2,\ldots,n)$ are dual variables associated with the balancing constraints in (2.1) and $\Phi(\boldsymbol{u}, \boldsymbol{v})$ is the dual function defined as

$$\Phi(\boldsymbol{u}, \boldsymbol{v}) = \inf_{\boldsymbol{x}\geq 0} L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) \tag{2.3}$$

and $L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ is the Lagrange function of the problem:

$$L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} u_j\Big(\sum_{i=1}^{m} x_{ij} - a_j\Big) + \sum_{i=1}^{m} v_i\Big(\sum_{j=1}^{n} x_{ij} - b_i\Big).$$

By rearranging terms in this expression we can obtain the following expression for the dual function

$$\begin{aligned} \Phi(\boldsymbol{u}, \boldsymbol{v}) &= -\sum_{j=1}^{n} u_j a_j - \sum_{i=1}^{m} v_i b_i + \sum_{i=1}^{m}\sum_{j=1}^{n} \inf_{\boldsymbol{x}\geq 0} x_{ij}\{c_{ij} + u_j + v_i\} \\ &= -\sum_{j=1}^{n} u_j a_j - \sum_{i=1}^{m} v_i b_i - \mathrm{Ind}_D(\boldsymbol{u}, \boldsymbol{v}), \end{aligned} \tag{2.4}$$

where

$$\mathrm{Ind}_D(\boldsymbol{u},\boldsymbol{v}) = \begin{cases} 0, & \text{when } c_{ij}+u_i+v_j \geq 0, \\ \infty, & \text{otherwise} \end{cases}$$

is the indicator function of the set $D = \{\boldsymbol{u},\boldsymbol{v} : c_{ij}+u_j+v_i \geq 0, i = 1,2,\ldots,m;\ j = 1,2,\ldots,n\}$ which is the feasible set of the dual problem.

Of course, by explicitely writing feasibility constraints for (2.2) we obtain the linear dual transportation problem with a fewer variables but with much higher number of constraints. This is bad news for textbook simplex method so many specialized algorithms were developed, one of them was simple-minded method of generalized gradient which started the development of NSO.

This method relies on subgradient of concave function $\Phi(\boldsymbol{u},\boldsymbol{v})$ which we can transform into convex just by changing signs and replacing inf with sup

$$\begin{aligned}\Phi(\boldsymbol{u},\boldsymbol{v}) &= \sum_{j=1}^{n} u_j a_j + \sum_{i=1}^{m} v_i b_i + \sum_{i=1}^{m}\sum_{j=1}^{n} \sup_{\boldsymbol{x}\geq 0} x_{ij}\{c_{ij}+u_j+v_i\} \\ &= \sum_{j=1}^{n} u_j a_j + \sum_{i=1}^{m} v_i b_i + \mathrm{Ind}_D(\boldsymbol{u},\boldsymbol{v}),\end{aligned}$$

and ask for its *minimization*.

According to convex analysis [64] the subdifferential $\partial_c\Phi(\boldsymbol{u},\boldsymbol{v})$ exists for any $\boldsymbol{v},\boldsymbol{u} \in \operatorname{int}\operatorname{dom}(\mathrm{Ind}_D)$, and in this case just equals to the (constant) vector $\boldsymbol{g}_L = (\boldsymbol{g}_{\boldsymbol{u}},\boldsymbol{g}_{\boldsymbol{v}}) = (\boldsymbol{a},\boldsymbol{b})$ of a linear objective in the interior of D. The situation becomes more complicated when $\boldsymbol{u},\boldsymbol{v}$ happens to be at the boundary of D, the subdifferential set ceases to be a singleton and becomes even unbounded, roughly speaking certain linear manifolds are added to $\boldsymbol{g}_L$ but we will not go into details here. The difficulty is that if we mimic gradient method of the kind

$$\begin{aligned}\boldsymbol{u}^{k+1} &= \boldsymbol{u}^k - \lambda \boldsymbol{g}_L^u = \boldsymbol{u}^k - \lambda \boldsymbol{a}, \\ \boldsymbol{v}^{k+1} &= \boldsymbol{v}^k - \lambda \boldsymbol{g}_L^v = \boldsymbol{v}^k - \lambda \boldsymbol{b}, \qquad k = 0,1,\ldots\end{aligned}$$

with a certain step size $\lambda > 0$, we inevitably violate the dual feasibility constraints as $\boldsymbol{a},\boldsymbol{b} > 0$. Than the dual function (2.4) becomes undefined and correspondently the subdifferential set becomes undefined as well.

There are at least two simple ways to overcome this difficulty. One is to incorporate in the gradient method certain operations which restore feasibility and the appropriate candidate for it is the orthogonal projection operation where one can make use of the special structure of constraints and sparsity. However it will still require computing projection operator of the kind $B^T(BB^T)^{-1}B$ for basis matrices B with rather uncertain number of iterations and of matrices of the size around $(n+m)\times(n+m)$. Neither computers speed nor memory sizes at that time

where not up to demands to solve problems of $n + m \approx 10^4$ which was required by GOSPLAN![1]

The second ingenious way was to take into account that if $\sum_{j=1}^{n} a_j = \sum_{i=1}^{m} b_i = V$, which is required anyway for solvability of the transportation problem in a closed form. The flow variables are uniformly bounded by V and the dual function can be redefined as

$$\begin{aligned}\Phi_V(\boldsymbol{u}, \boldsymbol{v}) &= \sum_{j=1}^{n} u_j a_j + \sum_{i=1}^{m} v_i b_i - \sum_{i=1}^{m}\sum_{j=1}^{n} \max_{0 \le \boldsymbol{x} \le V} x_{ij}\{c_{ij} + u_j + v_i\} \\ &= \sum_{j=1}^{n} u_j a_j + \sum_{i=1}^{m} v_i b_i + P_V(\boldsymbol{u}, \boldsymbol{v})\end{aligned}$$

where the penalty function $P_V(\boldsymbol{u}, \boldsymbol{v})$ is easily computed by component-wise maximization:

$$\begin{aligned}P_V(\boldsymbol{u}, \boldsymbol{v}) &= \sum_{i=1}^{m}\sum_{j=1}^{n} \max_{x_{ij} \in [0, V]} x_{ij}\{c_{ij} + u_j + v_i\} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} V\{c_{ij} + u_j + v_i\}_+\end{aligned}$$

where $\{\cdot\}_+ = \max\{0, \cdot\}$. Than the dual objective function becomes finite, the optimization problem—unconstrained and we can use simple subgradient method with very low requirements for memory and computations.

Actually even tighter bounds $x_{ij} \le \min(a_i, b_j)$ can be imposed on the flow variables which may be advantageous for computational reasons.

In both cases there is a fundamental problem of recovering optimal $n \times m$ primal solution from $n + m$ dual. This problem was studied by many authors and recent advances in this area can be studied from the excellent paper by Nedić and Ozdoglar [44]. Theoretically speaking, nonzero positive values of $c_{ij} + u_j^\star + v_i^\star$, where $\boldsymbol{u}^\star$, $\boldsymbol{v}^\star$ are the *exact* optimal solutions of the dual problem (2.2) signal that the corresponding optimal primal flow $x_{ij}^\star$ is equal to zero. Hopefully after excluding these variables we obtain nondegenerate basis and can compute the remaining variables by simple and efficient linear algebra, especially taking into account the unimodularity of basis.

However the theoretical gap between zeros and nonzeros is exponentially small even for modest length integer data therefore we need an accuracy unattainable by modern 64–128 bits hardware. Also the real-life computations are always

[1]The highest planning authority in USSR responsible for resource allocation and production planning.

accompanied by numerical noise and we face the hard choice in fact guessing which dual constraints are active and which are not.

To connect the transportation problem with NSO notice that the penalty function $P_V(\boldsymbol{u}, \boldsymbol{v})$ is finite with the subdifferential $\partial_c P_V(\boldsymbol{u}, \boldsymbol{v})$ which can be represented as a set of $n \times m$ matrices

$$\boldsymbol{g}_{ij} = \begin{cases} V, & \text{if } c_{ij} + u_j + v_i > 0, \\ \mathbf{0}, & \text{if } c_{ij} + u_j + v_i < 0, \\ \text{cone}(\mathbf{0}, V), & \text{if } c_{ij} + u_j + v_i = 0, \end{cases}$$

so the subdifferential set is a convex hull of up to $2^{(n+m)}$ extreme points—enormous number even for a modest size transportation problem. Nevertheless it is easy to get at least single member of subdifferential and consider the simplest version of subgradient method:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \lambda \bar{\boldsymbol{g}}^k, \qquad k = 0, 1, \ldots$$

where $\boldsymbol{x}^0$ is a given starting point, $\lambda > 0$—fixed step size and $\bar{\boldsymbol{g}}^k = \boldsymbol{g}^k / \|\boldsymbol{g}^k\|$ is a normalized subgradient $\boldsymbol{g}^k \in \partial f(\boldsymbol{x}^k)$. Of course we assume that $\boldsymbol{g}^k \neq \mathbf{0}$, otherwise, $\boldsymbol{x}^k$ is already a solution.

Of course, there is no hope of classical convergence result such that $\boldsymbol{x}^k \to \boldsymbol{x}^\star \in X_\star$, but the remarkable theorem of Shor [68] establishes that this simplest algorithm determines at least an approximate solution. As a major step in the development of different algorithmic ideas we can start with the subgradient algorithm due to Shor (see [71] for the overview and references to earliest works). Of course, there is no hope of classical convergence result such that $\boldsymbol{x}^k \to \boldsymbol{x}^\star \in X_\star$, but the remarkable theorem of Shor [68] establishes that this very simple algorithm provides an approximate solution of (COP) at least theoretically.

Theorem 2.1 *Let f is a finite convex function with a subdifferential ∂f and the sequence $\{\boldsymbol{x}^k\}$ is obtained by the recursive rule*

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \lambda \boldsymbol{g}_\nu^k, \qquad k = 0, 1, \ldots$$

with $\lambda > 0$ and $\boldsymbol{g}_\nu^k = \boldsymbol{g}^k / \|\boldsymbol{g}^k\|$, $\boldsymbol{g}^k \in \partial f(\boldsymbol{x}^k)$, $\boldsymbol{g}^k \neq \mathbf{0}$ is a normalized subgradient at the point $\boldsymbol{x}^k$. Then for any $\epsilon > 0$ there is an infinite set $Z_\epsilon \subset Z$ such that for any $k \in Z_\epsilon$

$$f(\tilde{\boldsymbol{x}}^k) = f(\boldsymbol{x}^k) \quad \text{and} \quad \text{dist}(\tilde{\boldsymbol{x}}^k, X_\star) \leq \lambda(1 + \epsilon)/2.$$

The statement of the theorem is illustrated on Fig. 2.1 together with the idea of the proof. The detailed proof of the theorem goes like following: Let $\boldsymbol{x}^\star \in X_\star$ and estimate

$$\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^\star\|^2 = \|\boldsymbol{x}^k - \boldsymbol{x}^\star - \lambda \boldsymbol{g}_\nu^k\|^2 = \|\boldsymbol{x}^k - \boldsymbol{x}^\star\|^2 + \lambda^2 - 2\lambda \bar{\boldsymbol{g}}^k (\boldsymbol{x}^k - \boldsymbol{x}^\star).$$

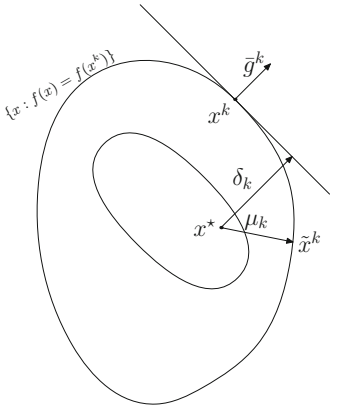

Fig. 2.1 The statement and the idea of the proof of Shor theorem

The last term in fact equals

$$\min_{z \in H_k} \|\boldsymbol{x}^\star - z\|^2 = \|\boldsymbol{x}^\star - \boldsymbol{z}^k\|^2 = \delta_k,$$

where $H_k = \{z : z\boldsymbol{g}_\nu^k = \boldsymbol{x}^k\boldsymbol{g}_\nu^k\}$ is a hyperplane, orthogonal to $\boldsymbol{g}_\nu^k$ and passing through the point $\boldsymbol{x}^k$, so

$$\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^\star\|^2 = \|\boldsymbol{x}^k - \boldsymbol{x}^\star\|^2 + \lambda^2 - 2\lambda\delta_k, \quad k = 0, 1, 2, \ldots$$

If $\lambda^2 - 2\lambda\delta_k \leq -\lambda^2\epsilon$ for any $k \in Z$ then

$$\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^\star\|^2 \leq \|\boldsymbol{x}^k - \boldsymbol{x}^\star\|^2 - \lambda^2\epsilon, \quad k = 0, 1, 2, \ldots,$$

therefore

$$0 \leq \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^\star\|^2 \leq \|\boldsymbol{x}^0 - \boldsymbol{x}^\star\|^2 \leq -k\lambda^2\epsilon \to -\infty \tag{2.5}$$

when $k \to \infty$. This contradiction proves that there is k_0 such that

$$\lambda^2 - 2\lambda\delta_{k_0} > -\lambda^2\epsilon \quad \text{or} \quad \delta_{k_0} < \lambda(1+\epsilon)/2.$$

To complete the proof notice that by convexity $f(\boldsymbol{z}^{k_0}) \geq f(\boldsymbol{x}^{k_0})$ and therefore

$$\begin{aligned}\min_{z: f(z) = f(\boldsymbol{x}^{k_0})} \|\boldsymbol{x}^\star - z\|^2 &= \|\boldsymbol{x}^\star - \bar{\boldsymbol{z}}^{k_0}\|^2 \\ &= \min_{z: f(z) \geq f(\boldsymbol{x}^{k_0})} \|\boldsymbol{x}^\star - z\|^2 \\ &\leq \|\boldsymbol{x}^\star - \boldsymbol{z}^{k_0}\|^2 = \delta_{k_0}.\end{aligned}$$

By setting $\tilde{\boldsymbol{x}}^0 = \boldsymbol{z}^{k_0}$ we obtain $\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}^0\|^2 < \lambda(1+\epsilon)/2$.

By replacing $\boldsymbol{x}^0$ in (2.5) by $\tilde{\boldsymbol{x}}^0$ and repeating the reasoning above we obtain $\tilde{\boldsymbol{x}}^1$ such that $\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}^1\|^2 < \lambda(1+\epsilon)/2$, then in the same manner $\tilde{\boldsymbol{x}}^2, \tilde{\boldsymbol{x}}^3$ and so on with $\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}^k\|^2 < \lambda(1+\epsilon)/2, k = 2, 3, \ldots$ which complete the proof. □

2.3 Complexity Results for Convex Optimization

At this section we describe the complexity results for nonsmooth convex optimization problems. Most of the results mentioned below can be found in books [9, 13, 48, 58, 63]. We start with the "**small dimensional problems**", when

$$N \geq n = \dim \boldsymbol{x},$$

where N is a number of oracle calls (number of subgradient calculations or/and calculations of separation hyperplane to some simple set at a given point).

Let's consider the convex optimization problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in Q, \end{cases} \tag{2.6}$$

where Q is a compact and simple set (it's significant here!). Based on at least N subgradient calculations (in general, oracle calls) we would like to find such a point $\boldsymbol{x}^N$ that

$$f\left(\boldsymbol{x}^N\right) - f_* \leq \varepsilon,$$

where $f_* = f(\boldsymbol{x}_*)$ is an optimal value of function in (2.6), $\boldsymbol{x}_*$ is the solution of (2.6). The lower and the upper bounds for the oracle complexity is (up to a multiplier, which has logarithmic dependence on some characteristic of the set Q)

$$N \sim n \ln\left(\Delta f / \varepsilon\right),$$

where

$$\Delta f = \sup_{\boldsymbol{x}, \boldsymbol{y} \in Q} \{f(\boldsymbol{y}) - f(\boldsymbol{x})\}.$$

The center of gravity method [43, 60] converges according to this estimate. The center of gravity method for $n = 1$ is a simple binary search method [12], but for $n > 1$ this method is hard to implement. The complexity of iteration is too high, because we require the center-of-gravity oracle [13]. Well known ellipsoid

method [48, 69] requires[2] $N = \tilde{O}\left(n^2 \ln\left(\Delta f / \varepsilon\right)\right)$ oracle calls and $O\left(n^2\right)$ iteration complexity. In [13, 76] a special version of cutting plane method was proposed. This method (Vayda's method) requires $N = \tilde{O}\left(n \ln\left(\Delta f / \varepsilon\right)\right)$ oracle calls and has iteration complexity $\tilde{\mathrm{O}}\left(n^{2.37}\right)$. In the work [42] there proposed a method with $N = \tilde{O}\left(n \ln\left(\Delta f / \varepsilon\right)\right)$ oracle calls and iteration complexity $\tilde{O}\left(n^2\right)$. Unfortunately, for the moment it's not obvious that this method is very practical one due to the large log-factors in $\tilde{O}$.

Based on ellipsoid method in the late 70th Khachiyan showed [38] that LP is in P in byte complexity. Let us shortly explain the idea. The main question is whether $A\boldsymbol{x} \leq \boldsymbol{b}$ is solvable or not, where $n = \dim \boldsymbol{x}$, $m = \dim \boldsymbol{b}$ and all elements of A and $\boldsymbol{b}$ are integers. We would like also to find one of the exact solutions $\boldsymbol{x}_*$. This problem up to a logarithmic factor in complexity is equivalent to the problem to find the exact solution of LP problem

$$\begin{cases} \text{minimize} & \langle \boldsymbol{c}, \boldsymbol{x} \rangle \\ \text{subject to} & A\boldsymbol{x} \leq \boldsymbol{b} \end{cases}$$

with integer A, $\boldsymbol{b}$ and $\boldsymbol{c}$. We consider only inequality constraints as it is known that to find the exact solution of $A\boldsymbol{x} = \boldsymbol{b}$ one can use polynomial Gauss elimination algorithm with $O\left(n^3\right)$ arithmetic operations (a.o.) complexity.

Let us introduce

$$\Lambda = \sum_{i,j=1,1}^{m,n} \log_2 |a_{ij}| + \sum_{i=1}^{m} \log_2 |b_i| + \log_2 (mn) + 1.$$

If $A\boldsymbol{x} \leq \boldsymbol{b}$ is compatible, then there exists such $\boldsymbol{x}_*$ that $\|\boldsymbol{x}_*\|_\infty \leq 2^\Lambda$, $A\boldsymbol{x}_* \leq \boldsymbol{b}$ otherwise

$$\min_{\boldsymbol{x}} \|(A\boldsymbol{x} - \boldsymbol{b})_+\|_\infty \geq 2^{-(\Lambda - 1)}.$$

Thus, the question of compatibility of $A\boldsymbol{x} \leq \boldsymbol{b}$ is equivalent to the problem of finding minimum of the following nonsmooth convex optimization problem

$$\begin{cases} \text{minimize} & \|(A\boldsymbol{x} - \boldsymbol{b})_+\|_\infty \\ \text{subject to} & \|\boldsymbol{x}_*\|_\infty \leq 2^\Lambda. \end{cases}$$

The approach of [38] is to apply ellipsoid method for this problem with $\varepsilon = 2^{-\Lambda}$. From the complexity of this method, it follows that in $O(n\Lambda)$-bit arithmetic with $\tilde{O}\left(mn + n^2\right)$ cost of PC memory one can find $\boldsymbol{x}_*$ (if it exists) in $\tilde{O}\left(n^3\left(n^2 + m\right)\Lambda\right)$ a.o.

[2] Here and below for all (large) n: $\tilde{O}(g(n)) \leq C \cdot (\ln n)^r g(n)$ with some constants $C > 0$ and $r \geq 0$. Typically, $r = 1$. If $r = 0$, then $\tilde{O}(\cdot) = O(\cdot)$.

Table 2.1 Optimal estimates for the number of oracle calls

$N \leq n$	$\lvert f(\mathbf{y}) - f(\mathbf{x})\rvert \leq M\lVert \mathbf{y}-\mathbf{x}\rVert$	$\lVert \nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\rVert_* \leq L\lVert \mathbf{y}-\mathbf{x}\rVert$
$f(\mathbf{x})$ convex	$O\left(\frac{M^2R^2}{\varepsilon^2}\right)$	$O\left(\sqrt{\frac{LR^2}{\varepsilon}}\right)$
$f(\mathbf{x})$ is $\mu-$ strongly convex in $\lVert\cdot\rVert$-norm	$\tilde{O}\left(\frac{M^2}{\mu\varepsilon}\right)$	$\tilde{O}\left(\sqrt{\frac{L}{\mu}}\left\lceil \ln\left(\frac{\mu R^2}{\varepsilon}\right)\right\rceil\right)$ $(\forall\, N)$

Note, that in the ideal arithmetic with real numbers it is still an open question [10] whether it is possible to find the exact solution of LP an problem (with the data given by real numbers) in polynomial time in the ideal arithmetic ($\pi \cdot e$—costs $O(1)$).

Now let us consider "**large dimensional problems**"

$$N \leq n = \dim \boldsymbol{x}.$$

Table 2.1 describes (for more details see [9, 13, 58]) optimal estimates for the number of oracle calls for the convex optimization problem (2.6) in the case when $N \leq n$. Now Q is not necessarily compact set.

Here R is a "distance" (up to a $\ln n$-factor) between starting point and the nearest solution

$$R = \tilde{O}\left(\left\|\boldsymbol{x}^0 - \boldsymbol{x}_*\right\|\right).$$

Let's describe optimal method in the most simple case: $Q = \mathbb{R}^n$, $\|\cdot\| = \|\cdot\|_2$ [52, 63]. The main iterative process is (for simplicity we'll denote arbitrary element of $\partial f(\boldsymbol{x})$ as $\nabla f(\boldsymbol{x})$)

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - h\nabla f\left(\boldsymbol{x}^k\right). \tag{2.7}$$

Assume that under $\boldsymbol{x} \in \bar{B}\left(\boldsymbol{x}_*; \sqrt{2}R\right)$

$$\|\nabla f(\boldsymbol{x})\| \leq M, \tag{2.8}$$

where $R = \left\|\boldsymbol{x}^0 - \boldsymbol{x}_*\right\|$.

Hence, from (2.7), (2.8) we have

$$\begin{aligned}\left\|\boldsymbol{x} - \boldsymbol{x}^{k+1}\right\|^2 &= \left\|\boldsymbol{x} - \boldsymbol{x}^k + h\nabla f\left(\boldsymbol{x}^k\right)\right\|^2 \\ &= \left\|\boldsymbol{x} - \boldsymbol{x}^k\right\|^2 + 2h\left\langle \nabla f\left(\boldsymbol{x}^k\right), \boldsymbol{x} - \boldsymbol{x}^k\right\rangle + h^2\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\|^2 \\ &\leq \left\|\boldsymbol{x} - \boldsymbol{x}^k\right\|^2 + 2h\left\langle \nabla f\left(\boldsymbol{x}^k\right), \boldsymbol{x} - \boldsymbol{x}^k\right\rangle + h^2M^2. \end{aligned} \tag{2.9}$$

Here we choose $\boldsymbol{x} = \boldsymbol{x}_*$ (if $\boldsymbol{x}_*$ isn't unique, we choose the nearest $\boldsymbol{x}_*$ to $\boldsymbol{x}^0$)

$$
\begin{aligned}
f\left(\frac{1}{N}\sum_{k=0}^{N-1}\boldsymbol{x}^k\right) - f_* &\leq \frac{1}{N}\sum_{k=0}^{N-1} f\left(\boldsymbol{x}^k\right) - f\left(\boldsymbol{x}_*\right) \\
&\leq \frac{1}{N}\sum_{k=0}^{N-1}\left\langle \nabla f\left(\boldsymbol{x}^k\right), \boldsymbol{x}^k - \boldsymbol{x}_*\right\rangle \\
&\leq \frac{1}{2hN}\sum_{k=0}^{N-1}\left\{\left\|\boldsymbol{x}_* - \boldsymbol{x}^k\right\|^2 - \left\|\boldsymbol{x}_* - \boldsymbol{x}^{k+1}\right\|^2\right\} + \frac{hM^2}{2} \\
&= \frac{1}{2hN}\left(\left\|\boldsymbol{x}_* - \boldsymbol{x}^0\right\|^2 - \left\|\boldsymbol{x}_* - \boldsymbol{x}^N\right\|^2\right) + \frac{hM^2}{2}.
\end{aligned}
$$

If

$$
h = \frac{R}{M\sqrt{N}}, \quad \bar{\boldsymbol{x}}^N = \frac{1}{N}\sum_{k=0}^{N-1}\boldsymbol{x}^k, \tag{2.10}
$$

then

$$
f\left(\bar{\boldsymbol{x}}^N\right) - f_* \leq \frac{MR}{\sqrt{N}}. \tag{2.11}
$$

Note that the precise lower bound for fixed steps first-order methods for the class of convex optimization problems with (2.8) [23]

$$
f\left(\boldsymbol{x}^N\right) - f_* \geq \frac{MR}{\sqrt{N+1}}.
$$

Inequality (2.11) means that (see also Table 2.1)

$$
N = \frac{M^2R^2}{\varepsilon^2}, \quad h = \frac{\varepsilon}{M^2}.
$$

So, one can mention that if we will use in (2.7)

$$
\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - h_k\nabla f\left(\boldsymbol{x}^k\right), \quad h_k = \frac{\varepsilon}{\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\|^2}, \tag{2.12}
$$

the result (2.11) holds with [52]

$$
\bar{\boldsymbol{x}}^N = \frac{1}{\sum_{k=0}^{N-1} h_k}\sum_{k=0}^{N-1} h_k\boldsymbol{x}^k.
$$

If we put in (2.12),

$$h_k = \frac{R}{\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\| \sqrt{N}},$$

like in (2.10), the result similar to (2.11) also holds

$$\min_{k=0,\ldots,N-1} f\left(\boldsymbol{x}^k\right) - f_* \leq \frac{MR}{\sqrt{N}}$$

not only for the convex functions, but also for quasiconvex functions [50, 62]:

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \max\{f(\boldsymbol{x}), f(\boldsymbol{y})\} \quad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in Q,\ \alpha \in [0,1].$$

Note that

$$0 \leq \frac{1}{2hk}\left(\left\|\boldsymbol{x}_* - \boldsymbol{x}^0\right\|^2 - \left\|\boldsymbol{x}_* - \boldsymbol{x}^k\right\|^2\right) + \frac{hM^2}{2}.$$

Hence, for all $k = 0, \ldots, N$,

$$\left\|\boldsymbol{x}_* - \boldsymbol{x}^k\right\|^2 \leq \left\|\boldsymbol{x}_* - \boldsymbol{x}^0\right\|^2 + h^2M^2k \leq 2\left\|\boldsymbol{x}_* - \boldsymbol{x}^0\right\|^2,$$

therefore

$$\left\|\boldsymbol{x}^k - \boldsymbol{x}_*\right\| \leq \sqrt{2}\left\|\boldsymbol{x}^0 - \boldsymbol{x}_*\right\|, \quad k = 0, \ldots, N. \tag{2.13}$$

The inequality (2.13) justifies that we need the assumption (2.8) holds only with $\boldsymbol{x} \in \bar{B}\left(\boldsymbol{x}_*; \sqrt{2}R\right)$.

For the general (constrained) case (2.6) we introduce a norm $\|\cdot\|$ and some prox-function $d(\boldsymbol{x}) \geq 0$, which is continuous and 1-strongly convex with respect to $\|\cdot\|$, i.e. $d(\boldsymbol{y}) - d(\boldsymbol{x}) - \langle d(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle \geq \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2$, for all $\boldsymbol{x}, \boldsymbol{y} \in Q$. We also introduce Bregman's divergence [9]

$$V[\boldsymbol{x}](\boldsymbol{y}) = d(\boldsymbol{y}) - d(\boldsymbol{x}) - \langle \nabla d(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle.$$

We set $R^2 = V[\boldsymbol{x}^0](\boldsymbol{x}_*)$, where $\boldsymbol{x}_*$—is solution of (2.6) (if $\boldsymbol{x}_*$ isn't unique then we assume that $\boldsymbol{x}_*$ is minimized $V[\boldsymbol{x}^0](\boldsymbol{x}_*)$. The natural generalization of iteration process (2.7) is the mirror descent algorithm [9, 46] which iterates as

$$\boldsymbol{x}^{k+1} = \operatorname{Mirr}_{\boldsymbol{x}^k}\left(h\nabla f\left(\boldsymbol{x}^k\right)\right),$$

where

$$\mathrm{Mirr}_{x^k}(\mathrm{v}) = \underset{x \in Q}{\mathrm{argmin}} \left\{ \left\langle v, x - x^k \right\rangle + V[x^k](x) \right\}.$$

For this iteration process instead of (2.9) we have

$$2V[x^{k+1}](x) \leq 2V[x^k](x) + 2h \left\langle \nabla f\left(x^k\right), x - x^k \right\rangle + h^2 M^2,$$

where $\|\nabla f(x)\|_* \leq M$ for all $x : V[x](x_*) \leq 2V[x^0](x_*) = 2R^2$, see also Sect. 2.5.

Analogues of formulas (2.10), (2.11), and (2.13) are also valid

$$f\left(\bar{x}^N\right) - f_* \leq \frac{\sqrt{2} M R}{\sqrt{N}},$$

where

$$\bar{x}^N = \frac{1}{N} \sum_{k=0}^{N-1} x^k, \quad h = \frac{\varepsilon}{M^2}$$

and

$$\left\| x^k - x_* \right\| \leq 2R, \quad k = 0, \ldots, N.$$

In [9] authors discuss how to choose $d(x)$ for different simple convex sets Q. One of these examples (unit simplex) will be considered below. Note, that in all these examples one can guarantee that [9]:

$$R \leq C\sqrt{\ln n} \cdot \left\| x_* - x^0 \right\|.$$

Note, that if $Q = \mathbb{R}^n, \|\cdot\| = \|\cdot\|$ then $d(x) = \frac{1}{2}\|x\|^2$, $V[x](y) = \frac{1}{2}\|y - x\|^2$,

$$\begin{aligned} x^{k+1} &= \mathrm{Mirr}_{x^k}\left(h \nabla f\left(x^k\right)\right) \\ &= \underset{x \in \mathbb{R}^n}{\mathrm{argmin}} \left\{ h \left\langle \nabla f\left(x^k\right), x - x^k \right\rangle + \frac{1}{2}\|x - x^k\|^2 \right\} \\ &= x^k - h \nabla f\left(x^k\right), \end{aligned}$$

that corresponds to the standard gradient-type iteration process (2.7).

Example 2.1 (Unit Simplex) We have

$$Q = S_n(1) = \left\{ \boldsymbol{x} \in \mathbb{R}^n_+ : \sum_{i=1}^n \boldsymbol{x}_i = 1 \right\}, \quad \|\nabla f(\boldsymbol{x})\|_\infty \le M_\infty, \quad \boldsymbol{x} \in Q,$$

$$\|\cdot\| = \|\cdot\|_1, \quad d(\boldsymbol{x}) = \ln n + \sum_{i=1}^n \boldsymbol{x}_i \ln \boldsymbol{x}_i, \quad h = M_\infty^{-1} \sqrt{2 \ln n / N},$$

$$\boldsymbol{x}_i^0 = 1/n, \quad i = 1, \ldots, n.$$

For $k = 0, \ldots, N-1$, $i = 1, \ldots, n$

$$\boldsymbol{x}_i^{k+1} = \frac{\exp\left(-h \sum\limits_{r=1}^k \nabla_i f(\boldsymbol{x}^r)\right)}{\sum\limits_{l=1}^n \exp\left(-h \sum\limits_{r=1}^k \nabla_l f(\boldsymbol{x}^r)\right)} = \frac{\boldsymbol{x}_i^k \exp\left(-h \nabla_i f\left(\boldsymbol{x}^k\right)\right)}{\sum\limits_{l=1}^n \boldsymbol{x}_l^k \exp\left(-h \nabla_l f\left(\boldsymbol{x}^k\right)\right)}.$$

The main result here is

$$f\left(\bar{\boldsymbol{x}}^N\right) - f_* \le M_\infty \sqrt{\frac{2 \ln n}{N}}, \quad \bar{\boldsymbol{x}}^N = \frac{1}{N} \sum_{k=0}^{N-1} \boldsymbol{x}^k.$$

Note, that if we use $\|\cdot\|_2$-norm and $d(\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^0\right\|^2$ here, we will have higher iteration complexity (2-norm projections on unit simplex) and

$$f\left(\bar{\boldsymbol{x}}^N\right) - f_* \le \frac{M_2}{\sqrt{N}}, \quad \|\nabla f(\boldsymbol{x})\| \le M_2, \quad \boldsymbol{x} \in Q.$$

Since typically $M_2 = \mathrm{O}\left(\sqrt{n} M_\infty\right)$, it is worth to use $\|\cdot\|_1$-norm.

Assume now that $f(\boldsymbol{x})$ in (2.6) is additionally μ-strongly convex in $\|\cdot\|_2$ norm:

$$f(\boldsymbol{y}) \ge f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{y} - \boldsymbol{x}\|^2 \text{ for all } \boldsymbol{x}, \boldsymbol{y} \in Q.$$

Let

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \operatorname{Mirr}_{\boldsymbol{x}^k}\left(h_k \nabla f\left(\boldsymbol{x}^k\right)\right) \\ &= \operatorname*{argmin}_{\boldsymbol{x} \in Q} \left\{ h_k \left\langle \nabla f\left(\boldsymbol{x}^k\right), \boldsymbol{x} - \boldsymbol{x}^k \right\rangle + \frac{1}{2} \left\|\boldsymbol{x} - \boldsymbol{x}^k\right\|^2 \right\}, \end{aligned}$$

where

$$h_k = \frac{2}{\mu \cdot (k+1)}, \quad d(\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^0\right\|^2, \quad \|\nabla f(\boldsymbol{x})\| \leq M, \quad \boldsymbol{x} \in Q.$$

Then [66]

$$f\left(\sum_{k=1}^{N} \frac{2k}{k(k+1)} \boldsymbol{x}^k\right) - f_* \leq \frac{2M^2}{\mu \cdot (k+1)}.$$

Hence (see also Table 2.1),

$$N \simeq \frac{2M^2}{\mu\varepsilon}.$$

This bound is also unimprovable up to a constant factor [48, 58].

2.4 Looking into the Blackbox

In this section we consider how the special structure of objective function can be used to solve NSO problems with the convergence rate $O\left(\frac{1}{k}\right)$, which is faster than the lower bound $O\left(\frac{1}{\sqrt{k}}\right)$ for general class of nonsmooth convex problems [48]. Nevertheless, there is no contradiction as additional structure is used and we are looking inside the blackbox.

2.4.1 Nesterov Smoothing

In this subsection, following [51], we consider the problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = h(\boldsymbol{x}) + \max\limits_{\boldsymbol{u} \in Q_2 \subset E_2} \{\langle A\boldsymbol{x}, \boldsymbol{u}\rangle - \phi(\boldsymbol{u})\} \\ \text{subject to} & \boldsymbol{x} \in Q_1 \subset E_1, \end{cases} \tag{2.14}$$

where $A : E_1 \to E_2^*$ is a linear operator, $\phi(\boldsymbol{u})$ is a continuous convex function on Q_2, Q_1, Q_2 are convex compacts, h is convex function with L_h-Lipschitz continuous gradient.

Let us consider an example of $f(\boldsymbol{x}) = \|A\boldsymbol{x} - \boldsymbol{b}\|_\infty$ with $A \in \mathbb{R}^{m \times n}$. Then

$$f(\boldsymbol{x}) = \max_{\boldsymbol{u} \in \mathbb{R}^m} \{\langle \boldsymbol{u}, A\boldsymbol{x} - \boldsymbol{b}\rangle : \|\boldsymbol{u}\|_1 \leq 1\},$$

$h = 0$, $E_2 = \mathbb{R}^m$, $\phi(\boldsymbol{u}) = \langle \boldsymbol{u}, \boldsymbol{b}\rangle$ and Q_2 is the ball in L_1-norm.

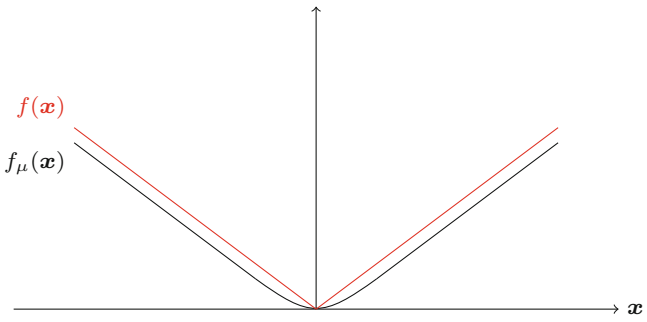

Fig. 2.2 Function $f_\mu(\boldsymbol{x})$ is a smooth approximation to nonsmooth function $f(\boldsymbol{x})$

The main idea of Nesterov is to add regularization inside the definition of f in (2.14). More precisely, a prox-function $d_2(\boldsymbol{u})$ (see definition in Sect. 2.3) is introduced for the set Q_2 and a smoothed counterpart $f_\mu(\boldsymbol{x})$ for f (See Fig. 2.2) is defined as

$$f_\mu(\boldsymbol{x}) = h(\boldsymbol{x}) + \max_{\boldsymbol{u}\in Q_2}\{\langle A\boldsymbol{x}, \boldsymbol{u}\rangle - \phi(\boldsymbol{u}) - \mu d_2(\boldsymbol{u})\}$$

and $\boldsymbol{u}_\mu(\boldsymbol{x})$ is the optimal solution of this maximization problem.

Theorem 2.2 ([51]) *The function $f_\mu(\boldsymbol{x})$ is well defined, convex and continuously differentiable at any $\boldsymbol{x} \in E_1$ with $\nabla f_\mu(\boldsymbol{x}) = \nabla h(\boldsymbol{x}) + A^*\boldsymbol{u}_\mu(\boldsymbol{x})$. Moreover, $\nabla f_\mu(\boldsymbol{x})$ is Lipschitz continuous with constant*

$$L_\mu = L_h + \frac{\|A\|_{1,2}^2}{\mu}.$$

Here the adjoint operator A^* is defined by equality $\langle A\boldsymbol{x}, \boldsymbol{u}\rangle = \langle A^*\boldsymbol{u}, \boldsymbol{x}\rangle$, $\boldsymbol{x} \in E_1, \boldsymbol{u} \in E_2$ and the norm of the operator $\|A\|_{1,2}$ is defined by $\|A\|_{1,2} = \max_{\boldsymbol{x},\boldsymbol{u}}\{\langle A\boldsymbol{x}, \boldsymbol{u}\rangle : \|\boldsymbol{x}\|_{E_1} = 1, \|\boldsymbol{u}\|_{E_2} = 1\}$.

Since Q_2 is bounded, $f_\mu(\boldsymbol{x})$ is a uniform approximation for the function f, namely, for all $\boldsymbol{x} \in Q_1$,

$$f_\mu(\boldsymbol{x}) \le f(\boldsymbol{x}) \le f_\mu(\boldsymbol{x}) + \mu D_2, \tag{2.15}$$

where $D_2 = \max\{d_2(\boldsymbol{u}) : \boldsymbol{u} \in Q_2\}$.

Then, the idea is to choose μ sufficiently small and apply accelerated gradient method to minimize $f_\mu(\boldsymbol{x})$ on Q_1. We use accelerated gradient method from [29, 30] which is different from the original method of [51].

Algorithm 2.1: Accelerated gradient method

Data: Objective $f(\boldsymbol{x})$, feasible set Q, Lipschitz constant L of the $\nabla f(\boldsymbol{x})$, starting point $\boldsymbol{x}^0 \in Q$, prox-setup: $d(\boldsymbol{x})$—1-strongly convex w.r.t. $\|\cdot\|_{E_1}$, $V[z](\boldsymbol{x}) := d(\boldsymbol{x}) - d(z) - \langle \nabla d(z), \boldsymbol{x} - z\rangle$, parameter $\varepsilon_L \in (0, 1/2)$.

Result: The point $\boldsymbol{y}^{k+1}$.

1 Set $k = 0$, $C_0 = \alpha_0 = 0$, $\boldsymbol{y}^0 = \boldsymbol{z}^0 = \boldsymbol{x}^0$.

2 **for** $k = 0, 1, \ldots$ **do**

3 Find α^{k+1} as the largest root of the equation

$$C_{k+1} := C_k + \alpha_{k+1} = L\alpha_{k+1}^2. \tag{2.16}$$

4 Calculate

$$\boldsymbol{x}^{k+1} = \frac{\alpha_{k+1}\boldsymbol{z}^k + C_k\boldsymbol{y}^k}{C_{k+1}}; \tag{2.17}$$

5

$$\boldsymbol{z}^{k+1} = \operatorname*{argmin}_{\boldsymbol{x}\in Q}\{V[\boldsymbol{z}^k](\boldsymbol{x}) + \alpha_{k+1}(f(\boldsymbol{x}^{k+1}) + \langle \nabla f(\boldsymbol{x}^{k+1}), \boldsymbol{x} - \boldsymbol{x}^{k+1}\rangle)\}; \tag{2.18}$$

6

$$\boldsymbol{y}^{k+1} = \frac{\alpha_{k+1}\boldsymbol{z}^{k+1} + C_k\boldsymbol{y}^k}{C_{k+1}}. \tag{2.19}$$

7 Set $k = k + 1$.

Theorem 2.3 ([29, 30]) *Let the sequences $\{\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k, \alpha_k, C_k\}$, $k \geq 0$ be generated by Algorithm 2.1. Then, for all $k \geq 0$, it holds that*

$$f(\boldsymbol{y}^k) - f^* \leq \frac{4LV[\boldsymbol{z}^0](\boldsymbol{x}^\star)}{(k+1)^2}. \tag{2.20}$$

Following the same steps as in the proof of Theorem 3 in [51], we obtain

Theorem 2.4 *Let Algorithm 2.1 be applied to minimize $f_\mu(\boldsymbol{x})$ on Q_1 with $\mu = \frac{2\|A\|_{1,2}}{N+1}\sqrt{\frac{D_1}{D_2}}$, where $D_1 = \max\{d_1(\boldsymbol{x}) : \boldsymbol{x} \in Q_1\}$. Then, after N iterations, we have*

$$0 \leq f(\boldsymbol{y}^N) - f_\star \leq \frac{4\|A\|_{1,2}\sqrt{D_1 D_2}}{N+1} + \frac{4L_h D_1}{(N+1)^2}. \tag{2.21}$$

Proof Applying Theorem 2.3 to f_μ, and using (2.15), we obtain

$$\begin{aligned} 0 &\leq f(\boldsymbol{y}^N) - f_\star \\ &\leq f_\mu(\boldsymbol{y}^N) + \mu D_2 - f_\mu(\boldsymbol{x}_\mu^\star) \\ &\leq \mu D_2 + \frac{4L_\mu D_1}{(N+1)^2} + \frac{4L_h D_1}{(N+1)^2} \\ &= \mu D_2 + \frac{4\|A\|_{1,2}^2 D_1}{\mu(N+1)^2} + \frac{4L_h D_1}{(N+1)^2}. \end{aligned}$$

Substituting the value of μ from the theorem statement, we finish the proof. □

A generalization of the smoothing technique for the case of noncompact sets Q_1, Q_2, which is especially interesting when dealing with problems dual to problems with linear constraints, can be found in [72]. Ubiquitous entropic regularization of optimal transport [18] can be seen as a particular case of the application of smoothing technique, especially in the context of Wasserstein barycenters [19, 31, 75].

2.4.2 Nemirovski Mirror Prox

In his paper [45], Nemirovski considers the problem (2.14) in the following form

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) = h(\boldsymbol{x}) + \max\limits_{\boldsymbol{u} \in Q_2 \subset E_2} \{\langle A\boldsymbol{x}, \boldsymbol{u}\rangle + \langle \boldsymbol{b}, \boldsymbol{u}\rangle\} \\ \text{subject to} & \boldsymbol{x} \in Q_1 \subset E_1 \end{cases} \tag{2.22}$$

pointing to the fact that this problem is as general as (2.14). Indeed, the change of variables $\boldsymbol{u} \leftarrow (\boldsymbol{u}, t)$ and the feasible set $Q_2 \leftarrow \{(\boldsymbol{u}, t) : \min_{\boldsymbol{u}' \in Q_2} \phi(\boldsymbol{u}') \leq t \leq \phi(\boldsymbol{u})\}$ allows to make ϕ linear. His idea is to consider the problem (2.22) directly as a convex-concave saddle point problem and associated weak *variational inequality* (VI).

$$\begin{aligned} &\text{Find} && z^\star = (\boldsymbol{x}^\star, \boldsymbol{u}^\star) \in Q_1 \times Q_2 \\ &\text{such that} && \langle \Phi(z), z^\star - z\rangle \leq 0 \ \text{ for all } z \in Q_1 \times Q_2, \end{aligned} \tag{2.23}$$

where the operator

$$\Phi(z) = \begin{pmatrix} \nabla h(\boldsymbol{x}) + A^* \boldsymbol{u} \\ -A\boldsymbol{x} - \boldsymbol{b} \end{pmatrix} \tag{2.24}$$

is monotone, i.e. $\langle \Phi(z^1) - \Phi(z^2), z^1 - z^2 \rangle \geq 0$, and Lipschitz continuous, i.e. $\|\Phi(z^1) - \Phi(z^2)\|_* \leq L\|z^1 - z^2\|$. With the appropriate choice of norm on $E_1 \times E_2$ and prox-function for $Q_1 \times Q_2$, see [45, Section 5], the Lipschitz constant for Φ can be estimated as $L = 2\|A\|_{1,2}\sqrt{D_1 D_2} + L_h D_1$.

Algorithm 2.2: Mirror prox

Data: General VI on a set $Q \subset E$ with operator $\Phi(z)$, Lipschitz constant L of $\Phi(z)$, prox-setup: $d(z)$, $V[z](w)$.

Result: $\widehat{w}^k = \frac{1}{k}\sum_{i=0}^{k-1} w^i$.

1 Set $k = 0$, $z^0 = \operatorname{argmin}_{z \in Q} d(z)$.

2 **for** $k = 0, 1, \ldots$ **do**

3 Calculate

$$w^k = \operatorname*{argmin}_{z \in Q} \left\{ \langle \Phi(z^k), z \rangle + L V[z^k](z) \right\}; \tag{2.25}$$

4

$$z^{k+1} = \operatorname*{argmin}_{z \in Q} \left\{ \langle \Phi(w^k), z \rangle + L V[z^k](z) \right\}. \tag{2.26}$$

5 Set $k = k + 1$.

Theorem 2.5 ([45]) *Assume that $\Phi(z)$ is monotone and LLC. Then, for any $k \geq 1$ and any $u \in Q$,*

$$\max_{z \in Q} \langle \Phi(z), \widehat{w}^k - z \rangle \leq \frac{L}{k} \max_{z \in Q} V[z^0](z). \tag{2.27}$$

Moreover, if the VI is associated with a convex-concave saddle point problem, i.e.

- $E = E_1 \times E_2$;
- $Q = Q_1 \times Q_2$ *with convex compact sets* $Q_1 \subset E_1$, $Q_2 \subset E_2$;
- $\Phi(z) = \Phi(x, u) = \begin{pmatrix} \nabla_x f(x, u) \\ -\nabla_u f(x, u) \end{pmatrix}$ *for a continuously differentiable function* $f(x, u)$ *which is convex in* $x \in Q_1$ *and concave in* $u \in Q_2$;

then

$$\left[\max_{u \in Q_2} f(\widehat{x}^k, u) - \min_{x \in Q_1} \max_{u \in Q_2} f(x, u)\right] + \left[\min_{x \in Q_1} \max_{u \in Q_2} f(x, u) - \min_{x \in Q_1} f(x, \widehat{u}^k)\right] \leq \frac{L}{k} \max_{z \in Q} V[z^0](z). \tag{2.28}$$

Choosing appropriately the norm in the space $E_1 \times E_2$ and applying the mirror prox algorithm to solve the problem (2.22) as a saddle point problem, we obtain that

the saddle point error in the left hand side of (2.28) decays as $\frac{2\|A\|_{1,2}\sqrt{D_1 D_2}+L_h D_1}{k}$. This is slightly worse than the rate in (2.20) since the accelerated gradient method allows the faster decay for the smooth part $h(\boldsymbol{x})$. An accelerated mirror prox method with the same rate as in (2.20) can be found in [16].

2.5 NSO in Large Dimensions

The optimization of nonsmooth functionals with constraints attracts widespread interest in large-scale optimization and its applications [8, 59]. Subgradient methods for NSO have a long history starting with the method for deterministic unconstrained problems and Euclidean setting in [70] and the generalization for constrained problems in [61], where the idea of steps switching between the direction of subgradient of the objective and the direction of subgradient of the constraint was suggested. A non-Euclidean extension, usually referred to as the mirror descent, originated in [46, 48] and was later analyzed in [5]. An extension for constrained problems was proposed in [48], see also recent version in [7]. To prove faster convergence rate of the mirror descent for strongly convex objective in an unconstrained case, the restart technique [47–49] was used in [35]. Usually, the step size and stopping rule for the mirror descent requires to know the Lipschitz constant of the objective function and constraint, if any. Adaptive step sizes, which do not require this information, are considered in [46] for problems without inequality constraints, and in [7] for constrained problems.

Formally speaking, we consider the following convex constrained minimization problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & q(\boldsymbol{x}) \le 0, \\ & \boldsymbol{x} \in X \subset E, \end{cases} \tag{2.29}$$

where X is a convex closed subset of a finite-dimensional real vector space E, $f : X \to \mathbb{R}$, $q : E \to \mathbb{R}$ are convex functions.

We assume q to be a nonsmooth Lipschitz continuous function and the problem (2.3) to be regular. The last means that there exists a point $\bar{\boldsymbol{x}}$ in relative interior of the set X, such that $q(\bar{\boldsymbol{x}}) < 0$.

Note that, despite the problem (2.29) contains only one inequality constraint, considered algorithms allow to solve more general problems with a number of constraints given as $\{q_i(\boldsymbol{x}) \le 0, i = 1, \dots, m\}$. The reason is that these constraints can be aggregated and represented as an equivalent constraint given by $\{q(\boldsymbol{x}) \le 0\}$, where $q(\boldsymbol{x}) = \max_{i=1,\dots,m} q_i(\boldsymbol{x})$.

We consider two adaptive mirror descent methods [4] for the problem (2.29). Both considered methods have complexity $O\left(\frac{1}{\varepsilon^2}\right)$ and optimal.

We consider algorithms, which are based on the mirror descent method. Thus, we start with the description of proximal setup and basic properties of the mirror descent step. Let E be a finite-dimensional real vector space and E^* be its dual. We denote the value of a linear function $q \in E^*$ at $\boldsymbol{x} \in E$ by $\langle \boldsymbol{q}, \boldsymbol{x} \rangle$. Let $\|\cdot\|_E$ be some norm on E, $\|\cdot\|_{E,*}$ be its dual, defined by $\|\boldsymbol{q}\|_{E,*} = \max_{\boldsymbol{x}} \{\langle \boldsymbol{q}, \boldsymbol{x} \rangle, \|\boldsymbol{x}\|_E \leq 1\}$. We use $\nabla f(\boldsymbol{x})$ to denote any subgradient of a function f at a point $\boldsymbol{x} \in \operatorname{dom} f$.

Given a vector $\boldsymbol{x} \in X^0$, and a vector $\boldsymbol{p} \in E^*$, the mirror descent step is defined as

$$\begin{aligned} \boldsymbol{x}^+ = \operatorname{Mirr}[\boldsymbol{x}](\boldsymbol{p}) :&= \operatorname*{argmin}_{z \in X} \{\langle \boldsymbol{p}, z \rangle + V[\boldsymbol{x}](z)\} \\ &= \operatorname*{argmin}_{z \in X} \{\langle \boldsymbol{p}, z \rangle + d(z) - \langle \nabla d(\boldsymbol{x}), z \rangle\}. \end{aligned} \tag{2.30}$$

We make the simplicity assumption, which means that $\operatorname{Mirr}[\boldsymbol{x}](\boldsymbol{p})$ is easily computable.

The following lemma [9] describes the main property of the mirror descent step.

Lemma 2.1 *Let f be some convex function over a set X, $h > 0$ be a step size, $\boldsymbol{x} \in X^0$. Let the point $\boldsymbol{x}^+$ be defined by $\boldsymbol{x}^+ = \operatorname{Mirr}[\boldsymbol{x}](h(\nabla f(\boldsymbol{x})))$. Then, for any $z \in X$,*

$$\begin{aligned} h\big(f(\boldsymbol{x}) - f(z)\big) &\leq h\langle \nabla f(\boldsymbol{x}), \boldsymbol{x} - z \rangle \\ &\leq \frac{h^2}{2}\|\nabla f(\boldsymbol{x})\|^2 + V[\boldsymbol{x}](z) - V[\boldsymbol{x}^+](z). \end{aligned} \tag{2.31}$$

The following analog of Lemma 2.1 for δ-subgradient $\nabla_\delta f$ holds.

Lemma 2.2 *Let f be some convex function over a set X, $h > 0$ be a step size, $\boldsymbol{x} \in X^0$. Let the point $\boldsymbol{x}^+$ be defined by $\boldsymbol{x}^+ = \operatorname{Mirr}[\boldsymbol{x}](h \cdot (\nabla_\delta f(\boldsymbol{x})))$. Then, for any $z \in X$,*

$$\begin{aligned} h \cdot \big(f(\boldsymbol{x}) - f(z)\big) &\leq h \cdot \langle \nabla_\delta f(\boldsymbol{x}), \boldsymbol{x} - z \rangle + h \cdot \delta \\ &\leq \frac{h^2}{2}\|\nabla_\delta f(\boldsymbol{x})\| + h \cdot \delta + V[\boldsymbol{x}](z) - V[\boldsymbol{x}^+](z). \end{aligned} \tag{2.32}$$

We consider the problem (2.29) in two different settings, namely, nonsmooth Lipschitz continuous objective function f and general objective function f, which is not necessarily Lipschitz continuous, e.g. a quadratic function. In both cases, we assume that q is nonsmooth and Lipschitz continuous

$$|q(\boldsymbol{x}) - q(\boldsymbol{y})| \leq M_q \|\boldsymbol{x} - \boldsymbol{y}\|_E, \quad \boldsymbol{x}, \boldsymbol{y} \in X. \tag{2.33}$$

Let $\boldsymbol{x}_*$ be a solution to (2.29). We say that a point $\tilde{\boldsymbol{x}} \in X$ is an *ε-solution* to (2.29) if

$$f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x}_*) \leq \varepsilon, \quad q(\tilde{\boldsymbol{x}}) \leq \varepsilon. \tag{2.34}$$

All methods considered in this section (Algorithms 2.3 and 2.4) are applicable in the case of using δ-subgradient instead of usual subgradient. For this case we can get an ε-solution $\tilde{\boldsymbol{x}} \in X$:

$$f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x}_*) \le \varepsilon + O(\delta), \quad q(\tilde{\boldsymbol{x}}) \le \varepsilon + O(\delta). \tag{2.35}$$

The methods we describe are based on the of Polyak's switching subgradient method [61] for constrained convex problems, also analyzed in [53], and the mirror descent method originated in [48]; see also [46].

2.5.1 Convex Nonsmooth Objective Function

In this subsection, we assume that f is a nonsmooth Lipschitz continuous function

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le M_f \|\boldsymbol{x} - \boldsymbol{y}\|_E, \quad \boldsymbol{x}, \boldsymbol{y} \in X. \tag{2.36}$$

Let $\boldsymbol{x}_*$ be a solution to (2.29) and assume that we know a constant $\Theta_0 > 0$ such that

$$d(\boldsymbol{x}_*) \le \Theta_0^2. \tag{2.37}$$

For example, if X is a compact set, one can choose $\Theta_0^2 = \max_{\boldsymbol{x} \in X} d(\boldsymbol{x})$.

Algorithm 2.3: Adaptive mirror descent (nonsmooth objective)

Data: Accuracy $\varepsilon > 0$, Θ_0 s.t. $d(\boldsymbol{x}_*) \le \Theta_0^2$.

Result: $\bar{\boldsymbol{x}}^k := \frac{\sum_{i \in I} h_i \boldsymbol{x}^i}{\sum_{i \in I} h_i}$.

$\boldsymbol{x}^0 = \operatorname{argmin}_{\boldsymbol{x} \in X} d(\boldsymbol{x})$.
Initialize the set I as empty set.
Set $k = 0$.
while $\sum_{j=0}^{k-1} \frac{1}{M_j^2} < \frac{2\Theta_0^2}{\varepsilon^2}$ **do**
  if $q(\boldsymbol{x}^k) \le \varepsilon$, **then**
    $M_k = \|\nabla f(\boldsymbol{x}^k)\|_{E,*}$;
    $h_k = \frac{\varepsilon}{M_k^2}$;
    $\boldsymbol{x}^{k+1} = \mathrm{Mirr}[\boldsymbol{x}^k](h_k \nabla f(\boldsymbol{x}^k))$ ("productive step");
    Add k to I.
  else
    $M_k = \|\nabla q(\boldsymbol{x}^k)\|_{E,*}$;
    $h_k = \frac{\varepsilon}{M_k^2}$;
    $\boldsymbol{x}^{k+1} = \mathrm{Mirr}[\boldsymbol{x}^k](h_k \nabla q(\boldsymbol{x}^k))$ ("non-productive step").
  Set $k = k + 1$.

Theorem 2.6 *Assume that inequalities* (2.33) *and* (2.36) *hold and a known constant* $\Theta_0 > 0$ *is such that* $d(\boldsymbol{x}_*) \leq \Theta_0^2$. *Then, Algorithm 2.3 stops after not more than*

$$k = \left\lceil \frac{2\max\{M_f^2, M_q^2\}\Theta_0^2}{\varepsilon^2} \right\rceil \tag{2.38}$$

iterations and $\bar{\boldsymbol{x}}^k$ *is an* ε*-solution to* (2.29) *in the sense of* (2.34).

Let us now show that Algorithm 2.3 allows to reconstruct an approximate solution to the problem, which is dual to (2.29). We consider a special type of problem (2.29) with q given by

$$q(\boldsymbol{x}) = \max_{i \in \{1,\ldots,m\}} \left\{ q_i(\boldsymbol{x}) \right\}. \tag{2.39}$$

Then, the dual problem to (2.29) is

$$\begin{cases} \text{maximize} & \varphi(\boldsymbol{\lambda}) \\ \text{subject to} & \lambda_i \geq 0, i = 1, \ldots, m, \end{cases} \tag{2.40}$$

where $\varphi(\boldsymbol{\lambda}) = \min\limits_{\boldsymbol{x} \in X} \left\{ f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i q_i(\boldsymbol{x}) \right\}$ and $\lambda_i \geq 0, i = 1, \ldots, m$ are Lagrange multipliers.

We slightly modify the assumption (2.37) and assume that the set X is bounded and that we know a constant $\Theta_0 > 0$ such that

$$\max_{\boldsymbol{x} \in X} d(\boldsymbol{x}) \leq \Theta_0^2.$$

As before, denote $[k] = \{j \in \{0, \ldots, k-1\}\}$, $J = [k] \setminus I$. Let $j \in J$. Then a subgradient of $q(\boldsymbol{x})$ is used to make the j-th step of Algorithm 2.3. To find this subgradient, it is natural to find an active constraint $i \in 1, \ldots, m$ such that $q(\boldsymbol{x}^j) = q_i(\boldsymbol{x}^j)$ and use $\nabla q(\boldsymbol{x}^j) = \nabla q_i(\boldsymbol{x}^j)$ to make a step. Denote $i(j) \in 1, \ldots, m$ the number of active constraint, whose subgradient is used to make a non-productive step at iteration $j \in J$. In other words, $q(\boldsymbol{x}^j) = q_{i(j)}(\boldsymbol{x}^j)$ and $\nabla q(\boldsymbol{x}^j) = \nabla q_{i(j)}(\boldsymbol{x}^j)$. We define an approximate dual solution on a step $k \geq 0$ as

$$\bar{\lambda}_i^k = \frac{1}{\sum\limits_{j \in I} h_j} \sum_{j \in J, i(j) = i} h_j, \quad i \in \{1, \ldots, m\}, \tag{2.41}$$

and modify Algorithm 2.3 to return a pair $(\bar{\boldsymbol{x}}^k, \bar{\boldsymbol{\lambda}}^k)$.

Theorem 2.7 *Assume that the set* X *is bounded, the inequalities* (2.33) *and* (2.36) *hold and a known constant* $\Theta_0 > 0$ *is such that* $d(\boldsymbol{x}_*) \leq \Theta_0^2$. *Then, modified*

Algorithm 2.3 stops after not more than

$$k = \left\lceil \frac{2\max\{M_f^2, M_q^2\}\Theta_0^2}{\varepsilon^2} \right\rceil$$

iterations and the pair $(\bar{\boldsymbol{x}}^k, \bar{\boldsymbol{\lambda}}^k)$ *returned by this algorithm satisfies*

$$f(\bar{\boldsymbol{x}}^k) - \varphi(\bar{\boldsymbol{\lambda}}^k) \le \varepsilon, \quad q(\bar{\boldsymbol{x}}^k) \le \varepsilon. \tag{2.42}$$

Now we consider an interesting example of huge-scale problem [55, 59] with a sparse structure. We would like to illustrate two important ideas. Firstly, consideration of the dual problem can simplify the solution, if it is possible to reconstruct the solution of the primal problem by solving the dual problem. Secondly, for a special sparse nonsmooth piecewise linear functions we suggest a very efficient implementation of one subgradient iteration [55]. In such cases simple subgradient methods (for example, Algorithm 2.3) can be useful due to the relatively inexpensive cost of iterations.

Recall (see e.g. [59]) that the truss topology design problem consists in finding the best mechanical structure resisting to an external force with an upper bound for the total weight of construction. Its mathematical formulation looks as follows:

$$\begin{cases} \text{minimize} & \langle \overline{\boldsymbol{f}}, \boldsymbol{z} \rangle \\ \text{subject to} & A(\boldsymbol{w})\boldsymbol{z} = \overline{\boldsymbol{f}}, \\ & \langle \boldsymbol{e}, \boldsymbol{w} \rangle = T, \\ & \boldsymbol{w} \in R_+^m, \end{cases} \tag{2.43}$$

where $\overline{\boldsymbol{f}}$ is a vector of external forces, $\boldsymbol{z} \in \mathbb{R}^{2n}$ is a vector of virtual displacements of n nodes in $\mathbb{R}^2$, $\boldsymbol{w}$ is a vector of m bars, and T is the total weight of construction ($\boldsymbol{e} = (1, 1, ..., 1)$). The compliance matrix $A(\boldsymbol{w})$ has the following form:

$$A(\boldsymbol{w}) = \sum_{i=1}^{m} \boldsymbol{w}_i \boldsymbol{a}_i \boldsymbol{a}_i^T,$$

where $\boldsymbol{a}_i \in \mathbb{R}^{2n}$ are the vectors describing the interactions of two nodes connected by an arc. These vectors are very sparse: for 2D-model they have at most 4 nonzero elements.

Let us rewrite the problem (2.43) as a linear programming problem.

$$\begin{aligned} \min_{\boldsymbol{z}, \boldsymbol{w}} \{ \langle \overline{\boldsymbol{f}}, \boldsymbol{z} \rangle : \; & A(\boldsymbol{w})\boldsymbol{z} = \overline{\boldsymbol{f}}, \; \boldsymbol{w} \ge 0, \; \langle \boldsymbol{e}, \boldsymbol{w} \rangle = T \} \\ &= \min_{\boldsymbol{w}} \{ \langle \overline{\boldsymbol{f}}, A^{-1}(\boldsymbol{w})\overline{\boldsymbol{f}} \rangle : \; \boldsymbol{w} \in \triangle(T) = \{ \boldsymbol{w} \ge 0, \; \langle \boldsymbol{e}, \boldsymbol{w} \rangle = T \} \} \\ &= \min_{\boldsymbol{w} \in \triangle(T)} \max_{\boldsymbol{z}} \{ 2\langle \overline{\boldsymbol{f}}, \boldsymbol{z} \rangle - \langle A(\boldsymbol{w})\boldsymbol{z}, \boldsymbol{z} \rangle \} \end{aligned}$$

$$
\begin{aligned}
&\geq \max_{z} \min_{w \in \triangle(T)} \{2\langle \overline{f}, z\rangle - \langle A(w)z, z\rangle\} \qquad (2.44)\\
&= \max_{z} \{2\langle \overline{f}, z\rangle - T \max_{1\leq i\leq m} \langle a_i, z\rangle^2\}\\
&= \max_{\lambda, y} \{2\lambda\langle \overline{f}, y\rangle - \lambda^2 T \max_{1\leq i\leq m} \langle a_i, y\rangle^2\}\\
&= \max_{y} \frac{\langle \overline{f}, y\rangle^2}{T \max\limits_{1\leq i\leq m} \langle a_i, y\rangle^2}\\
&= \frac{1}{T}\left(\max_{y}\{\langle \overline{f}, y\rangle : \max_{1\leq i\leq m} |\langle a_i, y\rangle| \leq 1\}\right)^2.
\end{aligned}
$$

Note that for the inequality in the forth line we do not need any assumption.

Denote by y^* the optimal solution of the optimization problem in the brackets. Then there exist multipliers $x^* \in \mathbb{R}^m_+$ such that

$$
\overline{f} = \sum_{i\in J_+} a_i x_i^* - \sum_{i\in J_-} a_i x_i^*, \qquad x_i^* = 0,\ i \notin J_+ \bigcap J_-, \qquad (2.45)
$$

where $J_+ = \{i : \langle a_i, y^*\rangle = 1\}$, and $J_- = \{i : \langle a_i, y^*\rangle = -1\}$. Multiplying the first equation in (2.45) by y^*, we get

$$
\langle \overline{f}, y^*\rangle = \langle e, x^*\rangle. \qquad (2.46)
$$

Note that the first equation in (2.45) can be written as

$$
\overline{f} = A(x^*)y^*. \qquad (2.47)
$$

Let us reconstruct now the solution of the primal problem. Denote

$$
w^* = \frac{T}{\langle e, x^*\rangle}\cdot x^*, \qquad z^* = \frac{\langle e, x^*\rangle}{T}\cdot y^*. \qquad (2.48)
$$

Then, in view of (2.47) we have $\overline{f} = A(w^*)z^*$, and $w^* \in \triangle(T)$. Thus, the pair (2.48) is feasible for the primal problem. On the other hand,

$$
\langle \overline{f}, z^*\rangle = \langle \overline{f}, \frac{\langle e, x^*\rangle}{T}\cdot y^*\rangle = \frac{1}{T}\cdot\langle e, x^*\rangle\cdot\langle \overline{f}, y^*\rangle = \frac{1}{T}\cdot\langle \overline{f}, y^*\rangle^2.
$$

Thus, the duality gap in the chain (2.44) is zero, and the pair (w^*, z^*), defined by (2.48) is the optimal solution of the primal problem.

The above discussion allows us to concentrate on the following (dual) linear programming problem:

$$\begin{cases} \text{maximize} & \langle \bar{\boldsymbol{f}}, \boldsymbol{y} \rangle \\ \text{subject to} & \max\limits_{1 \le i \le m} \langle \pm \boldsymbol{a}_i, \boldsymbol{y} \rangle \le 1 \\ & \boldsymbol{y} \in X, \end{cases} \tag{2.49}$$

which we can solve by the primal-dual Algorithm 2.3.

Assume that we have *local* truss: each node is connected only with few neighbors. It allows to apply the property of *sparsity* for vectors $\boldsymbol{a}_i$ $(1 \le i \le m)$. In this case the computational cost of each iteration grows as $O(\log_2 m)$ [55, 59].

In [55] a special class of huge-scale problems with sparse subgradient was considered. According to [55] for smooth functions this is a very rare feature. For example, for quadratic function $f(\boldsymbol{y}) = \frac{1}{2}\langle A\boldsymbol{y}, \boldsymbol{y} \rangle$ the gradient $\nabla f(\boldsymbol{y}) = A\boldsymbol{y}$ usually is dense even for a sparse matrix A.

However, the subgradient of nonsmooth function $f(\boldsymbol{y}) = \max_{1 \le i \le m} \langle \boldsymbol{a}_i, \boldsymbol{y} \rangle$ (see (2.49) above) are sparse provided that all vectors $\boldsymbol{a}_i$ share this property. This fact is based on the following observation. For the function $f(\boldsymbol{y}) = \max_{1 \le i \le m} \langle \boldsymbol{a}_i, \boldsymbol{y} \rangle$ with sparse matrix $A = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m]$ the vector $\nabla f(\boldsymbol{y}) = \boldsymbol{a}_{i(\boldsymbol{y})}$ is a subgradient at point $\boldsymbol{y}$. Then the standard subgradient step

$$\boldsymbol{y}_+ = \boldsymbol{y} - h \cdot \nabla f(\boldsymbol{y})$$

changes only a few entries of vector $\boldsymbol{y}$ and the vector $\boldsymbol{z}_+ = A^T \boldsymbol{y}_+$ differs from $\boldsymbol{z} = A^T \boldsymbol{y}$ also in a few positions only. Thus, the function value $f(\boldsymbol{y}_+)$ can be easily updated provided that we have an efficient procedure for recomputing the maximum of m values.

Note the objective functional in (2.49) is linear and the costs of iteration of Algorithm 2.3 and considered in [55] switching simple subgradient scheme is comparable. At the same time, the step productivity condition is simpler for Algorithm 2.3 as considered in [55] switching subgradient scheme. Therefore main observations for [55] are correct for Algorithm 2.3.

2.5.2 General Convex and Quasiconvex Objectives

In this subsection, we assume that the objective function f in (2.29) might not satisfy (2.36) and, hence, its subgradient could be unbounded. One of the examples is a quadratic function. We also assume that inequality (2.37) holds.

We further consider ideas in [53, 57] and adapt them for the problem (2.29), in a way that our algorithm allows to use non-Euclidean proximal setup, as does the mirror descent, and does not require to know the constant M_q. Following [53], given

a function f for each subgradient $\nabla f(\boldsymbol{x})$ at a point $\boldsymbol{y} \in X$, we define

$$v_f[\boldsymbol{y}](\boldsymbol{x}) = \begin{cases} \left\langle \dfrac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|_{E,*}}, \boldsymbol{x} - \boldsymbol{y} \right\rangle, & \nabla f(\boldsymbol{x}) \neq 0 \\ 0, & \nabla f(\boldsymbol{x}) = 0 \end{cases}, \quad \boldsymbol{x} \in X. \tag{2.50}$$

The following result gives complexity estimate for Algorithm 2.4 in terms of $v_f[\boldsymbol{x}_*](\boldsymbol{x})$. Below we use this theorem to establish complexity result for smooth objective f.

Theorem 2.8 *Assume that inequality (2.33) holds and a known constant $\Theta_0 > 0$ is such that $d(\boldsymbol{x}_*) \leq \Theta_0^2$. Then, Algorithm 2.4 stops after not more than*

$$k = \left\lceil \frac{2 \max\{1, M_q^2\} \Theta_0^2}{\varepsilon^2} \right\rceil \tag{2.51}$$

iterations and it holds that $\min_{i \in I} v_f[\boldsymbol{x}_*](\boldsymbol{x}^i) \leq \varepsilon$ and $q(\bar{\boldsymbol{x}}^k) \leq \varepsilon$.

Algorithm 2.4: Adaptive mirror descent (general convex objective)

Data: Accuracy $\varepsilon > 0$, Θ_0 s.t. $d(\boldsymbol{x}_*) \leq \Theta_0^2$.
Result: $\bar{\boldsymbol{x}}^k := \operatorname{argmin}_{\boldsymbol{x}^j, j \in I} f(\boldsymbol{x}^j)$.

$\boldsymbol{x}^0 = \operatorname{argmin}_{\boldsymbol{x} \in X} d(\boldsymbol{x})$.
Initialize the set I as empty set.
Set $k = 0$.
while $|I| + \sum_{j \in J} \frac{1}{\|\nabla q(\boldsymbol{x}^j)\|_{E,*}^2} < \frac{2\Theta_0^2}{\varepsilon^2}$ **do**
  if $q(\boldsymbol{x}^k) \leq \varepsilon$, **then**
    $h_k = \frac{\varepsilon}{\|\nabla f(\boldsymbol{x}^k)\|_{E,*}}$;
    $\boldsymbol{x}^{k+1} = \mathrm{Mirr}[\boldsymbol{x}^k](h_k \nabla f(\boldsymbol{x}^k))$ ("productive step");
    Add k to I.
  else
    $h_k = \frac{\varepsilon}{\|\nabla q(\boldsymbol{x}^k)\|_{E,*}^2}$;
    $\boldsymbol{x}^{k+1} = \mathrm{Mirr}[\boldsymbol{x}^k](h_k \nabla q(\boldsymbol{x}^k))$ ("non-productive step").
  Set $k = k + 1$.

To obtain the complexity of our algorithm in terms of the values of the objective function f, we define nondecreasing function

$$\omega(\tau) = \begin{cases} \max\limits_{\boldsymbol{x} \in X} \{f(\boldsymbol{x}) - f(\boldsymbol{x}_*) : \|\boldsymbol{x} - \boldsymbol{x}_*\|_E \leq \tau\}, & \tau \geq 0, \\ 0, & \tau < 0. \end{cases} \tag{2.52}$$

and use the following lemma from [53].

Lemma 2.3 *Assume that f is a convex function. Then, for any $\boldsymbol{x} \in X$,*

$$f(\boldsymbol{x}) - f(\boldsymbol{x}_*) \leqslant \omega(v_f[\boldsymbol{x}_*](\boldsymbol{x})). \tag{2.53}$$

Corollary 2.1 *Assume that the objective function f in (2.29) is given as $f(\boldsymbol{x}) = \max_{i\in\{1,\dots,m\}} f_i(\boldsymbol{x})$, where $f_i(\boldsymbol{x})$, $i = 1,\dots,m$ are differentiable with Lipschitz continuous gradient*

$$\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\|_{E,*} \leq L_i \|\boldsymbol{x} - \boldsymbol{y}\|_E \quad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in X, \quad i \in \{1,\dots,m\}. \tag{2.54}$$

Then $\bar{\boldsymbol{x}}^k$ is $\widetilde{\varepsilon}$-solution to (2.29) in the sense of (2.34), where

$$\widetilde{\varepsilon} = \max\{\varepsilon, \varepsilon \max_{i=1,\dots,m} \|\nabla f_i(\boldsymbol{x}_*)\|_{E,*} + \varepsilon^2 \max_{i=1,\dots,m} L_i/2\}.$$

Remark 2.1 According to [50, 58] the main lemma 2.3 holds for quasiconvex objective functions [62] too:

$$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \max\{f(\boldsymbol{x}), f(\boldsymbol{y})\} \text{ for all } \boldsymbol{x}, \boldsymbol{y}, \alpha \in [0,1].$$

This means that results of this subsection are valid for quasiconvex objectives.

Remark 2.2 In view of the Lipschitzness and, generally speaking, nonsmoothness of functional limitations, the obtained estimate for the number of iterations means that the proposed method is optimal from the point of view of oracle evaluations: $O\left(\frac{1}{\varepsilon^2}\right)$ iterations are sufficient for achieving the required accuracy ε of solving the problem for the class of target functionals considered in this section of the article. Note also that the considered Algorithm 2.3 applies to the considered classes of problems with constraints with convex objective functionals of different smoothness levels. However, the non-fulfillment, generally speaking, of the Lipschitz condition for the objective functional f does not allow one to substantiate the optimality of Algorithm 2.3 in the general situation (for example, with a Lipschitz continuous gradient). More precisely, situations are possible when the productive steps of the norm (sub)gradients of the objective functional $\|\nabla f(\boldsymbol{x}^k)\|_*$ are large enough and this will interfere with the speedy achievement of the stopping criterion of Algorithm 2.3.

2.6 Universal Methods

In this section we consider the problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in Q \subseteq E, \end{cases} \tag{2.55}$$

where Q is a convex set and f is a convex function with Hölder continuous subgradient

$$\|\nabla f(\boldsymbol{x}^1) - \nabla f(\boldsymbol{x}^2)\|_* \leq L_\nu \|\boldsymbol{x}^1 - \boldsymbol{x}^2\|^\nu \tag{2.56}$$

with $\nu \in [0, 1]$. The case $\nu = 0$ corresponds to NSO and the case $\nu = 1$ corresponds to smooth optimization. The goal of this section is to present the universal accelerated gradient method first proposed by Nesterov [56]. This method is a blackbox method which does not require the knowledge of constants ν, L_ν and works in accordance with the lower complexity bound

$$O\left(\left(\frac{L_\nu R^{1+\nu}}{\epsilon}\right)^{\frac{2}{1+3\nu}}\right)$$

obtained in [48].

The main idea is based on the observation that a nonsmooth convex function can be upper bounded by a quadratic objective function slightly shifted above (See Fig. 2.3). More precisely, for any $\boldsymbol{x}, \boldsymbol{y} \in Q$,

$$\begin{aligned} f(\boldsymbol{y}) &\leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L_\nu}{1+\nu}\|\boldsymbol{y} - \boldsymbol{x}\|^{1+\nu} \\ &\leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L(\delta)}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 + \delta, \end{aligned} \tag{2.57}$$

where

$$L(\delta) = \left(\frac{1-\nu}{1+\nu}\frac{1}{\delta}\right)^{\frac{1-\nu}{1+\nu}} L_\nu^{\frac{2}{1+\nu}}.$$

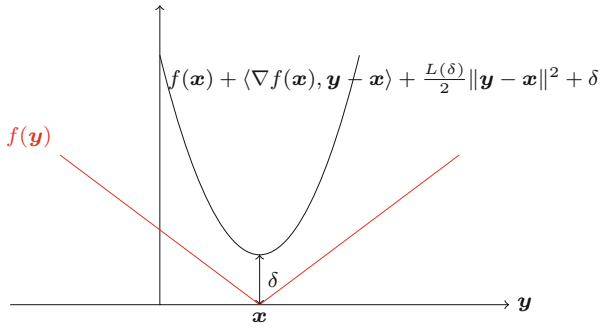

Fig. 2.3 Quadratic majorant of a nonsmooth function $f(\boldsymbol{x})$

The next idea is to apply an accelerated gradient method with backtracking procedure to adapt for the unknown $L(\delta)$ with appropriately chosen δ. The method we present is based on accelerated gradient method from [29, 30] and, thus is different from the original method of [56].

Algorithm 2.5: Universal accelerated gradient method

Data: Accuracy ϵ, starting point $\boldsymbol{x}^0 \in Q$, initial guess $L_0 > 0$, prox-setup: $d(\boldsymbol{x})$—1-strongly convex w.r.t. $\|\cdot\|_E$, $V[z](\boldsymbol{x}) := d(\boldsymbol{x}) - d(z) - \langle \nabla d(z), \boldsymbol{x} - z\rangle$.

Result: The point $\boldsymbol{y}^{k+1}$.

1 Set $k = 0$, $C_0 = \alpha_0 = 0$, $\boldsymbol{y}^0 = z^0 = \boldsymbol{x}^0$.

2 **for** $k = 0, 1, \ldots$ **do**

3 Set $M_k = L_k/2$.

4 **while**

$$f(\boldsymbol{y}^{k+1}) > f(\boldsymbol{x}^{k+1}) + \langle \nabla f(\boldsymbol{x}^{k+1}), \boldsymbol{y}^{k+1} - \boldsymbol{x}^{k+1}\rangle + \frac{M_k}{2}\|\boldsymbol{y}^{k+1} - \boldsymbol{x}^{k+1}\|^2 + \frac{\alpha_{k+1}\epsilon}{2C_{k+1}} \tag{2.58}$$

do

5 Set $M_k = 2M_k$, find α_{k+1} as the largest root of the equation

$$C_{k+1} := C_k + \alpha_{k+1} = M_k \alpha_{k+1}^2. \tag{2.59}$$

6 Calculate

$$\boldsymbol{x}^{k+1} = \frac{\alpha_{k+1} z^k + C_k \boldsymbol{y}^k}{C_{k+1}}; \tag{2.60}$$

7

$$z^{k+1} = \operatorname*{argmin}_{\boldsymbol{x}\in Q}\{V[z^k](\boldsymbol{x}) + \alpha_{k+1}(f(\boldsymbol{x}^{k+1}) + \langle \nabla f(\boldsymbol{x}^{k+1}), \boldsymbol{x} - \boldsymbol{x}^{k+1}\rangle)\}; \tag{2.61}$$

8

$$\boldsymbol{y}^{k+1} = \frac{\alpha_{k+1} z^{k+1} + C_k \boldsymbol{y}^k}{C_{k+1}}. \tag{2.62}$$

9 Set $L_{k+1} = M_k/2$, $k = k + 1$.

Inequality (2.57) guarantees that the backtracking procedure in the inner cycle is finite.

Theorem 2.9 ([56]) *Let f satisfy (2.56). Then,*

$$f(\boldsymbol{y}^{k+1}) - f_\star \leq \left(\frac{2^{2+4\nu} L_\nu^2}{\epsilon^{1-\nu} k^{1+3\nu}}\right)^{\frac{1}{1+\nu}} V[\boldsymbol{x}^0](\boldsymbol{x}^\star) + \frac{\epsilon}{2}. \tag{2.63}$$

Moreover, the number of oracle calls is bounded by

$$4(k+1) + 2\log_2\left((2V[\boldsymbol{x}^0](\boldsymbol{x}^\star))^{\frac{1-\nu}{1+3\nu}} \left(\frac{1}{\epsilon}\right)^{\frac{3(1-\nu)}{1+3\nu}} L_\nu^{\frac{4}{1+3\nu}}\right).$$

Translating this rate of convergence to the language of complexity, we obtain that to obtain a solution with an accuracy ϵ the number of iterations is no more than

$$O\left(\inf_{\nu\in[0,1]} \left(\frac{L_\nu}{\epsilon}\right)^{\frac{2}{1+3\nu}} \left(V[\boldsymbol{x}^0](\boldsymbol{x}^\star)\right)^{\frac{1+\nu}{1+3\nu}}\right),$$

i.e. is optimal.

In his paper, Nesterov considers a more general composite optimization problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) + h(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in Q \subseteq E, \end{cases} \tag{2.64}$$

where h is a simple convex function, and obtains the same complexity guarantees. Universal methods were extended for the case of strongly convex problems by a restart technique in [65], for nonconvex optimization in [33] and for the case of nonconvex optimization with inexact oracle in [26]. As we can see from (2.57), universal accelerated gradient method is connected to smooth problems with inexact oracle. The study of accelerated gradient methods with inexact oracle was first proposed in [20] and was very well developed in [11, 22, 26, 27] including stochastic optimization problems and strongly convex problems. A universal method with inexact oracle can be found in [28]. Experiments show [56] that universal method accelerates to $O\left(\frac{1}{k}\right)$ rate for nonsmooth problems with a special "smoothing friendly" (see Sect. 2.4) structure. This is especially interesting for traffic flow modeling problems, which possess such structure [3].

Now we consider universal analog of Nemirovski's proximal mirror method for variational inequalities with a Hölder continuous operator. More precisely, we consider universal extension of Algorithm 2.2 which allows to solve smooth and nonsmooth variational inequalities without the prior knowledge of the smoothness. Main idea of the this method is the adaptive choice of constants and level of

smoothness in minimized prox-mappings at each iteration. These constants are related to the Hölder constant of the operator and this method allows to find a suitable constant at each iteration.

Algorithm 2.6: Universal mirror prox

Data: General VI on a set $Q \subset E$ with operator $\Phi(z)$, accuracy $\varepsilon > 0$, initial guess $M_{-1} > 0$, prox-setup: $d(z)$, $V[z](\boldsymbol{w})$.

Result: $\widehat{\boldsymbol{w}}^k = \frac{1}{k}\sum_{i=0}^{k-1} \boldsymbol{w}^i$.

Set $k = 0$, $z^0 = \operatorname{argmin}_{z \in Q} d(z)$.

for $k = 0, 1, \ldots$ **do**

  Set $i_k = 0$.

  while

$$\langle \Phi(\boldsymbol{w}^k) - \Phi(z^k), \boldsymbol{w}^k - z^{k+1} \rangle > \frac{M_k}{2}\left(\|\boldsymbol{w}^k - z^k\|^2 + \|\boldsymbol{w}^k - z^{k+1}\|^2\right) + \frac{\varepsilon}{2} \tag{2.65}$$

  do

    Set $M_k = 2^{i_k - 1} M_{k-1}$.

    Calculate

$$\boldsymbol{w}^k = \operatorname*{argmin}_{z \in Q}\left\{\langle \Phi(z^k), z\rangle + M_k V[z^k](z)\right\}; \tag{2.66}$$

$$z^{k+1} = \operatorname*{argmin}_{z \in Q}\left\{\langle \Phi(\boldsymbol{w}^k), z\rangle + M_k V[z^k](z)\right\}. \tag{2.67}$$

    $i_k = i_k + 1$.

  Set $k = k + 1$.

Theorem 2.10 ([32]) *For any $k \geq 1$ and any $z \in Q$,*

$$\frac{1}{\sum_{i=0}^{k-1} M_i^{-1}} \sum_{i=0}^{k-1} M_i^{-1} \langle \Phi(\boldsymbol{w}^i), \boldsymbol{w}^i - z\rangle \leq \frac{1}{\sum_{i=0}^{k-1} M_i^{-1}} (V[z^0](z) - V[z^k](z)) + \frac{\varepsilon}{2}. \tag{2.68}$$

Note that if $\max_{z \in Q} V[z^0](z) \leq D$, we can construct the following adaptive stopping criterion for our algorithm

$$\frac{D}{\sum_{i=0}^{k-1} M_i^{-1}} \leq \frac{\varepsilon}{2}.$$

Next, we consider the case of Hölder continuous operator Φ and show that Algorithm 2.6 is universal. Assume for some $\nu \in [0, 1]$ and $L_\nu \geq 0$

$$\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{y})\|_* \leq L_\nu \|\boldsymbol{x} - \boldsymbol{y}\|^\nu, \quad \boldsymbol{x}, \boldsymbol{y} \in Q$$

holds. The following inequality is a generalization of (2.57) for VI. For any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in Q$ and $\delta > 0$,

$$\begin{aligned}\langle \Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{z}\rangle &\leq \|\Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x})\|_* \|\boldsymbol{y} - \boldsymbol{z}\| \\ &\leq L_\nu \|\boldsymbol{x} - \boldsymbol{y}\|^\nu \|\boldsymbol{y} - \boldsymbol{z}\| \\ &\leq \frac{1}{2}\left(\frac{1}{\delta}\right)^{\frac{1-\nu}{1+\nu}} L_\nu^{\frac{2}{1+\nu}} \left(\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \|\boldsymbol{y} - \boldsymbol{z}\|^2\right) + \frac{\delta}{2},\end{aligned}$$

where

$$L(\delta) = \left(\frac{1}{\delta}\right)^{\frac{1-\nu}{1+\nu}} L_\nu^{\frac{2}{1+\nu}}. \tag{2.69}$$

So, we have

$$\langle \Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{z}\rangle \leq \frac{L(\delta)}{2}\left(\|\boldsymbol{y} - \boldsymbol{x}\|^2 + \|\boldsymbol{y} - \boldsymbol{z}\|^2\right) + \delta. \tag{2.70}$$

Let us consider estimates of the necessary number of iterations are obtained to achieve a given quality of the variational inequality solution.

Corollary 2.2 (Universal Method for VI) *Assume that the operator Φ is Hölder continuous with constant L_ν for some $\nu \in [0, 1]$ and $M_{-1} \leq \left(\frac{2}{\varepsilon}\right)^{\frac{1-\nu}{1+\nu}} L_\nu^{\frac{2}{1+\nu}}$. Also assume that the set Q is bounded. Then, for all $k \geq 0$, we have*

$$\max_{z \in Q} \langle \Phi(z), \widehat{\boldsymbol{w}}_k - z\rangle \leq \frac{(2L_\nu)^{\frac{2}{1+\nu}}}{k\varepsilon^{\frac{1-\nu}{1+\nu}}} \max_{z \in Q} V[z^0](z) + \frac{\varepsilon}{2} \tag{2.71}$$

As it follows from (2.70), if $M_k \geq L(\frac{\varepsilon}{2})$, (2.65) holds. Thus, for all $i = 0, \ldots, k-1$, we have $M_i \leq 2 \cdot L(\frac{\varepsilon}{2})$ and

$$\frac{1}{\sum_{i=0}^{k-1} M_i^{-1}} \leq \frac{2L(\frac{\varepsilon}{2})}{k} \leq \frac{(2L_\nu)^{\frac{2}{1+\nu}}}{k\varepsilon^{\frac{1-\nu}{1+\nu}}},$$

Equation (2.71) holds. Here $L(\cdot)$ is defined in (2.69). □

Let us add some remarks.

Remark 2.3 Since the algorithm does not use the values of parameters ν and L_ν, we obtain the following iteration complexity bound

$$2 \inf_{\nu \in [0,1]} \left(\frac{2L_\nu}{\varepsilon} \right)^{\frac{2}{1+\nu}} \cdot \max_{z \in Q} V[z^0](z)$$

to achieve

$$\max_{z \in Q} \langle \Phi(z), \widehat{w}_k - z \rangle \leq \varepsilon.$$

Using the same reasoning as in [56], we estimate the number of oracle calls for Algorithm 2.6. The number of oracle calls on each iteration k is equal to $2i_k$. At the same time, $M_k = 2^{i_k - 2} M_{k-1}$ and, hence, $i_k = 2 + \log_2 \frac{M_k}{M_{k-1}}$. Thus, the total number of oracle calls is

$$\sum_{j=0}^{k-1} i_j = 4k + 2 \sum_{i=0}^{k-1} \log_2 \frac{M_j}{M_{j-1}} < 4k + 2 \log_2 \left(2L \left(\frac{\varepsilon}{2} \right) \right) - 2 \log_2 (M_{-1}), \tag{2.72}$$

where we used that $M_k \leq 2L(\frac{\varepsilon}{2})$.

Thus, the number of oracle calls of the Algorithm 2.6 does not exceed:

$$4 \inf_{\nu \in [0,1]} \left(\frac{2 \cdot L_\nu}{\varepsilon} \right)^{\frac{2}{1+\nu}} \cdot \max_{u \in Q} V[z^0](u) + 2 \inf_{\nu \in [0,1]} \log_2 2 \left(\left(\frac{2}{\varepsilon} \right)^{\frac{1-\nu}{1+\nu}} L_\nu^{\frac{2}{1+\nu}} \right) - 2 \log_2 (M_{-1}).$$

Remark 2.4 We can apply this method to convex-concave saddle problems of the form

$$\begin{cases} \text{minimize} & \max\limits_{y \in Q_2} f(x, y) \\ \text{subject to} & x \in Q_1, \end{cases} \tag{2.73}$$

where $Q_{1,2}$ are convex compacts in $\mathbb{R}^n$, f is convex in x and concave in y, there is $\nu \in [0, 1]$ and constants $L_{11,\nu}, L_{12,\nu}, L_{21,\nu}, L_{22,\nu} < +\infty$:

$$\|\nabla_x f(x + \Delta x, y + \Delta y) - \nabla_x f(x, y)\|_{1,*} \leq L_{11,\nu} \|\Delta x\|_1^\nu + L_{12,\nu} \|\Delta y\|_2^\nu,$$

$$\|\nabla_y f(x + \Delta x, y + \Delta y) - \nabla_y f(x, y)\|_{2,*} \leq L_{21,\nu} \|\Delta x\|_1^\nu + L_{22,\nu} \|\Delta y\|_2^\nu$$

for all $x, x + \Delta x \in Q_1$, $y, y + \Delta y \in Q_2$.

It is possible to achieve an acceptable approximation $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}}) \in Q_1 \times Q_2$ such that

$$\max_{\boldsymbol{y} \in Q_2} f(\widehat{\boldsymbol{x}}, \boldsymbol{y}) - \min_{\boldsymbol{x} \in Q_1} f(\boldsymbol{x}, \widehat{\boldsymbol{y}}) \leq \varepsilon \tag{2.74}$$

for the saddle point $(\boldsymbol{x}_*, \boldsymbol{y}_*) \in Q_1 \times Q_2$ of (2.73) in no more than

$$O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{2}{1+\nu}}\right)$$

iterations, which indicates the optimality of the proposed method, at least for $\nu = 0$ and $\nu = 1$. However, in practice experiments show that (2.74) can be achieved much faster due to the adaptability of the method.

2.7 Concluding Remarks

Modern numerical methods for nonsmooth convex optimization problems are typically based on the structure of the problem. We start with one of the most powerful example of such type. For the geometric median search problem there exists efficient method that significantly outperform described above lower complexity bounds [17]. In machine learning we typically meet the problems with hidden affine structure and small effective dimension (SVM) that allow us to use different smoothing techniques [1]. Description of one of these techniques (Nesterov's smoothing technique) one can find in this survey. The other popular technique is based on averaging of the function around the small ball with the center at the point in consideration [25]. A huge amount of data since applications lead to composite optimization problems with nonsmooth composite (LASSO). For this class of problems accelerated (fast) gradient methods are typically applied [6, 39, 54]. This approach (composite optimization) have been recently expanded for more general class of problems [73]. In different image processing applications one can find a lot of nonsmooth problems formulations with saddle point structure. That is the goal function has Legendre representation. In this case one can apply special versions of accelerated (primal-dual) methods [14, 15, 41]. Universal mirror prox method described above demonstrates the alternative approach which can be applied in rather general context. Unfortunately, the most of these tricks have proven to be beyond the scope of this survey. But we include in the survey the description of the universal accelerated gradient descent algorithm [73] which in the general case can also be applied to a wide variety of problems.

Another important direction in nonsmooth convex optimization is huge-scale optimization for sparse problems [55]. The basic idea that reduce huge dimension to nonsmoothness is as follows:

$$\langle \boldsymbol{a}_k, \boldsymbol{x} \rangle - b_k \leq 0, \quad k = 1, \ldots m, \quad m \gg 1$$

is equivalent to the single nonsmooth constraint:

$$\max_{k=1,\dots m} \{\langle \boldsymbol{a}_k, \boldsymbol{x}\rangle - b_k\} \leq 0.$$

We demonstrated this idea above on truss topology design example.

One should note that we concentrate in this survey only on deterministic convex optimization problems, but the most beautiful things in NSO is that stochasticity [24, 36, 37, 48] and online context [34] in general doesn't change (up to a logarithmic factor in the strongly convex case) anything in complexity estimates. As an example, of stochastic (randomized) approach one can mentioned the work [2] where one can find reformulation of Google problem as a nonsmooth convex optimization problem. Special randomized mirror descent algorithm allows to solve this problem almost independently on the number of vertexes.

Finally, let's note that in the decentralized distributed nonsmooth (stochastic) convex optimization for the last few years there appear optimal methods [40, 67, 74].

Acknowledgements The chapter was supported in its major parts by the grant 18-29-03071 mk from Russian Foundation for Basic Research. E. Nurminski acknowledges the partial support from the project 1.7658.2017/6.7 of Ministry of Science and Higher Professional Education in Sect. 2.2. The work of A. Gasnikov, P. Dvurechensky and F. Stonyakin in Sect. 2.4 was partially supported by Russian Foundation for Basic Research grant 18-31-20005 mol_a_ved. The work of F. Stonyakin in Sect. 2.5.1, Corollary 2.2, Remarks 2.3 and 2.4 was supported by Russian Science Foundation grant 18-71-00048.

References

1. Allen-Zhu, Z., Hazan, E.: Optimal black-box reductions between optimization objectives. In: Advances in Neural Information Processing Systems, pp. 1614–1622 (2016)
2. Anikin, A., Gasnikov, A., Gornov, A., Kamzolov, D., Maximov, Y., Nesterov, Y.: Effective numerical methods for huge-scale linear systems with double-sparsity and applications to PageRank. Proceedings of Moscow Institute of Physics and Technology. **7**(4), 74–94 (2015). arXiv:1508.07607
3. Baimurzina, D., Gasnikov, A., Gasnikova, E., Dvurechensky, P., Ershov, E., Kubentaeva, M., Lagunovskaya, A.: Universal Method of Searching for Equilibria and Stochastic Equilibria in Transportation Networks. Computational Mathematics and Mathematical Physics. **59**(1), 19–33 (2019). https://doi.org/10.1134/S0965542519010020. arXiv:1701.02473
4. Bayandina, A., Dvurechensky, P., Gasnikov, A., Stonyakin, F., Titov, A.: Mirror descent and convex optimization problems with non-smooth inequality constraints. In: Giselsson, P., Rantzer, A. (eds.) Large-Scale and Distributed Optimization, Chap. 8, pp. 181–215. Springer, Berlin (2018). https://doi.org/10.1007/978-3-319-97478-1_8. arXiv:1710.06612
5. Beck, A., Teboulle, M.: Mirror descent and nonlinear projected subgradient methods for convex optimization. Oper. Res. Lett. **31**(3), 167–175 (2003). https://doi.org/10.1016/S0167-6377(02)00231-6
6. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009). https://doi.org/10.1137/080716542

7. Beck, A., Ben-Tal, A., Guttmann-Beck, N., Tetruashvili, L.: The comirror algorithm for solving nonsmooth constrained convex problems. Oper. Res. Lett. **38**(6), 493–498 (2010). https://doi.org/10.1016/j.orl.2010.08.005
8. Ben-Tal, A., Nemirovski, A.: Robust truss topology design via semidefnite programming. SIAM J. Optim. **7**(4), 991–1016 (1997)
9. Ben-Tal, A., Nemirovski, A.: Lectures on Modern Convex Optimization (Lecture Notes). Personal web-page of A. Nemirovski (2015). http://www2.isye.gatech.edu/~nemirovs/Lect_ModConvOpt.pdf
10. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, Berlin (2012)
11. Bogolubsky, L., Dvurechensky, P., Gasnikov, A., Gusev, G., Nesterov, Y., Raigorodskii, A.M., Tikhonov, A., Zhukovskii, M.: Learning supervised pagerank with gradient-based and gradient-free optimization methods. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 4914–4922. Curran Associates, Red Hook (2016). ArXiv:1603.00717
12. Brent, R.: Algorithms for Minimization Without Derivatives. Dover Books on Mathematics. Dover, New York (1973). https://books.google.de/books?id=6Ay2biHG-GEC
13. Bubeck, S.: Convex optimization: algorithms and complexity. Found. Trends Mach. Learn. **8**(3–4), 231–357 (2015). https://arxiv.org/pdf/1405.4980.pdf
14. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vision **40**(1), 120–145 (2011)
15. Chen, Y., Lan, G., Ouyang, Y.: Optimal primal-dual methods for a class of saddle point problems. SIAM J. Optim. **24**(4), 1779–1814 (2014)
16. Chen, Y., Lan, G., Ouyang, Y.: Accelerated schemes for a class of variational inequalities. Math. Program. **165**(1), 113–149 (2017)
17. Cohen, M.B., Lee, Y.T., Miller, G., Pachocki, J., Sidford, A.: Geometric median in nearly linear time. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing, pp. 9–21. ACM, New York (2016)
18. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 2292–2300. Curran Associates, Red Hook (2013)
19. Cuturi, M., Doucet, A.: Fast computation of Wasserstein barycenters. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning, vol. 32, pp. 685–693. PMLR, Bejing (2014). http://proceedings.mlr.press/v32/cuturi14.html
20. d'Aspremont, A.: Smooth optimization with approximate gradient. SIAM J. Optim. **19**(3), 1171–1183 (2008). https://doi.org/10.1137/060676386
21. Demyanov, A., Demyanov, V., Malozemov, V.: Minmaxmin problems revisited. Optim. Methods Softw. **17**(5), 783–804 (2002). https://doi.org/10.1080/1055678021000060810
22. Devolder, O., Glineur, F., Nesterov, Y.: First-order methods of smooth convex optimization with inexact oracle. Math. Program. **146**(1), 37–75 (2014). https://doi.org/10.1007/s10107-013-0677-5
23. Drori, Y., Teboulle, M.: An optimal variants of Kelley's cutting-plane method. Math. Program. **160**(1–2), 321–351 (2016)
24. Duchi, J.: Introductory lectures on stochastic optimization. Park City Mathematics Institute, Graduate Summer School Lectures (2016)
25. Duchi, J.C., Bartlett, P.L., Wainwright, M.J.: Randomized smoothing for stochastic optimization. SIAM J. Optim. **22**(2), 674–701 (2012)
26. Dvurechensky, P.: Gradient method with inexact oracle for composite non-convex optimization (2017). arXiv:1703.09180
27. Dvurechensky, P., Gasnikov, A.: Stochastic intermediate gradient method for convex problems with stochastic inexact oracle. J. Optim. Theory Appl. **171**(1), 121–145 (2016). https://doi.org/10.1007/s10957-016-0999-6

28. Dvurechensky, P., Gasnikov, A., Kamzolov, D.: Universal intermediate gradient method for convex problems with inexact oracle. Optimization Methods and Software (accepeted) (2019). arXiv:1712.06036
29. Dvurechensky, P., Gasnikov, A., Kroshnin, A.: Computational optimal transport: complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, vol. 80, pp. 1367–1376 (2018). arXiv:1802.04367
30. Dvurechensky, P., Gasnikov, A., Omelchenko, S., Tiurin, A.: Adaptive similar triangles method: a stable alternative to Sinkhorn's algorithm for regularized optimal transport (2017). arXiv:1706.07622
31. Dvurechensky, P., Dvinskikh, D., Gasnikov, A., Uribe, C.A., Nedić, A.: Decentralize and randomize: faster algorithm for Wasserstein barycenters. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 10783–10792. Curran Associates, Inc. (2018). arXiv:1802.04367
32. Gasnikov, A., Dvurechensky, P., Stonyakin, F., Titov, A.: Adaptive proximal method for variational inequalities. Comput. Math. Math. Phys. **59**(5), 836–841 (2019)
33. Ghadimi, S., Lan, G., Zhang, H.: Generalized uniformly optimal methods for nonlinear programming. Journal of Scientific Computing. 79, 1854–1881 (2019) https://doi.org/10.1007/s10915-019-00915-4. arXiv:1508.07384. https://arxiv.org/abs/1508.07384
34. Hazan, E.: Introduction to online convex optimization. Found. Trends® Optim. **2**(3–4), 157–325 (2016)
35. Juditsky, A., Nemirovski, A.: First order methods for non-smooth convex large-scale optimization, I: general purpose methods. In: Sra, S., Nowozin, S. (ed.) Optimization for Machine Learning, pp. 121–184. MIT Press, Cambridge, MA (2012)
36. Juditsky, A., Nemirovski, A.: First order methods for nonsmooth convex large-scale optimization, I: general purpose methods. Optimization for Machine Learning, pp. 121–148 (2011)
37. Juditsky, A., Nemirovski, A.: First order methods for nonsmooth convex large-scale optimization, II: utilizing problems structure. Optimization for Machine Learning, pp. 149–183 (2011)
38. Khachiyan, L.G.: A polynomial algorithm in linear programming. In: Doklady Academii Nauk SSSR, vol. 244, pp. 1093–1096 (1979)
39. Lan, G.: Gradient sliding for composite optimization. Math. Program. **159**(1), 201–235 (2016). https://doi.org/10.1007/s10107-015-0955-5
40. Lan, G., Lee, S., Zhou, Y.: Communication-efficient algorithms for decentralized and stochastic optimization (2017). arXiv:1701.03961
41. Lan, G., Ouyang, Y.: Accelerated gradient sliding for structured convex optimization (2016). arXiv:1609.04905
42. Lee, Y.T., Sidford, A., Wong, S.C.W.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), pp. 1049–1065. IEEE, Piscataway (2015)
43. Levin, A.Y.: On an algorithm for the minimization of convex functions. In: Soviet Mathematics Doklady (1965)
44. Nedić, A., Ozdaglar, A.: Approximate primal solutions and rate analysis for dual subgradient methods. SIAM J. Optim. **19**(4), 1757–1780 (2009). https://doi.org/10.1137/070708111
45. Nemirovski, A.: Prox-method with rate of convergence $o(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. SIAM J. Optim. **15**(1), 229–251 (2004)
46. Nemirovskii, A.: Efficient methods for large-scale convex optimization problems. Ekonomika i Matematicheskie Metody **15** (1979) (in Russian)
47. Nemirovskii, A., Nesterov, Y.: Optimal methods of smooth convex minimization. USSR Comput. Math. Math. Phys. **25**(2), 21–30 (1985). https://doi.org/10.1016/0041-5553(85)90100-4
48. Nemirovsky, A., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. Wiley, New York (1983)

49. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Math. Doklady **27**(2), 372–376 (1983)
50. Nesterov, Y.: Effective Methods in Nonlinear Programming. Radio i Svyaz, Moscow (1989)
51. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. **103**(1), 127–152 (2005)
52. Nesterov, Y.: Primal-dual subgradient methods for convex problems. Math. Program. **120**(1), 221–259 (2009). https://doi.org/10.1007/s10107-007-0149-x. First appeared in 2005 as CORE discussion paper 2005/67
53. Nesterov, Y.: Introduction to Convex Optimization. MCCME, Moscow (2010)
54. Nesterov, Y.: Gradient methods for minimizing composite functions. Math. Program. **140**(1), 125–161 (2013). First appeared in 2007 as CORE discussion paper 2007/76
55. Nesterov, Y.: Subgradient methods for huge-scale optimization problems. Math. Program. **146**(1), 275–297 (2014). https://doi.org/10.1007/s10107-013-0686-4. First appeared in 2012
56. Nesterov, Y.: Universal gradient methods for convex optimization problems. Math. Program. **152**(1), 381–404 (2015). https://doi.org/10.1007/s10107-014-0790-0
57. Nesterov, Y.: Subgradient methods for convex functions with nonstandard growth properties (2016). http://www.mathnet.ru:8080/PresentFiles/16179/growthbm_nesterov.pdf
58. Nesterov, Y.: Lectures on Convex Optimization. Springer, Berlin (2018)
59. Nesterov, Y., Shpirko, S.: Primal-dual subgradient method for huge-scale linear conic problems. SIAM J. Optim. **24**(3), 1444–1457 (2014). https://doi.org/10.1137/130929345
60. Newman, D.: Location of the maximum on unimodal surfaces. J. Assoc. Comput. Mach. **12**, 395–398 (1965)
61. Polyak, B.: A general method of solving extremum problems. Soviet Math. Doklady **8**(3), 593–597 (1967)
62. Polyak, B.T.: Minimization of nonsmooth functionals. USSR Comput. Math. Math. Phys. **9**(3), 14–29 (1969). https://www.sciencedirect.com/science/article/abs/pii/0041555369900615
63. Polyak, B.: Introduction to Optimization. Optimization Software, New York (1987)
64. Rockafellar, R.: Convex Analysis. Priceton University, Princeton (1970)
65. Roulet, V., d'Aspremont, A.: Sharpness, restart and acceleration. In: Guyon, I., Luxburg U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 1119–1129. Curran Associates, Inc. (2017). arXiv:1702.03828
66. Lacost-Julien, S., Schmidt, M., Bach, F.: A simpler approach to obtaining $o(1/t)$ convergence rate for the projected stochastic subgradient method (2012). arxiv:1212.2002. http://arxiv.org/pdf/1212.2002v2.pdf
67. Scaman, K., Bach, F., Bubeck, S., Massoulié, L., Lee, Y.T.: Optimal algorithms for non-smooth distributed optimization in networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 2740–2749. Curran Associates, Inc. (2018). arXiv:1806.00291v1
68. Shor, N.: Minimization of Nondifferentiable Functions. Naukova Dumka, Kyiv (1979)
69. Shor, N.: Minimization Methods for Non-Differentiable Functions. Springer, Berlin (1985)
70. Shor, N.Z.: Generalized gradient descent with application to block programming. Kibernetika **3**(3), 53–55 (1967)
71. Shor, N.Z., Kiwiel, K.C., Ruszczynski, A.: Minimization Methods for Non-Differentiable Functions. Springer Series in Computational Mathematics, vol. 3. Springer, Berlin (2012)
72. Tran-Dinh, Q., Fercoq, O., Cevher, V.: A smooth primal-dual optimization framework for nonsmooth composite convex minimization. SIAM J. Optim. **28**(1), 96–134 (2018). https://doi.org/10.1137/16M1093094. arXiv:1507.06243
73. Tyurin, A., Gasnikov, A.: Fast gradient descent method for convex optimization problems with an oracle that generates a model of a function in a requested point. Comput. Math. Math. Phys. **59**(7), 1085–1097 (2019)
74. Uribe, C.A., Lee, S., Gasnikov, A., Nedić, A.: Optimal algorithms for distributed optimization (2017). arXiv:1712.00232

75. Uribe, C.A., Dvinskikh, D., Dvurechensky, P., Gasnikov, A., Nedić, A.: Distributed computation of Wasserstein barycenters over networks. In: 2018 IEEE Conference on Decision and Control (CDC) pp. 6544–6549. IEEE (2018). arXiv:1803.02933. https://doi.org/10.1109/CDC.2018.8619160.
76. Vaidya, P.M.: Speeding-up linear programming using fast matrix multiplication. In: 30th Annual Symposium on Foundations of Computer Science, 1989, p. 332–337 (1989)

Chapter 3
Standard Bundle Methods: Untrusted Models and Duality

Antonio Frangioni

Abstract We review the basic ideas underlying the vast family of algorithms for nonsmooth convex optimization known as "bundle methods". In a nutshell, these approaches are based on constructing models of the function, but lack of continuity of first-order information implies that these models cannot be trusted, not even close to an optimum. Therefore, many different forms of stabilization have been proposed to try to avoid being led to areas where the model is so inaccurate as to result in almost useless steps. In the development of these methods, duality arguments are useful, if not outright necessary, to better analyze the behaviour of the algorithms. In addition, in many relevant applications the function at hand is itself a dual one, so that duality allows to map back algorithmic concepts and results into a "primal space" where they can be exploited; in turn, structure in that space can be exploited to improve the algorithms' behaviour, e.g. by developing better models. We present an updated picture of the many developments around the basic idea along at least three different axes: form of the stabilization, form of the model, and approximate evaluation of the function.

3.1 Introduction

We will describe the general ideas behind a large class of algorithms for the convex minimization problem

$$f_* = \begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in X, \end{cases} \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is proper and convex but possibly nondifferentiable. The problem (3.1) is quite general because of the "minimal" assumptions on how f

A. Frangioni (✉)
Dipartimento di Informatica, Università di Pisa, Pisa, Italy
e-mail: frangio@di.unipi.it

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_3

is provided: any computational procedure (an *oracle*) that, given $\boldsymbol{x}$, returns the value $f(\boldsymbol{x})$ and information about the first-order behaviour of f at $\boldsymbol{x}$ under the form of a subgradient $z \in \partial f(\boldsymbol{x})$ (both can actually be *approximated*, cf. Sect. 3.5). As far as the feasible set X is concerned, the usual assumption is that, roughly speaking, it is not making the problem significantly more complex than what the unconstrained version would be; details are given in Sect. 3.4.3, but on first reading one may imagine X as defined by a "small" set of explicitly known linear/conic constraints. To simplify the notation, for most of the chapter we will take $X = \mathbb{R}^n$; the modifications required to extend the ideas to the constrained case are, usually, simple enough as to be better introduced separately from the main analysis. We immediately remark, however, that allowing for constraints is important in that it makes it possible to deal with extended-valued f, i.e., $\operatorname{dom} f \subset \mathbb{R}^n$. In the simplest case, if only feasible iterates are produced and, say, $X \subset \operatorname{int} \operatorname{dom} f$, then f is just never evaluated at points $\boldsymbol{x}$ where $f(\boldsymbol{x}) = \infty$. It is actually possible to allow this to happen, but the oracle for f then has to provide appropriate information. In other words, we can view (3.1) as the unconstrained minimization of the *essential objective* $f_X = f + \imath_X$, where $\imath_X$ is the indicator function of X; then, besides an oracle for the finite-valued f, we will need a—necessarily, somewhat different—oracle for the extended-valued $\imath_X$. For most of the chapter f will be therefore intended as finite-valued, with the other case discussed in Sect. 3.4.3.

The basic idea behind all *bundle methods* (BM) is that, being them iterative algorithms, they will construct a sequence $\{\boldsymbol{x}^i\}$ of iterates, hopefully converging towards some optimum $\boldsymbol{x}_*$ of (3.1). The oracle will be called at the points $\boldsymbol{x}^i$, producing the corresponding sequence of pairs $\{(f(\boldsymbol{x}^i), z^i \in \partial f(\boldsymbol{x}^i))\}$. Unlike algorithms for smooth optimization, that can work keeping information about a very restricted set of iterates—possibly even only the last one—BM have to resort to ideally collect and store *all* the previously generated information to work, although *compression* and *selection* procedures can usually be implemented (cf. Sect. 3.3.2). For a number of reasons to become apparent in due time, one customarily replaces $f(\boldsymbol{x}^i)$ with $\alpha^i = \langle z^i, \boldsymbol{x}^i \rangle - f(\boldsymbol{x}^i)$ to define the (lower) *bundle* $\mathcal{B} = \{(z^i, \alpha^i)\}$. Then, the *cutting-plane model*

$$\check{f}_{\mathcal{B}}(\boldsymbol{x}) = \max \left\{ \langle z^b, \boldsymbol{x} \rangle - \alpha^b \ : \ b \in \mathcal{B} \right\} \tag{3.2}$$

(with the useful shorthand "$b \in \mathcal{B}$" for "$(z^b, \alpha^b) \in \mathcal{B}$") is a global *lower model* for f, i.e., $\check{f}_{\mathcal{B}} \leq f$. Upon first reading one may assume $b = i$; however, in general not all pairs in $\mathcal{B}$ are directly related with an iterate, as we shall see, whence the different index. In addition, since $\mathcal{B}$ changes at each iteration it must be denoted as $\mathcal{B}^i$ which, if nothing else, justifies using a different index for its elements; we will try to simplify notation as much as possible by using, e.g., $\check{f}^i$ in place of $\check{f}_{\mathcal{B}^i}$. Note that (3.2) does not use (and hence $\mathcal{B}$ does not need to store) the original iterates $\boldsymbol{x}^i$, which is already a sufficient rationale for introducing the α^b; however, $\check{f}_{\mathcal{B}}$ is not the only possible (lower) model of f, and some of them actually do require storing the iterates (cf. Sect. 3.4.4). It is in general useful to avoid as much as possible to detail

which (lower) model one uses, so that different ones can be employed (cf. Sect. 3.4); we will therefore generically indicate the model as $\underline{f}_{\mathcal{B}}$, although $\underline{f}_{\mathcal{B}} = \check{f}_{\mathcal{B}}$ is by far the most common choice.

With $\underline{f}_{\mathcal{B}}$ at hand, the obvious idea is to directly use it to guide the selection of the new iterate. That is, the iterative scheme

$$\boldsymbol{x}^i \in \operatorname{argmin} \left\{ \underline{f}^i(\boldsymbol{x}) \ : \ \boldsymbol{x} \in X \right\}, \tag{3.3}$$

reminiscent of the most successful algorithms for nonlinear optimization, immediately springs to mind. Of course, the new pair $(\, \boldsymbol{z}^i \,,\, \alpha^i = \langle \boldsymbol{z}^i, \boldsymbol{x}^i \rangle - f(\boldsymbol{x}^i) \,)$ is then added to $\mathcal{B}^i$; on first reading one may assume that no information is ever removed from $\mathcal{B}^i$. With $\underline{f}^i = \check{f}^i$ this is the *cutting-plane method* (CPM) [60], whose attractive feature is that (3.3) can be written as

$$(\, \boldsymbol{x}^i \,,\, v^i \,) \in \operatorname{argmin} \left\{ v \ : \ v \geq \langle \boldsymbol{z}^b, \boldsymbol{x} \rangle - \alpha^b \quad b \in \mathcal{B}^i, \quad \boldsymbol{x} \in X \right\}, \tag{3.4}$$

i.e., an LP if X is a polyhedron and in general a problem that looks "easy enough" to solve, at least if $|\mathcal{B}^i|$ is "not too large". The formulation also highlights how the natural space for the *master problem* (MP) (3.3)/(3.4) is the *epigraphical space* of f, with the extra variable v accounting for f-values (and $v^i = \underline{f}^i(\boldsymbol{x}^i)$). It is not surprising that the CPM is globally convergent, given that any convex function is the supremum of all its affine minorants; the proof, however, is short and instructive enough to be worth reporting.

In the following, we will denote by $\operatorname{lev}_g(v) = x \in \mathbb{R}^n \ : \ g(x) \leq v$ the level set of a generic function g for the level $v \in \mathbb{R}$.

Theorem 3.1 *If the level sets of the initial model $\check{f}^1$ are bounded, then $\{ \boldsymbol{x}^i \}$ in the CPM (weakly) converges to an optimal solution $\boldsymbol{x}_*$ of* (3.1).

Proof As $\mathcal{B}^{i+1} \supseteq \mathcal{B}^i$, $\check{f}^i$ is monotonically nondecreasing in i, hence so are its level sets. Thus, them being bounded for $i = 1$ means they are always so, which makes (3.3) always well defined. Since $\check{f}^i \leq f$, this means that $f_* \geq v^i > -\infty$, and $\{ v^i \}$ is clearly nondecreasing as well. Then, the nonincreasing *record value* $f^i_{rec} = \min\{ f(\boldsymbol{x}^j) \ : \ j = 1, \ldots, i \}$ can be used to define the nonincreasing gap $g^i = f^i_{rec} - v^i \geq 0$. The aim is proving that $g^i \to 0$, which, via $f^i_{rec} \geq f_* \geq v^i$, immediately implies $f^i_{rec} \to f_*$, and therefore that, extracting subsequences if necessary, $\{ \boldsymbol{x}^i \} \to \boldsymbol{x}_*$: in fact, $f^1 \geq f^i_{rec} \geq v^i$, i.e., $\boldsymbol{x}^i \in \operatorname{lev}_{v^i} f^i \subseteq \operatorname{lev}_{v^1} f^1$, hence $\{ \boldsymbol{x}^i \}$ is a bounded sequence. This implies that $\{ \boldsymbol{z}^i \}$ is also bounded, as the image of a compact set under the subdifferential mapping is compact [57, Proposition XI.4.1.2]. Hence, assume $g^i \geq \varepsilon > 0$: for each $j < i$, $f(\boldsymbol{x}^j) \geq f^i_{rec}$ and $\check{f}^i(\boldsymbol{x}^i) \geq f(\boldsymbol{x}^j) + \langle \boldsymbol{z}^j, \boldsymbol{x}^i - \boldsymbol{x}^j \rangle$, which gives $0 > -\varepsilon \geq \langle \boldsymbol{z}^j, \boldsymbol{x}^i - \boldsymbol{x}^j \rangle$. Taking a subsequence if necessary $\| \boldsymbol{x}^i - \boldsymbol{x}^j \| \to 0$; since $\| \boldsymbol{z}^j \|$ is bounded the right-hand side has to converge to zero, yielding the desired contradiction. □

A nice feature of the above proof is that constraints $\boldsymbol{x} \in X$ do not even need to be mentioned; a compact X is actually advantageous, in that compactness of $\operatorname{lev}_{(\cdot)} \check{f}^1$

is clearly no longer required ($\text{lev}_{(\cdot)}(\check{f}^1 + 1_X)$ are surely compact). But for this aspect, even a cursory glance at the proof immediately suggest that the prototypical CPM is fraught with computational issues. First, it requires $\mathcal{B}$ to start "large enough" so that the model $\check{f}_{\mathcal{B}}$ is bounded below and (3.3) is well-defined, which is not trivial unless X is compact. Furthermore, there is no apparent way to control the size of $\mathcal{B}^i$ by removing "outdated" information. Already keeping compactness of the level sets while removing elements from $\mathcal{B}^i$ is nontrivial. Even worse, there seem to be no way to detect whether an iterate $\boldsymbol{x}^i$ belongs or not to the convergent subsequence crucial in the argument. Indeed, it is easy to prove that "apparently reasonable" removals can lead to cycling, as the following example shows.

Example 3.1 Consider Fig. 3.1, where f is the pointwise maximum of the three linear functions (a), (b) and (c), to be minimized over $X = [\, x_a \,,\, x_b \,]$ (compact). With $\mathcal{B}^1 = \{\, (c) \,\}$, assume (3.3) returns $x^1 = x_a$, yielding $\mathcal{B}^2 = \{\, (a) \,,\, (c) \,\}$. Now assume (3.3) returns $x^2 = x_b$, so that (b) is added to $\mathcal{B}^2$. In this moment it would seem harmless to delete (a) from $\mathcal{B}^1$: the linearization has been obtained in x_a, hence "very far" from the current x^2, and it is not active (it does not contribute to defining $\check{f}^2(x^2)$). However, doing so opens the possibility that subsequently $x^3 = x_a$ with (b) being removed from $\mathcal{B}^3$, yielding a cycle. The example may seem to hinge on the fact that the linearization (c) belongs to $\mathcal{B}$ without having been produced by the oracle, and therefore without having produced the corresponding function value which contributes to the record value. This may actually happen (cf. Sect. 3.5), but one may easily extend the example by adding another dimension and having (c) as the intersection of two linearizations, computed (exactly) at different points.

Besides illustrating the difficulty in managing $\mathcal{B}$, the previous example also shows what is perhaps the most damning characteristic of the CPM: the approach is inherently *unstable*, with subsequent iterates possibly "very far" from each other. This is known to cause slow convergence, as clearly illustrated by the following experiment. A problem is solved by the CPM, with arbitrary initial iterate ($\boldsymbol{x}^1 = \mathbf{0}$) and $\mathcal{B}^1 = \emptyset$, and the optimal solution $\boldsymbol{x}_*$ is recorded. Then, the problem is solved again, this time with $\boldsymbol{x}^1 = \boldsymbol{x}_*$ and adding to the MP in (3.3) the constraint $\|\boldsymbol{x} - \boldsymbol{x}_*\|_\infty \leq \delta$ for some δ, but still taking $\mathcal{B}^1 = \emptyset$. The results are reported in Table 3.1, where "r.it." is the ratio between the number of iterations required by the CPM with the added constraint, for the given value of δ, and these of the initial

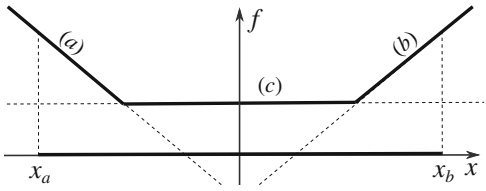

Fig. 3.1 Example of the CPM cycling

Table 3.1 A conceptual experiment illustrating instability of the CPM

δ	10^4	10^2	1.0	10^{-2}	10^{-4}	10^{-5}	10^{-6}
r.it.	1.07	1.12	0.86	0.77	0.56	0.19	0.04

CPM. To avoid unboundedness problems at early iterations, and for extra fairness, the MP of the CPM is actually solved with an extra constraint $\|\boldsymbol{x}\|_\infty \leq 10^4$; since $\|\boldsymbol{x}_*\|_\infty \approx 1$, this does not impact the correctness of the CPM.

Although these results are for a specific instance (the Lagrangian dual of a small-scale randomly generated nonempty and bounded LP), they are quite typical. Knowledge of $\boldsymbol{x}_*$, when only used to choose $\boldsymbol{x}^1 = \boldsymbol{x}_*$, is basically useless: the CPM will not perform significantly less iterations, and may easily do more. Restricting the search in a box around $\boldsymbol{x}_*$ can improve convergence, but only if the box is small enough. With a very small box, improvements of about two orders of magnitude are not unusual. All this starkly contrast with efficient algorithms for smooth optimization; ran with a starting point close to $\boldsymbol{x}_*$, these would converge extremely fast due to the use of second-order models having very good approximations of the curvature information at the optimum. A piecewise linear $\underline{f}_{\mathcal{B}}$ such as $\check{f}_{\mathcal{B}}$ has no inherent curvature information, and therefore has to construct it piecemeal by accruing first-order information in $\mathcal{B}$. It is possible to add "poorman's" second-order information to $\underline{f}_{\mathcal{B}}$ (cf. Sect. 3.4.4), but this has not so far improved performances in general. BM with reliable second-order-type models have been proposed, but they require considerably more sophisticated theory [77–79]; besides, they are implicitly based on the assumption that some second-order information exists that can be extracted, which may not be the case in all applications (f, not only $\underline{f}_{\mathcal{B}}$, can well be polyhedral, cf. Sect. 3.3.3). Therefore, we will concentrate here on the case where $\underline{f}_{\mathcal{B}}$ is an "unstable" model like $\check{f}_{\mathcal{B}}$ can be expected to be, with the corresponding unwelcome consequences: iterates do not have any locality properties and can "swing wildly" across the search space, meaning that often the new pair $(\boldsymbol{z}^i, \alpha^i)$, when added to $\mathcal{B}^i$, conveys little useful information, failing to effectively drive $\boldsymbol{x}^{i+1}$ towards $\boldsymbol{x}_*$. As a consequence, overall convergence of the CPM can be rather slow (although possibly with a surprising twist towards the end, cf. Sect. 3.2.1). This is why the CPM is more or less heavily modified, yielding the large family of (standard) BM described in this chapter.

The structure of the chapter is as follows. In Sect. 3.2 we discuss different forms of stabilization, all using the "primal" view of the problem (3.1), which try to address the issues illustrated above. Each time that a MP is formulated, a dual problem is implicitly defined; making it explicit is often quite useful for understanding the nuances of the approaches and improving their implementation, besides suggesting even different forms of stabilization, as discussed in Sect. 3.3. Section 3.4 presents the other, orthogonal approach that can significantly improve the practical convergence rate of a BM: exploiting specific structures in f to develop specialized models. Finally, since the cost of computing f can be considerable in some applications, another way of improving the practical efficiency of BM is

allowing to perform this computation only approximately, which is discussed in Sect. 3.5. Section 3.6 briefly reviews a number of issues that have not been addressed in the chapter and draws some conclusions.

3.2 Stabilization

The previous discussion has illustrated the need for *stabilizing* the CPM, i.e., ensuring that the iterates do not stray too far from a properly chosen point. However, in general the "right" point—ideally $\boldsymbol{x}_*$—is unknown, and therefore has to be estimated and revised iteratively. Hence, together with the sequence $\{ \boldsymbol{x}^i \}$ of iterates one has to consider the sequence $\{ \bar{\boldsymbol{x}}^i \}$ of *stability centers* which, as we shall see, is actually the one that matters most in terms of convergence properties of the algorithm. It is quite natural (although not strictly necessary [3]) to assume that the stability centers are chosen among the iterates, i.e., $\{ \bar{\boldsymbol{x}}^i \} \subseteq \{ \boldsymbol{x}^i \}$; a convenient consequence is that typically $\underline{f}^i(\bar{\boldsymbol{x}}^i) = f(\bar{\boldsymbol{x}}^i)$. Several different variants of BM correspond to different ways of ensuring that $\boldsymbol{x}^i$ is "near enough" to $\bar{\boldsymbol{x}}^i$. As the example has illustrated, having a "good" $\bar{\boldsymbol{x}}^i$ is not, by itself, enough: one also have to properly estimate "how near" $\boldsymbol{x}^i$ has to be kept. While one can expect the answer "as near as possible" to be correct when $\bar{\boldsymbol{x}}^i = \boldsymbol{x}_*$, in general this is not so, and an excessive stabilization is as detrimental as an insufficient one (cf. Fig. 3.2). Hence, each BM will also have some *stabilization parameters* controlling this aspect, again with a different meaning for each different variant.

3.2.1 Trust-Region Stabilization

A simple approach closely mimics our conceptual example by solving the *stabilized* MP

$$\boldsymbol{x}^i \in \operatorname{argmin} \left\{ \underline{f}^i(\boldsymbol{x}) \ : \ \|\boldsymbol{x} - \bar{\boldsymbol{x}}^i\| \leq \delta^i \right\}, \tag{3.5}$$

where the iterate is kept in a *trust region* (TR) around the current stability center; the (single, as in most cases) stabilization parameter is δ^i, the radius of the TR. Usually the norm in (3.5) is the L_∞ one, because then the natural "explicit form" of (3.5)

$$(\boldsymbol{x}^i \ , \ v^i) \in \operatorname{argmin} \left\{ v \ : \ v \geq \langle \boldsymbol{z}^b, \boldsymbol{x} \rangle - \alpha^b \quad b \in \mathcal{B}^i \quad , \quad \|\boldsymbol{x} - \bar{\boldsymbol{x}}^i\| \leq \delta^i \right\} \tag{3.6}$$

is an LP; this justifies the "BOXSTEP" name originally given to the *trust-region* BM (TRBM) [76], although the exact form of the TR is largely immaterial. Of course, rules to update $\bar{\boldsymbol{x}}^i$ and δ^i need be defined. For the latter, a simple boundedness condition $0 < \underline{\delta} \leq \delta^i \leq \bar{\delta} < \infty$ is sufficient. The former can be done in a natural

way with an Armijo-type condition:

$$f(\boldsymbol{x}^i) \leq f(\bar{\boldsymbol{x}}^i) + m(\underline{f}^i(\boldsymbol{x}^i) - f(\bar{\boldsymbol{x}}^i)) \quad \equiv \quad \Delta f^i \leq m\Delta^i \tag{3.7}$$

where $m \in (0,1)$ is fixed and $\Delta^i = \underline{f}^i(\boldsymbol{x}^i) - f(\bar{\boldsymbol{x}}^i) = v^i - f(\bar{\boldsymbol{x}}^i) < 0$, $\Delta f^i = f(\boldsymbol{x}^i) - f(\bar{\boldsymbol{x}}^i)$ are, respectively, the improvement estimated by the model and the actual one due to moving from $\bar{\boldsymbol{x}}^i$ to $\boldsymbol{x}^i$. If $\Delta^i = 0$, then $\bar{\boldsymbol{x}}^i$ is optimal for (3.1): in fact, $\bar{\boldsymbol{x}}^i$ is then optimal for the MP (although this does not necessarily mean that $\bar{\boldsymbol{x}}^i = \boldsymbol{x}^i$, as the MP can have multiple optimal solutions, cf. Example 3.1). Hence, $\bar{\boldsymbol{x}}^i$, which is in the *interior* of the TR, is also optimal for (3.3) where the TR constraint is removed, which immediately implies the result. As a consequence, $\Delta^i \leq \varepsilon$ is a convenient approximate stopping condition for the method, although one has to be careful that a small δ^i necessarily implies a small Δ^i. Whenever (3.7) holds the, *tentative point* $\boldsymbol{x}^i$ is "substantially better" than $\bar{\boldsymbol{x}}^i$, and one may reasonably set $\bar{\boldsymbol{x}}^{i+1} = \boldsymbol{x}^i$; this is usually called a *serious step* . Leaving the stability center unchanged, i.e., $\bar{\boldsymbol{x}}^{i+1} = \bar{\boldsymbol{x}}^i$, is instead called a *null step*. Clearly, (3.7) ensures that $\{ f(\bar{\boldsymbol{x}}^i) \}$ is a decreasing sequence, and in fact one typically uses $f(\bar{\boldsymbol{x}}^i)$ in place of f^i_{rec} (although the latter may be slightly better). The role of the null step is instead to ensure that $\underline{f}^i$ is improved "in the neighborhood of $\bar{\boldsymbol{x}}^i$", with the aim to ultimately attaining an accurate enough model so as to achieve descent. All in all, the method can be easily proven to be convergent.

Theorem 3.2 *If the level sets of f are bounded, then $\{ f(\bar{\boldsymbol{x}}^i) \} \to f_*$.*

Proof Clearly $\{ \bar{\boldsymbol{x}}^i \} \subset \text{lev}_{f(\bar{\boldsymbol{x}}^1)} f$ and therefore by the boundedness assumption it admits at least an accumulation point $\bar{\boldsymbol{x}}_\infty$; we want to prove that $f_\infty = f(\bar{\boldsymbol{x}}_\infty) = f_*$. The proof is divided into two distinct parts, according to the fact that $\{ \bar{\boldsymbol{x}}^i \}$ is or not a *finite* sequence.

Assume that the sequence is finite: there is a last serious step, after which only null steps are done with the fixed stability center $\bar{\boldsymbol{x}}_\infty$. Then, because $\delta^i \leq \bar{\delta} < \infty$, one is actually applying the CPM to (3.1) with the compact set $X := X \cap \{ \boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x} - \bar{\boldsymbol{x}}_\infty\| \leq \bar{\delta} \}$. Therefore, by Theorem 3.1 (extracting subsequences if necessary) $\{ \boldsymbol{x}^i \} \to \boldsymbol{x}_\delta$, with $\boldsymbol{x}_\delta$ an optimal solution to that problem. If $f(\boldsymbol{x}_\delta) = f(\bar{\boldsymbol{x}}_\infty)$, then $\bar{\boldsymbol{x}}_\infty$ is an optimal solution as well, and reasoning as before therefore an optimal solution of (3.1). Assume by contradiction that $f(\boldsymbol{x}_\delta) - f(\bar{\boldsymbol{x}}_\infty) = \Delta_\infty < 0$ instead. From the proof of Theorem 3.1, $g^i = f^i_{rec} - v^i \to 0$; since $f^i_{rec} \to f(\boldsymbol{x}_\delta)$, $v^i \to f(\boldsymbol{x}_\delta)$ as well. Hence, both $\Delta^i = v^i - f(\bar{\boldsymbol{x}}_\infty) \to \Delta_\infty$ and $f(\boldsymbol{x}^i) - f(\boldsymbol{x}_\infty) \to \Delta_\infty$: since $m < 1$, this contradicts the fact that (3.7) never holds.

Let us now turn to the case where $\{ \bar{\boldsymbol{x}}^i \}$ is an infinite sequence, converging (extracting subsequences if necessary) to $\bar{\boldsymbol{x}}_\infty$. Clearly, (3.7) then implies that $\Delta^i \to 0$. For all i and any fixed optimal solution $\boldsymbol{x}_*$ to (3.1) define $\Gamma^i = f_* - f(\bar{\boldsymbol{x}}^i) \leq 0$, and assume that $\Gamma_\infty = f_* - f(\bar{\boldsymbol{x}}_\infty) < 0$. As $\Gamma^i \leq \Gamma_\infty < 0$, clearly, $\|\bar{\boldsymbol{x}}^i - \boldsymbol{x}_*\| \geq \varepsilon$ for all i and some $\varepsilon > 0$. Define $\boldsymbol{x}^i(\alpha) = \alpha \boldsymbol{x}_* + (1-\alpha)\bar{\boldsymbol{x}}^i$: by convexity, $f(\boldsymbol{x}^i(\alpha)) \leq f(\bar{\boldsymbol{x}}^i) + \alpha\Gamma^i$. Also, let $\bar{\alpha}^i = \max\{\alpha : \|\boldsymbol{x}^i(\alpha) - \bar{\boldsymbol{x}}^i\| \leq \delta^i \}$: since $\delta^i \geq \underline{\delta} > 0$ and $\|\bar{\boldsymbol{x}}^i - \boldsymbol{x}_*\|$ is bounded away from 0, then $\bar{\alpha}^i$ is also bounded away

from 0. But since $\boldsymbol{x}^i(\bar{\alpha}^i)$ is feasible for (3.5), for which $\boldsymbol{x}^i$ is the optimal solution, and $\underline{f}^i \leq f$, one has $\underline{f}^i(\boldsymbol{x}^i) \leq \underline{f}^i(\boldsymbol{x}^i(\bar{\alpha}^i)) \leq f(\boldsymbol{x}^i(\bar{\alpha}^i)) \leq f(\bar{\boldsymbol{x}}^i) + \bar{\alpha}^i \Gamma^i$. Hence, $\Delta^i = \underline{f}^i(\boldsymbol{x}^i) - f(\bar{\boldsymbol{x}}^i) \leq \bar{\alpha}^i \Gamma^i$; the right-hand side is bounded away from zero, contradicting $\Delta^i \to 0$. □

The above proof purposely used direct and elementary arguments and is obtained under unnecessarily strict conditions. For instance, boundedness of the level sets is incompatible with $f_* = -\infty$, which instead happens in applications. Also, one may want more freedom about the size of the TR, say allowing $\delta^i \to 0$ as $\bar{\boldsymbol{x}}^i \to \boldsymbol{x}_*$. These extensions are possible, and the proof can be simplified in the process, using appropriate tools (cf. Sect. 3.3.4). Yet, the proof already clearly illustrates the basic machinery underlying many of BM convergence arguments. In particular, it is subdivided into two almost entirely distinct cases: that of finitely many serious steps, and that of infinitely many ones. In the former case, the algorithm becomes a standard CPM on the restricted feasible region and converges to an optimal solution of this problem: this has to be a global optimum, for otherwise at some point the descent condition (3.7) is triggered. In the latter case, the algorithm (restricted to the serious step subsequence) is a standard descent one, and it has to converge because whenever $\bar{\boldsymbol{x}}^i$ is "far" from $\boldsymbol{x}_*$, the descent Δ^i predicted by the model cannot vanish. This almost complete separation is also apparent from the fact that the two conditions on δ^i are separately required: $0 < \underline{\delta} \leq \delta^i$ is needed to ensure that $\boldsymbol{x}^i - \bar{\boldsymbol{x}}^i$ does not vanish, impairing global convergence of the $\{\bar{\boldsymbol{x}}^i\}$ sequence, whereas $\delta^i \leq \bar{\delta} < \infty$ is needed to ensure that $\{\boldsymbol{x}^i\}$ during a sequence of consecutive null steps actually remains inside a finite TR around the stability center. In some sense the separation is positive: for instance, it tells that one may entirely reset $\mathcal{B}^i$ after any serious step, as accumulation of information is only required to make sequences of consecutive null steps to work (not that this is a good idea in practice, cf. Sect. 3.3.2). However, in general this disconnect makes it harder to prove properties of the method, such as global efficiency estimates.

Of course, practitioners would be more interested in the practical effect of stabilization. An illustration is given in Fig. 3.2 for two specific problems. The figure compares how the distance from the optimal solution and the relative gap evolve during the CPM (INF) and the TRBM with three different (fixed) values of δ (10^3, 10^4, and 10^5).

The plots have several notable features, starting from the rather peculiar behaviour of the CPM. For the vast majority of the iterations, the algorithm seems to be making no progress: many of the first iterates $\boldsymbol{x}^i$ are far *worse* than the initial one $\boldsymbol{x}^1$, and there seems to be little, if any, sign of progress towards an optimum. However, information is indeed accrued during these iterations, and suddenly a tipping point is reached where the convergence behaviour drastically changes, becoming surprisingly quick at the end. Stabilizing may avoid the initial worsening of the iterations; even if it does not (right, $\delta = 10^5$), it typically results in the "quick tail" ensuing sooner. Stronger stabilization may (left) or not (right) result in better performances: a weaker stabilization may result in worse iterates at first, but a faster convergence overall. Indeed, it is clearly possible to over-stabilize

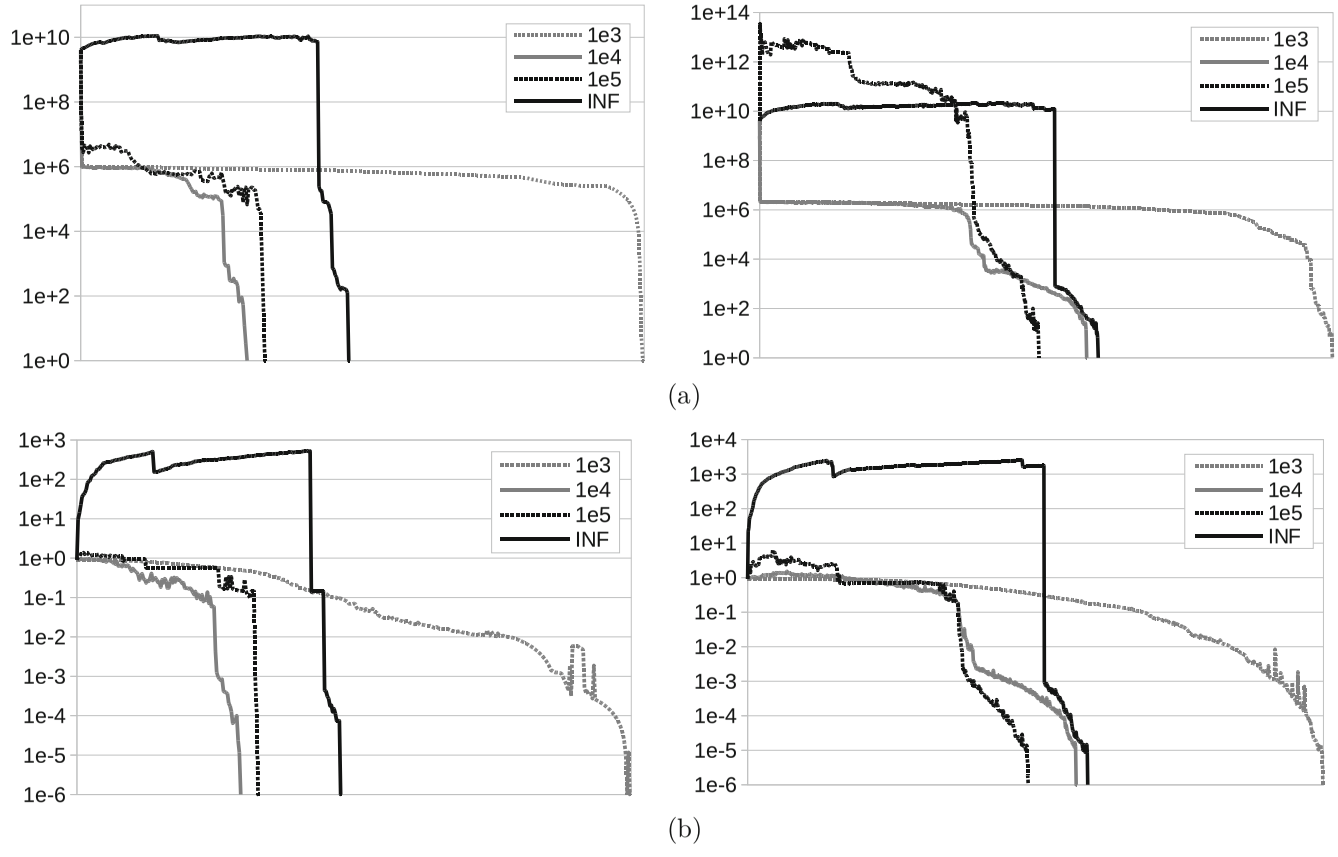

Fig. 3.2 Convergence plots of TRBM with different (fixed) values for δ. (**a**) Evolution of $\|\boldsymbol{x}^i - \boldsymbol{x}_*\|$. (**b**) Evolution of $(f(\boldsymbol{x}^i) - f_*)/f_*$

($\delta = 10^3$): the algorithm has then a much smoother convergence profile, but ultimately requires many more iterations. This is not surprising, in that a small TR intuitively corresponds to the fact that the algorithm behaves basically as a pure descent method (cf. Sect. 3.3.1): excessive stabilization does not allow to exploit the fact that the model $\underline{f}_{\mathcal{B}}$ is "global" instead of "local", and therefore potentially—provided that $\mathcal{B}$ contains enough information—capable of leading the iterate towards the global optimum, which is what happens in the "quick tail". All in all, the plots clearly show that "the right amount of stabilization" can have a positive impact; unfortunately, in general little can be said a-priori about how much stabilization is the right amount. This also depends on which stabilization device is employed, of which the TR is but one.

3.2.2 *Proximal Stabilization*

The *proximal BM* (PBM) replaces the TR with a penalty, as in

$$\boldsymbol{x}^i = \operatorname{argmin}\left\{ \underline{f}^i(\boldsymbol{x}) + \frac{\mu^i}{2}\|\boldsymbol{x} - \bar{\boldsymbol{x}}^i\|_2^2 \right\}, \tag{3.8}$$

where the stabilization parameter is now μ^i. The penalty term also ensures that $\boldsymbol{x}^i$ will not be "too far" from $\bar{\boldsymbol{x}}^i$, although the radius of the TR is only indirectly determined. Indeed, (3.8) could be viewed as the Lagrangian relaxation of (3.5) with respect to the TR constraint if the L_2-norm were used in the latter, and in principle given a δ^i one could always choose μ^i such that the two MP give the same solution, and vice-versa [57, Proposition XV.2.2.3]. The equivalence is only theoretical, since finding the value of μ^i equivalent to a given δ^i (or vice-versa) is not straightforward; not that there would be any reason for wanting to, since finding the "right" values of the two stabilization parameters in practice is roughly equally difficult. It is not entirely surprising, however, that in practice the PBM is sometimes found to be more efficient than the TRBM (e.g., [12, 41]). Indeed, the quadratic penalty term acts as a "poorman's Hessian", adding some (admittedly, very rough) second-order information to the piecewise $\check{f}_{\mathcal{B}}$; an in-depth computational evaluation of the practical behaviour of the PBM can be found e.g. in [15]. However, (3.8) is a QP, which may be more costly than the LP (3.5), potentially negating the advantage due to a faster convergence speed [40]. Yet, this can be partly counterbalanced (or even reversed [41]) by developing specialized QP algorithms that exploit the structure of the MP and its typical usage pattern [34].

An advantage of the stabilizing term is that it makes it easier to answer to an interesting question, i.e., "what would the master problem achieve if the model $\underline{f}_{\mathcal{B}}$ were exact?" That is, consider the *Moreau-Yosida regularization* ϕ_μ of f, perhaps

better written in terms of the *displacement* $\boldsymbol{d}$ from $\bar{\boldsymbol{x}}$:

$$\phi_\mu(\bar{\boldsymbol{x}}) = \min \left\{ f(\bar{\boldsymbol{x}} + \boldsymbol{d}) + \frac{\mu}{2}\|\boldsymbol{d}\|_2^2 \right\}. \tag{3.9}$$

This is an interesting object with useful properties, starting from $\phi_\mu \leq f$ (trivial since $\boldsymbol{d} = \boldsymbol{0}$ is feasible in (3.9)). The unique optimal solution $\boldsymbol{d}_*$ of (3.9) satisfies

$$\boldsymbol{0} \in \partial[\ f(\bar{\boldsymbol{x}}+\cdot)+\frac{\mu}{2}\|\cdot\|_2^2\ ](\boldsymbol{d}_*) \iff -\mu\boldsymbol{d}_* \in \partial f(\boldsymbol{x}) \ \text{ with } \ \boldsymbol{x} = \bar{\boldsymbol{x}}+\boldsymbol{d}_* \tag{3.10}$$

(note that we ignore the dependence of $\boldsymbol{d}_*$ and $\boldsymbol{x}$ on both $\bar{\boldsymbol{x}}$ and μ for notational simplicity); in other words, $\boldsymbol{z}_* = -\mu\boldsymbol{d}_*$ is a (very specific) subgradient at $\boldsymbol{x}$, and $\boldsymbol{x}$ itself is obtained by starting at $\bar{\boldsymbol{x}}$ and moving of a step $1/\mu$ along $-\boldsymbol{z}_*$. Therefore, $\boldsymbol{x}$ might appear to be produced by a subgradient-type approach, were it not that $\boldsymbol{z}_*$ is a subgradient at the *destination* $\boldsymbol{x}$ rather than at the starting point $\bar{\boldsymbol{x}}$. Yet, it turns out that moving from $\bar{\boldsymbol{x}}$ to $\boldsymbol{x}$ actually *is* a step of a *gradient* method: indeed, [57, Corollary XI.3.4.1] shows that ϕ_μ is differentiable, with $\nabla\phi_\mu(\bar{\boldsymbol{x}}) = \boldsymbol{z}_*$ [57, Theorem XV.4.1.4]. Note that this depends on *smoothness* of the stabilizing term rather than, as one may guess, its strong coercivity, i.e., uniqueness of $\boldsymbol{d}_*$. Hence, $\boldsymbol{d}_* = \boldsymbol{0}$ implies that $\bar{\boldsymbol{x}}$ is both a minimum of f and of ϕ_μ: indeed, minimizing ϕ_μ is equivalent to (3.1) [57, Theorem XV.4.1.7], with the obvious advantage that ϕ_μ is smooth. Thus, if (3.9) were efficiently solvable—which it isn't, as computing just one $\boldsymbol{d}_*$ for given $\bar{\boldsymbol{x}}$ and μ is as difficult as solving (3.1)—then one may run a *proximal point algorithm* (PPA), simply obtained by always setting $\bar{\boldsymbol{x}}^{i+1} = \bar{\boldsymbol{x}}^i + \boldsymbol{d}_*^i = \boldsymbol{x}^i$. With only minor requirements on μ^i—it must not to grow too fast, which would be analogous to $\delta^i \to 0$ very fast in the TRBM—and some technical conditions, the PPA can be shown to be a convergent algorithm. We will not go into the details of the convergence proof, which can be found e.g. in [57, Section XV.4.2], besides noting that the fact that one can always take the pre-determined step $1/\mu^i$ in a gradient method and still converge is not surprising considering that $\nabla\phi_\mu$ is Lipschitz continuous with constant μ (cf. again [57, Theorem XV.4.1.4]). Said otherwise, by necessity $\phi_\mu(\boldsymbol{x}) < \phi_\mu(\bar{\boldsymbol{x}})$ (or $\boldsymbol{d}_* = 0$ and the algorithm terminates), hence the step is surely a descent one; the devious trick here is that the step size μ is chosen beforehand, and *the function* ϕ_μ changes to reflect the choice, i.e., in such a way that $\nabla\phi_\mu(\bar{\boldsymbol{x}})$ provides the desired descent. However, all this is only conceptual, in that ϕ_μ is not readily available. What is relevant is rather the interpretation of the PBM in terms of a PPA: basically, if "the model were perfect", i.e., $\underline{f}_{\mathcal{B}} = f$, then each iteration would result in a serious step. In other words, the PBM can be seen as an approximated—but implementable, as opposed to conceptual—variant of the PPA, where sequences of consecutive null steps aim at computing $\nabla\phi_\mu(\bar{\boldsymbol{x}})$ "accurately enough", so that finally a serious step can be performed. This ties in well with the standard structure of convergence proofs, where sequences of consecutive null steps and the sequence of serious steps are analysed separately.

Besides being aesthetically pleasing, these results are also the basis of practical algorithmic developments, comprised some related to the real crux of the (P)BM,

which is appropriately (and dynamically) choosing the stabilization parameter. These are based on the idea that (3.9) could be generalized to

$$\phi_H(\bar{\boldsymbol{x}}) = \min \left\{ f(\bar{\boldsymbol{x}} + \boldsymbol{d}) + \frac{1}{2}\boldsymbol{d}^T H \boldsymbol{d} \right\} \tag{3.11}$$

depending on a whole matrix parameter $H \succ 0$, the standard Moreau–Yosida regularization then being just the special case for $H = \mu I$. Clearly it would be attractive to have H providing a better depiction of the second-order behaviour of f at $\bar{\boldsymbol{x}}$ that what the "poorman's matrix" μI can do. Of course, the Hessian cannot be used, but one may nonetheless consider quasi-Newton formulae. Say, with $\bar{\boldsymbol{z}}^i \in \partial f(\bar{\boldsymbol{x}}^i)$ and $\boldsymbol{z}^i \in \partial f(\boldsymbol{x}^i)$, it would be natural to select H_{i+1} so that the standard *quasi-Newton equation*

$$H_{i+1} \Delta \boldsymbol{x}^i = \Delta \boldsymbol{z}^i \tag{3.12}$$

is satisfied with $\Delta \boldsymbol{x}^i = \boldsymbol{x}^i - \bar{\boldsymbol{x}}^i$ and $\Delta \boldsymbol{z}^i = \boldsymbol{z}^i - \bar{\boldsymbol{z}}^i$, which is what one would do if f were differentiable; which it isn't, and this entirely breaks the theory upon which (3.12) relies. Yet, $\phi^i = \phi_{H^i}$ is differentiable, and therefore (3.12) would make sense with $\Delta \boldsymbol{z}^i = \nabla \phi^i(\boldsymbol{x}^i) - \nabla \phi^i(\bar{\boldsymbol{x}}^i)$, were it not for the fact that exactly computing gradients of ϕ^i requires solving a problem as difficult as the original (3.1). Here, however, one can cleverly exploit (3.10), immediately generalized as $\nabla \phi_H(\bar{\boldsymbol{x}}) = -H\boldsymbol{d}_* \in \partial f(\boldsymbol{x})$, to *find* $\boldsymbol{d}_*$—and therefore $\bar{\boldsymbol{x}}$—*given* $\boldsymbol{z} \in \partial f(\boldsymbol{x})$. Indeed, $-H\boldsymbol{d}_* = \boldsymbol{z}$ gives $(\bar{\boldsymbol{x}} - \boldsymbol{x}) = H^{-1}\boldsymbol{z}$, i.e., $\boldsymbol{z} = \nabla \phi_H(\boldsymbol{x} + H^{-1}\boldsymbol{z})$. In plain words, once a subgradient of f is known at any point $\boldsymbol{x}$, one can easily compute the point $\bar{\boldsymbol{x}}_H(\boldsymbol{z})$ such that $\boldsymbol{z} = \nabla \phi_H(\bar{\boldsymbol{x}}_H(\boldsymbol{z}))$. This *reversal operation* [70] suggests to use (3.12) indeed with $\Delta \boldsymbol{z}^i = \boldsymbol{z}^i - \bar{\boldsymbol{z}}^i = \nabla \phi^i(\bar{\boldsymbol{x}}^i(\boldsymbol{z}^i)) - \nabla \phi^i(\bar{\boldsymbol{x}}^i(\bar{\boldsymbol{z}}^i))$, but also with $\Delta \boldsymbol{x}^i = \bar{\boldsymbol{x}}^i(\boldsymbol{z}^i) - \bar{\boldsymbol{x}}^i(\bar{\boldsymbol{z}}^i)$ (with the obvious notation). This may give rise to various quasi-Newton approaches depending on the way in which (3.12) is approached, say with typical rank-one updates. We will not delve in these details, to be found in [70] and references therein, except for the specific case where H is constrained to have the "poorman's" form μI. This means that there is no hope that (3.12) be satisfied except in a least-squares sense, which yields

$$\mu_{i+1} = \|\Delta \boldsymbol{z}^i\|_2^2 \,/\, \langle \Delta \boldsymbol{z}^i, \Delta \boldsymbol{x}^i \rangle. \tag{3.13}$$

Of course, for (3.13) to make sense one must ensure that $\langle \Delta \boldsymbol{z}^i, \Delta \boldsymbol{x}^i \rangle > 0$, which can be done by an appropriate *curved search*, i.e., solving the MP with iteratively changing μ until the condition is attained [70]; this a natural enough approach for BM, already proposed e.g. in [88]. Under rather strong conditions (f differentiable or strongly convex), fast convergence (respectively, superlinear or two-step-superlinear) can be proven. Perhaps more importantly, the approach seems to improve practical performances with respect to other proposed strategies [61]. Also, the idea can be extended to making use of other available information for even better managing μ [85].

Yet, this does not imply that effective μ-management is completely understood. Formula (3.13) requires a specific care to ensure that $\langle \Delta z^i, \Delta x^i \rangle > 0$, and the theory is developed under the assumption that μ is only updated at serious step, whereas intuitively being able to increase μ after a few unsuccessful null steps could also be useful. Furthermore, all these approaches [61, 70, 85] are based on "local" behaviour of f, i.e., they do not explicitly depend on how "far" $\bar{x}^i$ is from x_* ($f(\bar{x}^i)$ from f_*), which may lead to sequences of "short" steps that slow down convergence (cf. $\delta = 10^3$ in Fig. 3.2). Although more "global" strategies can be devised [38], other stabilization approaches seem to be inherently better suited in this respect, as discussed next.

3.2.3 Level Stabilization

The idea of *level stabilization* is in some sense opposite to that of the previous approaches. In general, the issue is that $\underline{f}_{\mathcal{B}}$ is "too optimistic" a model of f, in that it dramatically underestimates the true value of f in a large part of the space. This lures the MP to points x^i such that $\underline{f}_{\mathcal{B}}(x^i) \ll f(\bar{x})$, often "unreasonably so", while $f(x^i) \gg f(\bar{x})$. The TRBM tries to set the TR in such a way as to exclude the points where $\underline{f}_{\mathcal{B}}(x) \ll f(\bar{x})$, while the PBM tries to limit their appeal by penalizing them on the basis of the distance from $\bar{x}$. In these cases, the amount of descent that the model will estimate for the next iteration, as measured by $\Delta^i = \underline{f}^i(x^i) - f(\bar{x}^i) < 0$, is a complex function of the stabilization parameters (δ^i and μ^i). A different approach is to fix beforehand how much descent the model should attain, which clearly has an intuitive appeal in the context of a descent method, i.e., to work in the level set $\mathrm{lev}_l \underline{f}_{\mathcal{B}}$ for some given *level parameter* $l < f(\bar{x})$. Such a set, however, may well be "large" (even unbounded), and therefore there has to be some way pick a specific point in there. In the spirit of BM, the intuitive idea is just that of keeping "close" to the stability center, which leads to the MP

$$x^i = \operatorname{argmin} \left\{ \|x - \bar{x}^i\| \,:\, \underline{f}^i(x) \le l^i \right\}. \tag{3.14}$$

An advantage of the resulting *proximal level BM* (PLBM) approach is that the stabilization parameter, l^i, has the scale of function values, which may make it easier to choose. For instance, if the optimal value f_* is *known*, then obviously l^i has to belong to the interval $[\, f(\bar{x}^i)\,,\ f_*\,]$ (actually, $[\, f^i_{rec}\,,\ f_*\,]$). The simple strategy of fixing any $\lambda \in (0, 1]$ and choosing $l^i = \lambda f(\bar{x}^i) + (1 - \lambda) f_*$ then works *even with very relaxed assumptions on the choice of* $\bar{x}^i$, such as by always doing serious step ($\bar{x}^{i+1} = x^i$) even if (3.7) does not hold, and even keeping $\bar{x}^i$ (possibly, $\notin X$) fixed [72]. The proof is somewhat technical and is not repeated here; what is relevant is that knowledge of f_* is not really required, as it can be replaced by its lower bound v^i obtained by solving the original unstabilized MP (3.3) (assumed finite). Solving the MP of the CPM but *not* directly using its optimal solution as the next

iterate is an interesting algorithmic concept, of which we will see other applications (cf. Sect. 3.2.5); here, it is rather the optimal value v^i that is used to compute the value of l^i, after which (3.14) provides $\boldsymbol{x}^i$. Of course, the disadvantage is having to solve two (related but different) MPs, which is not appealing in the case where they are rather costly (e.g., [97]). However, in some applications the cost of computing f far outweighs the MP cost, and therefore this approach may be competitive in that it provides a clear and principled way to choose the stabilization parameter, as opposed to the heuristic ones common for the TRBM and PBM, possibly resulting in better practical convergence.

In case one is not willing to compute v^i, or unable to do so (say, because (3.3) is unbounded below), the alternative is to choose l^i arbitrarily. The possible troubling consequence is that (3.14) may be empty, but this is actually not an issue; since $\underline{f}^i \leq f$, this means that $l^i < f_*$. Hence, if this happens the algorithm has found a provably correct lower bound on f_*, which can then be used in place of f_*/v^i to set the next target; clearly, then $l^{i+1} > l^i$, hopefully making (3.14) feasible. This is one of the specific traits of the PLBM, i.e., that it can provide valid lower approximations to f_*; in some cases this can be helpful. Actually, also TRBM and PBM may do this, since their next iterate $\boldsymbol{x}^i$ may in fact coincide with the optimal solution of (3.3), which is easy to detect; for instance, for TRBM this happens if $\boldsymbol{x}^i$ is in the interior of the TR. However, in these methods the occurrence is incidental and does not impact on the algorithm, while in the PLBM it is a crucial aspect. Hence, together with null step and serious step, the convergence analysis for the PLBM has to cater for these *level steps*; yet, this is easy. In fact, if, say, $l^{i+1} = \lambda f(\bar{\boldsymbol{x}}^i) + (1-\lambda)l^i$ whenever a level step happens, infinitely many level steps result in $l^i \to f(\bar{\boldsymbol{x}}^i)$, which means that $f(\bar{\boldsymbol{x}}^i) \to f_*$. Once this case is dealt with, the remaining analysis is analogous to the case where f_*/v^i are available, and not dissimilar from those of the TRBM and PBM.

Indeed, an interesting recent development is the *doubly stabilized BM* (DSBM) of [25], which has *both* proximal and level stabilization, i.e., MP

$$\boldsymbol{x}^i = \operatorname{argmin}\left\{ \underline{f}^i(\boldsymbol{x}) + \mu^i \|\boldsymbol{x} - \bar{\boldsymbol{x}}^i\|_2^2 \;:\; \underline{f}^i(\boldsymbol{x}) \leq l^i \right\}. \tag{3.15}$$

Two stabilization parameters are not necessarily more difficult to tune than one; actually, the converse may happen. Indeed, at any given iteration one among μ^i and l^i is "irrelevant": the obtained $\boldsymbol{x}^i$ is either that of (3.8) (a *proximal iteration*), or that of (3.14) (a *level iteration*), and it is easy (cf. (3.34)) to tell which of the two it is. Hence, the somewhat "more principled" level parameter l^i, which can exploit information about f_*, can be used to select the desired amount of descent, while μ^i can be used to select a "good" $\boldsymbol{x}^i$ in the (possibly, large) set $\operatorname{lev}_{l^i} \underline{f}_B$; the results of [25] are encouraging. Convergence theory is hardly much different from that of PBM: once the case of infinitely many level steps is ruled out, the algorithm is (almost, barring some fine details) exactly a PBM.

One may, however, argue that there is actually no need for the level stabilization in order to tune μ^i exploiting information about f_*. Firstly, any known guaranteed

lower bound $l \leq f_*$—such as v^i from (3.3), or directly obtained by the problem, cf. Sect. 3.3—can be directly incorporated into $\underline{f}_{\mathcal{B}}$ under the form of the "flat" linearization $(\mathbf{0}, l) \in \mathcal{B}$. This incurs in hardly any MP cost and it means that $\boldsymbol{x}^i$ will automatically exploit this information, which is indeed useful in practice; for instance, surely $v^i \geq l$. Furthermore, one might design μ-updating strategies that take into account this information and, say, try to ensure that $\underline{f}^i(\boldsymbol{x}^i) \leq l^i = \lambda f(\bar{\boldsymbol{x}}^i) + (1-\lambda) l$ exactly as in the DSBM. Somewhat different, but related, strategies can use information about the fact that $\boldsymbol{x}^i$ is, or not, "close" to a minimizer of $\underline{f}^i$ to properly increase or decrease μ^i (cf. e.g. [35]). Current consensus is that the l-updating strategies of PLBM are more robust, in particular in the constrained case ($X \neq \mathbb{R}^n$), while the PBM may be more efficient, especially in the unconstrained case; thus, the DSBM makes sense, as would any μ-updating strategy "simulating" it. All this highlights how proper tuning of the proximal parameters is still quite an open issue, and an area of active research. This also justifies why there is, among practitioners, a latent distrust of stabilization techniques, partly justifying the development of the alternative approaches of Sect. 3.2.5.

3.2.4 Center-Based Approaches

Another class of BM are based on the idea that, instead of aiming for the "extreme" point $\boldsymbol{x}^i$ minimizing $\underline{f}_{\mathcal{B}}$, one should target the "center" of a *localization set* $\mathcal{L}(g, l) = \{(\boldsymbol{x}, v) : g(\boldsymbol{x}) \leq v \leq l\} \subset \mathbb{R}^{n+1}$, which is the epigraphical version of the level set, for appropriately chosen g and l. For instance, the polyhedron

$$\mathcal{L}^i = \mathcal{L}(\check{f}^i, f^i_{rec}) = \left\{ (\boldsymbol{x}, v) : \langle \boldsymbol{z}^b, \boldsymbol{x} \rangle - \alpha^b \leq v \leq f^i_{rec} \;\; b \in \mathcal{B}^i \right\}$$

(cf. (3.4)) is clearly the best possible outer approximation—with the known data—of the epigraphical extension of the set of the optimal solutions $\mathcal{L}_* = \mathcal{L}(f, f_*) = \{(\boldsymbol{x}, v) : f(\boldsymbol{x}) = v = f_*\}$; it is defined by the linearizations in $\mathcal{B}^i$, plus the *hat cut* $v \leq f^i_{rec}$. This is obviously related with level-based BM, whose MP has feasible set is $\mathcal{L}(\underline{f}^i, l^i)$ (identical but for the hat cut). Each time the oracle is called at some $\boldsymbol{x}$ in (the projection of) $\mathcal{L}^i$, the generated information can be used to *cut away* some part of $\mathcal{L}^i$, obtaining a smaller $\mathcal{L}^{i+1}$. Indeed, if $f(\boldsymbol{x}) > \underline{f}^i(\boldsymbol{x})$ then the corresponding new linearization will at least cut away the point $(\boldsymbol{x}, \underline{f}^i(\boldsymbol{x})) \in \mathcal{L}^i$, while if $f(\boldsymbol{x}) < f^i_{rec}$ then the hat cut will be lowered; barring blatantly obvious bad choices of $\boldsymbol{x}$, at least one of the conditions must happen, and both potentially can. The idea is then to select $\boldsymbol{x}$ so that as "much as possible" of $\mathcal{L}^i$ is cut away at each iteration; intuitively, this corresponds to choosing $\boldsymbol{x}^i$ in "the center" of $\mathcal{L}^i$. Among the possible definitions of center of a polyhedron, a widely used one is the *analytic*

center (AC): the minimum of the *logarithmic barrier function*

$$(\boldsymbol{x}^i\ ,\ v^i\ ) = \operatorname{argmin}\left\{ -\log(\ f^i_{rec} - v\ ) - \sum_{b\in\mathcal{B}^i} \log(\ v - \langle \boldsymbol{z}^b, \boldsymbol{x}\rangle - \alpha^b\ ) \right\} \tag{3.16}$$

upon which *interior point* (IP) methods are based. It can be alternatively defined as the point $(\boldsymbol{x}^i\ ,\ v^i\ ) \in \mathcal{L}^i$ that *maximizes the product of the slacks* of the constraints; using it as the next iterate gives the *analytic center CPM* (ACCPM). Clearly, this means that $\mathcal{L}^i$ must have a nonempty interior and must be bounded; the latter is in general nontrivial, exactly as in the CPM. Due to the relationships with IP methods, the (approximate) computation of the AC can be performed by means of extremely well-understood and efficient methods. Also, the known methods can be adapted to efficiently update $(\boldsymbol{x}^i\ ,\ v^i\ )$ to $(\boldsymbol{x}^{i+1}\ ,\ v^{i+1}\ )$ when a new linearization enters $\mathcal{B}^i$ and/or the hat cut changes, a nontrivial feat because the former is typically no longer feasible, even less interior [49]. The upshot is that the ACCPM has favourable worst-case complexity estimates, and usually a regular convergence profile.

As other methods explicitly constructed to optimize the worst-case, however, the ACCPM is not always very fast in practice. One issue is that, as discussed in Sect. 3.2.1, when enough information has been accrued $\underline{f}^i$ can be quite accurate a model (especially if f itself is polyhedral), and therefore its optimum can be a promising point where to call the oracle; the ACCPM not using it may lead to missing out on the "fast tail" of the CPM. There are also some specific issues due to the fact that the AC of a polyhedron depends from its *algebraic representation* rather than from its true geometry. For instance, if a linearization $(\boldsymbol{z}^b\ ,\ \alpha^b\ )$ is generated multiple times (which happens in applications), this skews the AC to be "farther" from that. Conversely, the hat cut $v \le f^i_{rec}$ is the only inequality limiting v from above; as $|\mathcal{B}^i|$ grows the influence of the many cuts "pushing up v from below" may overwhelm that of the hat cut, which therefore tends to become almost active. Both cases may slow down the convergence, which is based on keeping $(\boldsymbol{x}^i\ ,\ v^i\ )$ firmly in the interior of $\mathcal{L}^i$, but specific adaptations can be devised to counter these effects [30]. Also, the issue of compactness of $\mathcal{L}^i$ can be faced by the *proximal ACCPM* (PACCPM), a "doubly stabilized" version [4] where a standard proximal term a-la (3.8) (thus introducing a proximal center $\bar{\boldsymbol{x}}$ and a proximal parameter μ) is added to (3.16), which is claimed to further improve the performances of the approach. Anyway, the ACCPM has not been widely adopted; this is likely due, above and beyond any other reason, to the need of specific sophisticated implementations for efficiently solving (3.16), which cannot therefore benefit from the regular advances of general-purpose LP/QP solvers.

It is possible to avoid the need of specialized approaches to solve the MP by using the *Chebychev center* (CC) instead, i.e., the center of the largest ball inside $\mathcal{L}^i$. For a generic polyhedron $\mathcal{P} = \{\ \boldsymbol{y} \in \mathbb{R}^k\ :\ \langle \boldsymbol{a}_h, \boldsymbol{y}\rangle \le b_h,\quad h \in H\ \}$, the CC is

the optimal solution of the LP

$$(\boldsymbol{y}, \sigma) = \operatorname{argmax}\{\sigma : \langle \boldsymbol{a}_h, \boldsymbol{y}\rangle + \|\boldsymbol{a}_h\|\sigma \leq b_h \; h \in H\} \tag{3.17}$$

(assuming it has any, which obviously requires $\mathcal{P}$ to be compact). When applied to $\mathcal{L}^i$, $\boldsymbol{y} = (\boldsymbol{x}, v)$ ($\mathbb{R}^k = \mathbb{R}^{n+1}$) and the hat cut $v \leq f^i_{rec}$ gives rise to a constraint of the form $v + \sigma \leq f^i_{rec}$, which is necessarily active in the optimal solution [81, Proposition 2.1]; this allows to substitute away v for $f^i_{rec} - \sigma$, which together with $v = -\sigma$ yields

$$(\boldsymbol{x}^i, v^i) \in \operatorname{argmin}\left\{ v : v \geq \frac{\langle \boldsymbol{z}^b, \boldsymbol{x}\rangle - \alpha^b - f^i_{rec}}{1 + \sqrt{\|\boldsymbol{z}^b\|^2 + 1}} \;\; b \in \mathcal{B}^i \right\}. \tag{3.18}$$

The notation is chosen to highlight the similarity with the MP (3.4) of the CPM: besides translating the right-hand side by f^i_{rec} (which is routinely done, cf. (3.21)), each constraint is just scaled by a factor depending only on $\|\boldsymbol{z}^b\|$. Using $\boldsymbol{x}^i$ from (3.18), which already can have a stabilization effect as in the ACCPM, gives the *Chebychev center CPM* (C^{3}PM) [58]. The modern take to the approach [81] views (3.18) as the finitely sampled version of the *Elzinga-Moore-Ouorou function*

$$\Psi(l) = \inf\left\{ v : v \geq \frac{\langle \boldsymbol{z}, \boldsymbol{x} - \boldsymbol{y}\rangle + f(\boldsymbol{y}) - l}{1 + \sqrt{\|\boldsymbol{z}\|^2 + 1}}, \;\; \boldsymbol{y} \in \mathbb{R}^n, \; \boldsymbol{z} \in \partial f(\boldsymbol{y}) \right\}. \tag{3.19}$$

Note that in (3.19) $\boldsymbol{x}$ and v are the variables upon which the minimization is performed (as in (3.18)), whereas $\boldsymbol{y}$ and $\boldsymbol{z}$ serve to index the infinitely many linear constraints. The function $\Psi(l)$ gives the negative of the radius of the largest sphere inscribed into $\mathcal{L}(f, l)$, and therefore is a *merit function* for (3.1): $\Psi(l) \leq 0$, and $\Psi(l) = 0$ if and only if $l = f_*$. Therefore, (3.1) is equivalent to finding l such that $\Psi(l) = 0$. One can then make Ψ a multivariate function by just setting $\Psi(\boldsymbol{x}) = \Psi(f(\boldsymbol{x}))$; this could be seen as awkward, were it not that it makes it possible to add a proximal term, yielding the MP

$$\min\left\{ v + \frac{\mu^i}{2}\|\boldsymbol{x} - \bar{\boldsymbol{x}}^i\|_2^2 : v \geq \frac{\langle \boldsymbol{z}^b, \boldsymbol{x}\rangle - \alpha^b - f(\bar{\boldsymbol{x}}^i)}{1 + \sqrt{\|\boldsymbol{z}^b\|^2 + 1}} \;\; b \in \mathcal{B}^i \right\}, \tag{3.20}$$

which has the usual advantage to have a finite solution even if $\mathcal{L}(\underline{f}^i, f(\bar{\boldsymbol{x}}^i))$ is unbounded. Clearly, this is for the C^{3}PM what the PBM is for the CPM; in other words, (3.20) with an "infinitely large" $\mathcal{B}^i$ a-la (3.19) defines the *Elzinga-Moore-Moreau-Yosida regularization*, which is to $\Psi(\boldsymbol{x})$ what the Moreau-Yosida regularization ϕ_μ (cf. (3.9)) is to f. Such a function has minima where $\Psi(\boldsymbol{x}) = 0$, and therefore minimizing it is equivalent to (3.1) [81]. It is not surprising, then, that the *proximal* C^{3}PM (PC^{3}PM) algorithm that minimizes it is very close to the PBM, down to the fine details of the solution of the MP (which is indeed identical save for the scaling factor $1 + \sqrt{\|\boldsymbol{z}^b\|^2 + 1}$), and whose convergence analysis proceeds

in the same way. Interestingly, a variant of the approach [81, Section 5] exists where a second LP is solved to compute the current maximum radius of the sphere, and this is used for tuning the proximal parameter μ^i, analogously to how the PLBM solves (3.3) to tune l^i. This is not incidental: that *target radius method* can be interpreted as a PLBM with a specific rule to define l^i [23].

Hence, both centers-based approaches, like the PLBM, benefit from adding a second proximal stabilizing device. While double stabilization has been reported to be superior to the (singly-stabilized) PBM in some cases [25, 81], this is not yet firmly established for all relevant applications.

3.2.5 Approximate CPM Approaches

All previous stabilization approaches are based on modifying the MP of the CPM, although in some cases that is *also* solved to help tuning the stabilization parameter(s). Another take is to keep the MP unchanged, but deal with its solution differently.

A first idea is solving (3.3) only *approximately*, a simple way of doing this being to employ a *subgradient-type* method (a "poorman's version" of the PBM, cf. Sect. 3.3.1), whose slow convergence and lack of effective stopping criteria mean that it is typically ran with a fixed number of iterations, reaching only an approximately optimal solution. This is actually natural enough when (3.1) itself is the dual of the problem one is actually interested in solving; as this is discussed in Sect. 3.3.3 we refrain from further delving into the subject now, pointing e.g. to [90] for details. Perhaps more interesting is the recent resurgence of ACCPM-type methods under the moniker of "*primal-dual column generation technique*" (PDCGT) [51]. The idea is again that of stopping "way before" the optimal solution of (3.3) is achieved, except doing this with an IP approach rather than with a subgradient-type one. This exploits the fact that IP methods approximately follow the *central path* of the polyhedron, which starts from the AC—exactly $\boldsymbol{x}^i$ of (3.16), if the hat cut is included in the formulation—and goes to the $\boldsymbol{x}^i$ of (3.3). Hence, by construction they produce a sequence of "well centred" iterates in $\mathcal{L}^i$, except in the v-dimension (that is minimized); thus, by stopping the IP method early on—which is also convenient computationally, as IP iterations are costly—one can obtain well-centred solution "in between" the AC and the CPM iterate. Since (feasible) primal-dual IP methods (unlike, say, subgradient-type methods) allow to measure the quality of the current iterate, the stabilization parameter can just be the gap ε below which the MP computation is terminated. A large ε produces iterates close to the AC, while a small ε produces iterates close to that of the standard CPM, which can be beneficial in the "fast tail" of the CPM when $\mathcal{B}^i$ is a "good" model. As in the ACCPM, however, for efficiency reasons nontrivial warm-starting strategies are needed each time the IP method is re-started after a new linearization is included in $\mathcal{B}^i$ [50].

The PDCGT tracks the iterative solution, via an IP method, of the MP from the AC of $\mathcal{L}^i$ to the CPM iterate $\boldsymbol{x}^i$ of (3.3), and "stops somewhere in the middle". The *in-out approach* (IOA) [11] takes a similar stance in a simpler way, using the previous iterate—which doubles as a stability center $\bar{\boldsymbol{x}}^i$—in lieu of the AC. That is, the optimal solution $\boldsymbol{x}^i$ of (3.3) is obtained (which does not depend on $\bar{\boldsymbol{x}}^i$), and then f is computed at $\bar{\boldsymbol{x}}^{i+1} = (1-\lambda^i)\bar{\boldsymbol{x}}^i + \lambda^i \boldsymbol{x}^i$ for some $\lambda^i \in (0, 1]$. The "in-out" moniker derives from the fact that $(\bar{\boldsymbol{x}}^i, f(\bar{\boldsymbol{x}}^i))$ is *inside* epi f, whereas $(\boldsymbol{x}^i, v^i)$ belongs to $\mathcal{L}^i$ which is an outer approximation, and therefore it is very likely *outside* epi f (if not, the algorithm terminates). By taking only a partial step towards $\boldsymbol{x}^i$ from $\bar{\boldsymbol{x}}^i$, the IOA tries to remain inside the epigraph. If this actually happens, then $f(\bar{\boldsymbol{x}}^{i+1}) \leq f(\bar{\boldsymbol{x}}^i) + \lambda^i \Delta^i$, where as usual $\Delta^i = v^i - f(\bar{\boldsymbol{x}}^i) < 0$; hence, a significant decrease of f is attained, a-la (3.7). Otherwise, the newly obtained linearization $(\boldsymbol{z}^i, \alpha^i)$ cuts away a part of $\mathcal{L}^i$. Clearly, the CPM is the special case where $\lambda^i = 1$ uniformly, and therefore convergence can be proven similarly to the CPM whenever λ^i does not become too small; the simple condition $\lambda^i \geq \underline{\lambda} > 0$ is used in [11]. As most other stabilization approaches, the IOA requires ways to dynamically tune the stabilization parameter λ^i. The recent large computational study in [83] deals with these aspects and proposes further variants where the next iterate is chosen along the *deflected* direction $(\boldsymbol{x}^i - \bar{\boldsymbol{x}}^i) + \beta^i \boldsymbol{z}^i$, where $\boldsymbol{z}^i$ is the subgradient at $\bar{\boldsymbol{x}}^i$. The IOA method is shown to be competitive with ones using piecewise linear penalty terms/trust regions (cf. Sect. 3.3.4).

It is worth remarking that the IOA is also related with the version of the PLBM where the MP of the CPM is solved, *prior* to (3.14), to compute the lower bound v^i out of which l^i is obtained. Indeed, the PLBM would obtain the same $\boldsymbol{x}^i$ as the IOA if it was using the (*upper*, cf. 3.5.2) model $\underline{f}^i$ such that $\underline{f}^i(\boldsymbol{x}) = (1-\lambda) f(\bar{\boldsymbol{x}}^i) + \lambda v^i$ if $\boldsymbol{x} = (1-\lambda)\bar{\boldsymbol{x}}^i + \lambda \boldsymbol{x}^i$, and $\underline{f}^i(\boldsymbol{x}) = \infty$ otherwise, instead as the cutting-plane one, in (3.14). To the best of our knowledge, this connection has never been explicitly made before.

A significant perceived benefit of both the PDCGT and the IOA for practitioners is that there is no need to modify the MP of the CPM; this may (or may not) also make them more efficient, since, say, an LP is solved instead of a QP like (3.8)/(3.14), and the LP does not have the extra bounds of (3.5). Of course, the approaches also share the issue of the CPM of requiring (3.3) to have a solution in the first place (e.g., X compact). Hybridizing them with a proximal/trust region approach, a-la [4], could solve this issue, but would do away with the benefit of working with an unsullied MP. Yet, in particular for the IP method used by PDCGT, the addition of a simple quadratic term in the objective function may not make it any significantly more difficult to solve, and conceivably even less so (cf. e.g. [17]). To the best of our knowledge, this has not been tested yet.

While the above recount summarizes many of the (simple) BM approaches in the literature, the discussion is purposely limited to the "primal" description of the problem. In many relevant applications (and, in fact, in general) the "dual" aspect is as much, if not more, important. Indeed, (3.1) itself can be a dual problem, whose aim is to help solving a primal one. Furthermore, the dual description is also useful

to understand and implement the approaches themselves; this is the subject of the next section.

3.3 Duality

As everywhere in convex analysis, duality is inescapable: even if one were trying to purposely avoid it, as we did in the previous section, it would still be there. In our case, this starts from the fact that every BM solves one (or more) MP, which is a convex program and therefore it has a dual. Most often, MPs are LPs or QPs, and therefore their duals are also straightforward to compute. Doing so is actually beneficial, both because the dual may be simpler to solve, and because it reveals details of the method that can be important to understand and improve it. Also, in some applications (3.1) is itself the dual of the problem one is actually interested to solve, and therefore the dual of the MP (and of (3.1)) is related to it. This section is devoted to discussing all these issues and their main conceptual and algorithmic consequences.

3.3.1 Dual Forms of the Master Problem

For discussing the dual forms of the MP, it is useful to introduce the *translated* model $\underline{f}_{\mathcal{B},\boldsymbol{x}}(\boldsymbol{d}) = \underline{f}_{\mathcal{B}}(\boldsymbol{x}+\boldsymbol{d}) - f(\boldsymbol{x})$ with respect to a point $\boldsymbol{x}$ (typically, the stability center $\bar{\boldsymbol{x}}$). This is a model of the *translated function* $f_{\bar{\boldsymbol{x}}}(\boldsymbol{d}) = f(\bar{\boldsymbol{x}}+\boldsymbol{d}) - f(\bar{\boldsymbol{x}})$ such that $f_{\bar{\boldsymbol{x}}}(\mathbf{0}) = 0$, with $\underline{f}_{\mathcal{B},\bar{\boldsymbol{x}}}(\mathbf{0}) = 0$ if any pair having $\boldsymbol{x}_b = \bar{\boldsymbol{x}}$ belongs to $\mathcal{B}$, as it usually (but not always) happens. A *displacement* (cf. (3.9)) $\boldsymbol{d}$ such as $\underline{f}_{\mathcal{B},\bar{\boldsymbol{x}}}(\boldsymbol{d}) < 0$ indicates a point $\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{d}$ where $\underline{f}_{\mathcal{B}}(\boldsymbol{x}) < f(\bar{\boldsymbol{x}})$, a crucial property throughout all of Sect. 3.2 (e.g., (3.7) and (3.20)). The effect of translation on $\mathcal{B}$ is trivial: it only amounts at replacing the α^b—the intercepts of the linearizations in the "default stability center" $\mathbf{0}$—with the *linearization errors*

$$\alpha^b(\bar{\boldsymbol{x}}) = f(\bar{\boldsymbol{x}}) - [\ f(\boldsymbol{x}^b) + \langle \boldsymbol{z}^b, \bar{\boldsymbol{x}} - \boldsymbol{x}^b \rangle\ ] = \alpha^b - \langle \boldsymbol{z}^b, \bar{\boldsymbol{x}} \rangle + f(\bar{\boldsymbol{x}}) \tag{3.21}$$

(just apply the definition to $f_{\bar{\boldsymbol{x}}}$). By convexity, $\alpha^b(\bar{\boldsymbol{x}}) \geq 0$, and

$$\boldsymbol{z}^b \in \partial_{\alpha^b(\bar{\boldsymbol{x}})} f(\bar{\boldsymbol{x}}), \tag{3.22}$$

where the ε-subdifferential $\partial_\varepsilon f(\bar{\boldsymbol{x}})$ contains all ε-subgradients of f at $\bar{\boldsymbol{x}}$, i.e., $\boldsymbol{z} \in \mathbb{R}^n$ such that $f(\boldsymbol{x}) \geq f(\bar{\boldsymbol{x}}) + \langle \boldsymbol{z}, \boldsymbol{x} - \bar{\boldsymbol{x}} \rangle - \varepsilon$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore, $\alpha^b(\bar{\boldsymbol{x}})$ is a measure of "how close" $\boldsymbol{z}^b$ is to be a subgradient of f at $\bar{\boldsymbol{x}}$. Although the definition (3.21)

uses the original iterates $\boldsymbol{x}^b$, it is not necessary to store them to re-compute the linearization errors when $\bar{\boldsymbol{x}}$ changes to any other $\boldsymbol{x}$, since they can be updated using the *information transport property*

$$\alpha^b(\boldsymbol{x}) = \langle \boldsymbol{z}^b, \bar{\boldsymbol{x}} - \boldsymbol{x} \rangle + \alpha^b(\bar{\boldsymbol{x}}) + (\, f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}})\,) \tag{3.23}$$

(just write (3.21) for $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$ and simplify out common terms). Since usually $\bar{\boldsymbol{x}}$ is clear from the context, for the sake of notational simplicity we will use α^b as much as possible. Doing so, the MP of the CPM using the translated model $\underline{f}_{\mathcal{B},\bar{\boldsymbol{x}}}$ is formally identical to (3.4), save that its optimal value need be increased by $f(\bar{\boldsymbol{x}})$, to account for the translation in $f/\underline{f}_{\mathcal{B}}$, to recover the original objective value. This provides a neat interpretation for its (linear) dual, i.e.,

$$[-]\ \min \left\{ \sum_{b\in\mathcal{B}^i} \alpha^b \theta^b \ :\ \sum_{b\in\mathcal{B}^i} \boldsymbol{z}^b \theta^b = \boldsymbol{0},\ \sum_{b\in\mathcal{B}^i} \theta^b = 1,\ \theta^b \geq 0\ \ b \in \mathcal{B}^i \right\}\ [-f(\bar{\boldsymbol{x}}^i)]. \tag{3.24}$$

Note that ordinarily (3.24) would be a maximization one with coefficients $-\alpha^b$ in the objective function; the change of sign reveals the problem as that of constructing $\boldsymbol{0}$ as convex combination of the $\boldsymbol{z}^b$ using "as much accurate as possible" information with respect to the point $\bar{\boldsymbol{x}}$ (although the latter actually changes nothing in this problem), also accounting for the fact that the offset has to be changed in sign, too. This intuitive interpretation can be stated exactly. It is crucial that the dual variables θ^b are convex combinators; since this will be quite common, we will denote by Θ the unitary simplex of appropriate dimension. The fact that $\boldsymbol{\theta} \in \Theta$ implies that

$$\sum_{b\in\mathcal{B}} \boldsymbol{z}^b \theta^b = \boldsymbol{z}(\boldsymbol{\theta}) \in \partial f_{\alpha(\boldsymbol{\theta})}(\bar{\boldsymbol{x}}), \quad \text{where} \quad \alpha(\boldsymbol{\theta}) = \sum_{b\in\mathcal{B}} \alpha^b \theta^b. \tag{3.25}$$

This can be obtained combining $\partial_\varepsilon \underline{f}_{\mathcal{B}}(\bar{\boldsymbol{x}}) \subseteq \partial_\varepsilon f(\bar{\boldsymbol{x}})$ (use [57, Proposition XI.1.3.1.(vii)] together with $\underline{f}^i \leq f$ and $\underline{f}^i(\bar{\boldsymbol{x}}) = f(\bar{\boldsymbol{x}})$) and

$$\partial_\varepsilon \underline{f}^i(\bar{\boldsymbol{x}}) = \left\{ \boldsymbol{z} = \sum_{b\in\mathcal{B}^i} \boldsymbol{z}^b \theta^b \ :\ \boldsymbol{\theta} \in \Theta,\ \ \sum_{b\in\mathcal{B}^i} \alpha^b \theta^b \leq \varepsilon \right\}$$

(see [57, Example XI.5.3]). The dual (3.24) can therefore be described in plain words as the problem of finding the smallest ε such that $\boldsymbol{0} \in \partial_\varepsilon \underline{f}^i(\bar{\boldsymbol{x}})$.

This interpretation carries over to the PBM: the explicit, translated form of (3.8) and its (quadratic) dual are, respectively,

$$(\boldsymbol{d}^i\ ,\ v^i\ ) = \operatorname{argmin}\left\{ v + \frac{\mu^i}{2}\|\boldsymbol{d}\|_2^2\ :\ v \geq \langle z^b, \boldsymbol{d}\rangle - \alpha^b\ \ b \in \mathcal{B}^i \right\}\ [+f(\bar{\boldsymbol{x}}^i)] \tag{3.26}$$

$$\boldsymbol{\theta}^i = \operatorname{argmin}\left\{ \frac{1}{2\mu^i}\|\sum_{b\in\mathcal{B}^i} z^b\theta^b\|_2^2 + \sum_{b\in\mathcal{B}^i}\alpha^b\theta^b\ :\ \boldsymbol{\theta} \in \Theta \right\}\ [-f(\bar{\boldsymbol{x}}^i)]\ . \tag{3.27}$$

The dual optimal solution $\boldsymbol{\theta}^i$ gives, via (3.25), the *aggregated linearization* $(\bar{z}^i = z(\boldsymbol{\theta}^i)\ ,\ \bar{\alpha}^i = \alpha(\boldsymbol{\theta}^i)\ )$ such that $\bar{z}^i \in \partial_{\bar{\alpha}^i} f(\bar{\boldsymbol{x}}^i)$; the complementary slackness conditions tie that to the optimal solution of (3.26) as

$$\boldsymbol{d}^i = -(1/\mu^i)\bar{z}^i, \qquad v^i = \langle \bar{z}^i, \boldsymbol{d}^i\rangle - \bar{\alpha}^i = -(1/\mu^i)\|\bar{z}^i\|_2^2 - \bar{\alpha}^i. \tag{3.28}$$

Thus, the dual problem requires finding an ε-subgradient $\bar{z}^i$, obtained as a convex combination of previously obtained ones, which has *both* a small norm *and* a small ε, with the relative weight of the two objective functions dictated by μ^i. In other words, (3.27) is the *augmented Lagrangian* of (3.24) with respect to the constraint requiring z^i to be $\mathbf{0}$; it is therefore not surprising, then, that the former always have a(n unique) solution, whereas the latter can be empty. Furthermore, the next iterate is then $\boldsymbol{x}^i = \bar{\boldsymbol{x}}^i + \boldsymbol{d}^i = \bar{\boldsymbol{x}}^i - (1/\mu^i)\bar{z}^i$, i.e., is obtained by doing a step $1/\mu^i$ along the approximated subgradient; this strongly links the PBM with (approximated, deflected) subgradient-type methods, cf. Sect. 3.3.2 and [19].

The fact that $\bar{z}^i \in \partial_{\bar{\alpha}^i} f(\bar{\boldsymbol{x}}^i)$ has several useful consequences. For instance, it immediately candidates it at being used as an alternative source of approximated subgradients of f to be used in (3.13) [85]. More importantly, however, it provides the stopping criterion of the method: $\bar{z}^i = \mathbf{0}$ and $\bar{\alpha}^i = 0$ imply that $\bar{\boldsymbol{x}}^i$ is optimal. In practice one therefore stops when $\|\bar{z}^i\|$ and $\bar{\alpha}^i$ are both "small". One can either use two distinct thresholds for the two quantities, or join both in a single criterion

$$\|\bar{z}^i\|_2^2/\bar{\mu} + \bar{\alpha}^i \leq \varepsilon, \tag{3.29}$$

where $\bar{\mu}$ is a scaling factor and ε is the final (absolute) accuracy required. This still requires to properly chose $\bar{\mu}$, but at least $\bar{\mu}$ and μ^i should be related; this means that (3.29) can be exploited to on-line tune μ^i, as discussed below.

However, what the above development mainly reveals is that BM have to properly balance two contrasting objectives: getting a "small" $\|\bar{z}^i\|$, and getting a "small" $\bar{\alpha}^i$. The CPM goes all the way towards the first, which basically means completely ignoring the "quality" of the first-order information with respect to $\bar{\boldsymbol{x}}^i$, with the known negative practical consequences. The opposite approach is well represented

by the following variant of MP [57, Section XI.2.4]:

$$\min \left\{ \| \sum_{b \in \mathcal{B}} z^b \theta^b \|_2^2 \ : \ \sum_{b \in \mathcal{B}} \alpha^b \theta^b \leq \varepsilon^i \quad , \quad \boldsymbol{\theta} \in \Theta \right\} \ . \tag{3.30}$$

One can see (3.27) as the Lagrangian relaxation of (3.30) having $1/\mu^i$ as Lagrangian multiplier, and therefore this would yield a BM with basically the same relationship to the PBM as the LBM has, except in the dual. In principle, for any given ε^i one could find a μ^i giving the same solution. The effect of a small ε^i in (3.30)—equivalently, a small μ^i in (3.30)—is therefore to discard all the first-order information with "large" α^b, so that the new iterate only takes into account information that is "quite accurate" at $\bar{\boldsymbol{x}}^i$. Indeed, (3.30) can be seen as a minor variant of the MP of *ε-descent* methods [57, Chapter IX], where $\mathcal{B}^i$ is exclusively used to construct an inner approximation of $\partial_{\varepsilon^i} f(\bar{\boldsymbol{x}}^i)$; then, (3.30) becomes the problem of finding the *steepest ε-descent direction* for the model, i.e., the least-norm vector in $\partial_{\varepsilon^i} \underline{f}^i(\bar{\boldsymbol{x}})$. Choosing the right value of the stabilization parameter ε^i—similarly to δ^i, μ^i, l^i, λ^i, ...—is crucial, since pure steepest descent methods have a rather bad practical behaviour even in the smooth case.

The issue with all BM is therefore to find the right value of the stabilization parameter(s) so as on one hand to include as much as possible non-local information to avoid the pitfalls of the steepest descent direction, and on the other hand not to trust the model too far beyond the region where it actually provides a reasonable depiction of the function's behaviour. For the PBM, this can be described in terms of finding the right point along the *proximal trajectory*, the family of solutions of (3.8) as a function of μ^i, which is a piecewise linear function, easily computed incrementally by solving a sequence of linear programs [45] or by sensitivity analysis techniques [34]. Exploring the proximal trajectory allows one to figure out how $\|\boldsymbol{z}^i\|$ and α^i change as μ^i does, and therefore can be the basis for handling μ^i.

Although similar relationship between the stabilization parameter and the locality of the used information should hold for other forms of BM, the different shape of the MP makes them less obvious to see. For reasons to become apparent in due time we postpone the discussion on the TRBM on Sect. 3.3.4. For the LBM, the explicit form of (3.14) and its dual are, respectively:

$$\boldsymbol{d}^i \in \operatorname{argmin} \left\{ \|\boldsymbol{d}\|_2^2/2 \ : \ l \geq \langle z^b, \boldsymbol{d} \rangle - \alpha^b \quad b \in \mathcal{B}^i \right\}, \tag{3.31}$$

$$\boldsymbol{\theta}^i \in \operatorname{argmin} \left\{ \| \sum_{b \in \mathcal{B}^i} z^b \theta^b \|_2^2/2 + \sum_{b \in \mathcal{B}^i} (l + \alpha^b) \theta^b \ : \ \boldsymbol{\theta} \geq \mathbf{0} \right\}, \tag{3.32}$$

(where α^b has to be intended as $\alpha^b(\bar{\boldsymbol{x}})$). Here again $\boldsymbol{d}^i = -\boldsymbol{z}(\boldsymbol{\theta}^i)$ holds as in (3.25), but $\boldsymbol{\theta}^i$ does not necessarily belong to Θ. Yet, the fact that necessarily $\boldsymbol{d}^i \neq \mathbf{0}$ implies that $\boldsymbol{\theta}^i \neq \mathbf{0}$ as well; thus, $\boldsymbol{\theta}^i / \langle \boldsymbol{\theta}^i, \boldsymbol{u} \rangle \in \Theta$ ($\boldsymbol{u}$ being the vector of all ones). In other words, $\boldsymbol{d}^i$ is still a scaled multiple of a convex combination of the z^b, although the

step size is no longer clearly related to the stabilization parameter. With a "small" l^i, (3.32) will have an incentive to only use z^b with "small" α^b (locally accurate information), whereas with a "large" l^i the role of the α^b becomes marginal. Also, note that, despite being a QP, (3.32) can be unbounded below as the objective function is not necessarily strictly convex (in fact, (3.31) can be empty). Similarly, the explicit form of the MP (3.15) of the DSBM and its dual are

$$(\boldsymbol{d}^i\ ,\ v^i\ ) = \min\left\{\ v + \frac{\mu^i}{2}\|\boldsymbol{d}\|_2^2\ :\ v \geq \langle z^b, \boldsymbol{d}\rangle - \alpha^b\ \ b \in \mathcal{B}^i,\ \ l^i \geq v\ \right\} \tag{3.33}$$

$$(\boldsymbol{\theta}^i\ ,\ \rho^i\ ) \in \operatorname{argmin}\{\frac{1}{2\mu^i}\|\sum_{b\in\mathcal{B}^i} z^b\theta^b\|_2^2 + \sum_{b\in\mathcal{B}^i}\alpha^b\theta^b + l^i\rho\ :\ \sum_{b\in\mathcal{B}^i}\theta^b - \rho = 1,$$
$$\rho \geq 0,\ \ \theta^b \geq 0\ \ b \in \mathcal{B}^i\ \}. \tag{3.34}$$

By complementary slackness, $\rho^i > 0$ implies $v^i = l^i$: in this case, (3.34) coincides with (3.32), in that $\rho^i = \langle \boldsymbol{\theta}^i, \boldsymbol{u}\rangle$ and therefore the objective function is identical, save for a constant term and the scaling factor μ^i on the quadratic term. If, instead, $v^i < l^i$ then $\rho^i = 0$ and (3.34) coincides with (3.27). Thus, ρ^i can be used to devise strategies to adjust l^i and/or μ^i [25]; this is but one of the many important uses of dual information, as discussed in the next section.

3.3.2 Algorithmic Uses of Duality

The dual concepts introduced in the previous section have many uses in the definition and analysis of BM. In particular, if $\boldsymbol{\theta}^i \in \Theta$ then the aggregated pair $(\bar{z}^i = z(\boldsymbol{\theta}^i)\ ,\ \bar{\alpha}^i = \alpha(\boldsymbol{\theta}^i)\ )$ satisfies $\bar{z}^i \in \partial_{\bar{\alpha}^i} f(\bar{\boldsymbol{x}}^i)$, and therefore can be inserted into $\mathcal{B}^i$. This is free for the PBM and the DSBM when $\rho^i = 0$; for the LBM, or the DSBM when $\rho^i > 0$, a simple scaling is needed, and an analogous technique can be used for PC^{3}PM.

The aggregated pair $(\bar{z}^i\ ,\ \bar{\alpha}^i\ )$ has not been obtained at any iterate $\boldsymbol{x}^i$, but this is not an issue; $\bar{\alpha}^i = \bar{\alpha}^i(\bar{\boldsymbol{x}}^i)$ can be updated via (3.23) when $\bar{\boldsymbol{x}}^i$ changes as all the other ones in $\mathcal{B}^i$. The important result is that $(\bar{z}^i\ ,\ \bar{\alpha}^i\ )$ can actually *substitute* all other information: if one were to set $\mathcal{B}^{i+1} = \{\,(\bar{z}^i\ ,\ \bar{\alpha}^i\ )\,\}$, then $(\boldsymbol{d}^{i+1}\ ,\ v^{i+1}\ ) = (\boldsymbol{d}^i\ ,\ v^i\ )$ in (3.26). Of course, one does not really want the solution to remain the same, in particular if a null step is being performed; this is not so because of the new information $(z^i\ ,\ \alpha^i\ )$ computed by evaluating $f(\boldsymbol{x}^i)$. It is easy to prove that even if one takes the minimal stance $\mathcal{B}^{i+1} = \bar{\mathcal{B}}^i = \{\,(\bar{z}^i\ ,\ \bar{\alpha}^i\ )\ ,\ (z^i\ ,\ \alpha^i\ )\,\}$—called the *poorman's bundle*—the PBM is still convergent; that is, an infinite sequence of consecutive null steps will result in $\|\bar{z}^i\| \to 0$ and $\bar{\alpha}^i \to 0$. The proof is simple and instructive enough to be worth reporting: it is based on the fact that (3.27) with $\bar{\mathcal{B}}^i$

is the simple problem

$$\min \left\{ h^i(\theta) = \frac{1}{2\mu^i}\|\theta\bar{z}^i + (1-\theta)z^i\|_2^2 + \theta\bar{\alpha}^i + (1-\theta)\alpha^i \ : \ \theta \in [0,1] \right\}, \tag{3.35}$$

whose optimal solution has the following closed-form expression:

$$\theta_*^i \ = \ \min\left\{ 1 \ , \ \max\left\{ 0 \ , \ \frac{\alpha^i - \bar{\alpha}^i - \langle z^i, \bar{z}^i - z^i\rangle/\mu^i}{\|\bar{z}^i - z^i\|_2^2/\mu^i} \right\}\right\}. \tag{3.36}$$

Since $h^i(1)$ is the optimal value of (3.27) at iteration i, one only has to show that $h^i(1) - h^i(\theta_*^i)$ decreases enough. This hinges on the fact that (3.7) *not* holding can be rewritten, by means of some simple algebra (cf. (3.28))

$$\Delta f^i \ > \ -mv^i = -m\left(-(1/\mu^i)\|\bar{z}^i\|_2^2 - \bar{\alpha}^i \right) \geq -mh^i(1), \tag{3.37}$$

from which it is easy to derive

$$h^i(1) - h^i(\theta_*^i) \geq \frac{(1-m)h^i(1)}{2} \min\left\{ 1 \ , \ \frac{(1-m)h^i(1)}{\|\bar{z}^i - z^i\|_2^2/\mu^i} \right\}. \tag{3.38}$$

By (3.38), the optimal value of (3.27) is decreasing, and it must necessarily converge to zero during an infinite sequence of consecutive null steps (at least if μ^i is not dramatically mishandled, e.g. just kept bounded away from 0).

Thus, the PBM is convergent provided that $(\bar{z}^i \ , \ \bar{\alpha}^i)$ is still a feasible solution of (3.34) (in the (z, α)-space) at iteration $i+1$, and, of course, $(z^i \ , \ \alpha^i) \in \mathcal{B}^{i+1}$. This immediately suggests the two standard forms of *bundle management*:

1. (*compression*) ensure that $(\bar{z}^i \ , \ \bar{\alpha}^i) \in \mathcal{B}^{i+1}$;
2. (*selection*) ensure that $b \in \mathcal{B}^{i+1}$ for all the $b \in \mathcal{B}^i$ such that $\theta^{i,b} > 0$.

Note that, by Carathéodory's theorem, there always exist a $\boldsymbol{\theta}^i$ with at most $n+1$ positive variables; hence, both strategies yield a finite bound over the size of $\mathcal{B}$, a significant advantage—at least in theory—over the non-stabilized CPM. Not unexpectedly, the practical side of bundle management is considerably more nuanced. For once, (3.38) only refers to the "tail" of the algorithm, where $\bar{\boldsymbol{x}}^i$ has reached (very close to) some optimal $\boldsymbol{x}_*$ and the PBM "only" have to *prove* this by driving both $\|\bar{z}^i\|$ and $\bar{\alpha}^i$ to 0. This all but ignores the "cruise" phase where $\boldsymbol{x}^i$ is closing in to $\boldsymbol{x}_*$. For that, (3.7) would imply a reasonably fast convergence *in the number of serious steps*, with of course the issue of how many null steps occur between two consecutive serious steps. Even ignoring this, the rate of convergence implied by (3.38) is sublinear, i.e., rather slow. This is one of the main reasons why iteration complexity of the PBM is $O(1/\varepsilon^3)$ [3, 64], even worse than the $O(1/\varepsilon^2)$ that any black-box algorithm of this type necessarily has to have. This is so bad

a convergence rate as to make it completely impractical to obtain anything more than moderately accurate solutions. Indeed, the PBM with "extreme" aggregation $\mathcal{B}^{i+1} = \bar{\mathcal{B}}^i$ is a minor variant of a *deflected* subgradient-type method [22]; in particular, it is closely related [7] with the so-called *volume algorithm* [8], that had spurred considerable interest in combinatorial optimization circles at the turn of the millenium. It had actually been known already [1] that these subgradient-type methods have—in theory—a working stopping criterion, which is important in some applications. However, (3.38) reveals how the advantage is only theoretical: in practice, convergence of subgradient methods is so slow that the only feasible stopping criterion is a limit on the number of iterations. Although they can still be attractive in some applications, this is only true under very mild requirements on the required accuracy (say, 10^{-3} to 10^{-4} relative) [44].

It is revealing to contrast this behaviour with that of the CPM as numerically illustrated in Sect. 3.2.1. There, although the algorithm has an erratic behaviour in the "cruise" phase (apparently failing to exhibit any convergence at all), the "tail" of the process is pleasingly fast. This is due to the fact that once enough information is accrued in $\mathcal{B}$ to make $\underline{f}_{\mathcal{B}}$ a good enough model at some optimal solution, the algorithm can efficiently close in to that. Such accumulation of information in $\mathcal{B}$ is essentially destroyed by extreme aggregation $\mathcal{B}^{i+1} = \bar{\mathcal{B}}^i$: although the process remains generally convergent, the speed can be as abysmal in practice as (3.38) predicts. In other words, extreme aggregation can hurt a BM precisely in what could otherwise be a strong point of its. Similarly, discarding a pair (z^b, α^b) as soon as $\theta^{i,b} = 0$ may considerably hurt performances; more appropriate (heuristic) rules are to discard it after that the multiplier has been zero for some (say, 20) consecutive iterations. In some tests, the "fast tail phase" has proven rather delicate, being impaired by even mildly aggressive selection rules or by imposing even seemingly loose limits on the maximum size of $\mathcal{B}$ [41]. Hence, at least in some applications it is better to shoulder the substantial burden of solving MP with a large $\mathcal{B}$ than trying to keep the latter small, as any reduction in MP cost is largely outweighed by the corresponding decrease of convergence speed.

Unfortunately, all these aspects are only characterized experimentally; all convergence arguments—and efficiency estimates—on PBM hinge on extreme aggregation. The complexity estimate can actually be improved to—the still sublinear—$O(\log(1/\varepsilon)(1/\varepsilon))$ with further assumptions on f (in particular, strong coercivity at the unique optimum) [28], but still the same bound holds for $\bar{\mathcal{B}}^i$ and for any arbitrarily large $\mathcal{B}^i$; hence, the theoretical worst-case analysis seems unable to capture some important aspects of the practical behaviour of BM, (fortunately) substantially underestimating convergence speed. This is not helped by the fact that convergence arguments, as discussed in Sect. 3.2.1, deal with the sequence of serious steps and with sub-sequences of consecutive null steps between two serious steps as two loosely related processes; after a serious step is declared the algorithm can basically be restarted from scratch, as the arguments allow to completely change $\mathcal{B}$ then. One recent effort to devise a convergence analysis of the PBM as an unique process is based on (in principle) avoiding the dichotomic distinction between serious step and null step [3]. This hinges on the introduction

of the—apparently weird—*merit function* $\zeta_\mu(\boldsymbol{x}) = 2f(\boldsymbol{x}) - \phi_\mu(\boldsymbol{x})$, with ϕ_μ the Moreau–Yosida regularization (3.9). The only nice properties of ζ_μ are that $\zeta_\mu \geq f$ and $\zeta_\mu(\boldsymbol{x}) = f(\boldsymbol{x}) \iff \boldsymbol{x}$ is optimal for (3.1); otherwise, the function is nondifferentiable and nonconvex. However, its upper approximation $\zeta_{\mathcal{B},\mu} \geq \zeta_\mu$ obtained by replacing f with $f_{\mathcal{B}}$ in (3.8) is *precisely* computed by solving (3.27), *comprised* the constant term "$-f(\bar{\boldsymbol{x}})$" that is usually ignored in the analysis of the PBM. Once $\boldsymbol{x}^i$ is produced by the MP and $f(\boldsymbol{x}^i)$ and $\boldsymbol{z}^i$ are computed by the oracle, it is possible to define the problem of minimizing $\zeta_{\mathcal{B},\mu}(\boldsymbol{x})$ for $\boldsymbol{x} \in [\bar{\boldsymbol{x}}^i, \boldsymbol{x}^i]$. Actually, doing so would require knowing the value of f at all points of the interval; this can be replaced by an *upper model* of f on the interval, typically $\lambda f(\bar{\boldsymbol{x}}^i) + (1-\lambda) f(\boldsymbol{x}^i)$ for $\boldsymbol{x} = \boldsymbol{x}(\lambda) = \lambda \bar{\boldsymbol{x}}^i + (1-\lambda)\boldsymbol{x}^i$. This allows to define a further upper approximation of $\zeta_{\mathcal{B},\mu}$, and $\bar{\boldsymbol{x}}^{i+1}$ can be easily chosen as the minima of this function on the interval $[\bar{\boldsymbol{x}}^i, \boldsymbol{x}^i]$; doing so one can prove that eventually $\zeta_\mu(\bar{\boldsymbol{x}}^i) - f(\bar{\boldsymbol{x}}^i) \to 0$, i.e., global convergence. This is potentially interesting in that $\bar{\boldsymbol{x}}^{i+1}$ can be chosen "in between" $\bar{\boldsymbol{x}}^i$ and $\boldsymbol{x}^i$, thereby generalizing the PBM at least insomuch as $f(\bar{\boldsymbol{x}}^{i+1}) > f(\bar{\boldsymbol{x}}^i)$ can happen. Unfortunately, the approach—at least with the natural upper model—turns out to actually only do either serious step or null step. Also, the efficiency analysis still uses arguments very close to (3.36), and therefore it does not seem of being any better able of properly evaluating the effect of information accrual. Besides, the practical efficiency of the method does not seem to be much different from that of the original PBM.

All in all, it can be argued that the currently available convergence and efficiency analyses fail to properly capture some aspects of the BM that are important in practice. Yet, the dual viewpoint is crucial for the understanding and the implementation of BM; this is even more so in the case where (3.1) is itself a dual problem, as discussed next.

3.3.3 Duality in the Original Problem

One important motivation for (3.1) is the case where

$$f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{b} \rangle + \max \{ \langle \boldsymbol{c} - \boldsymbol{x}A, \boldsymbol{u} \rangle \ : \ \boldsymbol{u} \in U \}, \tag{3.39}$$

i.e., f is the Lagrangian function of the problem

$$\max \{ \langle \boldsymbol{c}, \boldsymbol{u} \rangle \ : \ A\boldsymbol{u} = \boldsymbol{b}, \ \ \boldsymbol{u} \in U \}, \tag{3.40}$$

with respect to the explicit constraints. Customarily U is assumed compact, so that f is finite-valued; this is mainly to save on details, with extensions discussed in Sect. 3.4.3. Similarly, linearity of objective function and constraints can be relaxed with most of the results carrying over, albeit at the cost of considerably more cumbersome notation [71].

Evaluating f at some iterate $\boldsymbol{x}^i$ requires solving the *Lagrangian relaxation* (3.39) of (3.40). Any of its *optimal solutions* $\boldsymbol{u}^i$ gives $f(\boldsymbol{x}^i) = \langle \boldsymbol{c} - \boldsymbol{x}^i A, \boldsymbol{u}^i \rangle + \langle \boldsymbol{x}^i, \boldsymbol{b} \rangle$ and $\boldsymbol{z}^i = \boldsymbol{b} - A\boldsymbol{u}^i$; note that this yields $\alpha^i = \langle \boldsymbol{z}^i, \boldsymbol{x}^i \rangle - f(\boldsymbol{x}^i) = -\langle \boldsymbol{c}, \boldsymbol{u}^i \rangle$, a suggestive enough result. Indeed, the dual (3.24) of the MP of the CPM then (heavily exploiting linearity) becomes

$$\max \left\{ c \left(\sum_{b \in \mathcal{B}^i} \boldsymbol{u}^b \theta^b \right) \; : \; A \left(\sum_{b \in \mathcal{B}^i} \boldsymbol{u}^b \theta^b \right) = \boldsymbol{b}, \quad \boldsymbol{\theta} \in \Theta \right\}. \tag{3.41}$$

Hence, one is in fact considering the convex set $U_{\mathcal{B}} = \mathrm{conv}(\{ \boldsymbol{u}^b \; : \; b \in \mathcal{B} \})$, which would be an inner approximation of U if the latter were convex, and is solving (3.40) with U replaced by $U_{\mathcal{B}}$. Clearly, "with an infinitely large $\mathcal{B}$" one would be solving the *convexified relaxation* of (3.40)

$$\max \{ \langle \boldsymbol{c}, \boldsymbol{u} \rangle \; : \; A\boldsymbol{u} = \boldsymbol{b}, \quad \boldsymbol{u} \in \mathrm{conv}(U) \}, \tag{3.42}$$

equivalent to (3.40) if U is convex, and in some sense its best possible convex relaxation otherwise. Then, (3.41) is the *inner approximation* (a restriction) of (3.42) corresponding to the finite subset of solutions collected so far. The optimal value of (3.41) is thus a lower bound on that of (3.42), just as that of (3.3) is a lower bound on f_*; indeed, the *Lagrangian dual* (LD) (3.1) of (3.40) is equivalent to its convexified relaxation (3.42), a celebrated result [71] with many useful consequences [37]. This allows to give interesting interpretations to the results of Sect. 3.3.1, starting with the fact that the linearization error (3.21) becomes $\alpha^b(\bar{\boldsymbol{x}}) = \langle \boldsymbol{c} - \bar{\boldsymbol{x}} A, \boldsymbol{u}(\bar{\boldsymbol{x}}) \rangle - \langle \boldsymbol{c} - \bar{\boldsymbol{x}} A, \boldsymbol{u}^b \rangle$, where $\boldsymbol{u}(\bar{\boldsymbol{x}})$ is (any one of) the optimal solution of (3.39) with $\boldsymbol{x} = \bar{\boldsymbol{x}}$; basically, how much sub-optimal is the solution $\boldsymbol{u}^b$ with respect to the optimal one $\boldsymbol{u}(\bar{\boldsymbol{x}})$ with the *Lagrangian costs* (sometimes called *reduced costs*) $\boldsymbol{c} - \bar{\boldsymbol{x}} A$ corresponding to the current point $\bar{\boldsymbol{x}}$.

Hence, the LD (3.1) of (3.42)/(3.40)—which is the same, as the LD cannot distinguish a problem from its convexified relaxation—provides a way to solve (3.42) by iteratively accumulating solutions $\boldsymbol{u}^i \in U$ and explicitly constructing (the relevant part of) its feasible region. If, say, U is a finite set, then $\mathrm{conv}(U)$ is a polyhedron and only the finite set of its extreme points is required to fully represent it; hence, the LP (3.41) with a (possibly, very) large $\mathcal{B}$ is actually equivalent to (3.42). This is known as the *Dantzig–Wolfe reformulation* of (3.42), and it is well known that solving the LD by the CPM is equivalent to solving (3.42) by the *Dantzig–Wolfe decomposition algorithm* [37]. The Dantzig–Wolfe reformulation has "few" $(n + 1)$ constraints, but in principle exponentially many variables (columns in the LP); thus the Dantzig–Wolfe decomposition algorithm is also referred to as *column generation* (although the latter concept is in some sense slightly more general) [27]. Stabilizing the CPM is therefore also known as stabilizing the column generation

[11, 12]. For instance, for the PBM one can rewrite (3.27) as

$$\max\left\{\sum_{b\in\mathcal{B}^i}(\boldsymbol{c}\boldsymbol{u}^b)\theta^b+\langle\bar{\boldsymbol{x}},\boldsymbol{z}\rangle-\frac{1}{2\mu^i}||\,\boldsymbol{z}\,||_2^2\;:\;A\left(\sum_{b\in\mathcal{B}^i}\boldsymbol{u}^b\theta^b\right)-\boldsymbol{b}=\boldsymbol{z},\;\boldsymbol{\theta}\in\Theta\right\},$$

or, even more tellingly, as

$$\max\left\{\langle\boldsymbol{c},\boldsymbol{u}\rangle+\langle\bar{\boldsymbol{x}},\boldsymbol{b}-A\boldsymbol{u}\rangle-\frac{1}{2\mu^i}||\,A\boldsymbol{u}-\boldsymbol{b}\,||_2^2\;:\;\boldsymbol{u}\in U^i\right\}. \tag{3.43}$$

(with $U^i=U_{\mathcal{B}^i}$). Thus, the PBM can be read from the viewpoint of (3.42) as an *augmented Lagrangian* combined with an *inner linearization* approach where U is substituted by its approximation U^i. The aggregated pair $(\,\bar{\boldsymbol{z}}^i\,,\;\bar{\alpha}^i\,)$ is then associated with the point

$$\bar{\boldsymbol{u}}^i=\boldsymbol{u}(\boldsymbol{\theta}^i)\in\mathrm{conv}(U)\quad\text{ with }\quad\boldsymbol{u}(\boldsymbol{\theta})=\sum_{b\in\mathcal{B}}\boldsymbol{u}^b\theta^b \tag{3.44}$$

by $\bar{\boldsymbol{z}}^i=\boldsymbol{b}-A\bar{\boldsymbol{u}}^i$ and $\bar{\alpha}^i=\langle\boldsymbol{c}-\bar{\boldsymbol{x}}A,\boldsymbol{u}(\bar{\boldsymbol{x}})\rangle-\langle\boldsymbol{c}-\bar{\boldsymbol{x}}A,\bar{\boldsymbol{u}}^i\rangle$; convergence of the PBM can be read as the fact that $\{\,\bar{\boldsymbol{u}}^i\,\}\to\boldsymbol{u}_*$, with the latter optimal to (3.42) (an easy but instructive connection to formally prove).

It is, however, useful to delve a bit further into the equivalence between (3.1) and (3.42)—with the f of (3.39)—as doing so requires to introduce useful concepts, primarily the *Fenchel's conjugate* $f^*(\boldsymbol{z})=\sup_{\boldsymbol{x}}\{\,\langle\boldsymbol{z},\boldsymbol{x}\rangle-f(\boldsymbol{x})\,\}$ of f. The function f^* is convex by definition, even if f is not, and closed under very mild assumptions on f. The bi-conjugate f^{**} is the (closed) *convex envelope* of f, i.e., the smallest (in set-inclusion sense) closed convex function g such that epi $g\supseteq$ epi f; clearly, if f is closed convex then $f^{**}=f$. Geometrically, f^* characterizes all the affine functions supporting epi f, i.e., basically its (approximate) subgradients; indeed, a fundamental property of f^* is

$$\boldsymbol{z}\in\partial_\varepsilon f(\boldsymbol{x})\Longleftrightarrow\boldsymbol{x}\in\partial_\varepsilon f^*(\boldsymbol{z})\Longleftrightarrow f(\boldsymbol{x})+f^*(\boldsymbol{z})\le\langle\boldsymbol{z},\boldsymbol{x}\rangle+\varepsilon \tag{3.45}$$

for each $\varepsilon\ge 0$ [57, Proposition XI.1.2.1]. Coupled with *Fenchel's inequality* $\langle\boldsymbol{z},\boldsymbol{x}\rangle\le f(\boldsymbol{x})+f^*(\boldsymbol{z})$ for all $\boldsymbol{z}$, $\boldsymbol{x}$, this gives $\langle\boldsymbol{z},\boldsymbol{x}\rangle=f(\boldsymbol{x})+f^*(\boldsymbol{z})\Longleftrightarrow\boldsymbol{z}\in\partial f(\boldsymbol{x})\Longleftrightarrow\boldsymbol{x}\in\partial f^*(\boldsymbol{z})$; the immediate consequence is that $\alpha^b=\langle\boldsymbol{z}^b,\boldsymbol{x}^b\rangle-f(\boldsymbol{x}^b)=f^*(\boldsymbol{z}^b)\,(=-\langle\boldsymbol{c},\boldsymbol{u}^b\rangle$ for (3.39)). Also, since $(f(\cdot+\bar{\boldsymbol{x}}))^*(\boldsymbol{z})=f^*(\boldsymbol{z})-\langle\boldsymbol{z},\bar{\boldsymbol{x}}\rangle$ and $(f(\cdot)-v)^*(\boldsymbol{z})=f^*(\boldsymbol{z})+v$, one has for the translated function $f_{\bar{\boldsymbol{x}}}(\boldsymbol{d})=f(\bar{\boldsymbol{x}}+\boldsymbol{d})-f(\bar{\boldsymbol{x}})$ that $f^*_{\bar{\boldsymbol{x}}}(\boldsymbol{z})=f^*(\boldsymbol{z})-\langle\boldsymbol{z},\boldsymbol{x}\rangle+f(\bar{\boldsymbol{x}})$ (≥ 0 by Fenchel's inequality). Hence, $\alpha^b(\bar{\boldsymbol{x}})=\alpha^b-\langle\boldsymbol{z}^b,\bar{\boldsymbol{x}}\rangle+f(\bar{\boldsymbol{x}})=f^*_{\bar{\boldsymbol{x}}}(\boldsymbol{z}^b)$ (cf. (3.21)). Thus, clearly all dual problems of Sect. 3.3.1 are dealing with $f^*/f^*_{\bar{\boldsymbol{x}}}$, but when f is (3.39) they are also dealing with (3.42). The link is made explicit by the (opposite of the)

value function of (3.42)

$$v(z) = -\max\{\,\langle c, u\rangle \;:\; b - Au = z,\; u \in \mathrm{conv}(U)\,\}, \tag{3.46}$$

in fact

$$\begin{aligned} v^*(x) &= \max\{\,\langle z, x\rangle + \max\{\,\langle c, u\rangle \;:\; b - Au = z,\; u \in \mathrm{conv}(U)\,\}\} \\ &= \max\{\,\langle c, u\rangle + \langle x, b - Au\rangle \;:\; u \in \mathrm{conv}(U)\,\} \;=\; f(x). \end{aligned}$$

With the obvious $f^*(\mathbf{0}) = -f_*$, this confirms that the LD (3.1) of (3.40) is equivalent to its convexified relaxation (3.42): $v(\mathbf{0}) = -f_*$, with the change in sign only due to the insistence on minimization typical of convex optimization. Linearity of the objective function is by no means a crucial ingredient: with a generic objective function $c(u)$ in (3.40), the LD (3.1) is equivalent to $\max\{\,\tilde{c}(u) \;:\; Au = b\,\}$, where $\tilde{c} = (c + \imath_U)^{**}$. The result easily extends to inequality constraints $Au \le b$, yielding sign constraints $x \ge 0$ in (3.1); the generic nonlinear case $A(u) \le b$ requires considerably more complex notation, even in the convex case, although the results are in the same vein [71].

Thus, the conjugate f^* allows to express in a general way primal/dual relationships that would seem to be specific of the Lagrangian case (3.39). In particular, one can consider the (apparently weird) problem

$$\min\{\, f^*(z) \;:\; z = \mathbf{0}\,\} \tag{3.47}$$

as the dual of (3.1). This is quite a reasonable dual: its optimal value is $(-)\, f_*$, and it deals with dual objects, as $z \in \mathrm{dom}\, f^*$ if and only if $z \in \partial f(x)$ for some point x (cf. (3.45)). Furthermore, the Lagrangian relaxation of (3.47) with respect to the constraints "$z = \mathbf{0}$", using $\bar{x}$ as Lagrangian multipliers, is

$$\inf\{\, f^*(z) - \langle z, \bar{x}\rangle\,\} = (f^*)^*(\bar{x}) = f(\bar{x}). \tag{3.48}$$

Thus, the minimization in (3.48) is the equivalent to the maximization in (3.39): a Lagrangian relaxation that has to be solved to find the optimum z (respectively u), which is (provides) the subgradient. All interpretation of, say, the PBM as an inner linearization approach, where the computation of $f(x^i)$ provides a new point u^i that enlarges $U_{\mathcal{B}}$, can be recast in terms of generating an inner approximation of epi f^*, without a need for any special structure in f. This is described in some detail in the next section, in which conjugacy arguments—in particular *Fenchel's duality*—are used to devise more general stabilization devices than the proximal and TR ones. Yet, it is clear that the dual interpretation of BM is particularly useful for their applications to Lagrangian optimization (cf. [5, 6, 11, 12, 15, 16, 20, 31, 33, 38–41, 43, 46, 56, 67, 73, 83, 89, 91, 95] among the many others), because then generated dual information has a direct and crucial algorithmic use (e.g., [8, 9, 14, 21, 29, 32, 37, 42]).

3.3.4 Generalized Stabilization

As the previous section showed, devising and analysing BM requires—or at least significantly benefits from—considering the dual aspects of all involved concepts, starting from the MP. This would seem to make it harder to use less simple stabilizing terms, like TR constraints in any norm that is not L_1, L_2 or L_∞ or penalty functions that are not either piecewise linear or convex quadratic, just because then the dual of the MP cannot be obtained with familiar LP or QP duality. Yet, it is intuitively clear that these should in principle work as well (if not better) than the simple ones. Furthermore, there can be good reasons for wanting to use different stabilizing terms, which requires being able to express dual relationships beyond LP and QP. Being this a convex setting Lagrangian duality would seem to be the natural recourse, but its max/min form is more cumbersome than closed-form duals with only dual variables. The alternative is *Fenchel's duality*, mirably expressed by

$$\inf_{\boldsymbol{x}}\{\ f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})\ \} = -\inf_{z}\{\ f_1^*(z) + f_2^*(-z)\ \}, \tag{3.49}$$

which holds under mild assumptions (f_1 and f_2 closed convex and the intersection of their domains nonempty). Note the "$-z$", which comes from the standard form of the conjugate of a sum

$$(\ f_1(\cdot) + f_2(\cdot)\ )^*(\mathbf{0}) = \inf_{z_1,z_2}\{\ f_1^*(z_1) + f_2^*(-z_2)\ :\ z_1 + z_2 = \mathbf{0}\ \}$$

and that could not be noticed in Sect. 3.3.1 because the stabilizing terms are radially symmetric ($\|z\| = \|-z\|$). Thus, one may consider the generalized BM (GBM) with a *generalized stabilization term* D_μ [36], i.e., the MP

$$\boldsymbol{d}^i \in \operatorname{argmin}\left\{\ \underline{f}^i(\bar{\boldsymbol{x}} + \boldsymbol{d}) + D_{\mu^i}(\boldsymbol{d})\ \right\}, \tag{3.50}$$

and immediately derive its (Fenchel's) dual

$$\bar{z}^i \in \operatorname{argmin}\left\{\ (\underline{f}^i)^*(z) + \langle z, \bar{\boldsymbol{x}}\rangle + D^*_{\mu^i}(-z)\ \right\}. \tag{3.51}$$

This becomes more familiar when using $\underline{f}_{\mathcal{B}} = \check{f}_{\mathcal{B}}$, as for (3.2) one has

$$\check{f}^*_{\mathcal{B}}(z) = \min\left\{\ \sum_{b\in\mathcal{B}}\alpha^b\theta^b\ :\ \sum_{b\in\mathcal{B}}z^b\theta^b = z,\ \boldsymbol{\theta}\in\Theta\ \right\} \tag{3.52}$$

which, with (say) $D_\mu(\boldsymbol{d}) = \mu\|\boldsymbol{d}\|_2^2/2 \equiv D^*_\mu(z) = \|z\|_2^2/(2\mu)$ immediately reproduces (3.27). For the Lagrangian case (3.39), (3.51) becomes (cf. (3.43))

$$\bar{\boldsymbol{u}}^i \in \operatorname{argmin}\left\{\ \langle \boldsymbol{c}, \boldsymbol{u}\rangle + \langle \bar{\boldsymbol{x}}, \boldsymbol{b} - A\boldsymbol{u}\rangle - D^*_{\mu^i}(\,A\boldsymbol{u} - \boldsymbol{b}\,)\ :\ \boldsymbol{u}\in U^i\ \right\}$$

(again, note the change of sign in $z = b - Au$), or, in "explicit form"

$$\min \left\{ \sum_{b \in \mathcal{B}^i} (cu^b)\theta^b + \langle \bar{x}, z \rangle + D^*_{\mu^i}(-z) : A \left(\sum_{b \in \mathcal{B}^i} u^b \theta^b \right) - b = z, \ \theta \in \Theta \right\};$$

a *generalized augmented Lagrangian* of (3.42), with D^*_μ in the second-order term. Conjugacy arguments allow to derive primal-dual relationships that do not depend on the choice of D_μ (or $\underline{f}_{\mathcal{B}}$, for that matter), such as

$$-\bar{z}^i \in \partial D_{\mu^i}(d^i) \ \text{ and } \ d^i \in \partial D^*_{\mu^i}(-\bar{z}^i);$$

$$\bar{z}^i \in \partial \underline{f}^i(\bar{x} + d^i) \ \text{ and } \ \bar{x} + d^i \in \partial (\underline{f}^i)^*(\bar{z}^i);$$

$$\underline{f}^i(\bar{x} + d^i) + (\underline{f}^i)^*(\bar{z}^i) = \langle \bar{z}^i, \bar{x} + d^i \rangle \ \text{ and } \ D_{\mu^i}(d^i) + D^*_{\mu^i}(-\bar{z}^i) = -\langle \bar{z}^i, d^i \rangle.$$

These generalize most of the relationships that are needed to prove convergence of a PRB; for instance, one can prove the suggestive

$$\Delta^i = \underline{f}^i(\bar{x} + d^i) - f(\bar{x}) = (\underline{f}^i)^*(\bar{z}^i) - f^*(z^i) + \langle z^i - \bar{z}^i, \bar{x} + d^i \rangle$$

(with $z^i \in \partial f(x^i = \bar{x} + d^i)$), which gives a dual interpretation to the serious step condition (3.7). Note, however, that not all the relevant relationships of the PBM generalize; most notably, $d^i = -\bar{z}^i/\mu^i$ is *not* true in general, which prevents using some important arguments (basically, a GBM is not necessarily a subgradient-type method in the same was as the PBM is). Yet, convergence can still be proven, provided of course that D_μ has the right properties; those proposed in [36] are nicely symmetric with respect to the conjugacy operation:

1. for all $\mu > 0$, $D_\mu(\mathbf{0}) = 0$ and $\mathbf{0} \in \partial D_\mu(\mathbf{0}) \equiv D^*_\mu(\mathbf{0}) = 0$ and $\mathbf{0} \in \partial D^*_\mu(\mathbf{0})$;
2. for all $\mu > 0$ and $\varepsilon > 0$, $\mathrm{lev}_\varepsilon D_\mu$ is *compact* and $0 \in \mathrm{int}\, \mathrm{lev}_\varepsilon D_\mu \equiv \mathrm{lev}_\varepsilon D^*_\mu$ is *compact* and $0 \in \mathrm{int}\, \mathrm{lev}_\varepsilon D^*_\mu$;
3. for all $\mu' \geq \mu > 0$, $D_\mu \leq D_{\mu'} \equiv D^*_\mu \geq D^*_{\mu'}$;
4. $\lim_{\mu \to 0} D_\mu(d) = 0$ for all $d \equiv$ for all $\varepsilon > 0$, $\lim_{\mu \to 0} \inf\{D^*_\mu(z) : \|z\| \geq \varepsilon\} = +\infty$.

That is, both D_μ and D^*_μ must be non-negative and have bounded level sets with nonempty interior. Of course, some properties are only symmetric insomuch as it is allowed by conjugacy: D_μ has to be *increasing* in μ and converge pointwise to the constant zero function as $\mu \to 0$, which means that D^*_μ has to be *decreasing* in μ and converge "uniformly" to the indicator function of $\{\mathbf{0}\}$ as $\mu \to 0$. That is, D_μ becomes less and less stabilizing as $\mu \to 0$: (3.50) becomes more and more like (3.3), hence (3.51) becomes more and more like (3.24) ($\bar{z}^i$ is constrained to remain closer and closer to $\mathbf{0}$). It is easy to see that these properties are respected both by the proximal and by the TR stabilization; in particular

$D_\mu(\boldsymbol{d}) = 1_{\{\boldsymbol{d}:\|\boldsymbol{d}\|_1 \le 1/\mu\}} \equiv D^*_\mu(z) = \|z\|_\infty/\mu$ and $D_\mu(\boldsymbol{d}) = 1_{\{\boldsymbol{d}:\|\boldsymbol{d}\|_\infty \le 1/\mu\}} \equiv D^*_\mu(z) = \|z\|_1/\mu$ (if a norm $\|\cdot\|$ is used in the primal, then its dual norm $\|\cdot\|_*$ appears in the dual). Thus, (3.50)/(3.51) cover both the TRBM and the PBM, as well as with stabilizing terms that behave *both* as a distance and as the indicator of a ball. Also, one can have a TR in the dual, such as $D^*_\mu(z) = 1_{\{z:\|z\|_\infty \le \mu\}} \equiv D_\mu(\boldsymbol{d}) = \mu\|z\|_1$, a setting not really considered so far.

The above properties are the basic ones, but other assumptions are required to closely reproduce the convergence properties of the PBM. For instance, D_μ *strongly coercive* ($\lim_{\|\boldsymbol{d}\|\to\infty} D_\mu(\boldsymbol{d})/\|\boldsymbol{d}\| = +\infty$), which is equivalent D^*_μ finite everywhere, ensures that (3.50) is always bounded below/(3.51) is nonempty. The assumption can be avoided if boundedness is guaranteed in other ways, the simplest one being that a lower bound $\underline{f} \le f_*$ is *known* and explicitly inserted in $\mathcal{B}^i$ via the pair $(\mathbf{0},\ f(\bar{\boldsymbol{x}}^i) - \underline{f})$; in the case of (3.39) this is equivalent to inserting in $\mathcal{B}^i$ a $\underline{\boldsymbol{u}} \in \mathrm{conv}(U)$ such that $A\underline{\boldsymbol{u}} = \boldsymbol{b}$. Also, *smoothness in* $\mathbf{0}$ is important for the properties of the algorithm, although not symmetrically between D_μ and D^*_μ. In particular, $\nabla D_\mu(\mathbf{0}) = \mathbf{0}$ (which is equivalent to strict convexity of D^*_μ in $\mathbf{0}$) ensures that $\boldsymbol{d}^i = \mathbf{0}$ implies that $\bar{\boldsymbol{x}}$ is optimal for (3.1); if D_μ is not differentiable in $\mathbf{0}$ the algorithm is not guaranteed to converge to an optimum of the problem, and this has to be ensured by forcing $\mu^i \to 0$ along iterations. Instead, smoothness of D^*_μ in $\mathbf{0}$ (which is equivalent to strict convexity of D_μ in $\mathbf{0}$) is crucial for proving convergence under "extreme aggregation", directly generalizing (3.36); the results can actually be strengthened somewhat by requiring that the dependency on μ is "simple", i.e., that $D_\mu = \mu D \equiv D^*_\mu = D^*/\mu$ for some fixed D/D^* with the above properties. With a nonsmooth D^*_μ, in principle information cannot be discarded from $\mathcal{B}$ like for the CPM. Practical approaches to discard some information exist—it is always possible to entirely reset $\mathcal{B}$ at each serious step—but no finite bound on $|\mathcal{B}|$ can be established (which may not be too much of an issue in practice due to the possibly dire consequences of too aggressive removals, cf. Sect. 3.3.1).

All in all, (more or less strong) convergence results are available for many choices of D_μ/D^*_μ, potentially allowing to adapt stabilization to the application at hand. For instance, piecewise linear stabilizing terms with "few" pieces can be used to try to obtain a stabilization effect close to that of the PBM without paying the price of a quadratic MP [12]. Often the computational results show that the PBM has better practical convergence behaviour, and therefore is preferable [46]; however, the cost of making the MP a QP can be so high that piecewise linear functions result in better running times [40, 41]. Arguably, Fenchel's duality would not have been necessary to use piecewise linear functions, as the corresponding MP are LP ones; however, other forms of nonlinear stabilization have been proposed. For instance, *Bregman functions* [18] with the form $D_{\bar{\boldsymbol{x}}}(\boldsymbol{d}) = \psi(\bar{\boldsymbol{x}}+\boldsymbol{d}) - \psi(\bar{\boldsymbol{x}}) - \langle\nabla\psi(\bar{\boldsymbol{x}}), \boldsymbol{d}\rangle$ with ψ fixed, strictly convex, differentiable and with compact level sets, can be used to implicitly express the set X via a barrier-like approach, thus possibly making the MP easier to solve [63]. Also, other stabilization terms have been proposed in the context of solving (3.42) that could be adapted for GBM, such as the smooth approximations

of $\|\cdot\|_1$ [84] (below, left) and of $\|\cdot\|_\infty/\mu$ [52] (below, right, for $z \geq 0$)

$$D^*_\mu(z) = \sum_i \begin{cases} z_i^2/(2\mu), \text{ if } -\mu \leq z_i \leq \mu \\ |z_i| - \frac{\mu}{2}, \text{ otherwise} \end{cases}, \qquad D^*_\mu(z) = \ln \sum_i e^{z_i/\mu}.$$

Thus, quite a variety of stabilization terms can be employed, offering a vast trade-off between the theoretical/practical convergence properties of the BM and the cost of the MP. We also mention that a somehow more general approach is proposed in [80], where BM are interpreted, a-la (3.47), as the problem of computing $f^*(\mathbf{0})$. The information provided by the oracle is used to construct the epigraph of $f^*_{\mathcal{B}}$, an inner approximation of the epigraph of f^* (cf. (3.52)), and a MP is solved that finds the closest point of epi $f^*_{\mathcal{B}}$ to $(\mathbf{0}, 0)$ under a general norm $\|\cdot\|$. The GBM can be interpreted as an instance of this process where the norm is separable between the subgradient component and the linearization error component, i.e., $\|(z, \alpha)\| = D^*(z) + |\alpha|$, whereas the approach of [80] does not require this assumption. On the other hand, several important practical aspects of the method are not extensively discussed, and there is no computational indication that using more complex norms can significantly improve performances.

We finish this section mentioning that a generalized DSBM (cf. Sect. 3.2.3) should be possible with

$$\min \left\{ \underline{f}^i(\bar{\boldsymbol{x}} + \boldsymbol{d}) + D_{\mu^i}(\boldsymbol{d}) \ : \ \underline{f}^i(\bar{\boldsymbol{x}} + \boldsymbol{d}) \leq l^i \right\}. \tag{3.53}$$

Somewhat surprisingly, to derive a meaningful dual it is simpler to start with Lagrangian duality (as opposed to Fenchel's)

$$\begin{aligned} &\max_{\rho \geq 0} \{ -\rho l^i + \min \left\{ \left\{ (1+\rho) \underline{f}^i(\bar{\boldsymbol{x}} + \boldsymbol{d}) + D_{\mu^i}(\boldsymbol{d}) \right\} \right\} \\ &\qquad = [-] \min_{\rho \geq 0} \left\{ \rho l^i + (1+\rho)(\underline{f}^i)^*(z/(1+\rho)) + \langle z, \bar{\boldsymbol{x}} \rangle + D^*_{\mu^i}(-z) \right\}, \end{aligned}$$

although in the second step one does apply (3.50)/(3.51) (together with standard properties of the conjugate, among which $(\gamma f(\cdot))^*(z) = \gamma f^*(z/\gamma)$ for $\gamma > 0$). Then, using (3.2)/(3.52) for $\underline{f}_{\mathcal{B}} = \check{f}_{\mathcal{B}}$ one gets

$$\begin{aligned} \min\{\rho l^i + (1+\rho) \sum_{b \in \mathcal{B}} \alpha^b \theta^b + \langle z, \bar{\boldsymbol{x}} \rangle + D^*_{\mu^i}(-z) \ : \ &\sum_{b \in \mathcal{B}} z^b \theta^b = z/(1+\rho), \\ &\boldsymbol{\theta} \in \Theta, \quad \lambda \geq 0\}, \end{aligned}$$

that via the (nonlinear) rescaling $\boldsymbol{\theta} \leftarrow (1+\rho)\boldsymbol{\theta}$ finally becomes

$$\begin{aligned} \min \{\rho l^i + \sum_{b \in \mathcal{B}} \alpha^b \theta^b + \langle \sum_{b \in \mathcal{B}} z^b \theta^b, \bar{\boldsymbol{x}} \rangle + D^*_{\mu^i}(-\sum_{b \in \mathcal{B}} z^b \theta^b) \ : \ &\sum_{b \in \mathcal{B}} \theta^b = 1 + \rho, \\ &\boldsymbol{\theta} \geq \mathbf{0}, \quad \rho \geq 0\}, \end{aligned}$$

readily generalizing (3.34). To the best of our knowledge, this derivation is new; convergence of the GDSBM has not yet been firmly established, although it should follow easily enough by combining [36] with [25].

One property of all the stabilizing approaches discussed so far is that they are completely independent on the specific choice of f, comprised the fact that is has, or not, the form (3.39). While this is some sense an advantage, it also means that the stabilizing terms are not, on the outset, capable of *exploiting* any available information about the form of f. However, besides the stabilizing term D_μ, (3.50)/(3.51) also depend on the model $\underline{f}_{\mathcal{B}}$. So far we have mostly assumed the use of the cutting-plane model $\check{f}_{\mathcal{B}}$, but most of the convergence arguments only require very few specific properties from $\underline{f}_{\mathcal{B}}$ [36]. Indeed, the model can be chosen to exploit specific properties of f, as discussed in the next section.

3.4 Alternative Models

This section is devoted to improvements of the BM that pertain to using "better models" of f. Since all these are, in essence, orthogonal to the details of the stabilization, we will only present them in the context of the standard PBM (which is where, actually, they have for the most part been discussed), with the understanding that they could be applied to the other forms with some (possibly not entirely trivial) adjustment of the convergence analysis.

3.4.1 Quadratic Models

Following well-established approaches in nonlinear optimization, the first idea that would likely spring to mind is to use quadratic models of f, in order to capture its second-order behaviour. As already remarked, this is possible using sophisticated tools that are beyond the scope of this treatment (cf. e.g. (3.11)). Yet, some attempts have used simpler techniques that are based on the concept that $\underline{f}_{\mathcal{B}}$ should not "deviate too much" from $\check{f}_{\mathcal{B}}$.

One such model is the piecewise quadratic [2]

$$\check{f}_{\mathcal{B}} = \max \left\{ q^b(\boldsymbol{x}) = f(\boldsymbol{x}^b) + \langle \boldsymbol{z}^b, \boldsymbol{x} - \boldsymbol{x}^b \rangle + \epsilon^b \|\boldsymbol{x} - \boldsymbol{x}^b\|_2^2/2 \ : \ b \in \mathcal{B} \right\},$$

i.e., the pointwise maximum of the quadratic expansions q^b of f generated at each $\boldsymbol{x}^b$; note that, unlike for $\check{f}$, this clearly requires keeping the $\boldsymbol{x}^b$ in $\mathcal{B}$. This is in general not a valid lower model of f, unless all $\epsilon^b = 0$ in which case it falls back to $\check{f}_{\mathcal{B}}$; yet, it is easy to compute "small enough" ϵ^b such that $\check{f}_{\mathcal{B}}(\boldsymbol{x}^b) \leq f(\boldsymbol{x}^b)$ for all $\boldsymbol{x}^b$, i.e., the model never *knowingly overestimates* f. Actually, it is sufficient to guarantee the property only for a subset of the previous iterates, possibly only the

current stability center $\bar{\boldsymbol{x}}$. The model can be translated with respect to $\bar{\boldsymbol{x}}$, although this now requires

$$\breve{\alpha}^b = \alpha^b - \epsilon^b \|\bar{\boldsymbol{x}} - \boldsymbol{x}^b\|_2^2/2 \quad \text{and} \quad \breve{\boldsymbol{z}}^b = \boldsymbol{z}^v + \epsilon^b(\bar{\boldsymbol{x}} - \boldsymbol{x}^b),$$

which allows to define the "doubly stabilized" MP

$$\min \left\{ v + \mu^i \|\boldsymbol{d}\|_2^2/2 \ : \ v \geq \ \epsilon^b \|\boldsymbol{d}\|^2/2 + \langle \breve{\boldsymbol{z}}^b, \boldsymbol{d} \rangle - \breve{\alpha}^b \quad b \in \mathcal{B}^i, \ \gamma^i \|\boldsymbol{d}\|_2^2 \leq 2 \right\}$$

having both a proximal term weighted with μ^i and a TR one governed by γ^i. The rationale for the TR in the L_2-norm is that the problem is a quadratically constrained QP anyway, so there is no significant penalty in an extra quadratic constraint. The MP is actually a second-order cone program (SOCP); this is more easily seen computing its dual

$$\min \ \left\{ \frac{\|\sum_{b \in \mathcal{B}^i} \breve{\boldsymbol{z}}^b \theta^b\|_2^2}{2(\mu^i + \rho + \sum_{b \in \mathcal{B}^i} \theta^b \epsilon^b)} + \sum_{b \in \mathcal{B}^i} \breve{\alpha}^b \theta^b + \frac{\rho}{\gamma^i} \ : \ \boldsymbol{\theta} \in \Theta \ , \ \rho \geq 0 \right\},$$

where the apparently nasty fractional term in the objective function can be transformed into a rotated SOCP constraint with a well-known reformulation trick. Hence, the MP can be solved with off-the-shelf IP methods, at a cost comparable with a convex QP of the same size. All this allows to define a convergent BM, whose two stability parameters can be quite freely managed: indeed, as soon as at least one ϵ^b is strictly positive, one can even take $\mu^i = \rho^i = 0$, as the quadratic model is "self stabilizing". The convergence arguments follow the standard pattern of BM; the only nontrivial step is aggregation, as together with $\bar{\boldsymbol{z}}^i$ and $\bar{\alpha}^i$ one must also compute a $\bar{\boldsymbol{x}}^i$ to match, which requires some appropriate but overall simple computation. While the results seemed to show that this model was in fact capable of improving practical performances with respect to a standard PBM, this happened only with functions f that had the same piecewise quadratic nature as $\breve{f}_{\mathcal{B}}$.

Another recent take on the approach [56] is different in two key aspects:

1. it insists in having only *one* quadratic term by modifying the proximal term in (3.26) into $\boldsymbol{d}^T H^i \boldsymbol{d}$, a-la (3.11);
2. it insists on not *underestimating* the cutting-plane model too much.

The basic formula can be written in a "poorman's" setting (cf. Sect. 3.3.2), where one has the aggregated linearization $(\bar{\boldsymbol{z}}^i \ , \ \bar{\alpha}^i \ )$ and just one other linearization $(\boldsymbol{z}^i \ , \ \alpha^i \ )$; then,

$$\begin{aligned} \langle \bar{\boldsymbol{z}}^i, \boldsymbol{d} \rangle - \bar{\alpha}^i + \tfrac{1}{2} \boldsymbol{d}^T H \boldsymbol{d} \geq \langle \boldsymbol{z}^i, \boldsymbol{d} \rangle - \alpha^i - \varepsilon \quad \text{for all } \boldsymbol{d} \in \mathbb{R}^n \quad \equiv \\ H \geq \tfrac{1}{2(\alpha^i - \bar{\alpha}^i + \varepsilon)} (\bar{\boldsymbol{z}}^i - \boldsymbol{z}^i)(\bar{\boldsymbol{z}}^i - \boldsymbol{z}^i)^T. \end{aligned} \tag{3.54}$$

Note that $\bar{\alpha}^i \leq \alpha^i + \varepsilon$ must hold for (3.54) to have any chance to hold (set $\boldsymbol{d} = \boldsymbol{0}$), i.e., the scaling term must be positive; apart from that ε is "free" and can serve as a stabilization parameter. One can then build a semidefinite program (SDP) with as many semidefinite constraints of the form (3.54) as there are elements in $\mathcal{B}$ to find the least-curvature H that ensures that $\langle \bar{\boldsymbol{z}}^i, \boldsymbol{d} \rangle - \bar{\alpha}^i + \frac{1}{2}\boldsymbol{d}^T H \boldsymbol{d} \geq \check{f}^i(\boldsymbol{d}) - \varepsilon$; this can be shown in simple cases (f convex quadratic) to be reasonably related with the Hessian. Because solving the SDP at each step would be too costly, an approximate solution can be obtained by computing the singular value decomposition of an appropriate matrix (think that with $\bar{\boldsymbol{z}}^i - \boldsymbol{z}^b$ as columns) and taking "a few" of the columns corresponding to the largest singular values. This has been shown to be quite successful in improving practical convergence speed of the BM in one application.

Albeit interesting in their own right, the previous two models are "general-purpose": they do not make any assumption on f, and therefore arguably cannot exploit any of its specific properties. In the next sections we will instead describe models that exploit different forms of structure in f.

3.4.2 Disaggregate Models

Perhaps the most frequent structure in f is the sum one, i.e., $f(\boldsymbol{x}) = \sum_{k \in \mathcal{K}} f_k(\boldsymbol{x})$ where $\mathcal{K}$ is a finite index set. A prolific source of this kind of problems is the Lagrangian one, in which U in (3.40)—or, for that matter, conv(U) in (3.42)—is a Cartesian product: $U = \bigoplus_{k \in \mathcal{K}} U_k$, so that

$$\max \left\{ \sum_{k \in \mathcal{K}} \langle \boldsymbol{c}_k, \boldsymbol{u}_k \rangle \ : \ \sum_{k \in \mathcal{K}} A_k \boldsymbol{u}_k = \boldsymbol{b}, \quad \boldsymbol{u}_k \in U_k \quad k \in \mathcal{K} \right\} \tag{3.55}$$

for $\boldsymbol{u} = [\, \boldsymbol{u}_k \,]_{k \in \mathcal{K}}$, and therefore

$$f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{b} \rangle + \sum_{k \in \mathcal{K}} [\, f_k(\boldsymbol{x}) = \max \{ \, \langle \boldsymbol{c}_k - \boldsymbol{x} A_k, \boldsymbol{u}_k \rangle \ : \ \boldsymbol{u}_k \in U_k \, \} \,] \,. \tag{3.56}$$

For each $k \in \mathcal{K}$, any optimal solution $\boldsymbol{u}_k(\boldsymbol{x})$ of (3.56) provides the individual function value $f_k(\boldsymbol{x}) = \langle \boldsymbol{c}_k - \boldsymbol{x} A_k, \boldsymbol{u}_k(\boldsymbol{x}) \rangle$ and the individual subgradient $\boldsymbol{z}_k = -A_k \boldsymbol{u}_k(\boldsymbol{x}) \in \partial f_k(\boldsymbol{x})$. We immediately remark that there is a small (and intended) inconsistency between (3.56) and the original definition, in that in the former there actually are $|\mathcal{K}| + 1$ components of the sum, comprised the linear one $\langle \boldsymbol{x}, \boldsymbol{b} \rangle$; clearly such a term can (and should) be dealt with in a specific way, as discussed in details in Sect. 3.4.3. Disregarding this point for the time being, one could obviously define the *aggregated* function value and subgradient out of the individual $f_k(\boldsymbol{x})$ and $\boldsymbol{z}_k$, and then fall back to the previously developed theory. However, there is clearly another

alternative: defining *individual models* for each component, say the cutting-plane ones

$$\check{f}_k^i(\boldsymbol{x}) = \max\left\{ \langle z_k^b, \boldsymbol{x}\rangle - \alpha_k^b \;:\; b \in \mathcal{B}_k^i \right\} \leq f_k(\boldsymbol{x}) \tag{3.57}$$

depending on *individual bundles* $\mathcal{B}_k^i = \{\, (z_k^b, \alpha_k^b) = \langle z_k^b, \boldsymbol{x}^b\rangle - f_k(\boldsymbol{x}^b) = f_k^*(z_k^b) \,\}$. We can still refer to $\mathcal{B} = [\,\mathcal{B}_k\,]_{k\in\mathcal{K}}$ as "the bundle", and still avoid to distinguish between the untranslated α_k^b and the linearization errors $\alpha_k^b(\bar{\boldsymbol{x}}) = \alpha_k^b - \langle z_k^b, \bar{\boldsymbol{x}}\rangle + f_k(\bar{\boldsymbol{x}})$ (cf. (3.21)) unless strictly necessary. It is then immediate to define the *disaggregate master problem*

$$\min\left\{ \langle \boldsymbol{b}, \boldsymbol{d}\rangle + \sum_{k\in\mathcal{K}} v_k + \frac{\mu^i}{2}\|\boldsymbol{d}\|_2^2 \;:\; v_k \geq \langle z_k^b, \boldsymbol{d}\rangle - \alpha_k^b,\; b \in \mathcal{B}_k^i \;\; k \in \mathcal{K} \right\} \tag{3.58}$$

using the disaggregate model instead of the original (aggregated) one. It is quite obvious that, for the same set of information produced by evaluating f (all the f_k), (3.58) provides a better representation of f than (3.26). This is clearer when comparing the dual of (3.58)

$$\min\left\{ \frac{1}{2\mu^i}\|\boldsymbol{b} + \sum_{k\in\mathcal{K}}\sum_{b\in\mathcal{B}_k^i} z_k^b\theta_k^b\|_2^2 + \sum_{k\in\mathcal{K}}\sum_{b\in\mathcal{B}_k^i} \alpha_k^b\theta_k^b \;:\; \boldsymbol{\theta}_k \in \Theta_k \;\; k \in \mathcal{K} \right\} \tag{3.59}$$

with (3.27): in fact, its is obvious that the latter is the restriction of the former obtained by imposing that all the multipliers θ_k^b corresponding to all the individual subgradients attained at the same iterate b have the same value. Intuitively, "gluing together" the individual z_k^b into one aggregated z^b just because they have happened to have been produced at the same iteration is rather arbitrary, as they are in fact independent information about independent functions. Analogously, individual aggregated pairs $(\,\bar{z}_k^i\,,\,\bar{\alpha}_k^i\,)$ can be obtained out of the solutions $\boldsymbol{\theta}_k^i$ of (3.59), and independently inserted in each $\mathcal{B}_k^i$; despite all them having been obtained with multipliers corresponding to one specific MP solution, there is no reason why two different pairs should be later on constrained to each other. Nowhere this is clearer than in the Lagrangian case (3.56): in this case (3.59) is equivalent to

$$[\,\langle \bar{\boldsymbol{x}}, \boldsymbol{b}\rangle +] \tag{3.60}$$
$$\max\left\{ \sum_{k\in\mathcal{K}} \langle \boldsymbol{c}_k - \bar{\boldsymbol{x}}A_k, \boldsymbol{u}_k\rangle - \frac{1}{2\mu^i}\|\sum_{k\in\mathcal{K}} A_k\boldsymbol{u}_k - \boldsymbol{b}\|_2^2 : \boldsymbol{u}_k \in U_k^i \;\; k \in \mathcal{K} \right\}.$$

In other words, the feasible region of (3.60) is a Cartesian product of convex hulls, whereas that of the aggregated (3.43) is the convex hull of a Cartesian product: it is

very easy to see that the former set (for, ideally, the same $\mathcal{B}_k^i$) is much larger than the latter one. All this justifies why *disaggregate BM* using (3.57) typically converge much faster than aggregated ones, all the rest being equal [6, 14, 41]; indeed, convergence happens when enough information has been accrued that allows to express the optimal solution, and disaggregate BM make much better use of the gathered information.

Of course, there is a negative aspect in using disaggregate models: the master problems are larger (roughly, "by a factor of $|\mathcal{K}|$"), and therefore potentially (much) more costly to solve. Therefore, the trade-off between aggregated and disaggregate BM strongly depends on the relative weight of the MP cost and of the f_k computation cost. Often, the increase in convergence speed obtained by using a disaggregate model is worth the extra MP cost. Indeed, by converging much faster the disaggregate BM can actually end up collecting *less* information that the aggregated one (while making much better use of it), so that the disaggregate MP simply does not have the time to become too large. However, if the subproblems (3.56) are easy but "many", the cost of the disaggregate MP can become by far the computational bottleneck of the algorithm.

In order to face this issue, an intuitively promising approach is *partial aggregation*. That is, one may partition $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2 \cup \ldots \cup \mathcal{K}_h$ into h disjoint subsets, and then define the corresponding partly aggregated functions, subgradients and linearization errors. This is clearly possible, with the size of the MP now increasing "only" of a factor of h, at the cost of some (but less than in the fully aggregated case) arbitrary information aggregation. It is still unclear how to choose h, and how to distribute the different components across the partition. Some experiments [82, Chapter 2] seemed to show a potential for the approach, in that a small h was sufficient to significantly increases convergence speed with respect to the fully aggregated case, becoming comparable to that of the fully disaggregate case as a fraction of the latter's MP cost. However, even within the same class of problems the trade-off was very dependent on the specific type of instance, and it seemed hard to devise dependable guidelines. In this line of approach, it might be useful if the partition could be *dynamic*; this is indeed possible, as advocated in [96]. By arbitrarily choosing any $\mathcal{Z} \subseteq \mathcal{K}$ one may insert in (3.58) *partly aggregated cuts*

$$\sum_{k \in \mathcal{Z}} v_k \geq \langle \sum_{k \in \mathcal{Z}} \boldsymbol{z}_k^i, \boldsymbol{d} \rangle - \sum_{k \in \mathcal{Z}} \alpha_k^i. \tag{3.61}$$

Recent results [56] indicate that such an approach may be promising, in particular by using disaggregate cuts for a small set of "critical" components (whose subgradients seem to vary rapidly, thus exhibiting nondifferentiable behaviour), while all the remaining ones are aggregated into one component. A specific application where this technique makes especially sense is two-stage stochastic programs, since there a partly aggregated cut has a clear meaning in terms of sub-sampled estimate of the true subgradient (cf. Sect. 3.5.2). It is not surprising, then, that good results have been reported, e.g. with a level-type BM [101]. For problems with fixed recourse,

aggregation rules can be defined that benefit from information about the function and exploit ideas already presented in the stochastic programming community, again with a significant practical effect [98]. Yet, the implementation details required to achieve good results seem to be rather dependent on the specific application; this therefore remains an interesting, but still wide open, research line.

3.4.3 Constraints and Easy Components

The sum-funcion structure paves the way for further exploiting the specific structure of some of the components. We have actually already seen this happening: the function (3.56) has the linear component $f_0(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$, which in the MP (3.58)/(3.60) is *not* treated like the other f_k, but rather "directly inserted in the model". The principle is readily applicable each time one of the components has the appropriate structure. That is, assume for simplicity that $\mathcal{K} = \{0, 1\}$, where f_1 is produced by a standard oracle, whereas f_0 is "easy" in the sense that it can be *directly written into the MP*:

$$\min \left\{ f_0(\bar{\boldsymbol{x}} + \boldsymbol{d}) + v_1 + \frac{\mu^i}{2}\|\boldsymbol{d}\|_2^2 \;:\; v_1 \geq \langle \boldsymbol{z}_1^b, \boldsymbol{d} \rangle - \alpha_1^b \quad b \in \mathcal{B}_1^i \right\}. \tag{3.62}$$

This is how $f_0(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x} \rangle$ was dealt with in (3.58), and it is also the standard treatment of constraints in BM: with $f^0 = \imath_X$, this amounts to adding the constraints "$\bar{\boldsymbol{x}} + \boldsymbol{d} \in X$" to (3.8). Obviously, the general assumption is that (3.62) is not much more costly to solve than (3.58), which happens e.g. when X is defined by a "small" set of "simple" (say, linear or conic) constraints, as in the original (3.3). Clearly, a BM using (3.62) necessarily has to work if a BM using (3.58) was: the MP has better (indeed, "perfect") knowledge of f_0. Extension to any number of "easy" and "standard" component is immediate.

It is now appropriate to remark that constraints can be dealt with dynamically, so that a polyhedron X represented by a very large (say, exponential) number of constraints can still be used under the standard assumption that an efficient separation algorithm exist. A revealing case is that of a Lagrangian component over a non-compact polyhedron, i.e.,

$$f_0(\boldsymbol{x}) = \max\{ \langle \boldsymbol{c}_0 - \boldsymbol{x} A_0, \boldsymbol{u}_0 \rangle \;:\; \bar{U}_0 \boldsymbol{u}_0 \leq \bar{\boldsymbol{u}}_0 \} \tag{3.63}$$

with $\bar{U}_0/\bar{\boldsymbol{u}}_0$ matrix/vector of appropriate dimension, for which $f_0(\boldsymbol{x}^i) = +\infty$ can happen. This means that one *ray* ω^i of the polyhedron exists (and is identified by whatever LP solver is used to compute f_0) that is also an ascent direction, i.e., $\bar{U}_0 \omega^i \leq 0$ and $\langle \boldsymbol{c}_0 - \boldsymbol{x}^i A_0, \omega^i \rangle > 0$. Obviously, the very same ray will prove unboundedness for any other $\boldsymbol{x}$ such that $\langle \boldsymbol{c}_0 - \boldsymbol{x} A_0, \omega^i \rangle > 0$; in other words, ω^i defines the *constraint* $\langle \boldsymbol{c}_0, \omega^i \rangle \leq \langle \boldsymbol{x}, A_0 \omega^i \rangle$ that must be satisfied by all points in the domain of f_0. Thus, each time $f_0(\boldsymbol{x}^i) = +\infty$ a new constraint can be

added to the MP that "cuts away" $\boldsymbol{x}^i$, thereby necessarily changing its solution. Constraints are in fact a slightly different form of linearization describing epi f_0, which we can call "vertical" since the coefficient of v_0 is 0, and can be added to $\mathcal{B}_0^i$ instead of the standard ones; this only means that the corresponding dual variables θ^b in (3.27) do not participate to (have 0 coefficient in) the simplex constraint. Assuming that *the oracle only reports a finite set of rays* (say, the extreme ones) and that *vertical linearizations are never removed from* $\mathcal{B}_0$, then $f_0(\boldsymbol{x}^i) = +\infty$ can only happen a finite number of times, and the BM is still provably convergent. Similarly, constraints describing any polyhedron X for which a separation algorithm is available can be dynamically added to the MP whenever $\boldsymbol{x}^i \notin X$.

However, vertical linearizations/constraints are in some sense "more delicate": removing or aggregating them is not as easy as with subgradients. Removal is possible with the usual rules—as soon as a serious step is performed, $\mathcal{B}^i$ can be entirely reset of either type of linearization—but no finite bound on $|\mathcal{B}|$ can be established. Furthermore, all this only works under the assumption that the set of constraints is finite in the first place. It is easy to see where the catch is by thinking to (3.1) in which f is "simple" (say, linear) and X is given by a separation oracle only reporting vertical linearizations. If X is not a polyhedron, the CPM would still work—actually, this is the very setting in which it has been defined [60]—but it is completely possible that $f(\boldsymbol{x}^i) = \infty$ for *all* iterates $\boldsymbol{x}^i$. While this is not a problem for the CPM, it is typically so for a BM, which is based on tests like (3.7) to manage the stabilization center. Hence, either some mechanism is required that ensures that the BM obtains $f(\boldsymbol{x}^i) < \infty$ "frequently enough", or some alternative test has to be employed. Customarily, BM dealing with "complicated" constraints assume $X = \{\boldsymbol{x} \in \mathbb{R}^n : c(\boldsymbol{x}) \leq 0\}$, where *both* f and c are finite-valued; hence this does not exactly cover the previous example. Note that $c(\cdot)$ can w.l.o.g. be taken as a scalar function, since any finite set of convex constraints can be turned into one by taking their maximum, which is still convex (but nondifferentiable). Finiteness of $c(\cdot)$ is crucial to implement *infeasible* BM, which can make good use of unfeasible iterates $c(\boldsymbol{x}^i) > 0$; the required theoretical tool is the *improvement function* $h_{\bar{\boldsymbol{x}}}(\boldsymbol{x}) = \max\{ f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}}) , c(\boldsymbol{x}) \}$ such that $\bar{\boldsymbol{x}}$ solves the constrained (3.1) if and only if it is an unconstrained minimizer of $h_{\bar{\boldsymbol{x}}}$, the optimal value then being $h_{\bar{\boldsymbol{x}}}(\bar{\boldsymbol{x}}) = 0$ [87]. Basically, a standard unconstrained BM—allowing, in particular, aggregation—can then be used to minimize $h_{\bar{\boldsymbol{x}}}$; subgradients of both f and c computed at previous iterates are separately kept, analogously to (3.58), and transformed into subgradients of $h_{\bar{\boldsymbol{x}}}$ by simple formulæ. If an appropriate improvement in the value of $h_{\bar{\boldsymbol{x}}}$ is attained, the stability center is changed; this also changes the objective function, but again existing information—comprised aggregated one—can be used to compute valid approximate subgradients to the new $h_{\bar{\boldsymbol{x}}}$, allowing the method to continuously accrue information as in the standard case. The algorithm can then be shown to converge; furthermore, if a feasible iterate is ever produced, then all subsequent iterates remain feasible. Alternatively, filter techniques can be used [59]. Under stronger assumptions on X, *feasible* BM can be constructed: for instance, [68] requires *knowledge* (and hence, a fortiori, existence) of a *Slater point* $\boldsymbol{x}_{int}$

such that $c(\boldsymbol{x}_{int}) < 0$. Hence, whenever an unfeasible iterate $\boldsymbol{x}^i$ is obtained, an *interpolated point* can be defined—not unlike in the IOA (cf. Sect. 3.2.5)—in the segment $[\,\boldsymbol{x}_{int}\,,\,\boldsymbol{x}^i\,]$ that is feasible, and therefore whose objective function value can be used in the descent test. A similar approach has been used in [94] in the context of chance constrained optimization; the specific advantage is that the computation of the chance constraint requires a costly numerical procedure that can be ill-conditioned for "extreme" $\boldsymbol{x}^i$ with very high or low probability, whereas it is more reliable and efficient for points "in the middle". Yet, for some applications experiments have shown that infeasible starts can actually be beneficial [91, 93].

Returning to the original subject of this paragraph, it is clear that inserting f_0 in the MP is not limited to linear or indicator functions, but can be done whenever the MP remains "reasonably easy". This is often the case of Lagrangian functions, where the underlying Lagrangian subproblems can have special structures that make their dual function manageable without resorting to linearizations. An interesting example are *nonlinear multicommodity network design problems with congestion costs*, if only because they have been tackled twice, once with an ACCPM [5] and once with a PBM [73]. In the problem, the objective function is nonlinear because of many single-variable terms of the form $k(t) = t/(c-t)$, where t is the total flow on an arc of the underlying network and c is its capacity; this is the widely used *Kleinrock's delay function*. Once the linking constraints are relaxed, one is typically left with many single-variable optimization problems of the form $f(x) = \min\{\, t/(c-t) - xt \;:\; 0 \le t < c \,\}$ (the original problem being a minimization one), each one depending on one single Lagrangian multiplier x; this immediately reveals itself as *the opposite of the conjugate of Kleinrock's delay function*, $-k^*(x)$. Due to its simple form this can be computed with a closed formula: $k^*(x) = 1 + cx - 2\sqrt{cx}$ whenever $x \ge 1/c$. Hence, f_0 in (3.62) is a sum of those terms. Directly inserting these in the MP results in a problem that is no longer a QP; to address this issue, in [73] f_0 is replaced with its second-order approximation around the current stability center $\bar{\boldsymbol{x}}^i$, resulting in a hybrid BM/Newton's method. Only relatively minor changes are required in the convergence analysis, all using well-understood techniques from smooth optimization (basically, an appropriate line search); furthermore, the Newton's term directly stabilizes the approach, removing the need for the "artificial" proximal stabilization $\|\boldsymbol{d}\|_2^2$. This is even less of an issue for the ACCPM, whose MP (3.16) is already not a QP: adding the terms corresponding to f_0 in the KKT system of the IP method (itself again basically Newton's method) is easy. In both cases, inserting "exact" information about one (many) component(s) of f in the MP is shown to significantly improve performances in practice.

The approach does not even require the conjugate being easy to compute: as advocated in [41], any Lagrangian function whose form is "not more complex than that of the MP" lends itself to the treatment. That is, consider

$$\max \left\{ \langle \boldsymbol{c}_0, \boldsymbol{u}_0 \rangle + \langle \boldsymbol{c}_1, \boldsymbol{u}_1 \rangle \;:\; \bar{U}_0 \boldsymbol{u}_0 \le \bar{\boldsymbol{u}}_0 \;,\; \boldsymbol{u}_1 \in U_1 \;,\; A_0 \boldsymbol{u}_0 + A_1 \boldsymbol{u}_1 = \boldsymbol{b} \right\},$$

where f_0 is again (3.63). The key is, again, duality: while the dual of (3.62)

$$\min \left\{ \frac{1}{2\mu^i} \|\boldsymbol{b} - z_0 - \sum_{b \in \mathcal{B}_1^i} z_1^b \theta_1^b\|_2^2 + \sum_{b \in \mathcal{B}_1^i} \alpha_1^b \theta_1^b - \bar{\boldsymbol{x}} z_0 + (f_0)^*(z_0) \ : \ \theta_1 \in \Theta_1 \right\}$$

may at first look intimidating, f_0^* is, basically, nothing else than the original Lagrangian subproblem: in other words, the above can be rewritten as

$$\max\{\langle \boldsymbol{c}_0, \boldsymbol{u}_0 \rangle + \sum_{b \in \mathcal{B}_1^i} \alpha_1^b \theta_1^b + \langle \bar{\boldsymbol{x}}, z \rangle - \frac{1}{2\mu^i} \|z\|_2^2 : \ z = \boldsymbol{b} - \sum_{b \in \mathcal{B}_1^i} z_1^b \theta_1^b - A_0 \boldsymbol{u}_0,$$
$$\bar{U}_0 \boldsymbol{u}_0 \leq \bar{\boldsymbol{u}}_0, \quad \theta_1 \in \Theta_1\}. \tag{3.64}$$

The idea is therefore straightforward when seen in the dual MP: for the "standard" component the usual linearization is employed, whereas the "easy" one is basically *inserted unchanged in the (dual) MP*. This can be done beyond LPs; for instance, if the objective function $c_0(\boldsymbol{u}_0)$ were convex quadratic, then (3.64) would still be a convex QP, hence roughly as hard to solve as the original MP. Again, any BM working with an approximated model $\underline{f}_0$ will *a fortiori* work if the "true" f_0 is used, once a few minor details are taken care of. Of course, trade-offs reveal themselves in practice: (3.64) may be more costly to solve, especially at the beginning, but the part concerning f_0 will never grow in size, as opposed to the part concerning f_1. Furthermore, by having an "exact" model for one component one can expect faster convergence, often quite significantly so, as repeatedly reported in applications as diverse as multicommodity network design problems [41], stochastic unit commitment problems [89], chance constrained optimization [99] and SDP relaxations for hard combinatorial problems [46].

3.4.4 Specialized Dynamic Models

Most (but not all) specialized models of Sect. 3.4.3 are "static", in that all the information corresponding to f_0 is known at the beginning of the solution process. This is clearly not necessary, as the present section will show.

A specialized model is the basis of the *spectral* BM (SBM) [54] for solving SDP. This starts with the fact that the dual of the standard SDP

$$\max \{ \langle C, U \rangle \ : \ AU = b, \quad U \succeq 0 \}$$

under mild assumptions can be recast as the eigenvalue optimization problem

$$\min \{ f(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x} \rangle + \lambda_{\max}(C - \boldsymbol{x} A) \},$$

with $\lambda_{\max}(\cdot)$ indicating the maximum eigenvalue of a matrix, a convex nondifferentiable function. Each time $f(\boldsymbol{x})$ is computed, by standard linear algebra techniques, any eigenvector $\boldsymbol{w}$ associated with the maximal eigenvalue produces a subgradient $\boldsymbol{z} = \boldsymbol{b} - A(\boldsymbol{w}\boldsymbol{w}^T)$; that is, $\partial f(\boldsymbol{x})$ is spanned by all possible such eigenvectors, and therefore f is differentiable only if the maximum eigenvalue has multiplicity one. Rather than the standard $\check{f}$, the SBM uses

$$\underline{f}_{\mathcal{B}}(\boldsymbol{x}) = \max\{\, \langle \boldsymbol{b}, \boldsymbol{x}\rangle + \langle C - \boldsymbol{x}A, W\rangle \;:\; W \in W_{\mathcal{B}} \,\},$$

with $W_{\mathcal{B}} = \{\, W = \theta \bar{W}_{\mathcal{B}} + P_{\mathcal{B}} V P_{\mathcal{B}}^T \;:\; \theta + \mathrm{tr}(V) = 1, \quad V \succeq 0 \,\}$. At first read, one can take the columns of the matrix $P_{\mathcal{B}}$ as being the (orthogonalized) eigenvectors $\boldsymbol{w}^b$ computed at previous iterations, and $\bar{W}_{\mathcal{B}}$ as corresponding to the aggregated subgradient $\bar{\boldsymbol{z}}^i$, although updating $P_{\mathcal{B}}$ and $\bar{W}_{\mathcal{B}}$ at each iteration requires some care. All in all, minimizing $\underline{f}_{\mathcal{B}}$ is a SDP, and a small-scale one if the size of $P_{\mathcal{B}}$ is kept in check; hence, it can be efficiently solved by IP methods. Clearly, adding a quadratic stabilizing term (or a TR in the L_2-norm, for that matter) does not significantly change the computational cost of the MP. However, note that the efficiency of the MP solution is strictly related to the fact that the main matrix variable V has small size; this, for instance, may change if constraints $\boldsymbol{x} \in X$ (even simple bounds) are present, requiring the use of nontrivial techniques [55]. Not surprisingly, using the specialized model is much more efficient than using $\check{f}_{\mathcal{B}}$, and it can be competitive with IP methods in particular for solving sparse large-scale SDP.

In the somewhat different context of Lagrangian functions of structured problems, a quite general class of models has been proposed. The idea is that the standard Dantzig–Wolfe reformulation of $\mathrm{conv}(U)$, which gives rise to the standard cutting-plane model (cf. Sect. 3.3.3), is not the only possible formulation that lends itself to dynamic generation. Motivated by results on 0-1 reformulations of multicommodity network design problems [39], general requirements have been defined for any other "large" formulation of $\mathrm{conv}(U) = \{\, \boldsymbol{u} = C\boldsymbol{\theta} \;:\; \Gamma\boldsymbol{\theta} \le \boldsymbol{\gamma} \,\}$ that can be "constructed piecemeal" [40]. In this setting, the bundle is $\mathcal{B} = (\mathcal{B}^c, \mathcal{B}^r)$, where $\mathcal{B}^c$ is a subset of the variables $\boldsymbol{\theta}$ (columns of Γ and C), and $\mathcal{B}^r$ is a subset of the constraints (rows in Γ) which impact *at least one* variable in $\mathcal{B}^c$; this immediately defines the restrictions $\boldsymbol{\theta}_{\mathcal{B}}$, $\Gamma_{\mathcal{B}}$, $\boldsymbol{\gamma}_{\mathcal{B}}$ and $C_{\mathcal{B}}$ of the formulation. The first requirement is that any partial solution can always be completed with zeroes, i.e., $\Gamma_{\mathcal{B}}\bar{\boldsymbol{\theta}}_{\mathcal{B}} \le \boldsymbol{\gamma}_{\mathcal{B}}$ and $\boldsymbol{\theta} = [\,\bar{\boldsymbol{\theta}}_{\mathcal{B}}\,, \mathbf{0}\,]$ $\Longrightarrow \Gamma\boldsymbol{\theta} \le \boldsymbol{\gamma}$; this immediately implies that

$$U_{\mathcal{B}} = \left\{\, v = C_{\mathcal{B}}\boldsymbol{\theta}_{\mathcal{B}} \;:\; \Gamma_{\mathcal{B}}\boldsymbol{\theta}_{\mathcal{B}} \le \boldsymbol{\gamma}_{\mathcal{B}} \,\right\} \subseteq \mathrm{conv}(U),$$

and therefore that

$$\underline{f}_{\mathcal{B}}(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{b}\rangle + \max\left\{\, \langle \boldsymbol{c} - \boldsymbol{x}A, C_{\mathcal{B}}\boldsymbol{\theta}_{\mathcal{B}}\rangle \;:\; \Gamma_{\mathcal{B}}\boldsymbol{\theta}_{\mathcal{B}} \le \boldsymbol{\gamma}_{\mathcal{B}} \,\right\}$$

is an alternative lower model of f. Hence, the MP can be defined that uses this model; again, this is more easily seen in the dual, which is just (with the obvious notation)

$$\max \left\{ \langle \boldsymbol{c}, \boldsymbol{u} \rangle + \langle \bar{\boldsymbol{x}}, \boldsymbol{b} - A\boldsymbol{u} \rangle - \frac{1}{2\mu^i} || A\boldsymbol{u} - \boldsymbol{b} ||_2^2 \ : \ \boldsymbol{u} = C^i \boldsymbol{\theta}_{\mathcal{B}^i} \ : \ \Gamma^i \boldsymbol{\theta}_{\mathcal{B}^i} \leq \boldsymbol{\gamma}^i \right\}$$

(cf. (3.43)). The other necessary assumption is that, once the oracle is called in $\boldsymbol{x}^i$ and a new $\boldsymbol{u}^i$ is obtained, it can be efficiently used to update $\mathcal{B}^i$. This can be stated (intentionally vaguely) as follows: if $\bar{\boldsymbol{u}} \in \text{conv}(U) \setminus U_{\mathcal{B}}$, it must be easy to update $\mathcal{B}$ to a set $\mathcal{B}' \supset \mathcal{B}$ such that there exists $\mathcal{B}'' \supseteq \mathcal{B}'$ with $\bar{\boldsymbol{u}} \in U_{\mathcal{B}''}$. In plain words, one has to be able to see what are the missing variables and constraints in the formulation, and add at least some of them; this is basically a "dynamic version of the easy components approach". It is then possible to develop a BM using this approach, which generalizes the Dantzig–Wolfe decomposition/column generation; this has been proven efficient in several applications [39, 86, 89].

3.4.5 *Diminishing the MP Cost*

Many of the ideas discussed in the last sections lead to "large" MP; solving them can therefore easily become the computational bottleneck. Although the increased convergence speed may still make these large MP worthwhile, it is clear that techniques for lessening this computational cost could be crucial; some notable developments are discussed here.

The first have to do with the fact that $\boldsymbol{x}$ can be a "very long" vector. For instance, in the Lagrangian case (3.39), the constraints $A\boldsymbol{u} = \boldsymbol{b}$ can be exponentially many (of course, with an attached efficient separation routine), or anyway a very large number. Especially if the constraints are inequalities, though, one can expect only a small fraction of them actually being active at optimality, which means that only a small fraction of the components of $\boldsymbol{x}$ will be different from 0 at any optimal solution. One can then define a *dynamic* BM (DBM)—be it a PBM [10, 38, 42], a SBM [53], an ACCPM [5] or any other, even with specialized models [41], and comprised subgradient-type methods [44]—that has, basically, a simple *active-set* strategy on $\boldsymbol{x}$. At the beginning, only a small subset of the variables (constraints in the dual) is actually defined in the MP, which is therefore smaller and cheaper. An arbitrary number of iterations can be performed with $\boldsymbol{x}$ restricted to the active set; then, occasionally—but surely if convergence in the current subspace is detected—one has to check the entire subgradient to see if some components need to be added to the active set. In the Lagrangian case, this simply amounts at verifying which of the constraints (say) $A\boldsymbol{u} \leq \boldsymbol{b}$ are violated by the *aggregated primal solution* $\bar{\boldsymbol{u}}^i$ (cf. (3.44)). If no components are ever removed from the active set the DBM is trivially convergent: after a finite number of updates the active set is the full space, and "true convergence" begins. Careful removal is also possible with mild

assumptions on the separation process [10], although in practice the technique works well even without them. This has been shown to considerably improve performances, especially when the cost of the f computation is low [38, 41, 42, 44].

The approach of [67] rather deals with approximately solving MPs for sum-structured f (cf. Sect. 3.4.2). The idea is (again and again) quite simple when seen from the dual viewpoint: each of the (for simplicity) two components has distinct dual variables, say $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$. It is then easy to implement a *block descent approach*, where $\boldsymbol{\theta}_0$ is kept fixed to some (feasible) value and $\boldsymbol{\theta}_1$ is optimized, and then the roles are reversed. This may also work when the $\boldsymbol{\theta}_k$ are not convex multipliers, say as in (3.64), and potentially when there are more than two, although to the best of our knowledge this has never been studied. From the primal viewpoint, this means that one of the two models $\underline{f}_0$ and $\underline{f}_1$, in turn, is substituted with its aggregated linearization $(\bar{\boldsymbol{z}}_h^i, \bar{\alpha}_h^i)$. This may allow to use specialized solvers that exploit the individual structure of the two separate subproblems. For instance, $\underline{f}_0$ may be the indicator function of a "simple" set X, say a box, whereas $\underline{f}_1$ may be the standard cutting-plane model; then, the first subproblem is the optimization of a linear function on a box, whereas for the second specialized algorithms exist that are more efficient in the unconstrained case [34] (although the latter algorithm specifically deals with box constraints with a technique that is not entirely uncorrelated with the one we are discussing). A very small number of iterations, down to only one, may be sufficient to construct a direction $\boldsymbol{d}^i$ that allows to continue the BM, thus potentially reducing the MP cost. The approach may applied many different structures, see e.g. again [55].

Finally, it is clearly always possible to specialise well-known approaches to the specific structure of the MP. This is the case of [34] for active-set methods and of [74] for structure-exploiting IP methods applied to the parallel solution of the disaggregate MP (3.58)/(3.59).

3.5 Inexact and Incremental Approaches

Overall, the computational cost of the BM depends on both the number of iterations and their cost, in turn the sum of the MP cost and of the oracle cost. Clearly, reducing the number of iterations (improving convergence speed) is of paramount importance to reduce the total cost, and it has been therefore the focus of basically all the discussion so far. Indeed, often paying a larger MP cost to reduce the number of iteration is worth, although of course the cost of the MP must also be kept in check (cf. Sects. 3.3.2, 3.4.2, 3.4.4, and 3.4.5). What has not been discussed so far are methods to decrease the oracle cost. These would hardly seem to be subject of a general treatment in BM, since they clearly depend on the specific application giving rise to (3.1); however, general concepts of *approximate oracle* can be defined, whereby one loosens the requirement that $f(\boldsymbol{x})$ be computed exactly, and that $\boldsymbol{z} \in \partial f(\boldsymbol{x})$. This can be clearly beneficial, if only in the Lagrangian case (3.39)

where the oracle is an optimization problem (but then again, any (3.1) can be considered a dual problem, cf. (3.47)); allowing to solve it approximately should reasonably decrease its cost. This is the subject of the present section.

3.5.1 Inexact Approaches

The Lagrangian case is indeed a good one to inform the discussion: an approximate oracle for (3.39) (with $\boldsymbol{x} = \boldsymbol{x}^i$) might just compute any feasible solution $\underline{\boldsymbol{u}}^i \in U$, hopefully a "good" one. Using $\underline{\boldsymbol{u}}^i$ instead of the optimal solution $\boldsymbol{u}^i$ yields a lower bound $\underline{l}^i = \langle \boldsymbol{c} - \boldsymbol{x}^i A, \underline{\boldsymbol{u}}^i \rangle + \langle \boldsymbol{x}^i, \boldsymbol{b} \rangle \le f(\boldsymbol{x}^i)$, together with $\underline{\boldsymbol{z}}^i = \boldsymbol{b} - A\underline{\boldsymbol{u}}^i$ such that $\underline{\boldsymbol{z}}^i \in \partial_{\varepsilon^i} f(\boldsymbol{x}^i)$ for the error $\varepsilon^i = f(\boldsymbol{x}^i) - \underline{l}^i \ge 0$. That is, such an approximated oracle delivers ε-subgradients rather than subgradients, and lower approximations to the function value. Crucially, the (say) cutting-plane model constructed with this information is still is a valid lower model, which makes it surprisingly easy to define an *inexact BM* (IxBM). In fact, assuming that $\varepsilon^i \to 0$ "naturally" along the iterations, there is basically nothing to do [62]. In other words, as in the case of subgradient-type methods [22], what really counts for BM is the *asymptotic maximum error* $\varepsilon^\infty = \limsup_{i\to\infty} \varepsilon^i$: any "large" error $\varepsilon^i \gg \varepsilon^\infty$ occurring in the early iterations can be automatically corrected as the algorithm proceeds towards the optimum. This is an attractive feature in that, intuitively, it should not be required that the function be computed with high accuracy at the beginning of the algorithm, while the error reasonably need be reduced when approaching the optimal solution. However, such an *asymptotically exact* oracle is not necessary: a BM can converge under the quite minimal condition that $\varepsilon^i \le \bar{\varepsilon} < \infty$, with $\bar{\varepsilon}$ fixed but *not necessarily known*. Of course, in this case one can expect nothing better than a $\bar{\varepsilon}$-optimal solution [22, Observation 2.7].

In order to ensure convergence, though, some modifications are necessary. This stems from the fact that defining the linearization errors as $\underline{\alpha}^b(\bar{\boldsymbol{x}}) = \underline{l}^i - [\, \underline{l}^b + \langle \boldsymbol{z}^b, \bar{\boldsymbol{x}} - \boldsymbol{x}^b \rangle \,]$, i.e., using the lower estimates in place of the function values $f(\bar{\boldsymbol{x}}^i)$ and $f(\boldsymbol{x}^b)$, may lead to $\underline{\alpha}^b < 0$. Indeed, $\boldsymbol{z}^b$ is a $(\underline{\alpha}^b + \varepsilon^b)$-subgradient of f at $\bar{\boldsymbol{x}}$, with $\underline{\alpha}^b + \bar{\varepsilon} \ge \underline{\alpha}^b + \varepsilon^b \ge 0$; yet, $\bar{\varepsilon}$ and ε^b are unknown. In turn, when put e.g. in (3.28) this may lead to $v^i > 0$, i.e., $\boldsymbol{d}^i$ not being a descent direction. The point is that $\bar{\boldsymbol{x}}^i$ has been chosen as the stability center on the basis of $\underline{l}^i$, implicitly assuming it to be a reasonable approximation of the function value; yet, later on other information inserted in $\mathcal{B}^i$ reveals that in fact $\underline{l}^i \ll f(\bar{\boldsymbol{x}}^i)$. A possible solution, originally due to [66], is to exploit the fact that any BM has one (or more) proximal parameter(s), that can be almost freely adjusted. The idea is that whenever $v^i > 0$ the proximal parameter is adjusted so that the MP becomes *less* stabilized—say, μ^i is reduced in the PBM—so that $v^{i+1} < v^i$, hopefully becoming negative (enough). This is called a *noise reduction* (or noise attenuation) step, because the "noise" in the function computation is higher than the "signal" corresponding to v^i; by increasing v^i (in absolute value), the signal-to-noise ratio also increases (being the error bounded). With minimal care, a finite number of noise reduction steps leads to two possible

outcomes. The first is that the solution $\boldsymbol{x}^i$ of the MP becomes a solution of (3.3), i.e., a global minimum of the model $\underline{f}^i$; in this case, $\bar{\boldsymbol{x}}$ is $\bar{\varepsilon}$-optimal and the BM can stop (it actually has to, as there is no other recourse). Otherwise, v^i will eventually become "sufficiently negative", and the normal course of the BM can resume. This approach has been shown to work for the PBM under even looser assumptions on the oracle, i.e., $\underline{l}^i$ may not even be a guaranteed lower bound on $f(\boldsymbol{x}^i)$ and $\boldsymbol{z}^i$ may not even be a guaranteed ε-subgradient at $\boldsymbol{x}^i$, provided that the errors are suitably bounded [26, 93]. The PLBM has some different technicalities [24], in particular in the constrained case [92]; interestingly, the DSBM does not require a noise reduction step at all, since the level constraint always ensures that $v^i < 0$ [25, Section 4].

The previous analysis assumed no control on the oracle error, but this is not the only possible case. There have been different definitions of *controllable* inexact oracles [24, 26, 75, 92], but perhaps the most complete is that of the inexact *informative, cooperative* oracle of [96]. This takes in input, besides $\boldsymbol{x}$, *three* parameters $-\infty \leq \underline{\tau} \leq \overline{\tau} \leq \infty$ (the *lower and upper targets*, with $\overline{\tau} > -\infty$ and $\underline{\tau} < \infty$), and $0 \leq \varepsilon \leq \infty$ (the *accuracy*), and provides

$$\begin{aligned}
&\text{(i)} \quad \textit{function value information}\text{: } \textit{two} \text{ values } \underline{f} \text{ and } \bar{f} \text{ such that} \\
&\qquad -\infty \leq \underline{f} \leq f(\boldsymbol{x}) \leq \bar{f} \leq \infty, \quad \bar{f} - \underline{f} \leq \varepsilon, \text{ and} \\
&\qquad \textit{at least one} \text{ between } \bar{f} \leq \overline{\tau} \text{ and } \underline{f} \geq \underline{\tau} \text{ holds;} \qquad (3.65) \\
&\text{(ii)} \quad \textit{first-order information}\text{: if } \underline{f} > -\infty, \text{ a } \boldsymbol{z} \\
&\qquad \text{such that } f(\cdot) \geq \underline{f} + \langle \boldsymbol{z}, \cdot - \boldsymbol{x} \rangle.
\end{aligned}$$

It is always possible to attain (3.65), possibly at the cost of computing $f(\boldsymbol{x})$ with high accuracy, but the many parameters "allow to stop computation earlier". In particular, if $\bar{f} \leq \overline{\tau}$ then it is possible to return $\underline{f} = -\infty$, and hence *no linearization* $\boldsymbol{z}$ *at all*. This is motivated by the Lagrangian case in which (3.39) is *hard*, say a mixed-integer linear problem (MILP), whose solution process actually amounts at *three* different parts:

1. finding a feasible solution $\underline{\boldsymbol{u}} \in U$ (hence $\underline{f}$ and $\boldsymbol{z}$) by appropriate *heuristics*;
2. producing an upper bound $\bar{f}$ by the exact solution of some *relaxation* of (3.39), or a feasible solution of an appropriate dual problem;
3. if $\underline{f}$ and $\bar{f}$ are not "close enough", performing an arbitrary amount of branching and/or cutting and running 1. and 2. again.

The three parameters have different roles in stopping the process, and are not redundant. If $\varepsilon = \infty$, the thresholds $\overline{\tau}/\underline{\tau}$ may allow to stop after that step 2./1. above (respectively) have been ran, *possibly without even running the other one* (and therefore, in the case of $\overline{\tau}$, not even producing $\boldsymbol{z}$). If, instead, a finite ε is given, stopping requires both bounds, but is independent from which of the two thresholds

is satisfied. It is possible to set "minimal" values for the parameters ($\overline{\tau}$ and ε as large as possible, $\underline{\tau}$ as small as possible) that ensure convergence of a IxBM, thereby hopefully reducing the computational cost of (3.39) as much as possible. It has to be remarked, though, that doing so may potentially impact convergence speed, a trade-off that has not been well enough investigated in practice yet.

Oracle (3.65) is *collaborative* in that it must in principle be able to compute the function with arbitrary accuracy, although the BM can strive to keep the requirements at a minimum. Not in all cases it is possible, or reasonable, to do so: some oracles (problems) may only be solvable up to some specific accuracy $\bar{\varepsilon}$. Actually, there are *three* different ways in which this can happen. The first is that $\bar{\varepsilon}$ is explicitly known beforehand. Otherwise, the oracle may stop with $\varepsilon < \bar{f} - \underline{f} \leq \bar{\varepsilon}$, but still produce correct upper and lower estimate. Finally, the oracle can "cheat" by (say) reporting $\bar{f} = \underline{f}$, thus formally respecting $\bar{f} - \underline{f} \leq \varepsilon$, but doing so at the cost of returning incorrect information. It turns out [96, Section 4] that each of the three cases corresponds to an entirely different noise reduction step, where μ^i is decreased in response to a different condition; in all these cases, convergence of the IxBM to a $\bar{\varepsilon}$-optimal solution can be proven.

3.5.2 Incremental Approaches

Another (albeit strictly related) way in which the oracle cost can be reduced is specific to sum-functions (cf. Sect. 3.4.2). There, "the oracle" is actually a set of separate oracles, one for each $k \in \mathcal{K}$: in alternative/addition to allowing approximate computation in each of them separately, a rather drastic way of saving computation time is to *completely avoid to call some of them*. Hence, at each iteration one has (possibly, approximate) f-values and subgradients only for some subset $\mathcal{Z} \subseteq \mathcal{K}$ of the components, out of which the estimates $f_{\mathcal{Z}}(\boldsymbol{x}^i) = \sum_{k\in\mathcal{Z}} f_k(\boldsymbol{x}^i)$ and $\boldsymbol{z}_{\mathcal{Z}} = \sum_{k\in\mathcal{Z}} \boldsymbol{z}_k$ are obtained. A BM doing so is called *incremental* (IcBM) by analogy with incremental subgradient-type methods [13, 44, 65]. The latter are in turn closely related with *stochastic subgradient* methods for stochastic optimization and *mini-batch* approaches in machine learning; there, each f_k is a specific *realization* of a stochastic process or *sample* of a process to learn, again ideally drawn at random from an infinite set. Thus, for a random $\mathcal{Z}$, there can be hope that $\boldsymbol{z}_{\mathcal{Z}}$ be a reasonable estimate of the true (stochastic) subgradient, and hence a rationale for using it to define the step. In fact, convergence for these methods is perhaps more naturally proven in a probabilistic sense; deterministic results require to compute the "full" f ($\mathcal{Z} = \mathcal{K}$)—a *batch iteration* in ML parlance—often enough. This kind of analysis is not well suited for BM.

However, at least with the disaggregate model (cf. Sect. 3.4.2) it is easy enough to construct a IcBM by, basically, considering it a IxBM. Indeed, for each $k \notin \mathcal{Z}$ one can pretend that the model information at $\boldsymbol{x}^i$, say $\underline{f}_k^i(\boldsymbol{x}^i)$ and $\bar{\boldsymbol{z}}^i$, is the output of an approximate oracle; thus, it is possible to analyse IcBM using, say, the very general results of [26], as done in [31]. However, the IcBM— at least, with exact

individual sub-oracles—corresponds to a *controllable* oracle, in that by evaluating more and more components it is possible to arbitrarily reduce the error; hence, one would expect to be able to do without any noise reduction step. What is actually easy is declaring a null step by only evaluating a subset of all the components. In fact, since $\underline{f}_k^i(\boldsymbol{x}^i) \leq f_k(\boldsymbol{x}^i)$, clearly $\Delta f^i = \sum_{k\in\mathcal{K}}(\Delta f_k^i = f_k(\boldsymbol{x}^i) - \underline{f}_k^i(\boldsymbol{x}^i)) \geq \Delta f_{\mathcal{Z}}^i = \sum_{k\in\mathcal{Z}} \Delta f_k^i$. Hence, if $\Delta f_{\mathcal{Z}}^i > -m v^i$ implies that a fortiori (3.37) holds, and therefore (3.38) does. Declaring a serious step is instead trickier, as any unevaluated component may counterbalance the descent of all the evaluated ones with a very steep ascent. One possible strategy is to only perform incremental null step, while requiring a "full" iteration ($\mathcal{Z} = \mathcal{K}$) to declare a serious step, analogously to what incremental subgradients do. A different approach has been proposed in [96], under the assumption that all the f_k are Lipschitz continuous with *known* constant L_k. This allows to perform incremental serious step as well by using the *upper model*

$$\hat{f}_{\mathcal{P}}^k(\boldsymbol{x}) = \min \left\{ \sum_{p\in\mathcal{P}_k} f_k^p \theta_k^p + L_k \|\boldsymbol{s}_k\|_2 \ : \ \sum_{p\in\mathcal{P}_k} \boldsymbol{x}^p \theta_k^p + \boldsymbol{s}_k = \boldsymbol{x} \ , \ \boldsymbol{\theta}_k \in \Theta_k \right\},$$

where $\mathcal{P}_k$ is the *upper bundle* formed of pairs $(\boldsymbol{x}^p \, , \, f_k^p = f_k(\boldsymbol{x}^p))$. The upper bundle can be compressed similarly to the ordinary (lower) one $\mathcal{B}_k$, with its poor-man's version containing only $\bar{\boldsymbol{x}}$, thus making the computation of $\hat{f}_{\mathcal{P}}^k$ potentially inexpensive. With this expedient, an IcBM that need not necessarily compute all the components neither at null step nor at serious step can be defined, and its convergence analysed with quite standard results, basically those of [19]. It is also easy to combine the two techniques by only computing a subset of the components *and* do that only approximately, e.g. with oracle (3.65).

An even stronger version of IcBM requires that the MP is not solved, as usual, for all components together, but component-wise, somehow more in the spirit of incremental subgradient methods; this entails some complications [47]. Here one could, however, employ the approach of [67] discussed in Sect. 3.4.5, whereby only the dual variables of the currently "active" component are allowed to vary whereas all the others are kept fixed, so that all components but one are represented by one fixed linearization.

All in all, IcBM have already been shown to improve performances in practice [31, 48], but more work is required to characterize the many trade-offs they entail.

We finish this section with an apparently different, but in fact strongly related, way to decrease the function computation time: exploiting the fact that the oracles f_k are independent, and therefore can be computed *in parallel*. This can be done in the obvious master-slave fashion, which has obvious drawbacks. First, the MP is a sequential bottleneck, which by Amdahl's law limits the maximum achievable speedup [16], requiring specific efforts to decrease the MP cost (with the corresponding nontrivial trade-offs). Furthermore, subdividing the components between different processors so that the computation takes roughly the same time

can be reasonably easy if all components are alike, but in many applications some of them require considerably more effort than others. Thus, a truly *asynchronous* BM would be required. A proposal in this sense is [33], which however is tailored to the case where $|\mathcal{K}|$ is large, but each component actually depends on only a few of the variables $\boldsymbol{x}$. A general purpose asynchronous BM should be possible, in particular using the results of [96], but several theoretical and practical issues still have to be ironed out.

3.6 Conclusion

Bundle-type methods have now a quite long history, spanning over 40 years from [69, 76, 100], and almost 60 from the seminal [60]. This chapter shows that this time has not been wasted: motivated by the ever increasing requirements of applications, many variants have been proposed and analysed that can provide significant performance benefits. As a very quick summary, investigation has focussed on:

1. different forms of stabilization, with different trade-offs between the cost of the corresponding MP and the theoretical and practical convergence speed;
2. different forms of (lower, and recently also upper) models that better exploit the properties of the function at hand;
3. solution methods for the MP that provide trade-offs between the accuracy of the solution and the computational cost;
4. a detailed characterization of the accuracy with which (the different components of) f has (have) to be computed in order to be able to proceed with optimization.

Yet, several theoretical and practical issues still remain open. The understanding of efficiency of standard BM is still rather partial, with the only available results depicting the almost hopelessly slow method corresponding to full aggregation—basically, a subgradient-type one—and therefore completely failing to capture facets of the practical convergence like the "fast tail". Furthermore, almost all efficiency estimate treat null step and serious step as almost entirely unrelated processes, whereas intuitively the practical efficiency of BM precisely hinges on the fact that they are not. In general, dealing with the stabilization parameter(s) remains more of an art than a science, thereby making BM rather susceptible to breaking down due to mismanagement of the algorithmic parameters; this limits their application potential, due to the difficulty of providing a "black-box" implementation that can be used by an inexperienced user without knowledge of its inner working and a significant parameter tuning phase. Besides the stabilization and related algorithmic parameters, this also applies to the fact that there are many variants of BM regarding to the stabilization technique, the model and the computation of the function; finding the right one for one's application, and capturing the proper trade-offs between all these aspects, is a rather complex process currently requiring specific knowledge and skills. It is therefore perhaps not surprising that there are not many available BM software packages, and that their practical use in applications is rather limited

in comparison to more "stable" algorithmic techniques like simplex and IP methods for linear/quadratic/conic programming. Admittedly, this also has to do with the inherent complexity of choosing, say, the right Lagrangian relaxation of one's problem, as opposed to just writing the model and using standard tools, a more general issue having more to do with the currently available modelling tools and solvers than with the specific characteristics of BM in particular.

Also, this chapter only deals with "standard" BM for convex problems. Significant research, often motivated by specific application like Machine Learning, has been poured into BM for nonconvex problems, or "nonstandard" ones trying to make better use of whatever available second-order information may be (if any). Thus, our treatment does not cover many other important facets of research in BM. Yet, we have hopefully shown that "standard" BM for convex optimization are a vast, diverse, and interesting class of algorithms with many relevant applications, and therefore a worthy research subject.

Acknowledgements I'm very grateful to Wim van Ackooij, Christoph Helmberg, Welington de Oliveira, Claudia Sagastizábal, and Jerome Malick for their useful suggestions that helped me to improve the contents and presentation of this work. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773897 "plan4res".

References

1. Anstreicher, K., Wolsey, L.: Two "well-known" properties of subgradient optimization. Math. Program. **120**(1), 213–220 (2009)
2. Astorino, A., Frangioni, A., Gaudioso, M., Gorgone, E.: Piecewise quadratic approximations in convex numerical optimization. SIAM J. Optim. **21**(4), 1418–1438 (2011)
3. Astorino, A., Frangioni, A., Fuduli, A., Gorgone, E.: A nonmonotone proximal bundle method with (potentially) continuous step decisions. SIAM J. Optim. **23**(3), 1784–1809 (2013)
4. Babonneau, F., Beltran, C., Haurie, A., Tadonki, C., Vial, J.P.: Proximal-ACCPM: A Versatile Oracle Based Optimization Method. Advances in Computational Management Science. Springer, Berlin (2007)
5. Babonneau, F., Vial, J.P.: ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. Math. Program. **120**, 179–210 (2009)
6. Bacaud, L., Lemaréchal, C., Renaud, A., Sagastizábal, C.: Bundle methods in stochastic optimal power management: a disaggregate approach using preconditionners. Comput. Optim. Appl. **20**(3), 227–244 (2001)
7. Bahiense, L., Maculan, N., Sagastizábal, C.: The volume algorithm revisited: Relation with bundle methods. Math. Program. **94**(1), 41–69 (2002)
8. Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Math. Program. **87**(3), 385–399 (2000)
9. Belloni, A., Diniz, A., Maceira, M., Sagastizábal, C.: Bundle relaxation and primal recovery in unit-commitment problems. the brazilian case. Ann. Oper. Res. **120**(1–4), 21–44 (2003)
10. Belloni, A., Sagastizábal, C.: Dynamic bundle methods. Math. Program. **120**(2), 289–311 (2009)
11. Ben-Ameur, W., Neto, J.: Acceleration of cutting-plane and column generation algorithms: applications to network design. Networks **49**(1), 3–17 (2007)

12. Ben Amor, H., Desrosiers, J., Frangioni, A.: On the choice of explicit stabilizing terms in column generation. Discret. Appl. Math. **157**(6), 1167–1184 (2009)
13. Bertsekas, D., Nedić, A.: Incremental subgradient methods for nondifferentiable optimization. SIAM J. Optim. **12**(1), 109–138 (2001)
14. Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C.: Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment. IEEE Trans. Power Syst. **18**, 313–323 (2003)
15. Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N., Vanderbeck, F.: Comparison of bundle and classical column generation. Math. Program. **113**(2), 299–344 (2008)
16. Cappanera, P., Frangioni, A.: Symmetric and asymmetric parallelization of a cost-decomposition algorithm for multi-commodity flow problems. INFORMS J. Comput. **15**(4), 369–384 (2003)
17. Castro, J., Cuesta, J.: Quadratic regularizations in an interior-point method for primal block-angular problems. Math. Program. **2**(130), 415–445 (2011)
18. Chen, G., Teboulle, M.: Convergence analysis of a proximal-like minimization algorithm using Bregman functions. SIAM J. Optim. **3**(3), 538–543 (1993)
19. Correa, R., Lemaréchal, C.: Convergence of some algorithms for convex minimization. Math. Program. **62**(2), 261–275 (1993)
20. Crainic, T., Frangioni, A., Gendron, B.: Bundle-based relaxation methods for multicommodity capacitated fixed charge network design problems. Discret. Appl. Math. **112**, 73–99 (2001)
21. Daniildis, A., Lemaréchal, C.: On a primal-proximal heuristic in discrete optimization. Math. Program. **104**, 105–128 (2005)
22. d'Antonio, G., Frangioni, A.: Convergence analysis of deflected conditional approximate subgradient methods. SIAM J. Optim. **20**(1), 357–386 (2009)
23. de Oliveira, W.: Target radius methods for nonsmooth convex optimization. Oper. Res. Lett. **45**, 659–664 (2017)
24. de Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on demand accuracy. Optim. Methods Softw. **29**(6), 1180–1209 (2014)
25. de Oliveira, W., Solodov, M.: A doubly stabilized bundle method for nonsmooth convex optimization. Math. Program. **156**(1), 125–159 (2016)
26. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Program. **148**, 241–277 (2014)
27. Desaulniers, G., Desrosiers, J., Solomon, M. (eds.): Column Generation. Springer, Berlin (2005)
28. Du, Y., Ruszczyński, A.: Rate of convergence of the bundle method. J. Optim. Theory Appl. **173**(3), 908–922 (2017)
29. Dubost, L., Gonzalez, R., Lemaréchal, C.: A primal-proximal heuristic applied to French unit-commitment problem. Math. Program. **104**(1), 129–151 (2005)
30. du Merle, O., Goffin, J.L., Vial, J.P.: On improvements to the analytic center cutting plane method. Comput. Optim. Appl. **11**, 37–52 (1998)
31. Emiel, G., Sagastizábal, C.: Incremental like bundle methods with applications to energy planning. Comput. Optim. Appl. **46**(2), 305–332 (2009)
32. Feltenmark, S., Kiwiel, K.: Dual applications of proximal bundle methods, including lagrangian relaxation of nonconvex problems. SIAM J. Optim. **10**(3), 697–721 (2000)
33. Fischer, F., Helmberg, C.: A parallel bundle framework for asynchronous subspace optimisation of nonsmooth convex functions. SIAM J. Optim. **24**(2), 795–822 (2014)
34. Frangioni, A.: Solving semidefinite quadratic problems within nonsmooth optimization algorithms. Comput. Oper. Res. **21**, 1099–1118 (1996)
35. Frangioni, A.: Dual-ascent methods and multicommodity flow problems. Ph.D. thesis, TD 5/97, Dipartimento di Informatica, Università di Pisa, Pisa, Italy (1997)
36. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **13**(1), 117–156 (2002)
37. Frangioni, A.: About lagrangian methods in integer optimization. Ann. Oper. Res. **139**(1), 163–193 (2005)

38. Frangioni, A., Gallo, G.: A bundle type dual-ascent approach to linear multicommodity min cost flow problems. INFORMS J. Comput. **11**(4), 370–393 (1999)
39. Frangioni, A., Gendron, B.: 0–1 reformulations of the multicommodity capacitated network design problem. Discret. Appl. Math. **157**(6), 1229–1241 (2009)
40. Frangioni, A., Gendron, B.: A stabilized structured Dantzig-Wolfe decomposition method. Math. Program. **104**(1), 45–76 (2013)
41. Frangioni, A., Gorgone, E.: Generalized bundle methods for sum-functions with "easy" components: applications to multicommodity network design. Math. Program. **145**(1), 133–161 (2014)
42. Frangioni, A., Lodi, A., Rinaldi, G.: New approaches for optimizing over the semimetric polytope. Math. Program. **104**(2–3), 375–388 (2005)
43. Frangioni, A., Gentile, C., Lacalandra, F.: Solving unit commitment problems with general ramp contraints. Int. J. Electr. Power Energy Syst. **30**, 316–326 (2008)
44. Frangioni, A., Gendron, B., Gorgone, E.: On the computational efficiency of subgradient methods: a case study with lagrangian bounds. Math. Program. Comput. **9**(4), 573–604 (2017)
45. Fuduli, A., Gaudioso, M.: Tuning strategy for the proximity parameter in convex minimization. J. Optim. Theory Appl. **130**(1), 95–112 (2006)
46. Gaar, E.: Efficient implementation of sdp relaxations for the stable set problem. Ph.D. thesis, Alpen-Adria-Universität Klagenfurt, Fakultät für Technische Wissenschaften, Klagenfurt, Austria (2018)
47. Gaudioso, M., Giallombardo, G., Miglionico, G.: An incremental method for solving convex finite min-max problems. Math. Oper. Res. **31**, 173–187 (2006)
48. Gaudioso, M., Giallombardo, G., Miglionico, G.: On solving the lagrangian dual of integer programs via an incremental approach. Comput. Optim. Appl. **44**, 117–138 (2007)
49. Goffin, J.L., Haurie, A., Vial, J.P.: Decomposition and nondifferentiable optimization with the projective algorithm. Manag. Sci. **38**, 284–302 (1992)
50. Gondzio, J., González-Brevis, P.: A new warmstarting strategy for the primal-dual column generation method. Math. Program. **152**, 113–146 (2015)
51. Gondzio, J., González-Brevis, P., Munari, P.: New developments in the primal-dual column generation technique. EURO J. Oper. Res. **224**, 41–51 (2013)
52. Grigoriadis, M., Khachiyan, L.: An exponential function reduction method for block angular convex programs. Networks **26**(2), 59–68 (1995)
53. Helmberg, C.: A cutting plane algorithm for large scale semidefinite relaxations. In: Grötschel, M. (ed.) The Sharpest Cut, MPS-SIAM Series on Optimization, pp. 233–256. SIAM/MPS (2004)
54. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**(3), 673–696 (2000)
55. Helmberg, C., Kiwiel, K.: A spectral bundle method with bounds. Math. Program. **93**, 173–194 (2002)
56. Helmberg, C., Pichler, A.: Dynamic scaling and submodel selection in bundle methods for convex optimization. Optim. Online 6180 (2017)
57. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II. No. 306 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin (1996)
58. Elzinga, J., Moore, T.G.: A central cutting plane algorithm for the convex programming problem. Math. Program. **8**, 134–145 (1975)
59. Karas, E., Ribeiro, A., Sagastizábal, C., Solodov, M.: A bundle-filter method for nonsmooth convex constrained optimization. Math. Program. **116**(1), 297–320 (2009)
60. Kelley, J.: The cutting-plane method for solving convex programs. J. Soc. Ind. Appl. Math. **8**(4), 703–712 (1960)
61. Kiwiel, K.: Proximity control in bundle methods for convex nondiferentiable minimization. Math. Program. **46**(1–3), 105–122 (1990)
62. Kiwiel, K.: Approximations in proximal bundle methods and decomposition of convex programs. J. Optim. Theory Appl. **84**, 529–548 (1995)

63. Kiwiel, K.: A bundle bregman proximal method for convex nondifferentiable optimization. Math. Program. **85**(2), 241–258 (1999)
64. Kiwiel, K.: Efficiency of proximal bundle methods. J. Optim. Theory Appl. **104**(3), 589–603 (2000)
65. Kiwiel, K.: Convergence of approximate and incremental subgradient methods for convex optimization. SIAM J. Optim. **14**(3), 807–840 (2003)
66. Kiwiel, K.: A proximal bundle method with approximate subgradient linearizations. SIAM J. Optim. **16**(4), 1007–1023 (2006)
67. Kiwiel, K.: An alternating linearization bundle method for convex optimization and nonlinear multicommodity flow problems. Math. Program. **130**(1), 59–84 (2011)
68. Kiwiel, K., Lemaréchal, C.: An inexact bundle variant suited to column generation. Math. Program. **118**(1), 177–206 (2009)
69. Lemaréchal, C.: An extension of Davidon methods to nondifferentiable problems. Math. Program. Study **3**, 95–109 (1975)
70. Lemaréchal, C., Sagastizábal, C.: Variable metric bundle methods: from conceptual to implementable forms. Math. Program. **76**(3), 393–410 (1997)
71. Lemaréchal, C., Renaud, A.: A geometric study of duality gaps, with applications. Math. Program. **90**, 399–427 (2001)
72. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**(1), 111–147 (1995)
73. Lemaréchal, C., Ouorou, A., Petrou, G.: A bundle-type algorithm for routing in telecommunication data networks. Comput. Optim. Appl. **44**, 385–409 (2009)
74. Lubin, M., Martin, K., C.G. Petra, B.S.: On parallelizing dual decomposition in stochastic integer programming. Oper. Res. Lett. **41**, 252–258 (2013)
75. Malick, J., de Oliveira, W., Zaourar, S.: Uncontrolled inexact information within bundle methods. EURO J. Comput. Optim., **5**(1–2), 5–29 (2017)
76. Marsten, R., Hogan, W., Blankenship, J.: The BOXSTEP method for large-scale optimization. Oper. Res. **23**(3), 389–405 (1975)
77. Mifflin, R., Sagastizábal, C.: On VU-theory for functions with primal-dual gradient structure. SIAM J. Optim. **11**(2), 547–571 (2000)
78. Mifflin, R., Sagastizábal, C.: A VU-algorithm for convex minimization. Math. Program. **104**(2–3), 583–608 (2005)
79. Mifflin, R., Sun, D., Qi, L.: Quasi-Newton bundle-type methods for nondifferentiable convex optimization. SIAM J. Optim. **8**(2), 583–603 (1998)
80. Nurminski, E.: Separating planes algorithms for convex optimization. Math. Program. **76**, 373–391 (1997)
81. Ouorou, A.: A proximal cutting plane method using Chebychev center for nonsmooth convex optimization. Math. Program. **119**(2), 239–271 (2009)
82. Parriani, T.: Decomposition methods and network design problems. Ph.D. thesis, Dottorato di Ricerca in Automatica e Ricerca Operativa, Alma Mater Studiorum – Università di Bologna (2014)
83. Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F.: Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS J. Comput. **30**(2), 339–360 (2018)
84. Pinar, M., Zenios, S.: Parallel decomposition of multicommodity network flows using a linear-quadratic penalty algorithm. ORSA J. Comput. **4**(3), 235–249 (1992)
85. Rey, P., Sagastizábal, C.: Dynamical adjustment of the prox-parameter in variable metric bundle methods. Optimization **51**(2), 423–447 (2002)
86. Sadykov, R., Vanderbeck, F.: Column generation for extended formulations. EURO J. Comput. Optim. **1**(1–2), 81–115 (2013)
87. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**(1), 146–169 (2005)

88. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**, 121–152 (1992)
89. Scuzziato, M., Finardi, E., Frangioni, A.: Comparing spatial and scenario decomposition for stochastic hydrothermal unit commitment problems. IEEE Trans. Sustainable Energy **9**(3) 1307–1317 (2018)
90. Subramanian, S., Sherali, H.: An effective deflected subgradient optimization scheme for implementing column generation for large-scale airline crew scheduling problems. INFORMS J. Comput. **20**(4), 565–578 (2008)
91. van Ackooij, W.: Decomposition approaches for block-structured chance-constrained programs with application to hydro-thermal unit commitment. Math. Methods Oper. Res. **80**(3), 227–253 (2014)
92. van Ackooij, W., de Oliveira, W.: Level bundle methods for constrained convex optimization with various oracles. Comput. Optim. Appl. **57**(3), 555–597 (2014)
93. van Ackooij, W., Sagastizábal, C.: Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. SIAM J. Optim. **24**(2), 733–765 (2014)
94. van Ackooij, W., de Oliveira, W.: Convexity and optimization with copulæ structured probability constraints. Optimization **65**(7), 1349–1376 (2016)
95. van Ackooij, W., Malick, J.: Decomposition algorithm for large-scale two-stage unit-commitment. Ann. Oper. Res. **238**(1), 587–613 (2016)
96. van Ackooij, W., Frangioni, A.: Incremental bundle methods using upper models. SIAM J. Optim. **28**(1), 379–410 (2018)
97. van Ackooij, W., Frangioni, A., de Oliveira, W.: Inexact stabilized Benders' decomposition approaches: with application to chance-constrained problems with finite support. Comput. Optim. Appl. **65**(3), 637–669 (2016)
98. van Ackooij, W., de Oliveira, W., Song, Y.: An adaptive partition-based level decomposition for solving two-stage stochastic programs with fixed recourse. INFORMS J. Comput. **30**(1), 57–70 (2018)
99. van Ackooij, W., Berge, V., de Oliveira, W., Sagastizábal, C.: Probabilistic optimization via approximate p-efficient points and bundle methods. Comput. Oper. Res. **77**, 177–193 (2017)
100. Wolfe, P.: A method of conjugate subgradients for minimizing nondifferentiable functions. Math. Program. Study **3**, 143–173 (1975)
101. Wolf, C., Fábián, C., Koberstein, A., Suhl, L.: Applying oracles of on-demand accuracy in two-stage stochastic programming—a computational study. EURO J. Oper. Res. **239**, 437–448 (2014)

Chapter 4
A Second Order Bundle Algorithm for Nonsmooth, Nonconvex Optimization Problems

Hermann Schichl and Hannes Fendl

Abstract In this chapter we extend the SQP-approach of the well-known bundle-Newton method for nonsmooth unconstrained minimization and the second order bundle method by Fendl and Schichl (A feasible second order bundle algorithm for nonsmooth, nonconvex optimization problems with inequality constraints: I. derivation and convergence. arXiv:1506.07937, 2015, preprint) to the general nonlinearly constrained case. Instead of using a penalty function or a filter or an improvement function to deal with the presence of constraints, the search direction is determined by solving a convex quadratically constrained quadratic program to obtain good iteration points. Furthermore, global convergence of the method is shown under certain mild assumptions.

4.1 Introduction

We assume in this chapter that the optimization problem (1.1) takes the following form

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & F_i(\boldsymbol{x}) \leq 0 \quad \text{for all } i = 1, \ldots, m, \\ & G_j(\boldsymbol{x}) = 0 \quad \text{for all } j = 1, \ldots, k, \\ & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{4.1}$$

where $f, F_i, G_j : \mathbb{R}^n \to \mathbb{R}$ are locally Lipschitz continuous (LLC). Since $F_i(\boldsymbol{x}) \leq 0$ for all $i = 1, \ldots, m$ and $G_j(\boldsymbol{x}) = 0$ for all $j = 1, \ldots, k$ if and only if

H. Schichl (✉) · H. Fendl
University of Vienna, Faculty of Mathematics, Wien, Austria
e-mail: hermann.schichl@univie.ac.at; hannes.fendl2@unicreditgroup.at

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_4

$$F(\boldsymbol{x}) := \max_{i=1,\dots,m} c_i F_i(\boldsymbol{x}) \le 0, \quad G(\boldsymbol{x}) := \sum_{j=1}^{k} \tilde{c}_j |G_j(\boldsymbol{x})| = 0,$$

with constants $c_i > 0$ and $\tilde{c}_j > 0$, and F and G are still LLC [55, pp. 969, Theorem 6(a)], we can reformulate (4.1) equivalently as

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & F(\boldsymbol{x}) \le 0, \\ & G(\boldsymbol{x}) = 0, \\ & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{4.2}$$

Since LLC functions are differentiable almost everywhere, both f and F may have kinks and therefore already the attempt to solve an unconstrained nonsmooth optimization (NSO) problem by a smooth solver (e.g., by a line search algorithm or by a trust region method) by just replacing the gradient by a subgradient, fails in general [73, pp. 461–462]. The fact that equality constraints are involved in addition poses the problem that, in general, no interior points of the feasible set exist. Therefore, the usual approach, as taken in [16] and [15], to construct a descent sequence of strictly feasible points fails as well.

To provide a short overview, methods other than bundle algorithms that are able to solve NSO problems include the R-algorithm [67], proximal point methods [63], or stochastic algorithms that try to approximate the subdifferential. In the following we will present a few implementations of these methods.

A few support at most linear constraints, e.g., the algorithm PMIN [43, 45], solves linearly constrained minimax optimization problems, i.e., the objective function must be maximum of twice continuously differentiable functions. The robust gradient sampling algorithm for nonsmooth nonconvex optimization [11] approximates the whole subdifferential at each iteration [10] and does not make null steps. The MATLAB code HANSO [61] combines ideas from BFGS algorithms [42] and from the gradient sampling algorithm for solving nonsmooth unconstrained optimization problems. The derivative-free bundle method (DFBM) [1], where "derivate-free" means that no derivate information is used explicitly, can solve linearly constrained nonsmooth problems. The subgradients are approximated by finite differences in this algorithm [2]. DFBM is an essential part of the programming library for global and nonsmooth optimization GANSO [4]. The discrete gradient method DGM for nonsmooth nonconvex unconstrained optimization [5] is a bundle-like method that does not compute subgradients, but approximates them by discrete gradients. The quasisecant method QSM for minimizing nonsmooth nonconvex functions [3] combines ideas both from bundle methods and from the gradient sampling method [11].

The oracle based optimization engine OBOE [70] is based on the analytic center cutting-plane method [58], which is an interior point framework.

There exist only a few algorithms that can also handle nonconvex constraints. The robust sequential quadratic programming algorithm extends the gradient sampling algorithm [13] for nonconvex, nonsmooth constrained optimization. SolvOpt [29] and `ralg` [40] (only in Python) are implementations of the R-algorithm [67]. SolvOpt handles the constraints by automatically adapting the penalty parameter, while `ralg` uses a filter technique.

In [35] a brief, excellent description of the main ideas of many of the mentioned methods are given. For further information visit the online decision tree for NSO software [31].

The proximal point methods [14, 62, 63] were initially developed for convex problems. However, efficient methods for nonconvex problems have emerged recently, e.g. [8]. They construct sequences by iteratively applying the proximal point operator from nonsmooth analysis, and they are very efficient in situations where the proximal points can be calculated analytically. In other situations they are hardly applicable.

Among the subgradient based algorithms, the bundle algorithms are the most successful for solving general nonsmooth problems. They are iterative methods that only need to compute one element $\boldsymbol{g}$ of the subdifferential $\partial f(\boldsymbol{x})$ (and possible $\partial F(\boldsymbol{x})$) per iteration, which in practice often is readily computable by algorithmic differentiation [20]. For computing the search direction, they collect information about the function (e.g., subgradients) from previous iterations. This collected information is referred to as "the bundle".

Bundle methods were originally developed for convex problems. A good introduction to NSO which treats the convex, unconstrained case in great detail is [6, pp. 106]. Very detailed standard references for nonsmooth nonconvex optimization are [36] and [50], which both in particular discuss constrained problems extensively.

We next summarize the most prominent bundle algorithms. The multiobjective proximal bundle method for nonconvex NSO (MPBNGC) [49] is a first order method that uses the improvement function $h_{x_k}(\boldsymbol{x}) := \max\{f(\boldsymbol{x}) - f(\boldsymbol{x}_k), F(\boldsymbol{x})\}$ for the handling of the constraints [50]. The algorithms in [54–56] support a nonconvex objective function as well as nonconvex constraints. NOA [39] is a NSO algorithm that handles nonconvex constraints by using a penalty function or an improvement function, while in the special case of convex constraints it offers an alternative treatment by the constraint linearization technique of [37]. The limited memory bundle algorithm for inequality constrained nondifferentiable optimization [34] combines LMBM [21] with the feasible directions interior point technique [24, 25] for dealing with the constraints. The search direction is determined by solving a linear system. Special bundle methods for DC functions [26, 27] have recently shown promising results.

In addition a few bundle algorithms can only handle convex constraints. The bundle trust algorithm [65, 66], which also supports a nonconvex objective function, handles the constraints by using the constraint linearization technique of [37]. The bundle filter algorithm [17] is only applicable to convex optimization problems

and it computes the search direction by solving an LP. The bundle-filter method for nonsmooth convex constrained optimization [30] is based on an improvement function, while the infeasible bundle method for nonsmooth convex constrained optimization [64] is also based on an improvement function, but it uses neither a penalty function nor a filter.

In addition, there are some bundle algorithms that support nonconvex objective functions but at most linear constraints, e.g., the variable metric bundle method PVAR [47, 72], PBUN [44, 48, 71], and the proximal bundle method [38]. The focus of the limited memory bundle method LMBM [21–23] is the solution of large-scale nonsmooth nonconvex unconstrained optimization problems. This is done by combining ideas from the variable metric bundle method [47, 72] and limited memory variable metric methods [12]. For a bound constrained version see [32, 33].

All the algorithms above make use first order information of the objective function and the constraints only. Nevertheless, there are some bundle methods that use second order information, since they are Newton-like methods (at least in some sense) and which only support the handling of linear constraints. The quasi-Newton bundle-type method for nondifferentiable convex optimization [57] generalizes the idea of quasi-Newton methods to NSO and it converges superlinearly for strongly convex functions (under some additional technical assumptions). The bundle-Newton method for nonsmooth unconstrained minimization [46] supports a nonconvex objective function and is based on an SQP–approach, and it is the only method for solving NSO problems that are known to us which uses Hessian information. Furthermore, its rate of convergence is superlinear for strongly convex, twice continuously differentiable functions. Moreover, a description of the implementation PNEW of the bundle-Newton method can be found in [44]). For the solution of eigenvalue problems a second order bundle method has been developed in [60].

In this work we extend the bundle-Newton method to a second order bundle algorithm for the problem (4.1) by using additional second order information of the objective function and the constraints. Furthermore, we use an extension of the SQP–approach of the bundle-Newton method for computing the search direction for the constrained case and combine it with the idea of quadratic constraint approximation, as it is used, e.g., in the sequential quadratically constrained quadratic programming method by [68] (this method is not a bundle method), in the hope to obtain good feasible iterates. Therefore, we have to solve a strictly feasible convex quadratically constrained quadratic problem (QCQP) for computing the search direction. Using such a QCQP for computing the search direction yields a line search condition for accepting infeasible points as trial points (which is different to that in, e.g., [56]). One of the most important properties of the convex QP (that is used to determine the search direction) with respect to a bundle method is its strong duality (e.g., for a meaningful termination criterion, for global convergence, etc.) which is also true in the case of strictly feasible convex QCQPs (cf. Sect. 4.4.2).

The chapter is organized as follows. In Sect. 4.2 we recall the basics of an SQP-method which is a common technique in smooth optimization and we summarize the most important facts about NSO theory. In Sect. 4.3 we give the theoretical

foundation of our second order bundle algorithm and afterwards we present the algorithm and the line search in detail. Finally, we show the convergence of the line search and the global convergence of the algorithm in Sect. 4.4.

Throughout the chapter we use the following notation. We denote the nonnegative real numbers by $\mathbb{R}_{\geq 0} := \{\boldsymbol{x} \in \mathbb{R} : \boldsymbol{x} \geq 0\}$. We denote the space of all symmetric $n \times n$-matrices by $\mathbb{R}^{n\times n}_{\text{sym}}$. For $\boldsymbol{x} \in \mathbb{R}^n$ we denote the Euclidean norm of $\boldsymbol{x}$ by $\|\boldsymbol{x}\|$, and for $A \in \mathbb{R}^{n\times n}_{\text{sym}}$ we denote the spectral norm of A by $\|A\|$. Furthermore, we denote the smallest resp. the largest eigenvalue of a positive definite matrix $A \in \mathbb{R}^{n\times n}$ by $\lambda_{\min}(A)$ resp. $\lambda_{\max}(A)$.

4.2 Optimization Theory

The idea of the second order bundle method is to build a special nonsmooth piecewise quadratic local model similar to the smooth quadratic model employed in SQP–methods. Therefore, we will summarize in the following section some well-known facts and the basics of an SQP-method. Afterwards, we present the most important facts about NSO theory.

4.2.1 Smooth Optimality Conditions and SQP

Theorem 4.1 ([51]) *Let $f, F_i, G_j : \mathbb{R}^n \to \mathbb{R}$ (with $i = 1, \ldots, m$ and $j = 1, \ldots, k$) be continuously differentiable and $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ be a solution of the smooth optimization problem*

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subjectto} & F_i(\boldsymbol{x}) \leq 0 \quad \text{for all } i = 1, \ldots, m, \\ & G_j(\boldsymbol{x}) = 0 \quad \text{for all } j = 1, \ldots, k, \\ & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{4.3}$$

Then there exist $\kappa \geq 0$, $\boldsymbol{\lambda} \geq \mathbf{0}$, and $\boldsymbol{\mu}$ with

$$\begin{aligned} &\kappa \nabla f(\hat{\boldsymbol{x}})^T + \nabla F(\hat{\boldsymbol{x}})^T \boldsymbol{\lambda} + \nabla G(\hat{\boldsymbol{x}})^T \boldsymbol{\mu} = \mathbf{0}, \\ &\lambda_i F_i(\hat{\boldsymbol{x}}) = 0, \text{ for all } i = 1, \ldots, m, \\ &\kappa = 1 \text{ or } (\kappa = 0, (\boldsymbol{\lambda}, \boldsymbol{\mu}) \neq \mathbf{0}). \end{aligned} \tag{4.4}$$

An SQP–method for finding a solution of the optimization problem (4.3) minimizes the quadratic approximation of the Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^m_{\geq 0} \to \mathbb{R}$ given by $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f(\boldsymbol{x}) + \boldsymbol{\lambda}^T F(\boldsymbol{x}) + \boldsymbol{\mu}^T G(\boldsymbol{x})$, subject to linearizations of

the constraints and then it uses the obtained minimizer as the new iteration point (or it performs a line search between the current iteration point and the obtained minimizer to determine the new iteration point). Since quadratic information is necessary for this approach, we demand $f, F_i, G_j : \mathbb{R}^n \to \mathbb{R}$ (with $i = 1, \ldots, m$ and $j = 1, \ldots, k$) to be C^2 in this subsection.

Proposition 4.1 *Let the Mangasarian–Fromowitz constraint qualifications [51] be satisfied, and let the Hessian of the Lagrangian with respect to the $\boldsymbol{x}$-components $\nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = \nabla^2 f(\boldsymbol{x}) + \sum_{i=1}^m \nabla^2 F_i(\boldsymbol{x})\lambda_i$ be positive definite on the tangent space of the constraints [59, pp. 531, Assumption 18.1]. Then the **SQP-step** for optimization problem (4.3) is given by the solution of the QP*

$$\begin{aligned} f(\boldsymbol{x}) + \min_{\boldsymbol{d}} \nabla f(\boldsymbol{x})\boldsymbol{d} + \tfrac{1}{2}\boldsymbol{d}^T \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}, \boldsymbol{\lambda})\boldsymbol{d} \\ \text{subject to } F_i(\boldsymbol{x}) + \nabla F_i(\boldsymbol{x})\boldsymbol{d} \le 0 \quad \text{for all } i = 1, \ldots, m, \\ G_j(\boldsymbol{x}) + \nabla G_j(\boldsymbol{x})\boldsymbol{d} = 0 \quad \text{for all } j = 1, \ldots, k. \end{aligned} \tag{4.5}$$

Remark 4.1 A difficulty of an infeasible SQP-method (e.g., SNOPT [19])—i.e. infeasible iteration points $\boldsymbol{x}_k$ may occur—is that the linear constraints of the QP (4.5) can be infeasible [59, pp. 535, 18.3 Algorithmic development]. Note that this difficulty does not arise for a feasible SQP–method (e.g., FSQP by [41])—i.e. only feasible iteration points $\boldsymbol{x}_k$ are accepted—as then $\boldsymbol{d} = \boldsymbol{0}$ is always feasible for the QP (4.5). Nevertheless, in this case it can be difficult to obtain feasible points that make good progress towards a solution (cf. Remark 4.3). In the presence of nonlinear equality constraints, however, a feasible SQP–method cannot be employed.

4.2.2 Nonsmooth Optimality Conditions

We gather information on the optimality conditions of the NSO problem (4.1) with LLC functions $f, F_i, G_j : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, k$. In addition to the subdifferential ∂f as introduced in Definition 1.8, we define the set $\partial^2 f(\boldsymbol{x}) \subseteq \mathbb{R}^{n\times n}_{\text{sym}}$ of the substitutes for the Hessian of f at $\boldsymbol{x}$ by

$$\partial^2 f(\boldsymbol{x}) := \begin{cases} \{G\}, & \text{if the Hessian } G \text{ of } f \text{ at } \boldsymbol{x} \text{ exists,} \\ \mathbb{R}^{n\times n}_{\text{sym}}, & \text{otherwise.} \end{cases} \tag{4.6}$$

We summarize the most important properties of the Clarke directional derivative and the subdifferential. The following result is taken from [7].

Proposition 4.2 *The subdifferential $\partial f(\boldsymbol{x})$ is nonempty, convex and compact. Furthermore, $\partial f : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$, where $\mathcal{P}(\mathbb{R}^n)$ denotes the power set of $\mathbb{R}^n$, is locally bounded and upper semicontinuous.*

From [28] we get

Theorem 4.2 (First Order Nonsmooth Optimality Conditions) *Let $\hat{\boldsymbol{x}}$ be a local minimizer of (4.1) and $f, F_i, G_j : \mathbb{R}^n \to \mathbb{R}$ (with $i = 1, \ldots, m$ and $j = 1, \ldots, k$) be Lipschitz continuous in a neighborhood of $\hat{\boldsymbol{x}}$. Then there exist $\kappa \geq 0$, $\boldsymbol{\lambda} \geq \mathbf{0}$, and $\boldsymbol{\mu}$ with*

$$\begin{aligned} &\mathbf{0} \in \kappa \partial f(\hat{\boldsymbol{x}}) + \sum_{i=1}^{m} \lambda_i \partial F_i(\hat{\boldsymbol{x}}) + \sum_{j=1}^{k} \mu_j \partial G_j(\hat{\boldsymbol{x}}), \\ &\lambda_i F_i(\hat{\boldsymbol{x}}) = 0 \text{ for all } i = 1, \ldots, m, \\ &\kappa = 1 \text{ or } (\kappa = 0, (\boldsymbol{\lambda}, \boldsymbol{\mu}) \neq \mathbf{0}). \end{aligned}$$

Furthermore, if the (nonsmooth) constraint qualification

$$\text{for all } (\boldsymbol{y}, \boldsymbol{z}) \in N(F(\hat{\boldsymbol{x}})) \times \mathbb{R}^k : \ \mathbf{0} \notin \sum_{i=1}^{m} y_i \partial F_i(\hat{\boldsymbol{x}}) + \sum_{j=1}^{k} z_j \partial G_j(\hat{\boldsymbol{x}}) \tag{4.7}$$

is satisfied, then we can always set $\kappa = 1$. Here

$$N(\boldsymbol{u}) := \{\boldsymbol{x} \in \mathbb{R}^m_{\geq 0} \mid \boldsymbol{x}^T \boldsymbol{u} = 0\}.$$

Corollary 4.1 *Let the constraint qualification (4.7) be satisfied for (4.1), then the necessary optimality condition for the equivalent reformulation (4.2) reads as follows: there exist $\lambda \geq 0$ and μ with*

$$\mathbf{0} \in \partial f(\hat{\boldsymbol{x}}) + \lambda \partial F(\hat{\boldsymbol{x}}) + \mu \partial G(\hat{\boldsymbol{x}}), \quad \lambda F(\hat{\boldsymbol{x}}) = 0, \quad F(\hat{\boldsymbol{x}}) \leq 0, \quad G(\hat{\boldsymbol{x}}) = 0. \tag{4.8}$$

Proof Inserting into Theorem 4.2 with $m = 1$ and $k = 1$. □

Remark 4.2 The algorithms in [54–56] (for solving nonlinearly constrained NSO problems) use a fixed point theorem about certain upper semicontinuous point to set mappings by Merrill [53] as optimality condition which is different to an approach with the optimality conditions in Theorem 4.2 or Corollary 4.1.

4.3 Derivation of the Method

In this section we discuss the theoretical basics of our second order bundle algorithm and we give a detailed presentation of the algorithm and the line search.

4.3.1 Theoretical Basics

We assume in this section that the functions $f, F, G : \mathbb{R}^n \to \mathbb{R}$ are LLC, $\boldsymbol{g}_j \in \partial f(\boldsymbol{y}_j)$, $\hat{\boldsymbol{g}}_j \in \partial F(\boldsymbol{y}_j)$, $\check{\boldsymbol{g}}_j \in \partial G(\boldsymbol{y}_j)$, and let G_j, $\hat{G}_j$, and $\check{G}_j$ be approximations of elements in $\partial^2 f(\boldsymbol{y}_j)$, $\partial^2 F(\boldsymbol{y}_j)$, and $\partial^2 G(\boldsymbol{y}_j)$ (cf. (4.6)), respectively.

The algorithm searches for a point that satisfies the first order optimality conditions (4.8). We propose an infeasible point version of the second order bundle method in [15, 16]. If we are at the iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$ (with iteration index k), we want to compute the next trial point (i.e. the search direction) by approximating the objective function f and the constraints F and G at $\boldsymbol{x}_k$ by piecewise quadratic functions and then perform a single SQP-step, as defined in Proposition 4.1.

Definition 4.1 Let $J_k \subseteq \{1, \dots, k\}$. We define quadratic approximations of f, F, and G in $\boldsymbol{y}_j \in \mathbb{R}^n$ with damping parameters ρ_j, $\hat{\rho}_j$, and $\check{\rho}_j \in [0, 1]$ for $j \in J_k$ by

$$\begin{aligned} f_j^\sharp(\boldsymbol{x}) &:= f(\boldsymbol{y}_j) + \boldsymbol{g}_j^T(\boldsymbol{x} - \boldsymbol{y}_j) + \tfrac{1}{2}\rho_j(\boldsymbol{x} - \boldsymbol{y}_j)^T G_j(\boldsymbol{x} - \boldsymbol{y}_j), \\ F_j^\sharp(\boldsymbol{x}) &:= F(\boldsymbol{y}_j) + \hat{\boldsymbol{g}}_j^T(\boldsymbol{x} - \boldsymbol{y}_j) + \tfrac{1}{2}\hat{\rho}_j(\boldsymbol{x} - \boldsymbol{y}_j)^T \hat{G}_j(\boldsymbol{x} - \boldsymbol{y}_j), \\ G_j^\sharp(\boldsymbol{x}) &:= G(\boldsymbol{y}_j) + \check{\boldsymbol{g}}_j^T(\boldsymbol{x} - \boldsymbol{y}_j) + \tfrac{1}{2}\check{\rho}_j(\boldsymbol{x} - \boldsymbol{y}_j)^T \check{G}_j(\boldsymbol{x} - \boldsymbol{y}_j), \end{aligned} \tag{4.9}$$

and the corresponding gradients by

$$\begin{aligned} \boldsymbol{g}_j^\sharp(\boldsymbol{x}) &:= \nabla f_j^\sharp(\boldsymbol{x})^T = \boldsymbol{g}_j + \rho_j G_j(\boldsymbol{x} - \boldsymbol{y}_j), \\ \hat{\boldsymbol{g}}_j^\sharp(\boldsymbol{x}) &:= \nabla F_j^\sharp(\boldsymbol{x})^T = \hat{\boldsymbol{g}}_j + \hat{\rho}_j \hat{G}_j(\boldsymbol{x} - \boldsymbol{y}_j), \\ \check{\boldsymbol{g}}_j^\sharp(\boldsymbol{x}) &:= \nabla G_j^\sharp(\boldsymbol{x})^T = \check{\boldsymbol{g}}_j + \check{\rho}_j \check{G}_j(\boldsymbol{x} - \boldsymbol{y}_j). \end{aligned} \tag{4.10}$$

We define the piecewise quadratic approximations of f, F, and G in $\boldsymbol{x}_k \in \mathbb{R}^n$ by

$$f_k^\square(\boldsymbol{x}) := \max_{j \in J_k} f_j^\sharp(\boldsymbol{x}), \quad F_k^\square(\boldsymbol{x}) := \max_{j \in J_k} F_j^\sharp(\boldsymbol{x}), \quad G_k^\square(\boldsymbol{x}) := \max_{j \in J_k} G_j^\sharp(\boldsymbol{x}). \tag{4.11}$$

Hence we approximate the objective function f at $\boldsymbol{x}_k$ by $f_k^\square$ and the constraints F and G at $\boldsymbol{x}_k$ by $F_k^\square$ and $G_k^\square$, respectively, in the optimization problem (4.2) and then we perform a single SQP-step to the resulting optimization problem

$$\begin{cases} \text{minimize} & f_k^\square(\boldsymbol{x}) \\ \text{subject to} & F_k^\square(\boldsymbol{x}) \le 0, \\ & G_k^\square(\boldsymbol{x}) = 0, \\ & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{4.12}$$

It is important to observe here that the local model for the nonsmooth problem (4.2) is the piecewise quadratic nonsmooth problem (4.12). This problem in turn can, however, since $G(\mathbf{x}) \geq 0$ be written as a smooth QCQP.

Proposition 4.3 *The SQP-step* $(\boldsymbol{d}, \hat{v}) \in \mathbb{R}^{n+1}$ *for* (4.12) *is given by the solution of the QP*

$$\begin{aligned} f(\boldsymbol{x}_k) + \min_{\boldsymbol{d},\hat{v}} \; & \left(\hat{v} + \tfrac{1}{2}\boldsymbol{d}^T W^k \boldsymbol{d}\right) \\ \text{subject to } & -\left(f(\boldsymbol{x}_k) - f_j^k\right) + \boldsymbol{d}^T \boldsymbol{g}_j^k \leq \hat{v} \text{ for all } j \in J_k, \\ & F(\boldsymbol{x}_k) - \left(F(\boldsymbol{x}_k) - F_j^k\right) + \boldsymbol{d}^T \hat{\boldsymbol{g}}_j^k \leq 0 \text{ for all } j \in J_k, \\ & G(\boldsymbol{x}_k) - \left(G(\boldsymbol{x}_k) - G_j^k\right) + \boldsymbol{d}^T \check{\boldsymbol{g}}_j^k \leq 0 \text{ for all } j \in J_k, \end{aligned} \tag{4.13}$$

where

$$\begin{aligned} f_j^k &:= f_j^\sharp(\boldsymbol{x}_k), \quad \boldsymbol{g}_j^k := \boldsymbol{g}_j^\sharp(\boldsymbol{x}_k) \overset{(4.10)}{=} \boldsymbol{g}_j + \rho_j G_j(\boldsymbol{x}_k - \boldsymbol{y}_j), \\ F_j^k &:= F_j^\sharp(\boldsymbol{x}_k), \quad \hat{\boldsymbol{g}}_j^k := \hat{\boldsymbol{g}}_j^\sharp(\boldsymbol{x}_k) \overset{(4.10)}{=} \hat{\boldsymbol{g}}_j + \hat{\rho}_j \hat{G}_j(\boldsymbol{x}_k - \boldsymbol{y}_j), \\ G_j^k &:= G_j^\sharp(\boldsymbol{x}_k), \quad \check{\boldsymbol{g}}_j^k := \check{\boldsymbol{g}}_j^\sharp(\boldsymbol{x}_k) \overset{(4.10)}{=} \check{\boldsymbol{g}}_j + \check{\rho}_j \check{G}_j(\boldsymbol{x}_k - \boldsymbol{y}_j), \\ W^k &:= \sum_{j \in J_{k-1}} \lambda_j^{k-1} \rho_j G_j + \sum_{j \in J_{k-1}} \mu_j^{k-1} \hat{\rho}_j \hat{G}_j + \sum_{j \in J_{k-1}} \nu_j^{k-1} \check{\rho}_j \check{G}_j, \end{aligned} \tag{4.14}$$

and λ_j^{k-1}, μ_j^{k-1}, *and* ν_j^{k-1} *denote the Lagrange multipliers with respect to* f, F, *and* G, *respectively, at iteration* $k-1$ *for* $j \in J_{k-1}$.

Proof We rewrite (4.12) as a smooth optimization problem by using (4.11), relaxing the equality constraint using the parameter σ_k. If we are at the iteration point $(\boldsymbol{x}_k, u_k) \in \mathbb{R}^{n+1}$ with $u_k := f(\boldsymbol{x}_k)$ in this smooth reformulation, then, according to (4.5) as well as using (4.14), the SQP-step for this problem is given by the solution of the QP (4.13). □

The problem in solving this SQP-step problem lies in the inequality constraint for G which might render the quadratic program infeasible, so it is useful to relax the constraint using a parameter $\sigma_k \in [0, 1]$.

Definition 4.2 The SQP-step $(\boldsymbol{d}, \hat{v}) \in \mathbb{R}^{n+1}$ for fixed parameter $\sigma_k \in [0,1]$ of (4.12) is given by the solution of the QP

$$\begin{aligned} f(\boldsymbol{x}_k) + \min_{\boldsymbol{d},\hat{v}} \; & \left(\hat{v} + \tfrac{1}{2}\boldsymbol{d}^T W^k \boldsymbol{d}\right) \\ \text{subject to } & -\left(f(\boldsymbol{x}_k) - f_j^k\right) + \boldsymbol{d}^T \boldsymbol{g}_j^k \le (1-\sigma_k)\hat{v} \text{ for all } j \in J_k, \\ & F(\boldsymbol{x}_k) - \left(F(\boldsymbol{x}_k) - F_j^k\right) + \boldsymbol{d}^T \hat{\boldsymbol{g}}_j^k \le 0 \text{ for all } j \in J_k, \\ & G(\boldsymbol{x}_k) - \left(G(\boldsymbol{x}_k) - G_j^k\right) + \boldsymbol{d}^T \check{\boldsymbol{g}}_j^k \le G(\boldsymbol{x}_k) + \sigma_k \hat{v} \text{ for all } j \in J_k, \end{aligned} \tag{4.15}$$

where f_j^k, $\boldsymbol{g}_j^k$, F_j^k, $\hat{\boldsymbol{g}}_j^k$, G_j^k, $\check{\boldsymbol{g}}_j^k$, W^k, λ_j^{k-1}, μ_j^{k-1}, and ν_j^{k-1} are as in Proposition 4.3.

Since $f_j^\sharp$, $F_j^\sharp$, and $G_j^\sharp$ are only global underestimators for convex f, F, and G, respectively, and $\rho_j = \hat{\rho}_j = \check{\rho} = 0$ and since $f_k^\square$, $F_k^\square$, and $G_k^\square$ approximate f, F, and G, respectively, only well for trial points close to $\boldsymbol{x}_k$, we decrease the activity of non local information (e.g., non local subgradients) by the following definition.

Definition 4.3 We define the localized approximation errors of f resp. F and G by

$$\begin{aligned} \alpha_j^k &:= \max\{|f(\boldsymbol{x}_k) - f_j^k|, \gamma_1 (s_j^k)^{\omega_1}\}, \\ A_j^k &:= \max\{|F(\boldsymbol{x}_k) - F_j^k|, \gamma_2 (s_j^k)^{\omega_2}\}, \\ B_j^k &:= \max\{|G(\boldsymbol{x}_k) - G_j^k|, \gamma_3 (s_j^k)^{\omega_3}\}, \end{aligned} \tag{4.16}$$

where

$$s_j^k := \|\boldsymbol{y}_j - \boldsymbol{x}_j\| + \sum_{i=j}^{k-1} \|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\| \tag{4.17}$$

denotes a locality measure for $j = 1, \ldots, k$ with fixed parameters $\gamma_i > 0$ and $\omega_i \ge 1$ for $i = 1, 2, 3$.

Straightforward calculations show

Proposition 4.4 *The locality measure s_j^k has the following properties*

$$s_j^k + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| = s_j^{k+1}, \quad s_j^k \ge \|\boldsymbol{y}_j - \boldsymbol{x}_k\| \text{ for all } j = 1, \ldots, k. \tag{4.18}$$

Like the bundle-Newton method by [46], our algorithm uses a convex search direction problem and therefore we modify (4.15) in the following sense.

Proposition 4.5 *If we generalize* (4.15) *by using the localized approximation errors* (4.16) *and replacing W^k by a positive definite modification $\overline{W}_p^k$ (e.g., the Gill-*

Murray factorization by [18]*), then the generalized version of* (4.15) *reads*

$$\begin{aligned} f(\boldsymbol{x}_k) + \min_{\boldsymbol{d},\hat{v}} \; & \hat{v} + \tfrac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d} \\ \text{subject to } & -\alpha_j^k + \boldsymbol{d}^T \boldsymbol{g}_j^k \leq (1-\sigma_k)\hat{v} \text{ for all } j \in J_k, \\ & F(\boldsymbol{x}_k) - A_j^k + \boldsymbol{d}^T \hat{\boldsymbol{g}}_j^k \leq 0 \text{ for all } j \in J_k, \\ & G(\boldsymbol{x}_k) - B_j^k + \boldsymbol{d}^T \check{\boldsymbol{g}}_j^k \leq G(\boldsymbol{x}_k) + \sigma_k \hat{v} \text{ for all } j \in J_k. \end{aligned} \tag{4.19}$$

Remark 4.3 The standard SQP approach for smooth optimization problems suffers from the Maratos effect [52], which, in general, prevents infeasible SQP-methods from getting a descent in the merit function and feasible SQP-methods from finding (good) feasible points [69, pp. 1003]. The way we have chosen to avoid the Maratos effect is by quadratic approximation of the constraints [68].

Remark 4.3 leads to the following idea. Let $\overline{G}_j^k, \hat{\overline{G}}_j^k, \check{\overline{G}}_j^k \in \mathbb{R}^{n\times n}_{\text{sym}}$ be positive definite (e.g., positive definite modifications of $G_j \in \partial^2 f(\boldsymbol{y}_j)$, $\hat{G}_j \in \partial^2 F(\boldsymbol{y}_j)$, and $\check{G}_j \in \partial^2 F(\boldsymbol{y}_j)$, respectively; also cf. Remark 4.9). Then we can try to determine the search direction by solving the convex QCQP

$$\begin{aligned} f(\boldsymbol{x}_k) + \min_{\boldsymbol{d},\hat{v}} \; & \hat{v} + \tfrac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d} \\ \text{subject to } & -\alpha_j^k + \boldsymbol{d}^T \boldsymbol{g}_j^k + \tfrac{1}{2}\boldsymbol{d}^T \overline{G}_j^k \boldsymbol{d} \leq (1-\sigma_k)\hat{v} \text{ for all } j \in J_k, \\ & F(\boldsymbol{x}_k) - A_j^k + \boldsymbol{d}^T \hat{\boldsymbol{g}}_j^k + \tfrac{1}{2}\boldsymbol{d}^T \hat{\overline{G}}_j^k \boldsymbol{d} \leq 0 \text{ for all } j \in J_k, \\ & -B_j^k + \boldsymbol{d}^T \check{\boldsymbol{g}}_j^k + \tfrac{1}{2}\boldsymbol{d}^T \check{\overline{G}}_j^k \boldsymbol{d} \leq \sigma_k \hat{v} \text{ for all } j \in J_k \end{aligned} \tag{4.20}$$

instead of the QP (4.19), i.e. instead of just demanding that the first order approximations are feasible, we demand that the first order approximations must be the more feasible, the more we move away from $\boldsymbol{x}_k$.

Example 4.1 We consider the optimization problem (4.2) with $f(\boldsymbol{x}) := x_2$, and $F(\boldsymbol{x}) := \max\{\min\{F_1(\boldsymbol{x}), F_2(\boldsymbol{x})\}, F_3(\boldsymbol{x})\}$, where $F_1(\boldsymbol{x}) := x_1^2 + x_2^2$, $F_2(\boldsymbol{x}) := -x_1 + x_2^2$, and $F_3(\boldsymbol{x}) := x_1 - 2$, and we assume that we are at the iteration point $\boldsymbol{x}_k := \boldsymbol{0}$.

Since $\hat{F}(\boldsymbol{x}) := \max\{F_2(\boldsymbol{x}), F_3(\boldsymbol{x})\}$ is convex, and since an easy examination yields that $F(\boldsymbol{x}) \leq 0 \Longleftrightarrow \hat{F}(\boldsymbol{x}) \leq 0$, the feasible set of our optimization problem (4.2) is convex. Therefore, the linearity of f implies that our optimization problem has the unique minimizer $\hat{\boldsymbol{x}} := (2, -\sqrt{2})$.

The quadratic approximation of F with respect to $\boldsymbol{x}_k$ in the QCQP (4.20) reads $F_1(\boldsymbol{x}_k + \boldsymbol{d}) \leq 0$, i.e. $\boldsymbol{d} = \boldsymbol{0}$ is the only feasible point for the QCQP (4.20)

and therefore its solution, although $\boldsymbol{x}_k = \mathbf{0}$ is not a stationary point for our optimization problem (for this consider f), resp. much less a minimizer (since $\hat{\boldsymbol{x}}$ is the unique minimizer of our optimization problem). As it can be seen, e.g., from considering the restriction of F to $x_2 = 0$, the reason for the occurrence of $\boldsymbol{d} = \mathbf{0}$ at $\boldsymbol{x}_k$ is the nonconvexity of F (which is a result of the presence of the min-function in F), although the feasible set is convex.

Notice that if we substitute F by $\hat{F}$ in the constraint of our optimization problem, which yields the same feasible set, the difficulty which we described above does not occur.

Remark 4.4 If $F(\boldsymbol{x}_k) \le 0$, ((4.19) as well as) (4.20) is always feasible and therefore we do not have to deal with infeasible search direction problems as they occur in infeasible SQP–methods (cf. Remark 4.1). Nevertheless, we have to demand $F(\boldsymbol{x}_k) < 0$, since otherwise it can happen that $\boldsymbol{d}_k = \mathbf{0}$ is the only feasible point and therefore the solution of (4.20), but $\boldsymbol{x}_k$ is not stationary for (4.2) as Example 4.1 showed. This is similar to difficulties arising in smooth problems at saddle points of the constraints. If no strictly feasible point can be found for F_i then it is advisable to treat it like the constraints arising from the equality constraints. There, the positive right hand side (for $\sigma_k > 0$) removes the problem.

Now we state the dual search direction problem which plays an important role for proving the global convergence of the method (cf. Sect. 4.4.2).

Proposition 4.6 *The dual problem of the QCQP* (4.20) *is given by*

$$\begin{aligned} f(\boldsymbol{x}_k) - \min_{\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu}} \tfrac{1}{2}\Big\| H_k(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu})\Big(\sum_{j\in J_k} \lambda_j \boldsymbol{g}_j^k + \mu_j \hat{\boldsymbol{g}}_j^k + \nu_j \check{\boldsymbol{g}}_j^k\Big)\Big\|^2 \\ + \sum_{j\in J_k} \lambda_j \alpha_j^k + \mu_j A_j^k + \nu_j B_j^k - \Big(\sum_{j\in J_k} \mu_j\Big) F(\boldsymbol{x}_k) \\ \text{subject to } \lambda_j \ge 0,\ \mu_j \ge 0,\ \nu_j \ge 0 \text{ for all } j \in J_k, \\ \sum_{j\in J_k} (1-\sigma_k)\lambda_j + \sigma_k \nu_j = 1, \end{aligned} \tag{4.21}$$

where $H_k(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu}) := \big(\overline{W}_p^k + \sum_{j\in J_k} \lambda_j \overline{G}_j^k + \mu_j \hat{\overline{G}}_j^k + \nu_j \check{\overline{G}}_j^k\big)^{-\frac{1}{2}}$. *If* $F(\boldsymbol{x}_k) < 0$, *then the duality gap is zero, and, furthermore, if we denote the minimizer of the dual problem* (4.21) *by* $(\boldsymbol{\lambda}^k, \boldsymbol{\mu}^k, \boldsymbol{\nu}^k)$, *then the minimizer* $(\boldsymbol{d}_k, \hat{v}_k)$ *of the primal QCQP* (4.20) *satisfies*

$$\boldsymbol{d}_k = -(\overline{W}_p^k + \sum_{j\in J_k} \lambda_j^k \overline{G}_j^k + \mu_j^k \hat{\overline{G}}_j^k + \nu_j^k \check{\overline{G}}_j^k)^{-1}\Big(\sum_{j\in J_k} \lambda_j^k \boldsymbol{g}_j^k + \mu_j^k \hat{\boldsymbol{g}}_j^k + \nu_j^k \check{\boldsymbol{g}}_j^k\Big),$$

$$\hat{v}_k = \Big(\sum_{j\in J_k} \lambda_j^k \boldsymbol{g}_j^k\Big)^T \boldsymbol{d}_k - \sum_{j\in J_k} \lambda_j^k \alpha_j^k + \tfrac{1}{2}\boldsymbol{d}_k^T\Big(\sum_{j\in J_k} \lambda_j^k \overline{G}_j^k\Big)\boldsymbol{d}_k$$

$$= -\boldsymbol{d}_k^T \overline{W}_p^k \boldsymbol{d}_k - \tfrac{1}{2}\boldsymbol{d}_k^T \Big(\sum_{j\in J_k} \lambda_j^k \overline{G}_j^k + \mu_j^k \hat{\overline{G}}_j^k + \nu_j^k \check{\overline{G}}_j^k \Big) \boldsymbol{d}_k$$
$$- \sum_{j\in J_k} \big(\lambda_j^k \alpha_j^k + \mu_j^k A_j^k + \nu_j^k B_j^k \big) - \Big(\sum_{j\in J_k} \mu_j^k \Big) \big(- F(\boldsymbol{x}_k) \big)$$
$$\leq 0.$$

Proof The Lagrangian of (4.20) is given by $L(\boldsymbol{d}, \hat{v}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) := \hat{v} + \frac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d} + \sum_{j\in J_k} \lambda_j F_j^1(\boldsymbol{d}, \hat{v}) + \sum_{j\in J_k} \mu_j F_j^2(\boldsymbol{d}, \hat{v}) + \sum_{j\in J_k} \nu_j F_j^3(\boldsymbol{d}, \hat{v})$, where $F_j^1(\boldsymbol{d}, \hat{v}) := -\alpha_j^k + \boldsymbol{d}^T \boldsymbol{g}_j^k + \frac{1}{2}\boldsymbol{d}^T \overline{G}_j^k \boldsymbol{d} - (1-\sigma_k)\hat{v}$, $F_j^2(\boldsymbol{d}, \hat{v}) := F(\boldsymbol{x}_k) - A_j^k + \boldsymbol{d}^T \hat{\boldsymbol{g}}_j^k + \frac{1}{2}\boldsymbol{d}^T \hat{\overline{G}}_j^k \boldsymbol{d}$, and $F_j^3(\boldsymbol{d}, \hat{v}) := -B_j^k + \boldsymbol{d}^T \check{\boldsymbol{g}}_j^k + \frac{1}{2}\boldsymbol{d}^T \check{\overline{G}}_j^k \boldsymbol{d} - \sigma_k \hat{v}$. Consequently, the equality constraint of the dual problem reads

$$\Big(\overline{W}_p^k + \sum_{j\in J_k} \lambda_j \overline{G}_j^k + \mu_j \hat{\overline{G}}_j^k + \nu_j \check{\overline{G}}_j^k \Big) \boldsymbol{d} + \sum_{j\in J_k} \lambda_j \boldsymbol{g}_j^k + \mu_j \hat{\boldsymbol{g}}_j^k + \nu_j \check{\boldsymbol{g}}_j^k = 0,$$
$$\sum_{j\in J_k} (1-\sigma_k)\lambda_j + \sigma_k \nu_j = 1. \tag{4.22}$$

Rewriting $\frac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d} = -\frac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d} + \boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d}$ in L, scooping $\boldsymbol{d}$ in the latter summand and $\hat{v}$, these terms vanish according to (4.22). Now, expressing $\boldsymbol{d}$ in (4.22) and inserting it into L yield the desired form of the dual objective function.

Since the primal problem is convex and (because of the assumption $F(\boldsymbol{x}_k) < 0$) strictly feasible, strong duality holds due to [9, Subsection 5.2.3]. Therefore the optimal primal and dual objective function values coincide and we can express $\hat{v}_k$ using this equality. Using (4.22), the optimality conditions for the QCQP (4.20) and straightforward calculations yield the desired formulas for $\hat{v}_k$. □

4.3.2 The Bundle Algorithm

The method described in Algorithm 4.1 below works according to the following scheme. After choosing a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$ that is strictly feasible with respect to the F-constraint and $\sigma_1 \in (0, 1)$ and after setting up a few positive definite matrices, we compute the localized approximation errors. Then we solve a convex QCQP to determine the search direction, where the quadratic constraints of the QCQP serve to obtain preferably feasible points that yield a good descent and a good approximation of the equality constraint. After computing the aggregated data and the predicted descent as well as testing the termination criterion, we perform a line search (see Algorithm 4.2) on the ray given by the search direction. This yields a trial point $\boldsymbol{y}_{k+1}$ that has the following property: either $\boldsymbol{y}_{k+1}$ is strictly feasible with respect to F and the objective function and constraint violation in G achieve sufficient descent (serious step) or $\boldsymbol{y}_{k+1}$ is strictly feasible and the model

of the objective function changes sufficiently (null step with respect to the objective function) or $\boldsymbol{y}_{k+1}$ is not strictly feasible and the model of the constraints changes sufficiently (null step with respect to the constraint). Afterwards we update the iteration point $\boldsymbol{x}_{k+1}$ and the information stored in the bundle. Now, we repeat this procedure until the termination criterion is satisfied.

The initial parameters for the algorithm are given in Table 4.1.

Table 4.1 Initial parameters

General	Default	Description
$\mathbf{x}_1 \in \mathbb{R}^n$		strictly feasible initial point
$\mathbf{y}_1 = \mathbf{x}_1$		initial trial point
$\varepsilon \geq 0$		final optimality tolerance
$M \geq 2$	$M = n + 3$	maximal bundle dimension
$t_0 \in (0, 1)$	$t_0 = 0.001$	initial lower bound for step size of serious step in line search
$\hat{t}_0 \in (0, 1)$	$\hat{t}_0 = 0.001$	scaling parameter for t_0
$m_L \in (0, \frac{1}{2})$	$m_L = 0.01$	descent parameters for serious step in line search
$\ell_L \in (0, \frac{1}{2})$	$\ell_L = m_L$	
$m_R \in (m_L, 1)$	$m_R = 0.5$	
$m_f \in [0, 1]$	$m_f = 0.01$	parameter for change of model of objective function for short serious and null steps in line search
$m_F, m_G \in (0, 1)$	$m_F = 0.01$ $m_G = m_F$	parameters for change of models of constraints for short serious and null steps in line search
$\zeta \in (0, \frac{1}{2})$	$\zeta = 0.01$	coefficient for interpolation in line search
$\vartheta \geq 1$	$\vartheta = 1$	exponent for interpolation in line search
$\sigma_1 \in (0, 1]$	$\sigma_1 = 0.9$	initial weight for the equality constraint
$\tau_0 \in [0, 1)$	$\tau_0 = 0.5$	minimal update parameter for σ_k
$\tau_1 \in (\tau_0, 1)$	$\tau_1 = 0.90$	minimal improvement factor for σ_k
$C_S > 0$	$C_S = 10^{50}$	upper bound of the distance between $\mathbf{x}_k$ and $\mathbf{y}_k$
$C_G, \hat{C}_G, \check{C}_G > 0$	$C_G = 10^{50}$ $\hat{C}_G = \check{C}_G = C_G$	upper bound of the norm of the damped matrices $\{\rho_j G_j\}$ ($\lvert\rho_j G_j\rvert \leq C_G$) resp. $\{\hat{\rho}_j \hat{G}_j\}$ and $\{\check{\rho}_j \check{G}_j\}$
$\bar{C}_G, \bar{\hat{C}}_G, \bar{\check{C}}_G > 0$	$\bar{C}_G = C_G$ $\bar{\hat{C}}_G = \bar{\check{C}}_G = \bar{C}_G$	upper bound of the norm of the matrices $\{\bar{G}_j^k\}$ and $\{\bar{G}^k\}$ ($\max(\lvert\bar{G}_j^k\rvert, \lvert\bar{G}^k\rvert) \leq \bar{C}_G$) resp. $\{\bar{\hat{G}}_j^k\}$, $\{\bar{\hat{G}}^k\}$ and $\{\bar{\check{G}}_j^k\}$, $\{\bar{\check{G}}_j\}$
$i_\rho \geq 0$	$i_\rho = 3$	selection parameter for ρ_{k+1} (Remark 4.5)
$i_l, i_m, i_r \geq 0$		line search selection, matrix selection, and bundle reset parameters (Remark 4.5)
$\gamma_1, \gamma_2, \gamma_3 > 0$	$\gamma_1 = \gamma_2 = \gamma_3 = 1$	coefficients for locality measures
$\omega_1, \omega_2, \omega_3 \geq 1$	$\omega_1 = \omega_2 = \omega_3 = 2$	exponents for locality measures

Algorithm 4.1: Second order bundle method

Data: Choose the initial parameters, which will not be changed during the algorithm, according to Table 4.1.

Step 0. *(Initialization)* Set the initial values of the data which gets changed during the algorithm:

$$\begin{aligned} i_n &= 0 \ \text{(\# subsequent null and short steps)},\\ i_s &= 0 \ \text{(\# subsequent serious steps)},\\ i_d &= 0 \ \text{(\# subsequent steps without equality improvement)},\\ J_1 &= \{1\} \ \text{(set of bundle indices)}. \end{aligned}$$

Compute the following information at the initial trial point

$$f_p^1 = f_1^1 = f(\boldsymbol{y}_1), \tag{4.23}$$

$$\boldsymbol{g}_p^1 = \boldsymbol{g}_1^1 = \boldsymbol{g}(\boldsymbol{y}_1) \in \partial f(\boldsymbol{y}_1), \tag{4.24}$$

$$G_p^1 = G_1 \text{ approximating } G(\boldsymbol{y}_1) \in \partial^2 f(\boldsymbol{y}_1), \tag{4.25}$$

$$F_p^1 = F_1^1 = F(\boldsymbol{y}_1) < 0 \ \ (\boldsymbol{y}_1 \text{ strictly feasible according to assumption}), \tag{4.26}$$

$$\hat{\boldsymbol{g}}_p^1 = \hat{\boldsymbol{g}}_1^1 = \hat{\boldsymbol{g}}(\boldsymbol{y}_1) \in \partial F(\boldsymbol{y}_1), \tag{4.27}$$

$$\hat{G}_p^1 = \hat{G}_1 \text{ approximating } \hat{G}(\boldsymbol{y}_1) \in \partial^2 F(\boldsymbol{y}_1), \tag{4.28}$$

$$G_p^1 = G_1^1 = G(\boldsymbol{y}_1), \tag{4.29}$$

$$\check{\boldsymbol{g}}_p^1 = \check{\boldsymbol{g}}_1^1 = \check{\boldsymbol{g}}(\boldsymbol{y}_1) \in \partial G(\boldsymbol{y}_1), \tag{4.30}$$

$$\check{G}_p^1 = \check{G}_1 \text{ approximating } \check{G}(\boldsymbol{y}_1) \in \partial^2 G(\boldsymbol{y}_1), \tag{4.31}$$

and set

$$\check{s}_p^1 = \hat{s}_p^1 = s_p^1 = s_1^1 = 0 \text{ (locality measure)}, \tag{4.32}$$

$$\check{\rho}_1 = \hat{\rho}_1 = \rho_1 = 1 \text{ (damping parameter)},$$

$$\bar{v}^1 = \bar{\kappa}^1 = 1 \text{ (Lagrange multiplier for optimality condition)},$$

$$k = 1 \text{ (iterator)}.$$

Step 1. *(Determination of the matrices for the QCQP)*
if (step $k-1$ and $k-2$ were serious steps) $\wedge$ ($\lambda_{k-1}^{k-1} = 1 \vee \underbrace{i_s > i_r}_{\text{bundle reset}}$) **then**

$$W = G_k + \bar{\kappa}^k \hat{G}_k + \bar{\nu}^k \check{G}_k, \tag{4.33}$$

else

$$W = G_p^k + \bar{\kappa}^k \hat{G}_p^k + \bar{\nu}^k \check{G}_p^k, \tag{4.34}$$

end

if $i_n \leq i_m + i_l$ **then**
$\overline{W}_p^k =$ "positive definite modification of W",
else

$$\overline{W}_p^k = \overline{W}_p^{k-1}, \tag{4.35}$$

end

if $i_n < i_m + i_l$ (i.e. # of subsequent null and short steps $<$ the fixed number $i_m + i_l$) **then**

$$(\overline{G}^k, \overline{\hat{G}}^k, \overline{\check{G}}^k) = \text{"positive definite modification of } (G_p^k, \hat{G}_p^k, \check{G}_p^k)\text{"}, \tag{4.36}$$
$$(\overline{G}_j^k, \overline{\hat{G}}_j^k, \overline{\check{G}}_j^k) = \text{"positive definite modification of } (G_j, \hat{G}_j, \check{G}_j)\text{" for all } j \in J_k,$$

else if $i_n = i_m + i_l$ **then**

$$\begin{aligned}(\overline{G}^k, \overline{\hat{G}}^k, \overline{\check{G}}^k) &= \text{"positive definite modification of } (G_p^k, \hat{G}_p^k, \check{G}_p^k)\text{"},\\ (\overline{G}_j^k, \overline{\hat{G}}_j^k, \overline{\check{G}}_j^k) &= (\overline{G}^k, \overline{\hat{G}}^k, \overline{\check{G}}^k) \text{ for all } j \in J_k,\end{aligned} \tag{4.37}$$

else (i.e. at least $i_m + i_l$ subsequent null and short steps were executed)

$$\begin{aligned}(\overline{G}^k, \overline{\hat{G}}^k, \overline{\check{G}}^k) &= (\overline{G}^{k-1}, \overline{\hat{G}}^{k-1}, \overline{\check{G}}^{k-1}),\\ (\overline{G}_j^k, \overline{\hat{G}}_j^k, \overline{\check{G}}_j^k) &= (\overline{G}^{k-1}, \overline{\hat{G}}^{k-1}, \overline{\check{G}}^{k-1}) \text{ for all } j \in J_k.\end{aligned} \tag{4.38}$$

end
Step 2. *(Computation of the localized approximation errors)*

$$\begin{aligned}\alpha_j^k &:= \max\{|f(\boldsymbol{x}_k) - f_j^k|, \gamma_1 (s_j^k)^{\omega_1}\},\\ \alpha_p^k &:= \max\{|f(\boldsymbol{x}_k) - f_p^k|, \gamma_1 (s_p^k)^{\omega_1}\},\end{aligned} \tag{4.39}$$

$$
\begin{aligned}
A_j^k &:= \max\{|F(\boldsymbol{x}_k) - F_j^k|, \gamma_2(s_j^k)^{\omega_2}\},\\
A_p^k &:= \max\{|F(\boldsymbol{x}_k) - F_p^k|, \gamma_2(\hat{s}_p^k)^{\omega_2}\}, \qquad (4.40)\\
B_j^k &:= \max\{|G(\boldsymbol{x}_k) - G_j^k|, \gamma_3(s_j^k)^{\omega_3}\},\\
B_p^k &:= \max\{|G(\boldsymbol{x}_k) - G_p^k|, \gamma_3(\check{s}_p^k)^{\omega_3}\}. \qquad (4.41)
\end{aligned}
$$

Step 3. *(Determination of the search direction)* Compute the solution $(\boldsymbol{d}_k, \hat{v}_k) \in \mathbb{R}^{n+1}$ of the (convex) QCQP

$$
\begin{cases}
\underset{\boldsymbol{d},\hat{v}}{\text{minimize}} & \hat{v} + \frac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d}\\
\text{subject to} & -\alpha_j^k + \boldsymbol{d}^T \boldsymbol{g}_j^k + \frac{1}{2}\boldsymbol{d}^T \overline{G}_j^k \boldsymbol{d} \le (1-\sigma_k)\hat{v} \quad \text{for } j \in J_k,\\
& -\alpha_p^k + \boldsymbol{d}^T \boldsymbol{g}_p^k + \frac{1}{2}\boldsymbol{d}^T \overline{G}^k \boldsymbol{d} \le (1-\sigma_k)\hat{v} \quad \text{if } i_s \le i_r,\\
& F(\boldsymbol{x}_k) - A_j^k + \boldsymbol{d}^T \hat{\boldsymbol{g}}_j^k + \frac{1}{2}\boldsymbol{d}^T \hat{\overline{G}}_j^k \boldsymbol{d} \le 0 \quad \text{for } j \in J_k,\\
& F(\boldsymbol{x}_k) - A_p^k + \boldsymbol{d}^T \hat{\boldsymbol{g}}_p^k + \frac{1}{2}\boldsymbol{d}^T \hat{\overline{G}}^k \boldsymbol{d} \le 0 \quad \text{if } i_s \le i_r,\\
& -B_j^k + \boldsymbol{d}^T \check{\boldsymbol{g}}_j^k + \frac{1}{2}\boldsymbol{d}^T \check{\overline{G}}_j^k \boldsymbol{d} \le \sigma_k \hat{v} \qquad \text{for } j \in J_k,\\
& -B_p^k + \boldsymbol{d}^T \check{\boldsymbol{g}}_p^k + \frac{1}{2}\boldsymbol{d}^T \check{\overline{G}}^k \boldsymbol{d} \le \sigma_k \hat{v} \qquad \text{if } i_s \le i_r,
\end{cases} \qquad (4.42)
$$

and its corresponding multiplier $(\lambda^k, \lambda_p^k, \mu^k, \mu_p^k, \nu^k, \nu_p^k) \in \mathbb{R}^{3(|J_k|+1)}_{\ge 0}$, i.e.

$$
\boldsymbol{d}_k = -H_k^2\Big(\sum_{j\in J_k} \lambda_j^k \boldsymbol{g}_j^k + \mu_j^k \hat{\boldsymbol{g}}_j^k + \nu_j^k \check{\boldsymbol{g}}_j^k + \lambda_p^k \boldsymbol{g}_p^k + \mu_p^k \hat{\boldsymbol{g}}_p^k + \nu_p^k \check{\boldsymbol{g}}_p^k\Big), \qquad (4.43)
$$

$$
\begin{aligned}
\hat{v}_k = &-\boldsymbol{d}_k^T \overline{W}_p^k \boldsymbol{d}_k\\
&- \tfrac{1}{2}\boldsymbol{d}_k^T\Big(\sum_{j\in J_k} \lambda_j^k \overline{G}_j^k + \mu_j^k \hat{\overline{G}}_j^k + \nu_j^k \check{\overline{G}}_j^k + \lambda_p^k \overline{G}^k + \mu_p^k \hat{\overline{G}}^k + \nu_p^k \check{\overline{G}}^k\Big)\boldsymbol{d}_k\\
&- \sum_{j\in J_k} (\lambda_j^k \alpha_j^k + \mu_j^k A_j^k + \nu_j^k B_j^k) - \lambda_p^k \alpha_p^k - \mu_p^k A_p^k - \nu_p^k B_p^k\\
&- \Big(\sum_{j\in J_k} \mu_j^k + \mu_p^k\Big)\big(-F(\boldsymbol{x}_k)\big), \qquad (4.44)
\end{aligned}
$$

where

$$
H_k := \Big(\overline{W}_p^k + \sum_{j\in J_k} \lambda_j^k \overline{G}_j^k + \mu_j^k \hat{\overline{G}}_j^k + \nu_j^k \check{\overline{G}}_j^k + \lambda_p^k \overline{G}^k + \mu_p^k \hat{\overline{G}}^k + \nu_p^k \check{\overline{G}}^k\Big)^{-\frac{1}{2}}. \qquad (4.45)
$$

Set

$$\bar{\lambda}^{k+1} := \sum_{j\in J_k} \lambda_j^k + \lambda_p^k, \quad (\zeta_j^k, \zeta_p^k) := \begin{cases} \frac{(\lambda_j^k, \lambda_p^k)}{\bar{\lambda}^{k+1}}, & \text{for } \bar{\lambda}^{k+1} > 0, \\ 0, & \text{for } \bar{\lambda}^{k+1} = 0, \end{cases} \tag{4.46}$$

$$\bar{\kappa}^{k+1} := \sum_{j\in J_k} \mu_j^k + \mu_p^k, \quad (\kappa_j^k, \kappa_p^k) := \begin{cases} \frac{(\mu_j^k, \mu_p^k)}{\bar{\kappa}^{k+1}}, & \text{for } \bar{\kappa}^{k+1} > 0, \\ 0, & \text{for } \bar{\kappa}^{k+1} = 0, \end{cases} \tag{4.47}$$

$$\bar{\nu}^{k+1} := \sum_{j\in J_k} \nu_j^k + \nu_p^k, \quad (\eta_j^k, \eta_p^k) := \begin{cases} \frac{(\nu_j^k, \nu_p^k)}{\bar{\nu}^{k+1}}, & \text{for } \bar{\nu}^{k+1} > 0, \\ 0, & \text{for } \bar{\nu}^{k+1} = 0, \end{cases} \tag{4.48}$$

if $i_s > i_r$ **then**
$i_s = 0$ (bundle reset).
end

Step 4. *(Aggregation)* We set for the aggregation of information of the objective function

$$(\tilde{\boldsymbol{g}}_p^k, \tilde{f}_p^k, G_p^{k+1}, \tilde{s}_p^k) = \sum_{j\in J_k} \zeta_j^k (\boldsymbol{g}_j^k, f_j^k, \rho_j G_j, s_j^k) + \zeta_p^k (\boldsymbol{g}_p^k, f_p^k, G_p^k, s_p^k), \tag{4.49}$$

$$\tilde{\alpha}_p^k = \max\left(|f(\boldsymbol{x}_k) - \tilde{f}_p^k|, \gamma_1 (\tilde{s}_p^k)^{\omega_1}\right), \tag{4.50}$$

and for the aggregation of information of the constraints

$$(\tilde{\hat{\boldsymbol{g}}}_p^k, \tilde{F}_p^k, \hat{G}_p^{k+1}, \tilde{\hat{s}}_p^k) = \sum_{j\in J_k} \kappa_j^k (\hat{\boldsymbol{g}}_j^k, F_j^k, \hat{\rho}_j \hat{G}_j, \hat{s}_j^k) + \kappa_p^k (\hat{\boldsymbol{g}}_p^k, F_p^k, \hat{G}_p^k, \hat{s}_p^k), \tag{4.51}$$

$$\tilde{A}_p^k = \max\{|F(\boldsymbol{x}_k) - \tilde{F}_p^k|, \gamma_2 (\tilde{\hat{s}}_p^k)^{\omega_2}\}, \tag{4.52}$$

$$(\tilde{\check{\boldsymbol{g}}}_p^k, \tilde{G}_p^k, \check{G}_p^{k+1}, \tilde{\check{s}}_p^k) = \sum_{j\in J_k} \eta_j^k (\check{\boldsymbol{g}}_j^k, G_j^k, \check{\rho}_j \check{G}_j, \check{s}_j^k) + \eta_p^k (\check{\boldsymbol{g}}_p^k, G_p^k, \check{G}_p^k, \check{s}_p^k), \tag{4.53}$$

$$\tilde{B}_p^k = \max\{|G(\boldsymbol{x}_k) - \tilde{G}_p^k|, \gamma_3 (\tilde{\check{s}}_p^k)^{\omega_3}\}, \tag{4.54}$$

and we set

$$\begin{aligned} v_k = -\boldsymbol{d}_k^T \big(\overline{W}_p^k + \tfrac{1}{2} \sum_{j\in J_k} \lambda_j^k \overline{G}_j^k + \mu_j^k \tilde{\overline{G}}_j^k + \nu_j^k \check{\overline{G}}_j^k + \lambda_p^k \overline{G}^k + \mu_p^k \hat{\overline{G}}^k + \nu_p^k \check{\overline{G}}^k\big) \boldsymbol{d}_k \\ - \bar{\lambda}^{k+1} \tilde{\alpha}_p^k - \bar{\kappa}^{k+1} \big(\tilde{A}_p^k - F(\boldsymbol{x}_k)\big) - \bar{\nu}^{k+1} \tilde{B}_p^k, \end{aligned} \tag{4.55}$$

$$w_k = \tfrac{1}{2}\|H_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{v}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)\|^2 + \bar{\lambda}^{k+1}\tilde{\alpha}_p^k + \bar{\kappa}^{k+1}\big(\tilde{A}_p^k - F(\boldsymbol{x}_k)\big) + \bar{v}^{k+1}\tilde{B}_p^k. \tag{4.56}$$

Step 5. *(Termination criterion)*
if $\max(w_k, \sigma_k, G(\boldsymbol{x}_k)) \le \varepsilon$ or $i_d > i_m$ **then**
stop.
end

Step 6. *(Line search)* We compute step sizes $0 \le t_L^k \le t_R^k \le 1$ and $t_0^k \in (0, t_0]$ by using the line search described in Algorithm 4.2 and we set

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k \quad \text{(strictly feasible with respect to } F \text{ by the line search)}, \tag{4.57}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k, \tag{4.58}$$

$$\begin{aligned}
&f_{k+1} = f(\boldsymbol{y}_{k+1}),\\
&\boldsymbol{g}_{k+1} = \boldsymbol{g}(\boldsymbol{y}_{k+1}) \in \partial f(\boldsymbol{y}_{k+1}),\\
&G_{k+1} \text{ approximating } G(\boldsymbol{y}_{k+1}) \in \partial^2 f(\boldsymbol{y}_{k+1}),\\
&F_{k+1} = F(\boldsymbol{y}_{k+1}),\\
&\hat{\boldsymbol{g}}_{k+1} = \hat{\boldsymbol{g}}(\boldsymbol{y}_{k+1}) \in \partial F(\boldsymbol{y}_{k+1}),\\
&\hat{G}_{k+1} \text{ approximating } \hat{G}(\boldsymbol{y}_{k+1}) \in \partial^2 F(\boldsymbol{y}_{k+1}),\\
&G_{k+1}^o = G(\boldsymbol{y}_{k+1}),\\
&\check{\boldsymbol{g}}_{k+1} = \check{\boldsymbol{g}}(\boldsymbol{y}_{k+1}) \in \partial G(\boldsymbol{y}_{k+1}),\\
&\check{G}_{k+1} \text{ approximating } \check{G}(\boldsymbol{y}_{k+1}) \in \partial^2 G(\boldsymbol{y}_{k+1}).
\end{aligned} \tag{4.59}$$

Step 7. *(Update)*
if $i_n \le i_\rho$ **then**

$$\rho_{k+1} = \min\{1, \tfrac{C_G}{|G_{k+1}|}\}, \tag{4.60}$$

else

$$\rho_{k+1} = 0.$$

end
We set

$$\begin{aligned}
\hat{\rho}_{k+1} &= \min\{1, \tfrac{\hat{C}_G}{|\hat{G}_{k+1}|}\},\\
\check{\rho}_{k+1} &= \min\{1, \tfrac{\check{C}_G}{|\check{G}_{k+1}|}\},
\end{aligned} \tag{4.61}$$

if $t_L^k \geq t_0^k$ (serious step) **then**

$$\begin{aligned} i_n &= 0, \\ i_s &= i_s + 1, \end{aligned}$$

if $|G_{k+1}^o / G_{k-i_d-i_n}^o| \leq \tau_1$ **then**

$$\begin{aligned} \sigma_{k+1} &= \max\{\tau_0, |G_{k+1}^o / G_{k-i_d-i_n}^o|\}\sigma_k, \\ i_d &= 0, \end{aligned} \tag{4.62}$$

else

$$\begin{aligned} \sigma_{k+1} &= \sigma_k, \\ i_d &= i_d + 1, \end{aligned} \tag{4.63}$$

end

else (no serious step, i.e. null or short step)

$$\begin{aligned} \sigma_{k+1} &= \sigma_k, \\ i_n &= i_n + 1. \end{aligned} \tag{4.64}$$

end

Compute the updates of the locality measure

$$s_j^{k+1} = s_j^k + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| \quad \text{for } j \in J_k, \tag{4.65}$$

$$s_{k+1}^{k+1} = \|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|, \tag{4.66}$$

$$s_p^{k+1} = \tilde{s}_p^k + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|, \tag{4.67}$$

$$\hat{s}_p^{k+1} = \tilde{\hat{s}}_p^k + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|, \tag{4.68}$$

$$\check{s}_p^{k+1} = \tilde{\check{s}}_p^k + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|. \tag{4.69}$$

Compute the updates for the objective function approximation, for $j \in J_k$,

$$\begin{aligned} f_j^{k+1} &= f_j^k + \boldsymbol{g}_j^{k\,T}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \tfrac{1}{2}\rho_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T G_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \\ f_{k+1}^{k+1} &= f_{k+1} + \boldsymbol{g}_{k+1}^T(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \\ &\quad + \tfrac{1}{2}\rho_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})^T G_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \end{aligned} \tag{4.70}$$

$$f_p^{k+1} = \tilde{f}_p^k + \tilde{\boldsymbol{g}}_p^{k\,T}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T G_p^{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \tag{4.71}$$

and for the constraints, for $j \in J_k$

$$
\begin{aligned}
F_j^{k+1} &= F_j^k + \hat{\boldsymbol{g}}_j^{k\,T}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \tfrac{1}{2}\hat{\rho}_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T \hat{G}_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k),\\
F_{k+1}^{k+1} &= F_{k+1} + \hat{\boldsymbol{g}}_{k+1}^T(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})\\
&\quad + \tfrac{1}{2}\hat{\rho}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})^T \hat{G}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \qquad (4.72)\\
F_p^{k+1} &= \tilde{F}_p^k + \tilde{\hat{\boldsymbol{g}}}_p^{k\,T}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T \hat{G}_p^{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \qquad (4.73)
\end{aligned}
$$

$$
\begin{aligned}
G_j^{k+1} &= G_j^k + \check{\boldsymbol{g}}_j^{k\,T}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \tfrac{1}{2}\check{\rho}_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T \check{G}_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k),\\
G_{k+1}^{k+1} &= G_{k+1}^o + \check{\boldsymbol{g}}_{k+1}^T(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})\\
&\quad + \tfrac{1}{2}\check{\rho}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})^T \check{G}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \qquad (4.74)\\
G_p^{k+1} &= \tilde{G}_p^k + \tilde{\check{\boldsymbol{g}}}_p^{k\,T}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T \check{G}_p^{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k). \qquad (4.75)
\end{aligned}
$$

Compute the updates for the subgradient of the objective function approximation

$$
\begin{aligned}
\boldsymbol{g}_j^{k+1} &= \boldsymbol{g}_j^k + \rho_j G_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \quad \text{for } j \in J_k,\\
\boldsymbol{g}_{k+1}^{k+1} &= \boldsymbol{g}_{k+1} + \rho_{k+1} G_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \qquad (4.76)\\
\boldsymbol{g}_p^{k+1} &= \tilde{\boldsymbol{g}}_p^k + G_p^{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \qquad (4.77)
\end{aligned}
$$

and for the constraints

$$
\begin{aligned}
\hat{\boldsymbol{g}}_j^{k+1} &= \hat{\boldsymbol{g}}_j^k + \hat{\rho}_j \hat{G}_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \quad \text{for } j \in J_k, \qquad (4.78)\\
\hat{\boldsymbol{g}}_{k+1}^{k+1} &= \hat{\boldsymbol{g}}_{k+1} + \hat{\rho}_{k+1}\hat{G}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \qquad (4.79)\\
\hat{\boldsymbol{g}}_p^{k+1} &= \tilde{\hat{\boldsymbol{g}}}_p^k + \hat{G}_p^{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \qquad (4.80)\\
\check{\boldsymbol{g}}_j^{k+1} &= \check{\boldsymbol{g}}_j^k + \check{\rho}_j \check{G}_j(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \quad \text{for } j \in J_k, \qquad (4.81)
\end{aligned}
$$

$$
\begin{aligned}
\check{\boldsymbol{g}}_{k+1}^{k+1} &= \check{\boldsymbol{g}}_{k+1} + \check{\rho}_{k+1}\check{G}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \qquad (4.82)\\
\check{\boldsymbol{g}}_p^{k+1} &= \tilde{\check{\boldsymbol{g}}}_p^k + \check{G}_p^{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k). \qquad (4.83)
\end{aligned}
$$

Choose $J_{k+1} \subseteq \{k - M + 2, \ldots, k + 1\} \cap \{1, 2, \ldots\}$ with $k + 1 \in J_{k+1}$. Set $k = k + 1$ and go to Step 1.

Remark 4.5 As in [16], the approximation of elements in $\partial^2 f(\boldsymbol{y})$, $\partial^2 F(\boldsymbol{y})$, and $\partial^2 G(\boldsymbol{y})$ only need to satisfy mild conditions for convergence. However, the speed of convergence will be influenced. Also, the parameters i_m and i_r as well as the additional parameter i_l are only needed for proving convergence. Since in practice we usually terminate an algorithm, if a maximal number of iterations `Nit_max` is exceeded, we always choose $i_m = i_n = i_l = \texttt{Nit\_max} + 1$ in our implementation. The case distinction for the choice of W according to (4.33) resp. (4.34) is only necessary for showing superlinear convergence for strongly convex, twice continuously differentiable functions. The choice $i_\rho = 3$ is due to empirical observations. A numerically meaningful choice of the matrices $\overline{G}_j^k$, $\overline{\hat{G}}_j^k$, $\overline{\hat{G}}^k$, $\overline{\check{G}}_j^k$ and $\overline{\check{G}}^k$, that occur in (4.36) is discussed in [16].

Proposition 4.7 *We have for all* $k \geq 0$

$$\|H_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)\|^2 = \boldsymbol{d}_k^T\big(\overline{W}_p^k + \sum_{j\in J_k}\lambda_j^k\overline{G}_j^k + \mu_j^k\overline{\hat{G}}_j^k + \nu_j^k\overline{\check{G}}_j^k + \lambda_p^k\overline{G}^k + \mu_p^k\overline{\hat{G}}^k + \nu_p^k\overline{\check{G}}^k\big)\boldsymbol{d}_k, \tag{4.84}$$

$$w_k = -\tfrac{1}{2}\boldsymbol{d}_k^T\overline{W}_p^k\boldsymbol{d}_k - v_k. \tag{4.85}$$

Proof Because of

$$H_k^{-2} = \overline{W}_p^k + \sum_{j\in J_k}\lambda_j^k\overline{G}_j^k + \mu_j^k\overline{\hat{G}}_j^k + \nu_j^k\overline{\check{G}}_j^k + \lambda_p^k\overline{G}^k + \mu_p^k\overline{\hat{G}}^k + \nu_p^k\overline{\check{G}}^k$$

due to (4.45) and

$$\boldsymbol{d}_k = -H_k^2(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)$$

due to (4.43), (4.46)–(4.49), (4.51), and, (4.53), easy calculations yield (4.84). Furthermore, (4.85) holds due to (4.84), (4.55) and (4.56). □

Remark 4.6 If we consider a nonsmooth inequality constrained optimization problem (i.e. we drop the constraint $G(\boldsymbol{x}) = 0$ in optimization the problem (4.2)), then our formula for v_k from (4.55) reduces to the formula for v_k in [16].

4.3.3 The Line Search

We extend the line search of [16] to the equality constrained case in the line search described in Algorithm 4.2. For obtaining a clear arrangement of the line search, we compute data concerning the objective function and the con-

straints in `ComputeObjectiveData`, and `ComputeConstraintDataF` and `ComputeConstraintDataG`, respectively. Before formulating the line search in detail, we give a brief overview of its functionality.

Starting with the step size $t = 1$, we check if the point $\boldsymbol{x}_k + t\boldsymbol{d}_k$ is strictly feasible with respect to F. If so and if additionally the objective function and the constraint violation in G decrease sufficiently in this point and t is not too small, then we take $\boldsymbol{x}_k + t\boldsymbol{d}_k$ as new iteration point in Algorithm 4.1 (serious step). Otherwise, if the point $\boldsymbol{x}_k + t\boldsymbol{d}_k$ is strictly feasible with respect to F, the constraint violation in G decreases sufficiently, and the model of the objective function changes sufficiently, we take $\boldsymbol{x}_k + t\boldsymbol{d}_k$ as new trial point (short/null step with respect to the objective function). If $\boldsymbol{x}_k + t\boldsymbol{d}_k$ is not strictly feasible or the constraint violation in G does not decrease sufficiently, but the model of the constraints changes sufficiently (in particular here the quadratic approximation of the constraints comes into play), we take $\boldsymbol{x}_k + t\boldsymbol{d}_k$ as new trial point (short/null step with respect to the constraint). After choosing a new step size $t \in [0, 1]$ by interpolation, we iterate this procedure.

Algorithm 4.2: Line search

Step 0. *(Initialization)* Choose $\zeta \in (0, \frac{1}{2})$ as well as $\vartheta \geq 1$ and set $t_L = 0$ as well as $t = t_U = 1$.

Step 1. *(Modification of either t_L or t_U)*

if $F(\boldsymbol{x}_k + t\boldsymbol{d}_k) < 0$ **then**
  if $f(\boldsymbol{x}_k + t\boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + m_L(1 - \sigma_k)v_k t$
      and $G(\boldsymbol{x}_k + t\boldsymbol{d}_k) \leq G(\boldsymbol{x}_k) + \ell_L \sigma_k v_k t$ **then**
    $t_L = t$,
  else
    $t_U = t$,
  end
else
  $t_U = t$,
  $t_0 = \hat{t}_0 t_U$, (4.86)
end
if $t_L \geq t_0$ **then**
  $t_R = t_L$,
  `return` (serious step).
end

Step 2. *(Decision of return)*

if $i_n < i_l$ **then**
  if $F(\boldsymbol{x}_k + t\boldsymbol{d}_k) < 0$ **then**
    $[g, G, \ldots] =$ `ComputeObjectiveData(t, ...)`,

```
        if Z = true then
            t_R = t,
            return (short/null step: change of model of the objective,
                    function),
        else
            [ğ, Ğ, ...] = ComputeConstraintDataG(t, ...),
            if Ž = true then
                t_R = t,
                return (short/null step: change of model of the equality
                        constraint),
            end
        end
    else
        [ĝ, Ĝ, ...] = ComputeConstraintDataF(t, ...),
        if Ẑ = true then
            t_R = t,
            return (short/null step: change of model of the inequality
                    constraint),
        end
    end
elseif i_n ≥ i_l then
    [g, G, ...] = ComputeObjectiveData(t, ...),
    if Z = false then
        [ğ, Ğ, ...] = ComputeConstraintDataG(t, ...),
    end
    if F(x_k + t d_k) < 0 and (Z or Ž) = true then
        t_R = t,
        return (short/null step: change of model of obj. or eq.const.).
    end
end
```

Step 3. *(Interpolation)* Choose $t \in [t_L + \zeta(t_U - t_L)^\vartheta, t_U - \zeta(t_U - t_L)^\vartheta]$.
Step 4. *(Loop)* Go to Step 1.

```
function [g, G, ...] =ComputeObjectiveData(t, ...)

  g = g(x_k + t d_k) ∈ ∂f(x_k + t d_k),
  G = approximation of G(x_k + t d_k) ∈ ∂²f(x_k + t d_k),
```

$$\rho = \begin{cases} \min\{1, \frac{C_G}{\|G\|}\}, & \text{for } i_n \le 3, \\ 0, & \text{else,} \end{cases}$$

$$f = f(\boldsymbol{x}_k + t\boldsymbol{d}_k) + (t_L - t)\boldsymbol{g}^T\boldsymbol{d}_k + \tfrac{1}{2}\rho(t_L - t)^2\boldsymbol{d}_k^T G\boldsymbol{d}_k, \tag{4.87}$$

$$\beta = \max\{|f(\boldsymbol{x}_k + t_L\boldsymbol{d}_k) - f|, \gamma_1|t_L - t|^{\omega_1}\|\boldsymbol{d}_k\|^{\omega_1}\}, \tag{4.88}$$

$$\overline{G} = \text{“positive definite modification of } G\text{”}, \tag{4.89}$$

$$Z = -\beta + \boldsymbol{d}_k^T\big(\boldsymbol{g} + \rho(t_L - t)G\boldsymbol{d}_k\big) \ge m_R v_k + m_f \cdot (-\tfrac{1}{2}\boldsymbol{d}_k^T\overline{G}\boldsymbol{d}_k), \quad \texttt{and } (t - t_L)\|\boldsymbol{d}_k\| \le C_S. \tag{4.90}$$

`function` $[\hat{\boldsymbol{g}}, \hat{G}, \ldots] =$`ComputeConstraintDataF(`$t$`, ...)`

$$\hat{\boldsymbol{g}} = g(\boldsymbol{x}_k + t\boldsymbol{d}_k) \in \partial F(\boldsymbol{x}_k + t\boldsymbol{d}_k),$$

$$\hat{G} = \text{ approximation of } \hat{G}(\boldsymbol{x}_k + t\boldsymbol{d}_k) \in \partial^2 F(\boldsymbol{x}_k + t\boldsymbol{d}_k),$$

$$\hat{\rho} = \min\{1, \tfrac{\hat{C}_G}{\|\hat{G}\|}\},$$

$$F = F(\boldsymbol{x}_k + t\boldsymbol{d}_k) + (t_L - t)\hat{\boldsymbol{g}}^T\boldsymbol{d}_k + \tfrac{1}{2}\hat{\rho}(t_L - t)^2\boldsymbol{d}_k^T\hat{G}\boldsymbol{d}_k, \tag{4.91}$$

$$\hat{\beta} = \max\{|F(\boldsymbol{x}_k + t_L\boldsymbol{d}_k) - F|, \gamma_2|t_L - t|^{\omega_2}\|\boldsymbol{d}_k\|^{\omega_2}\}, \tag{4.92}$$

$$\overline{\hat{G}} = \text{“positive definite modification of } \hat{G}\text{”}, \tag{4.93}$$

$$\hat{Z} = F(\boldsymbol{x}_k + t_L\boldsymbol{d}_k) - \hat{\beta} + \boldsymbol{d}_k^T\big(\hat{\boldsymbol{g}} + \hat{\rho}(t_L - t)\hat{G}\boldsymbol{d}_k\big) \ge m_F \cdot (-\tfrac{1}{2}\boldsymbol{d}_k^T\overline{\hat{G}}\boldsymbol{d}_k), \quad \texttt{and } (t - t_L)\|\boldsymbol{d}_k\| \le C_S. \tag{4.94}$$

`function` $[\check{\boldsymbol{g}}, \check{G}, \ldots] =$`ComputeConstraintDataG(`$t$`, ...)`

$$\check{\boldsymbol{g}} = \check{g}(\boldsymbol{x}_k + t\boldsymbol{d}_k) \in \partial G(\boldsymbol{x}_k + t\boldsymbol{d}_k),$$

$$\check{G} = \text{ approximation of } \check{G}(\boldsymbol{x}_k + t\boldsymbol{d}_k) \in \partial^2 G(\boldsymbol{x}_k + t\boldsymbol{d}_k),$$

$$\check{\rho} = \min\{1, \tfrac{\check{C}_G}{\|\check{G}\|}\},$$

$$G = G(\boldsymbol{x}_k + t\boldsymbol{d}_k) + (t_L - t)\check{\boldsymbol{g}}^T\boldsymbol{d}_k + \tfrac{1}{2}\check{\rho}(t_L - t)^2\boldsymbol{d}_k^T\check{G}\boldsymbol{d}_k, \tag{4.95}$$

$$\check{\beta} = \max\{|G(\boldsymbol{x}_k + t_L\boldsymbol{d}_k) - G|, \gamma_3|t_L - t|^{\omega_3}\|\boldsymbol{d}_k\|^{\omega_3}\}, \tag{4.96}$$

$$\overline{\check{G}} = \text{“positive definite modification of } \check{G}\text{”}, \tag{4.97}$$

$$\check{Z} = G(\boldsymbol{x}_k + t_L\boldsymbol{d}_k) - \check{\beta} + \boldsymbol{d}_k^T\big(\check{\boldsymbol{g}} + \check{\rho}(t_L - t)\check{G}\boldsymbol{d}_k\big) \ge m_G \cdot (-\tfrac{1}{2}\boldsymbol{d}_k^T\overline{\check{G}}\boldsymbol{d}_k), \quad \texttt{and } (t - t_L)\|\boldsymbol{d}_k\| \le C_S. \tag{4.98}$$

Remark 4.7 The parameter i_l is only necessary for proving global convergence of Algorithm 4.1 (to be more precise, it is only needed to show that a short or null step which changes the model of the objective function is executed in Lemma 4.17). If we choose $i_l = 0$, then only a change of the model of the objective function yields a short or null step. In fact we have i_l steps in Algorithm 4.1 in which we can use any meaningful criterion for terminating the line search, and (4.86) is like in [16].

The step sizes returned by the line search correspond to the points $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k$ and $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k = \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k$.

Only iteration points that are strictly feasible with respect to F are accepted in the line search

$$F(\boldsymbol{x}_k + t_L^k \boldsymbol{d}_k) < 0. \tag{4.99}$$

Nevertheless, trial points may be infeasible with respect to F (if $i_n < i_l$).

Proposition 4.8 *Let*

$$\hat{\alpha}_p^k := \sum_{j \in J_k} \zeta_j^k \alpha_j^k + \zeta_p^k \alpha_p^k,$$

$$\hat{A}_p^k := \sum_{j \in J_k} \kappa_j^k A_j^k + \kappa_p^k A_p^k, \quad \hat{B}_p^k := \sum_{j \in J_k} \eta_j^k B_j^k + \eta_p^k B_p^k, \tag{4.100}$$

$$\hat{w}_k := \tfrac{1}{2} \| H_k(\bar{\lambda}^{k+1} \tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1} \tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1} \tilde{\check{\boldsymbol{g}}}_p^k) \|^2 + \bar{\lambda}^{k+1} \hat{\alpha}_p^k + \bar{\kappa}^{k+1} \big(\hat{A}_p^k - F(\boldsymbol{x}_k) \big) + \bar{\nu}^{k+1} \hat{B}_p^k, \tag{4.101}$$

(Note: $\hat{w}_k$ is the optimal function value of the dual problem (4.21)). Then we have at iteration k of Algorithm 4.1

$$\hat{v}_k \le v_k \le 0 \le w_k \le \hat{w}_k. \tag{4.102}$$

Proof For $\gamma > 0$ and $\omega \ge 1$ the functions $\xi \mapsto \gamma |\xi|^\omega$ and $(\xi_1, \xi_2) \mapsto \max\{\xi_1, \xi_2\}$ are convex and thus we have $\gamma \big(\sum_{i=1}^k t_i \| \boldsymbol{x}_i \| \big)^\omega \le \sum_{i=1}^k t_i (\gamma \| \boldsymbol{x}_i \|^\omega)$ and $\max\{\sum_{i=1}^k t_i \boldsymbol{x}_i, \sum_{i=1}^k t_i \boldsymbol{y}_i\} \le \sum_{i=1}^k t_i \max\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$. For $\bar{\lambda}^{k+1} > 0$ we have that $\sum_{j \in J_k} \zeta_j^k + \zeta_p^k = 1$ and $\zeta_j^k \ge 0$ for $j \in J_k$ and $\zeta_p^k \ge 0$ holds by (4.46) for the solution of the dual problem (4.21) of the QCQP (4.42). This implies $f(\boldsymbol{x}_k) = \sum_{j \in J_k} \zeta_j^k f(\boldsymbol{x}_k) + \zeta_p^k f(\boldsymbol{x}_k)$, and hence $\tilde{\alpha}_p^k \le \hat{\alpha}_p^k$ follows from (4.50), (4.49), (4.39) and (4.100). If $\bar{\lambda}^{k+1} = 0$ this fact is trivial. Similar arguments prove $\tilde{A}_p^k \le \hat{A}_p^k$ and $\tilde{B}_p^k \le \hat{B}_p^k$. Consequently, we get $0 \le \bar{\lambda}^{k+1} \tilde{\alpha}_p^k + \bar{\kappa}^{k+1} \tilde{A}_p^k + \bar{\nu}^{k+1} \tilde{B}_p^k \le \bar{\lambda}^{k+1} \hat{\alpha}_p^k + \bar{\kappa}^{k+1} \hat{A}_p^k + \bar{\lambda}^{k+1} \hat{B}_p^k$. Now, we obtain the w_k-estimate of (4.102) due to (4.56), (4.101) and (4.46)–(4.48). Because of (4.100) and (4.46)–(4.48) we have $0 \ge -\bar{\lambda}^{k+1} \tilde{\alpha}_p^k - \bar{\kappa}^{k+1} \tilde{A}_p^k - \bar{\nu}^{k+1} \tilde{B}_p^k \ge -\sum_{j \in J_k} (\lambda_j^k \alpha_j^k + \mu_j^k A_j^k + \nu_j^k B_j^k) - \lambda_p^k \alpha_p^k - \mu_p^k A_p^k - \nu_p^k B_p^k$, and, therefore, we obtain the v_k-estimate of (4.102) by using (4.55), (4.46)–(4.48), (4.44), (4.85), the positive definiteness of $\overline{W}_p^k$ and (4.102). □

Like in [15, Proposition 9] we conclude that $v_k < 0$ if the line search is performed at iteration k of Algorithm 4.1, and that

$$-\tfrac{1}{2}\boldsymbol{d}_k^T \hat{\bar{G}}_{\boldsymbol{x}_k + t\boldsymbol{d}_k} \boldsymbol{d}_k < 0 \tag{4.103}$$

if there occurs a step size t with $F(\boldsymbol{x}_k + t\boldsymbol{d}_k) \geq 0$ in the line search.

Analogous to [15, Proposition 10] we can also prove that, provided the line search Algorithm 4.2 terminates with conditions (4.90), (4.94), or (4.98), the old search direction $\boldsymbol{d}_k$ of iteration k is sufficiently infeasible for the new QCQP (4.42) at iteration $k+1$.

4.4 Convergence

In the following section we give an outline of the proof of the convergence of the line search and the global convergence of the bundle algorithm.

4.4.1 Convergence of the Line Search

For convergence we will need to assume an additional property that is somewhat weaker than semismoothness (see Definition 1.11) and was given in [54].

Definition 4.4 Let $h : \mathbb{R}^n \to \mathbb{R}^m$ be Lipschitz on a ball around $\boldsymbol{x} \in \mathbb{R}^n$. We call h *weakly upper semismooth at* $\boldsymbol{x}$, if for all $\boldsymbol{d} \in \mathbb{R}^n$ and all sequences $\{t_k\}$ with $t_k \downarrow 0$ and $\{\boldsymbol{g}_k\}$ with $\boldsymbol{g}_k \in \partial h(\boldsymbol{x} + t_k \boldsymbol{d})$

$$\liminf_{k\to\infty} \boldsymbol{g}_k^T \boldsymbol{d} \geq \limsup_{t\downarrow 0} \frac{h(\boldsymbol{x} + t\boldsymbol{d}) - h(\boldsymbol{x})}{t}.$$

Proposition 4.9 *Let f and G be weakly upper semismooth, then the line search (Algorithm 4.2) terminates after finitely many steps with $t_L^k = t_L$, $t_R^k = t$ and $t_0^k > 0$.*

Proof If $F(\boldsymbol{x}_k + t\boldsymbol{d}_k) < 0$ for all $t \in [0, 1]$, then we can prove analogously to [46, pp. 379, Proof of Lemma 2.3] indirectly by assuming that infinitely many steps are taken. So in every step i we have $f(\boldsymbol{x}_k + t_i\boldsymbol{d}_k) - f(\boldsymbol{x}_k) > m_L t_i(1-\sigma_k)v_k$ or $G(\boldsymbol{x}_k + t_i\boldsymbol{d}_k) - G(\boldsymbol{x}_k) > m_L t_i \sigma_k v_k$. Hence, at least one of these conditions is satisfied infinitely often. For this condition weak upper semismoothness then leads to a contradiction.

Otherwise, since F is continuous and $F(\boldsymbol{x}_k) < 0$, there exists a largest $\tilde{t} > 0$ with $F(\boldsymbol{x}_k + \tilde{t}\boldsymbol{d}_k) = 0$ and $F(\boldsymbol{x}_k + s\boldsymbol{d}_k) < 0$ for all $s < \tilde{t}$. Therefore, after sufficiently many iterations in the line search (Algorithm 4.2) (Note that the interval $[t_L, t_U]$ is shrinking at each iteration of the line search), there only occur t_L, t_0, t_U with $0 \leq t_L < t_U < \tilde{t}$ and $0 < t_0 < t_U < \tilde{t}$ (i.e. from now on all $\boldsymbol{x}_k + t\boldsymbol{d}_k$ with $t \in \{t_L, t_U\}$

are feasible with respect to F) and consequently t_0 (where $\boldsymbol{x}_k + t_0 \boldsymbol{d}_k$ is also feasible with respect to F) does not change anymore (cf. (4.86)). Hence, here we also have exactly the same situation as in the proof [46, Proof of Lemma 2.3], where the only difference is the following additional argument to obtain the inequality at the bottom of [46, pp. 379]. Since $m_f \in [0, 1]$ due to the initialization of Algorithm 4.1 and since $\overline{G}$ is positive definite due to (4.89), the negation of the condition in (4.90) that corresponds to the change of the model of the objective function yields $-\beta + \boldsymbol{d}_k^T \left(\boldsymbol{g} + \rho(t_L - t) G \boldsymbol{d}_k\right) < m_R(1 - \sigma_k) v_k + m_f \cdot (-\frac{1}{2} \boldsymbol{d}_k^T \overline{G} \boldsymbol{d}_k) \leq m_R(1 - \sigma_k) v_k$. □

Remark 4.8 The proof of Proposition 4.9 only relies on f and G being weakly upper semismooth, the continuity of F and the strict feasibility of $\boldsymbol{x}_k$ with respect to F. In particular, F does not need to be weakly upper semismooth.

4.4.2 Global Convergence

For investigating the global convergence of Algorithm 4.1 we will closely follow the global convergence proof of the second order bundle method in [15] with modifications which concern the equality constrained case. We assume

$$\varepsilon = 0, \qquad \lambda_j^k = 0, \qquad \mu_j^k = 0, \qquad \nu_j^k = 0 \quad \text{for all } j \notin J_k. \tag{4.104}$$

We give a brief overview of the main steps of the proof. In Proposition 4.10 we express the p-tilde data (as, e.g., $\tilde{\boldsymbol{g}}_p^k$, $\tilde{\tilde{\boldsymbol{g}}}_p^k$, $\tilde{\check{\boldsymbol{g}}}_p^k, \dots$) as convex combinations in which no p-data (as, e.g., $\boldsymbol{g}_p^k$, $\hat{\boldsymbol{g}}_p^k$, $\check{\boldsymbol{g}}_p^k, \dots$) occurs. In Theorem 4.3 we show that if Algorithm 4.1 stops at iteration k, then the current iteration point $\boldsymbol{x}_k$ is stationary for the optimization problem (4.2). From then on we assume that the algorithm does not terminate (cf. (4.114)). Next, we deduce bounds for $\{(\overline{W}_p^k)^{-1}\}$ and $\{\overline{W}_p^k + \bar{\lambda}^{k+1} \overline{G}^k + \bar{\kappa}^{k+1} \hat{\overline{G}}^k + \bar{\mu}^{k+1} \check{\overline{G}}^k\}$ in Corollary 4.11, which will be essential in the following. Then, in Proposition 4.12, we show that if some boundedness assumptions are satisfied and the limit inferior of the sequence $\{\max\{w_k, \sigma_k, \|\boldsymbol{x}_k - \bar{\boldsymbol{x}}\|\}\}$ is zero, where $\bar{\boldsymbol{x}}$ denotes any accumulation point of the sequence of iteration points $\{\boldsymbol{x}_k\}$, then $\bar{\boldsymbol{x}}$ is stationary for the optimization problem (4.2), where the proof relies on Carathéodory's theorem as well as on the local boundedness and the upper semicontinuity of the subdifferentials ∂f, ∂F, and ∂G. Due to the negativity of v_k, which holds due to (4.102), we obtain the statement $t_L^k v_k \to 0$ in Proposition 4.13. In Proposition 4.14 we show some properties of the shifted sequences $\{\boldsymbol{x}_{k+i}\}$, $\{w_{k+i}\}$ and $\{t_L^{k+i}\}$, where we have to take care of the dependence of $(\boldsymbol{\lambda}^k, \lambda_p^k, \boldsymbol{\mu}^k, \mu_p^k, \boldsymbol{\nu}^k, \nu_p^k)$, which we noticed before, in the proof. Finally, we prove that under some additional boundedness assumptions the limit inferior of the sequence $\{\max\{w_k, \|\boldsymbol{x}_k - \bar{\boldsymbol{x}}\|\}\}$ is always zero and therefore Proposition 4.12 yields Theorem 4.4, which states that each accumulation point $\bar{\boldsymbol{x}}$ of the sequence of iteration points $\{\boldsymbol{x}_k\}$ is stationary for the optimization problem (4.2), under the assumption that the sequence $\{\sigma_k\}$ tends to zero.

Proposition 4.10 *Assume that Algorithm 4.1 has not stopped before iteration k with $k \geq 1$. Then*

$$(1 - \sigma_k)\bar{\lambda}^{k+1} + \sigma_k \bar{v}^{k+1} = 1. \tag{4.105}$$

If $\bar{\lambda}^{k+1} > 0$, then there exist $\hat{\zeta}_j^k \in \mathbb{R}$ for $j = 1, \ldots, k$ with

$$\hat{\zeta}_j^k \geq 0, \quad \sum_{j=1}^{k} \hat{\zeta}_j^k = 1, \quad (G_p^{k+1}, \tilde{\boldsymbol{g}}_p^k, \tilde{s}_p^k) = \sum_{j=1}^{k} \hat{\zeta}_j^k (\rho_j G_j, \boldsymbol{g}_j^k, s_j^k). \tag{4.106}$$

If $\bar{\lambda}^{k+1} = 0$, then the last equation in (4.106) holds with

$$\hat{\zeta}_j^k := 0 \quad \text{for all } j = 1, \ldots, k. \tag{4.107}$$

If $\bar{\kappa}^{k+1} > 0$, then there exist $\hat{\kappa}_j^k \in \mathbb{R}$ for $j = 1, \ldots, k$ with

$$\hat{\kappa}_j^k \geq 0, \quad \sum_{j=1}^{k} \hat{\kappa}_j^k = 1, \quad (\hat{G}_p^{k+1}, \tilde{\hat{\boldsymbol{g}}}_p^k, \tilde{\hat{s}}_p^k) = \sum_{j=1}^{k} \hat{\kappa}_j^k (\hat{\rho}_j \hat{G}_j, \hat{\boldsymbol{g}}_j^k, s_j^k). \tag{4.108}$$

If $\bar{\kappa}^{k+1} = 0$, then the last equation in (4.108) holds with

$$\hat{\kappa}_j^k := 0 \quad \text{for all } j = 1, \ldots, k. \tag{4.109}$$

If $\bar{v}^{k+1} > 0$, then there exists $\hat{\eta}_j^k \in \mathbb{R}$ for $j = 1, \ldots, k$ with

$$\hat{\eta}_j^k \geq 0, \quad \sum_{j=1}^{k} \hat{\eta}_j^k = 1, \quad (\check{G}_p^{k+1}, \tilde{\check{\boldsymbol{g}}}_p^k, \tilde{\check{s}}_p^k) = \sum_{j=1}^{k} \hat{\eta}_j^k (\check{\rho}_j \check{G}_j, \check{\boldsymbol{g}}_j^k, s_j^k). \tag{4.110}$$

If $\bar{v}^{k+1} = 0$, then the last equation in (4.110) holds with

$$\hat{\eta}_j^k := 0 \quad \text{for all } j = 1, \ldots, k. \tag{4.111}$$

Proof We get (4.105) from the equality condition of the dual QCQP (4.21). Equations (4.108) and (4.109) can be proved by a simple extension of the proof by induction in [15, Proposition 12]. The remaining claims follow with direct calculations from the definitions (4.46)–(4.48), (4.49), (4.53), (4.59), (4.60), and (4.61). □

Theorem 4.3 *If Algorithm 4.1 stops at iteration k with $\bar{\lambda}^{k+1} > 0$ and $i_d \leq i_m$, then there exist $\bar{\kappa}^{k+1}, \bar{v}^{k+1} \geq 0$ such that (4.8) holds for $(\boldsymbol{x}_k, \frac{\bar{\kappa}^{k+1}}{\bar{\lambda}^{k+1}}, \frac{\bar{v}^{k+1}}{\bar{\lambda}^{k+1}})$, i.e. $\boldsymbol{x}_k$ is stationary for the optimization problem (4.2).*

Proof Since Algorithm 4.1 stops at iteration k, step 5 of the algorithm with $i_d \leq i_m$, (4.104) and (4.102) imply $w_k = 0$ which is equivalent to

$$\begin{aligned} &\tfrac{1}{2}\|H_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)\|^2 = 0, \\ &\wedge\ \ \bar{\lambda}^{k+1}\tilde{\alpha}_p^k = 0 \ \ \wedge\ \ \bar{\kappa}^{k+1}\tilde{A}_p^k = 0 \ \ \wedge\ \ \bar{\nu}^{k+1}\tilde{B}_p^k = 0 \\ &\wedge\ \ \bar{\kappa}^{k+1}\big(-F(\boldsymbol{x}_k)\big) = 0, \end{aligned} \tag{4.112}$$

due to (4.56), (4.50), $\bar{\kappa}^{k+1} \geq 0$ and $F(\boldsymbol{x}_k) \leq 0$. Using the regularity of H_k, (4.50) and (4.49), we obtain from (4.112)

$$\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k = \boldsymbol{0}, \quad \tilde{s}_p^k = 0. \tag{4.113}$$

Furthermore, for $\bar{\kappa}^{k+1} > 0$ we obtain from (4.112), (4.52) and (4.51) that $\tilde{\hat{s}}_p^k = 0$ and hence we have either $\bar{\kappa}^{k+1} = 0$ or $\bar{\kappa}^{k+1} > 0 \ \wedge\ \tilde{\hat{s}}_p^k = 0$. In addition, for $\bar{\nu}^{k+1} > 0$ we get from (4.112), (4.54) and (4.53) that $\tilde{\check{s}}_p^k = 0$ and hence we have either $\bar{\nu}^{k+1} = 0$ or $\bar{\nu}^{k+1} > 0 \ \wedge\ \tilde{\check{s}}_p^k = 0$.

We set $\bar{\boldsymbol{x}} := \boldsymbol{x}_k$, $L := k$, $\bar{\boldsymbol{y}}_j := \boldsymbol{y}_j$, $\bar{s}_j := s_j^k$. Then for $\overline{G}_j := \rho_j G_j$, $\bar{\boldsymbol{g}}_j := \boldsymbol{g}_j$, $\bar{\lambda}_j := \bar{\lambda}^{k+1}\hat{\zeta}_j^k$, and $\bar{\boldsymbol{q}} := \bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k$ resp. for $\bar{\kappa}^{k+1} > 0$, $\overline{G}'_j := \hat{\rho}_j\hat{G}_j$, $\bar{\boldsymbol{g}}'_j := \hat{\boldsymbol{g}}_j$, $\bar{\lambda}'_j := \hat{\kappa}_j^k$, and $\bar{\boldsymbol{q}}' := \tilde{\hat{\boldsymbol{g}}}_p^k$, resp. for $\bar{\nu}^{k+1} > 0$, $\overline{G}''_j := \check{\rho}_j\check{G}_j$, $\bar{\boldsymbol{g}}''_j := \check{\boldsymbol{g}}_j$, $\bar{\lambda}''_j := \hat{\eta}_j^k$, and $\bar{\boldsymbol{q}}'' := \tilde{\check{\boldsymbol{g}}}_p^k$ the assumptions of [15, Proposition 13] are satisfied (by using Proposition 4.10), and therefore we obtain $\tilde{\boldsymbol{g}}_p^k \in \partial f(\boldsymbol{x}_k)$, $\tilde{\hat{\boldsymbol{g}}}_p^k \in \partial F(\boldsymbol{x}_k)$, and $\tilde{\check{\boldsymbol{g}}}_p^k \in \partial G(\boldsymbol{x}_k)$. Now, using (4.113) we calculate

$$\boldsymbol{0} \in \partial f(\boldsymbol{x}_k) + \frac{\bar{\kappa}^{k+1}}{\bar{\lambda}^{k+1}}\partial F(\boldsymbol{x}_k) + \frac{\bar{\nu}^{k+1}}{\bar{\lambda}^{k+1}}\partial G(\boldsymbol{x}_k).$$

In addition, since $\sigma_k = 0$, there must be $\tau_0 = 0$, and $G_\ell^o = 0$ for some $\ell \leq k$, and since all $v_i \leq 0$ by (4.62) and the construction of the line search, we get $G(\boldsymbol{x}_k) = 0$. □

From now on, we demand that we have for all k

$$w_k > 0. \tag{4.114}$$

Proposition 4.11 *If $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ is bounded, then $\{(\overline{W}_p^k)^{-1}\}$ and $\{H_k\}$ are bounded for all $k \geq 1$ with some positive constant $C_0 > 0$*

$$\|(\overline{W}_p^k)^{-1}\| \leq C_0. \tag{4.115}$$

If $\{\bar{\kappa}^{k+1}\}$ *and* $\{\bar{\nu}^{k+1}\}$ *are bounded and* $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ *is uniformly positive definite, then* $\{H_k^{-1}\}$ *is bounded and*

$$\|\overline{W}_p^k + \bar{\lambda}^{k+1}\overline{G}^k + \bar{\kappa}^{k+1}\bar{\hat{G}}^k + \bar{\nu}^{k+1}\bar{\check{G}}^k\| \leq C_1, \tag{4.116}$$

for all $k \geq 1$ *with some positive constant* $C_1 > 0$.

Proof Since $(\overline{W}_p^k)^{-\frac{1}{2}} = \big((\overline{W}_p^k)^{-1}\big)^{\frac{1}{2}}$ is bounded due to assumption, $\{(\overline{W}_p^k)^{-1}\}$ is bounded due to [15, Proposition 15] and therefore (4.115) holds with some positive constant $C_0 > 0$, which is equivalent to the uniform positive definiteness of $\{\overline{W}_p^k\}$ again due to [15, Proposition 15]. Since $H_k^{-2} - \overline{W}_p^k$ is positive definite for all $(\boldsymbol{\lambda}^k, \boldsymbol{\mu}^k, \boldsymbol{\nu}^k) \geq \mathbf{0}$ with $(1-\sigma_k)\big(\sum_{j\in J_k}\lambda_j^k + \lambda_p^k\big) + \sigma_k\big(\sum_{j\in J_k}\nu_j^k + \nu_p^k\big) = 1$ due to (4.45), we obtain $\|H_k^2\| \leq C_0$, which is equivalent to $\{H_k\}$ being bounded, due to (4.115) and [15, Propositions 14 and 15].

Since $\{\bar{\kappa}^{k+1}\}$ and $\{\bar{\nu}^{k+1}\}$ are bounded due to assumption, there exists a positive constant $\chi_0 > 0$ with $\bar{\kappa}^{k+1}, \bar{\nu}^{k+1} \leq \chi_0$ for all $k \geq 1$. Since $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ is uniformly positive definite due to assumption, $\{(\overline{W}_p^k)^{\frac{1}{2}}\}$ is bounded, which is equivalent to $\{\overline{W}_p^k\}$ being bounded, due to [15, Proposition 15], i.e. $\|\overline{W}_p^k\| \leq \chi_1$ for some positive constant $\chi_1 > 0$ and for all $k \geq 1$. Therefore, we obtain the boundedness of $\|H_k^{-2}\| \leq \chi_1 + \bar{C}_G + \chi_0(\bar{\hat{C}}_G + \bar{\check{C}}_G)$ due to (4.45), (4.47) and the initialization of Algorithm 4.1, which is equivalent to $\{H_k^{-1}\}$ being bounded due to [15, Proposition 15]. Furthermore, setting $C_1 := \chi_1 + \bar{C}_G + \chi_0(\bar{\hat{C}}_G + \bar{\check{C}}_G)$ yields (4.116) due to (4.47) and the initialization of Algorithm 4.1 □

From now on let the following assumption be satisfied.

Assumption 4.1 *Let* (4.114) *be satisfied. Furthermore, let* $\{(x_k, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})\}$ *be bounded,* $\sigma_k \to 0$*, and assume there exists* $\bar{\boldsymbol{x}} \in \mathbb{R}^N$ *with* $\sigma(\bar{\boldsymbol{x}}) = 0$*, where* $\sigma : \mathbb{R}^N \to \mathbb{R}$

$$\sigma(\boldsymbol{x}) := \liminf_{k\to\infty} \max\{w_k, \|\boldsymbol{x}_k - \boldsymbol{x}\|\}. \tag{4.117}$$

Moreover, let $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ *be uniformly positive definite.*

Next, we present Lemmas 4.1–4.7, which we need for proving Proposition 4.13.

Lemma 4.1 (Convergence of Basic Sequences) *There exist an infinite set* $K' \subset \{1, 2, \ldots, \}$ *and* $0 \leq \bar{\kappa}, \bar{\nu} \in \mathbb{R}$ *such that*

$$\bar{\lambda}^{k+1} \xrightarrow{K'} 1, \quad \bar{\kappa}^{k+1} \xrightarrow{K'} \bar{\kappa}, \quad \bar{\nu}^{k+1} \xrightarrow{K'} \bar{\nu}, \tag{4.118}$$

$$\boldsymbol{x}_k \xrightarrow{K'} \bar{\boldsymbol{x}}, \quad w_k \xrightarrow{K'} 0. \tag{4.119}$$

Proof Since we have $0 = \sigma(\bar{\boldsymbol{x}}) = \liminf_{k\to\infty} \max\{w_k, \|\boldsymbol{x}_k - \bar{\boldsymbol{x}}\|\}$ due to assumption and (4.117) and since $w_k \geq 0$, due to (4.102), and due to boundedness of the sequences considered, there exist convergent subsequences satisfying (4.119) and (4.118). Since $0 \geq \bar{\kappa}^{k+1}, \bar{\nu}^{k+1}$ for all k we see that the limits are nonnegative, as well. Because of (4.105) we conclude from the boundedness of $\{\nu^{k+1}\}$ and $\sigma_k \to 0$ that $\bar{\lambda}^{k+1}$ is bounded as well. Taking subsequences again, let $\bar{\lambda}$ be the limit of the $\bar{\lambda}^{k+1}$. Taking limits in (4.105) proves $\bar{\lambda} = 1$. □

Lemma 4.2 (Lagrange Multipliers) *Let* $I := \{1, 2, \ldots, N+2\}$ *(Note:* $|I| = N+2$*),* $S := \{(\boldsymbol{g}_j^k, s_j^k) : j = 1, \ldots, k\} \subseteq \mathbb{R}^{n+1}$*,* $\hat{S} := \{(\hat{\boldsymbol{g}}_j^k, s_j^k) : j = 1, \ldots, k\} \subseteq \mathbb{R}^{n+1}$*, and* $\check{S} := \{(\check{\boldsymbol{g}}_j^k, s_j^k) : j = 1, \ldots, k\} \subseteq \mathbb{R}^{n+1}$*. Then for* $i \in I$ *and* $k \geq 1$ *there exist* $\zeta^{k,i}, \kappa^{k,i}, \eta^{k,i} \in \mathbb{R}$ *and* $(\boldsymbol{g}^{k,i}, s^{k,i}) \in S$*,* $(\hat{\boldsymbol{g}}^{k,i}, \hat{s}^{k,i}) \in \hat{S}$*,* $(\check{\boldsymbol{g}}^{k,i}, \check{s}^{k,i}) \in \check{S}$ *such that*

$$\tilde{\boldsymbol{g}}_p^k = \sum_{i\in I} \zeta^{k,i} \boldsymbol{g}^{k,i}, \qquad \tilde{s}_p^k = \sum_{i\in I} \zeta^{k,i} s^{k,i}, \qquad 1 = \sum_{i\in I} \zeta^{k,i}, \quad \zeta^{k,i} \geq 0, \tag{4.120}$$

$$\tilde{\hat{\boldsymbol{g}}}_p^k = \sum_{i\in I} \kappa^{k,i} \hat{\boldsymbol{g}}^{k,i}, \qquad \tilde{\hat{s}}_p^k = \sum_{i\in I} \kappa^{k,i} \hat{s}^{k,i},$$

$$(1 = \sum_{i\in I} \kappa^{k,i} \ \wedge \ \kappa^{k,i} \geq 0) \text{ or } (\kappa^{k,i} = 0 \ \textit{ for all } i \in I), \tag{4.121}$$

$$\tilde{\check{\boldsymbol{g}}}_p^k = \sum_{i\in I} \eta^{k,i} \check{\boldsymbol{g}}^{k,i}, \qquad \tilde{\check{s}}_p^k = \sum_{i\in I} \eta^{k,i} \check{s}^{k,i},$$

$$(1 = \sum_{i\in I} \eta^{k,i} \ \wedge \ \eta^{k,i} \geq 0) \text{ or } (\eta^{k,i} = 0 \ \textit{ for all } i \in I). \tag{4.122}$$

In particular, we have

$$\sum_{i\in I} \kappa^{k,i} = \begin{cases} 1, \text{ if } \bar{\kappa}^{k+1} > 0, \\ 0, \text{ if } \bar{\kappa}^{k+1} = 0, \end{cases} \qquad \sum_{i\in I} \eta^{k,i} = \begin{cases} 1, \text{ if } \bar{\nu}^{k+1} > 0, \\ 0, \text{ if } \bar{\nu}^{k+1} = 0. \end{cases} \tag{4.123}$$

Proof We have $(\tilde{\boldsymbol{g}}_p^k, \tilde{s}_p^k) \in \operatorname{conv} S$ due to (4.114) and (4.106). Due to the theorem of Carathéodory for $i \in I$ and $k \geq 1$ there exist $(\boldsymbol{g}^{k,i}, s^{k,i}) \in S$ and $\zeta^{k,i} \in \mathbb{R}$ such that (4.120) holds. Furthermore, we have $(\tilde{\hat{\boldsymbol{g}}}_p^k, \tilde{\hat{s}}_p^k) \in \operatorname{conv} \hat{S}$ for $\bar{\kappa}^{k+1} > 0$ and $(\tilde{\hat{\boldsymbol{g}}}_p^k, \tilde{\hat{s}}_p^k) = \boldsymbol{0}$ for $\bar{\kappa}^{k+1} = 0$ due to (4.114) and (4.108). In the case $\bar{\kappa}^{k+1} > 0$ there exist $(\hat{\boldsymbol{g}}^{k,i}, \hat{s}^{k,i}) \in \hat{S}$, $\kappa^{k,i} \in \mathbb{R}$ for $i \in I$ with $1 = \sum_{i\in I} \kappa^{k,i}$, $\kappa^{k,i} \geq 0$ and $(\tilde{\hat{\boldsymbol{g}}}_p^k, \tilde{\hat{s}}_p^k) = \sum_{i\in I} \kappa^{k,i} (\hat{\boldsymbol{g}}^{k,i}, \hat{s}^{k,i})$ due to Carathéodory's theorem. In the case $\bar{\kappa}^{k+1} = 0$ choosing $\kappa^{k,i} := 0$ for all $i \in I$ yields $(\tilde{\hat{\boldsymbol{g}}}_p^k, \tilde{\hat{s}}_p^k) = \boldsymbol{0} = \sum_{i\in I} \kappa^{k,i} (\hat{\boldsymbol{g}}^{k,i}, \hat{s}^{k,i})$. Hence, (4.121) holds, which immediately implies the right equation in (4.123). Showing (4.122) and the right equation in (4.123) is done analogously. □

Lemma 4.3 (Equality Condition) *We have* $G(\boldsymbol{x}_k) \to 0$.

Proof From (4.62) we get for every k where σ_k changes that $\sigma_{k+1}/\sigma_k \geq |G(\boldsymbol{x}_{k+1})/G(\boldsymbol{x}_k)|$. For the other k we also find that inequality, since by construction of the line search the sequence $G(\boldsymbol{x}_k)$ is monontone decreasing. This implies that $\sigma_k/\sigma_0 \geq |G(\boldsymbol{x}_k)/G(\boldsymbol{x}_0)|$. Since $\sigma_k \to 0$ we conclude that $G(\boldsymbol{x}_k) \to 0$. □

Lemma 4.4 (Assignment) *There exists $j(k,i) \in \{1,\ldots,k\}$ (i.e. a function $j : \{k \in \mathbb{N} : k \geq 1\} \times I \to \{1,\ldots,k\}$) with $\boldsymbol{g}^{k,i} = \boldsymbol{g}^k_{j(k,i)}$, $s^{k,i} = s^k_{j(k,i)}$, $\hat{\boldsymbol{g}}^{k,i} = \hat{\boldsymbol{g}}^k_{j(k,i)}$, $\hat{s}^{k,i} = \hat{s}^k_{j(k,i)}$, $\check{\boldsymbol{g}}^{k,i} = \check{\boldsymbol{g}}^k_{j(k,i)}$, and $\check{s}^{k,i} = \check{s}^k_{j(k,i)}$.*

Proof Use $(\boldsymbol{g}^{k,i}, s^{k,i}) \in S$, $(\hat{\boldsymbol{g}}^{k,i}, \hat{s}^{k,i}) \in \hat{S}$, and $(\check{\boldsymbol{g}}^{k,i}, \check{s}^{k,i}) \in \check{S}$ for $i \in I$ and $k \geq 1$ from Lemma 4.2. □

Lemma 4.5 (Trial Point Convergence and Implications) *For all $i \in I$ there exist $\bar{\boldsymbol{y}}_i \in \mathbb{R}^N$ and (an infinite set) $K \subset K'$ with*

$$\boldsymbol{y}_{j(k,i)} \xrightarrow{K} \bar{\boldsymbol{y}}_i, \tag{4.124}$$

$$(\boldsymbol{g}_{j(k,i)}, \hat{\boldsymbol{g}}_{j(k,i)}, \check{\boldsymbol{g}}_{j(k,i)}) \xrightarrow{K} (\bar{\boldsymbol{g}}_i, \bar{\hat{\boldsymbol{g}}}_i, \bar{\check{\boldsymbol{g}}}_i) \in \partial f(\bar{\boldsymbol{y}}_i) \times \partial F(\bar{\boldsymbol{y}}_i) \times \partial G(\bar{\boldsymbol{y}}_i), \tag{4.125}$$

$$\begin{aligned} (\rho_{j(k,i)} G_{j(k,i)}, \zeta^{k,i}) &\xrightarrow{K} (\overline{G}_i, \bar{\lambda}_i), \\ (\hat{\rho}_{j(k,i)} \hat{G}_{j(k,i)}, \kappa^{k,i}) &\xrightarrow{K} (\overline{\hat{G}}_i, \bar{\kappa}_i), \\ (\check{\rho}_{j(k,i)} \check{G}_{j(k,i)}, \eta^{k,i}) &\xrightarrow{K} (\overline{\check{G}}_i, \bar{\nu}_i). \end{aligned} \tag{4.126}$$

Proof Since $\|\boldsymbol{y}_{j(k,i)}\| \leq \|\boldsymbol{x}_{j(k,i)}\| + C_S$ holds for all $i \in I$ and for all $k \geq 1$, the assumption of the boundedness of $\{\boldsymbol{x}_k\}$ yields that $\{\boldsymbol{y}_{j(k,i)}\}_{k\geq 1, i\in I}$ is bounded, hence has a convergent subsequence. Furthermore, the local boundedness of ∂f, ∂F, and ∂G (cf. Proposition 4.2) implies that the sets

$$\begin{aligned} B_1 &:= \{\boldsymbol{g} \in \partial f(\boldsymbol{y}_{j(k,i)}) : \ \boldsymbol{y}_{j(k,i)} \in \mathbb{R}^N, \ k \geq 1, \ k \in K_1, \ i \in I\}, \\ B_2 &:= \{\hat{\boldsymbol{g}} \in \partial F(\boldsymbol{y}_{j(k,i)}) : \ \boldsymbol{y}_{j(k,i)} \in \mathbb{R}^N, \ k \geq 1, \ k \in K_1, \ i \in I\}, \\ B_3 &:= \{\check{\boldsymbol{g}} \in \partial G(\boldsymbol{y}_{j(k,i)}) : \ \boldsymbol{y}_{j(k,i)} \in \mathbb{R}^N, \ k \geq 1, \ k \in K_1, \ i \in I\} \end{aligned}$$

are bounded. Therefore, $B_1 \times B_2 \times B_3$ is bounded and consequently there exists a convergent subsequence $\{\boldsymbol{g}_{j(k,i)}, \hat{\boldsymbol{g}}_{j(k,i)}, \check{\boldsymbol{g}}_{j(k,i)}\} \in \partial f(\boldsymbol{y}_{j(k,i)}) \times \partial F(\boldsymbol{y}_{j(k,i)}) \times \partial G(\boldsymbol{y}_{j(k,i)})$. The upper semicontinuity of ∂f, ∂F, and ∂G (cf. Proposition 4.2) and (4.124) imply that for all $i \in I$ (4.125) holds.

Since $\rho_{j(k,i)}, \hat{\rho}_{j(k,i)}, \check{\rho}_{j(k,i)} \in (0,1]$ due to (4.60), (4.61) and $C_G, \hat{C}_G, \check{C}_G > 0$, we obtain $\rho_{j(k,i)}\|G_{j(k,i)}\| \leq C_G$, $\hat{\rho}_{j(k,i)}\|\hat{G}_{j(k,i)}\| \leq \hat{C}_G$, and $\check{\rho}_{j(k,i)}\|\check{G}_{j(k,i)}\| \leq \check{C}_G$, which yields that all the sequences $\{\rho_{j(k,i)}\|G_{j(k,i)}\|\}$, $\{\hat{\rho}_{j(k,i)}\|\hat{G}_{j(k,i)}\|\}$, and $\{\check{\rho}_{j(k,i)}\|\check{G}_{j(k,i)}\|\}$ are bounded. Due to (4.120), (4.121), and (4.122) the sequences $\{\zeta^{k,i}\}$, $\{\kappa^{k,i}\}$, and $\{\eta^{k,i}\}$ are bounded.

Consequently, there exist convergent subsequences such that (4.126) holds. □

Lemma 4.6 (Complementarity Condition) *We have*

$$\sum_{i\in I}\bar\lambda_i\big(\bar{\boldsymbol g}_i+\overline{G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)+\bar\kappa\sum_{i\in I}\bar\kappa_i\big(\bar{\hat{\boldsymbol g}}_i+\overline{\hat G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)$$
$$+\bar\nu\sum_{i\in I}\bar\nu_i\big(\bar{\check{\boldsymbol g}}_i+\overline{\check G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)=\boldsymbol 0, \tag{4.127}$$

$$\zeta^{k,i}s^{k,i}\xrightarrow{K}0, \tag{4.128}$$

$$\kappa^{k,i}\hat s^{k,i}\xrightarrow{K}0,\quad \text{if } \bar\kappa>0, \tag{4.129}$$

$$\eta^{k,i}\check s^{k,i}\xrightarrow{K}0,\quad \text{if } \bar\nu>0. \tag{4.130}$$

Furthermore, the complementarity condition $\bar\kappa F(\bar{\boldsymbol x})=0$ holds.

Proof We calculate $\tilde{\boldsymbol g}_p^k\xrightarrow{K}\sum_{i\in I}\bar\lambda_i\big(\bar{\boldsymbol g}_i+\overline{G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)$, $\tilde{\hat{\boldsymbol g}}_p^k\xrightarrow{K}\sum_{i\in I}\bar\kappa_i\big(\bar{\hat{\boldsymbol g}}_i+\overline{\hat G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)$, and $\tilde{\check{\boldsymbol g}}_p^k\xrightarrow{K}\sum_{i\in I}\bar\nu_i\big(\bar{\check{\boldsymbol g}}_i+\overline{\check G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)$ by using (4.120)–(4.122), Lemma 4.4, (4.14), (4.119), (4.124), (4.125), and (4.126). Since $\{\bar\kappa^{k+1}\}$ and $\{\bar\nu^{k+1}\}$ are bounded and $\{(\overline{W}_p^k)^{-\frac12}\}$ is uniformly positive definite (both due to assumption), Corollary 4.11 implies the boundedness of $\{H_k^{-1}\}$. Because of (4.119), (4.56) and (4.102), we have $\|H_k(\bar\lambda^{k+1}\tilde{\boldsymbol g}_p^k+\bar\kappa^{k+1}\tilde{\hat{\boldsymbol g}}_p^k+\bar\nu^{k+1}\tilde{\check{\boldsymbol g}}_p^k)\|\xrightarrow{K}0$, which implies (4.127) due to the regularity of H_k, (4.118) and the uniqueness of a limit and $\tilde\alpha_p^k\xrightarrow{K}0$, which implies (4.128) due to (4.50), (4.120), Lemma 4.4, and (4.17), as well as $\bar\kappa^{k+1}F(\boldsymbol x_k)\xrightarrow{K}0$, which implies $0=\bar\kappa F(\bar{\boldsymbol x})$ due to (4.118), the continuity of F and (4.119), as well as $\bar\kappa^{k+1}\tilde A_p^k\xrightarrow{K}0$ which implies for $\bar\kappa>0$ that (4.129) holds due to (4.118), (4.52), (4.121), Lemma 4.4, (4.121) and (4.17). Also $\bar\nu^{k+1}\tilde B_p^k\xrightarrow{K}0$ and $G(\bar{\boldsymbol x})=0$ due to Lemma 4.3. Now (4.130) follows analogously to before. □

Lemma 4.7 (Subdifferential Elements) *We have*

$$\sum_{i\in I}\bar\lambda_i\big(\bar{\boldsymbol g}_i+\overline{G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)\in\partial f(\bar{\boldsymbol x}),$$

$$\begin{cases}\sum\limits_{i\in I}\bar\kappa_i\big(\bar{\hat{\boldsymbol g}}_i+\overline{\hat G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)\in\partial F(\bar{\boldsymbol x}), & \text{if } \bar\kappa>0,\\ \{\boldsymbol 0\}=\bar\kappa\,\partial F(\bar{\boldsymbol x}), & \text{if } \bar\kappa=0,\end{cases}$$

$$\begin{cases}\sum\limits_{i\in I}\bar\nu_i\big(\bar{\check{\boldsymbol g}}_i+\overline{\check G}_i(\bar{\boldsymbol x}-\bar{\boldsymbol y}_i)\big)\in\partial G(\bar{\boldsymbol x}), & \text{if } \bar\nu>0,\\ \{\boldsymbol 0\}=\bar\nu\,\partial G(\bar{\boldsymbol x}), & \text{if } \bar\nu=0.\end{cases}$$

Proof Since (4.120) holds for all $k\in K$, (4.126) implies $\sum_{i\in I}\bar\lambda_i=1$. Due to (4.126), we have $\lim_K\sum_{i\in I}\kappa^{k,i}=\sum_{i\in I}\bar\kappa_i$. If $\bar\kappa>0$, then—because of (4.118)

and since K is an infinite set—there exists $\hat{k} \in K$ such that $|\bar{\kappa}^{k+1} - \bar{\kappa}| < \frac{\bar{\kappa}}{2}$, which implies $0 < \frac{\bar{\kappa}}{2} < \bar{\kappa}^{k+1}$ for all $k \in \hat{K}$, where $\hat{K} := \left\{k \in K : k \geq \hat{k}\right\}$ is an infinite set. Therefore, we obtain $\sum_{i\in I} \kappa^{k,i} = 1$ for all $k \in \hat{K}$ due to (4.123), i.e. $\{\sum_{i\in I} \kappa^{k,i}\}_{k\in\hat{K}}$ is constant on $\hat{K}$ and hence we have $\lim_{\hat{K}} \sum_{i\in I} \kappa^{k,i} = 1$, hence the sequence $\{\sum_{i\in I} \kappa^{k,i}\}_{k\in K}$ converges to 1, as well, since we already know that it is convergent. Consequently, $\sum_{i\in I} \bar{\kappa}_i = 1$.

Due to (4.128) the sequence $\{\zeta^{k,i} s^{k,i}\}_{k\in K}$ is convergent and therefore necessarily bounded, i.e. there exists $C > 0$ with $0 \leq s^{k,i} \leq \frac{C}{\zeta^{k,i}}$ due to Lemma 4.4 as well as (4.17) and therefore $\{s^{k,i}\}_{k\in K}$ is bounded due to (4.126) for $\bar{\lambda}_i \neq 0$, where at least one such $\bar{\lambda}_i$ exists because $\sum_{i\in I} \bar{\lambda}_i = 1$. Since the locality measure is monotone due to (4.18), $\{s^{k,i}\}_{k\in K}$ is monotone. Consequently, $\{s^{k,i}\}_{k\in K}$ is convergent for $\bar{\lambda}_i \neq 0$, i.e. there exists $s_i := \lim_K s^{k,i}$. Therefore, (4.128), (4.126) and $\bar{\lambda}_i \neq 0$ implies $s_i = 0$. Hence, we obtain for $\bar{\lambda}_i \neq 0$ that $\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i\| = 0$ due to Lemma 4.4, (4.18), (4.124) and (4.119). For $\bar{\kappa} > 0$ and $\bar{\nu} > 0$ similar proofs using (4.129) and (4.130) show that $\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i\| = 0$ for $\bar{\kappa}_i \neq 0$ or $\bar{\nu}_i \neq 0$.

If we set

$$\bar{\boldsymbol{q}} := \sum_{i\in I} \bar{\lambda}_i \big(\bar{\boldsymbol{g}}_i + \overline{G}_i(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i)\big), \qquad \bar{s}_i := \left\{\begin{matrix} \|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i\| & \text{for } \bar{\lambda}_i = 0 \\ 0 & \text{for } \bar{\lambda}_i \neq 0 \end{matrix}\right\},$$

$$\bar{\boldsymbol{q}}' := \sum_{i\in I} \bar{\kappa}_i \big(\bar{\hat{\boldsymbol{g}}}_i + \overline{\hat{G}}_i(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i)\big), \qquad \bar{s}'_i := \left\{\begin{matrix} \|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i\| & \text{for } \bar{\kappa}_i = 0 \\ 0 & \text{for } \bar{\kappa}_i \neq 0 \end{matrix}\right\},$$

and analogously

$$\bar{\boldsymbol{q}}'' := \sum_{i\in I} \bar{\nu}_i \big(\bar{\check{\boldsymbol{g}}}_i + \overline{\check{G}}_i(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i)\big), \qquad \bar{s}''_i := \left\{\begin{matrix} \|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}_i\| & \text{for } \bar{\nu}_i = 0 \\ 0 & \text{for } \bar{\nu}_i \neq 0 \end{matrix}\right\},$$

then the assumptions of [15, Proposition 13] are satisfied and therefore we obtain the first two desired results. Since F and G are LLC, $\partial F(\bar{\boldsymbol{x}})$ and $\partial G(\bar{\boldsymbol{x}})$ are in particular bounded due to Proposition 4.2, and consequently we obtain $\bar{\kappa}\partial F(\bar{\boldsymbol{x}}) = \{\boldsymbol{0}\}$ in the case $\bar{\kappa} = 0$ and $\bar{\nu}\partial G(\bar{\boldsymbol{x}}) = \{\boldsymbol{0}\}$ if $\bar{\nu} = 0$. □

Proposition 4.12 *Let Assumption 4.1 be satisfied. Then there exist $\bar{\kappa}, \bar{\nu} \in \mathbb{R}_{\geq 0}$ such that (4.8) holds for $(\bar{\boldsymbol{x}}, \bar{\kappa}, \bar{\nu})$, i.e. if the sequence of iteration points and Lagrange multipliers is bounded, the sequence σ_k tends to zero, and the sequence of iteration points has an accumulation point with $\sigma(\bar{\boldsymbol{x}}) = 0$, then this accumulation point is stationary for the optimization problem (4.2).*

Proof Due to (4.99), the continuity of F and (4.119), we obtain $F(\bar{\boldsymbol{x}}) \leq 0$. Due to Lemma 4.6, the complementarity condition $\bar{\kappa} F(\bar{\boldsymbol{x}}) = 0$ holds. Using (4.127) and Lemma 4.7, we calculate $\boldsymbol{0} \in \partial f(\bar{\boldsymbol{x}}) + \bar{\kappa}\partial F(\bar{\boldsymbol{x}}) + \bar{\nu}\partial G(\bar{\boldsymbol{x}})$. $G(\bar{\boldsymbol{x}}) = 0$ by Lemma 4.3. □

Proposition 4.13 *Let* (4.114) *be satisfied. If there exist* $\bar{\boldsymbol{x}} \in \mathbb{R}^N$ *and* $K \subset \{1, 2, \dots\}$ *with* $\boldsymbol{x} \xrightarrow{K} \bar{\boldsymbol{x}}$, *then*

$$t_L^k v_k \xrightarrow{K} 0. \tag{4.131}$$

Proof [46, Proof of Lemma 3.5(ii)]. □

Proposition 4.14 *Let* (4.114) *be satisfied, let the sequence of (symmetric, positive definite matrices)* $\{H_k\}$ *be bounded and assume that there exists an infinite subset* $K \subset \{1, 2, \dots\}$ *and* $\bar{\boldsymbol{x}} \in \mathbb{R}^N$ *with*

$$\boldsymbol{x}_k \xrightarrow{K} \bar{\boldsymbol{x}}. \tag{4.132}$$

Then we have for all $i \geq 0$

$$\boldsymbol{x}_{k+i} \xrightarrow{k \xrightarrow{K} \infty} \bar{\boldsymbol{x}}. \tag{4.133}$$

If additionally $\sigma(\bar{\boldsymbol{x}}) > 0$ *holds, then we have for all* $i \geq 0$

$$t_L^{k+i} \xrightarrow{k \xrightarrow{K} \infty} 0, \tag{4.134}$$

and for fixed $\varepsilon_0 > 0$ *and for all fixed* $r \geq 0$ *there exists* $\tilde{k} \geq 0$ *such that*

$$w_{k+i} \geq \tfrac{\sigma(\bar{\boldsymbol{x}})}{2}, \quad t_L^{k+i} < \varepsilon_0 \tag{4.135}$$

for all $k > \tilde{k}$, $k \in K$ *and* $0 \leq i \leq r$.

Proof We show (4.133) by induction: the base case holds for $i = 0$ due to assumption (4.132). Now, let the induction hypothesis be satisfied for $i \geq 0$. We have

$$\boldsymbol{d}_{k+i} = H_{k+i}^2(\tilde{\boldsymbol{g}}_p^{k+i} + \bar{\kappa}^{k+i+1}\tilde{\tilde{\boldsymbol{g}}}_p^{k+i} + \bar{\nu}^{k+i+1}\tilde{\tilde{\boldsymbol{g}}}_p^{k+i}) \tag{4.136}$$

due to (4.43), (4.47), (4.49) and (4.51) as well as

$$\begin{aligned} &\tfrac{1}{2}\|H_{k+i}(\tilde{\boldsymbol{g}}_p^{k+i} + \bar{\kappa}^{k+i+1}\tilde{\tilde{\boldsymbol{g}}}_p^{k+i} + \bar{\nu}^{k+i+1}\tilde{\tilde{\boldsymbol{g}}}_p^{k+i})\|^2 \\ &\quad \leq \boldsymbol{d}_{k+i}^T \overline{W}_p^{k+i} \boldsymbol{d}_{k+i} + \tfrac{1}{2}\boldsymbol{d}_{k+i}^T\Big(\sum_{j \in J_{k+i}} \lambda_j^{k+i}\overline{G}_j^{k+i} + \lambda_p^{k+i}\overline{G}^{k+i} \\ &\qquad + \mu_j^{k+i}\bar{\hat{G}}_j^{k+i} + \mu_p^{k+i}\bar{\hat{G}}^{k+i} + \nu_j^{k+i}\bar{\check{G}}_j^{k+i} + \nu_p^{k+i}\bar{\check{G}}^{k+i}\Big)\boldsymbol{d}_{k+i} \end{aligned} \tag{4.137}$$

due to (4.84) and the positive definiteness of $\overline{W}_p^{k+i}$ as well as

$$\alpha_p^{k+i} + \bar{\kappa}^{k+i+1} A_p^{k+i} + \bar{\kappa}^{k+i+1}\big(-F(\boldsymbol{x}_{k+i})\big) + \bar{v}^{k+i+1} B_p^{k+i} \geq 0 \tag{4.138}$$

due to (4.50), (4.47), (4.52) and (4.99). Now, using (4.57), (4.136), (4.137), adding (4.138), using (4.55), the boundedness of $\{H_k\}$ (by assumption), $t_L^{k+i} \in [0,1]$ and (4.131) yields $\|\boldsymbol{x}_{k+(i+1)} - \boldsymbol{x}_{k+i}\| \overset{K}{\to} 0$, and therefore $\|\boldsymbol{x}_{k+(i+1)} - \bar{\boldsymbol{x}}\| \overset{K}{\to} 0$ follows from the induction hypothesis.

We show (4.134) by contradiction: suppose that (4.134) is false, i.e. there exists $i \geq 0$, $\bar{t} > 0$, $\bar{K} \subset K$: $t_L^{k+i} \geq \bar{t}$ for all $k \in \bar{K}$. Since $0 \leq \bar{t} w_{k+i} \leq -t_L^{k+i} v_{k+i} \overset{\bar{K}}{\to} 0$ due to (4.102), (4.56), (4.84), $t_L^{k+i} \in [0,1]$, (4.55) and (4.131), we have $w_{k+i} \overset{\bar{K}}{\to} 0$ and therefore we obtain $\sigma(\bar{\boldsymbol{x}}) = 0$ due to (4.117) and (4.133), which is a contradiction to the assumption $\sigma(\bar{\boldsymbol{x}}) > 0$.

We show (4.135). Let $r \geq 0$ be fixed and $0 \leq i \leq r$. Since we have $\frac{\sigma(\bar{\boldsymbol{x}})}{2} < \lim_K w_{k+i}$ due to the assumption $\sigma(\bar{\boldsymbol{x}}) > 0$, (4.117) and (4.133), because of (4.134) and because $\varepsilon_0 > 0$ is a fixed number by assumption, there exist $k_i \geq 0$ with $\frac{\sigma(\bar{\boldsymbol{x}})}{2} \leq w_{k+i}$ and $t_L^{k+i} < \varepsilon_0$ for all $k > k_i$ with $k \in K$. Now, setting $\tilde{k} := \max\{k_i : 0 \leq i \leq r\}$ yields (4.135). □

Proposition 4.15 *Let $\{\bar{\kappa}^{k+1}\}$ and $\{\bar{v}^{k+1}\}$ be bounded and let $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ be bounded and uniformly positive definite. For $k \geq 1$ we define $Z_k : \mathbb{R}^3_{\geq 0} \to \mathbb{R}^{N\times N}$*

$$Z_k(r,s,t) := \big(\overline{W}_p^k + r\overline{G}^k + s\hat{\overline{G}}^k + t\check{\overline{G}}^k\big)^{-\frac{1}{2}}. \tag{4.139}$$

Then we have for all $k \geq 1$

$$\begin{aligned} &\|Z_k(\bar{\lambda}^{k+2}, \bar{\kappa}^{k+2}, \bar{v}^{k+2}) - Z_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{v}^{k+1})\| \\ &\quad \leq C(|\bar{\lambda}^{k+2} - \bar{\lambda}^{k+1}| + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}| + |\bar{v}^{k+2} - \bar{v}^{k+1}|), \end{aligned} \tag{4.140}$$

for some positive constant C.

Proof We define for all $k \geq 1$

$$Y_k(r,s,t) := \big(\overline{W}_p^k + r\overline{G}^k + s\hat{\overline{G}}^k + t\check{\overline{G}}^k\big)^{-1}, \tag{4.141}$$

and therefore we have $\|Y_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{v}^{k+1})^{-1}\| \leq C_1$ for all $k \geq 1$ (4.141) and (4.116), which is equivalent to $\{Y_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{v}^{k+1})\}$ being uniformly positive definite due to [15, Proposition 15], i.e. there exists $\tilde{C}_2 > 0$ with $\lambda_{\min}(Y_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{v}^{k+1})) \geq \tilde{C}_2$. Consequently, we obtain for all $k \geq 1$

$$\frac{1}{(\lambda_{\min}(Y_k(\bar{\lambda}^{k+2}, \bar{\kappa}^{k+2}, \bar{v}^{k+2})))^{\frac{1}{2}} + (\lambda_{\min}(Y_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{v}^{k+1})))^{\frac{1}{2}}} \leq \tfrac{1}{2}\tilde{C}_2^{-\frac{1}{2}}$$

and hence we estimate for all $k \geq 1$

$$\begin{aligned}\|Y_k(\bar{\lambda}^{k+2}, \bar{\kappa}^{k+2}, \bar{\nu}^{k+2})^{\frac{1}{2}} - Y_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})^{\frac{1}{2}}\| \\ \leq C_2 \|Y_k(\bar{\lambda}^{k+2}, \bar{\kappa}^{k+2}, \bar{\nu}^{k+2}) - Y_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})\|,\end{aligned} \tag{4.142}$$

due to [15, Proposition 14], where we set $C_2 := \frac{1}{2}\tilde{C}_2^{-\frac{1}{2}} > 0$.

Defining

$$X_k(r, s, t) := \overline{W}_p^k + r\overline{G}^k + s\hat{\overline{G}} + t\check{\overline{G}}, \tag{4.143}$$

then X_k is continuously differentiable, and $\overline{W}_p^k \preceq X_k(r, s, t)$. Therefore, we estimate for all $k \geq 1$ and for all $s \geq 0$

$$\|Y_k(r, s, t)\| \leq C_0 \tag{4.144}$$

due to (4.141), (4.143), (4.115), and [15, Propositions 14, 20, and 21].

Straightforward calculations show that $Y_k(r, s, t)$ is continuously differentiable with bounded derivative, hence Lipschitz continuous with some positive Lipschitz constant C'.

Now, we estimate for all $k \geq 1$

$$\begin{aligned}\|Z_k(\bar{\lambda}^{k+2}, \bar{\kappa}^{k+2}, \bar{\nu}^{k+2}) - Z_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})\| \\ \leq C(|\bar{\lambda}^{k+2} - \bar{\lambda}^{k+1}| + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}| + |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}|)\end{aligned}$$

due to (4.139), (4.141), and (4.142). □

From now on let the following assumption be satisfied.

Assumption 4.2 *Let (4.114) be satisfied. Furthermore, let us assume that the sequence $\{(\boldsymbol{x}_k, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})\}$ is bounded, let the sequence (of symmetric, positive definite matrices) $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ be bounded and uniformly positive definite, and let $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ be any accumulation point of $\{\boldsymbol{x}_k\}$, i.e. there exists (an infinite set) $K \subset \{1, 2, \dots\}$ with*

$$\boldsymbol{x}_k \xrightarrow{K} \bar{\boldsymbol{x}}, \tag{4.145}$$

and demand

$$\begin{aligned}\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1} &\xrightarrow{K} 0, \\ \bar{\nu}^{k+2} - \bar{\nu}^{k+1} &\xrightarrow{K} 0,\end{aligned} \tag{4.146}$$

as well as $\sigma_k \to 0$ and

$$t_0^{\inf} := \inf_{k \geq 0} t_0^k > 0 \tag{4.147}$$

(cf. Remark 4.7). Let, furthermore, be $i_d \leq i_m$ in all iterations of the algorithm.

The following Lemmas 4.8–4.19, are tantamount for proving Theorem 4.4.

Lemma 4.8 (Bounded Basic Sequences) *The following boundedness statements hold:*

$$\{\boldsymbol{y}_k\}, \{\rho_k G_k\}, \{\hat{\rho}_k \hat{G}_k\}, \{\check{\rho}_k \check{G}_k\} \text{ and } \{\boldsymbol{g}_k\} \text{ are bounded,} \tag{4.148}$$

$$\{H_k\} \text{ is bounded,} \tag{4.149}$$

$$\{\boldsymbol{g}_k^k\}, \{H_k \boldsymbol{g}_k^k\} \text{ and } \{\alpha_k^k\} \text{ are bounded.} \tag{4.150}$$

Proof (4.148) holds as this statement was shown in the proof of Lemma 4.5, where only the assumption of the boundedness of $\{\boldsymbol{x}_k\}$ was used. Since $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ is bounded by assumption, (4.149) holds due to Corollary 4.11. Due to (4.76), the boundedness of $\{\boldsymbol{x}_k\}$ and (4.148) resp. $\|H_k \boldsymbol{g}_k^k\| \leq \|H_k\| \cdot \|\boldsymbol{g}_k^k\|$ and (4.149) resp. (4.39), (4.70), (4.66), (4.59), the Cauchy–Schwarz inequality and the fact that f is continuous on (the whole) $\mathbb{R}^n$, we obtain (4.150). □

Lemma 4.9 (Bounded Aggregate Sequences) *We define*

$$\tau_k := \bar{\lambda}^{k+1} \tilde{\alpha}_p^k + \bar{\kappa}^{k+1} \big(\tilde{A}_p^k - F(\boldsymbol{x}_k)\big) + \bar{\nu}^{k+1} \tilde{B}_p^k \geq 0, \tag{4.151}$$

then

$$\begin{aligned} &\{w_k\}, \{\tilde{\boldsymbol{g}}_p^k\}, \{\hat{\tilde{\boldsymbol{g}}}_p^k\}, \{\check{\tilde{\boldsymbol{g}}}_p^k\}, \{\tilde{\alpha}_p^k\}, \{\bar{\kappa}^{k+1} \tilde{A}_p^k\}, \{\bar{\nu}^{k+1} \tilde{B}_p^k\}, \\ &\quad \{H_k(\bar{\lambda}^{k+1} \tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1} \hat{\tilde{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1} \check{\tilde{\boldsymbol{g}}}_p^k)\}, \text{ and } \{\tau_k\} \text{ are bounded.} \end{aligned} \tag{4.152}$$

Proof Since $(\boldsymbol{\lambda}, \lambda_p, \boldsymbol{\mu}, \mu_p, \boldsymbol{\nu}, \nu_p) \in \mathbb{R}^{3(|J_k|+1)}$ with

$$\lambda_j := \left\{ \begin{array}{ll} \frac{1}{1-\sigma_k}, & \text{for } j = k \\ 0, & \text{for } j \in J_k \setminus \{k\} \end{array} \right\}, \quad \mu_j, \nu_j := 0 \text{ for all } j \in J_k, \ \ \lambda_p, \mu_p, \nu_p := 0 \tag{4.153}$$

is feasible for the (dual) problem (4.21) for $k \geq 1$ (Note: since $\sigma_k \to 0$ we can assume w.l.o.g. $(1 - \sigma_k) > 0$), we obtain (note: $\hat{w}_k$ is the optimal function value of (4.21)) due to (4.101), (4.49), (4.100), (4.51), (4.47) and inserting the feasible point from (4.153) that $\hat{w}_k \leq \frac{1}{2}\|H_k \boldsymbol{g}_k^k\|^2 + \alpha_k^k$. Hence, due to (4.56) and (4.102), we estimate

$$\begin{aligned} 0 \leq & \tfrac{1}{2}\|H_k(\bar{\lambda}^{k+1} \tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1} \hat{\tilde{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1} \check{\tilde{\boldsymbol{g}}}_p^k)\|^2 \\ & + \bar{\lambda}^{k+1} \tilde{\alpha}_p^k + \bar{\kappa}^{k+1} \big(\tilde{A}_p^k - F(\boldsymbol{x}_k)\big) + \bar{\nu}^{k+1} \tilde{B}_p^k \\ \leq & \tfrac{1}{2}\|H_k \boldsymbol{g}_k^k\|^2 + \alpha_k^k, \end{aligned}$$

and therefore (4.150) as well as the nonnegativity of $\tilde{\alpha}_p^k$, $\bar{\kappa}^{k+1}$, $\tilde{A}_p^k - F(\boldsymbol{x}_k)$, and $\tilde{B}_p^k$ due (4.50), (4.46)–(4.48), (4.52), (4.54) resp. (4.99) imply that $\{w_k\}$, $\{\bar{\lambda}^{k+1}\tilde{\alpha}_p^k\}$, $\{\bar{\kappa}^{k+1}\tilde{A}_p^k\}$, $\{\bar{\nu}^{k+1}\tilde{B}_p^k\}$, $\{H_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)\}$ and $\{\tau_k\}$ are bounded. Now, consider the proof of Lemma 4.6. There we only used the first consequence $x_k \xrightarrow{K} \bar{x}$ of (4.119) (and this property is also satisfied here due to (4.145)) of the assumption $\sigma(\bar{\boldsymbol{x}}) = 0$ for showing the convergence of $\tilde{\boldsymbol{g}}_p^k$, $\tilde{\hat{\boldsymbol{g}}}_p^k$, and $\tilde{\check{\boldsymbol{g}}}_p^k$ on a subsequence. Consequently, $\tilde{\boldsymbol{g}}_p^k$, $\tilde{\hat{\boldsymbol{g}}}_p^k$, and $\tilde{\check{\boldsymbol{g}}}_p^k$ are also bounded here. □

Lemma 4.10 (σ is Finite) *$\sigma(\bar{\boldsymbol{x}})$ is finite.*

Proof This is true due to (4.117), the assumption of the boundedness of $\{\boldsymbol{x}_k\}$ and (4.152). □

Lemma 4.11 (Cauchy Sequences) *We have*

$$s_p^{k+1} - \tilde{s}_p^k \xrightarrow{K} 0, \qquad \hat{s}_p^{k+1} - \tilde{\hat{s}}_p^k \xrightarrow{K} 0, \qquad \check{s}_p^{k+1} - \tilde{\check{s}}_p^k \xrightarrow{K} 0, \tag{4.154}$$

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \xrightarrow{K} 0, \quad F(\boldsymbol{x}_{k+1}) - F(\boldsymbol{x}_k) \xrightarrow{K} 0, \quad G(\boldsymbol{x}_{k+1}) - G(\boldsymbol{x}_k) \xrightarrow{K} 0, \tag{4.155}$$

$$f_p^{k+1} - \tilde{f}_p^k \xrightarrow{K} 0, \qquad F_p^{k+1} - \tilde{F}_p^k \xrightarrow{K} 0, \qquad G_p^{k+1} - \tilde{G}_p^k \xrightarrow{K} 0, \tag{4.156}$$

$$\boldsymbol{\Delta}_k \xrightarrow{K} 0, \tag{4.157}$$

where

$$\begin{aligned}\boldsymbol{\Delta}_k := H_{k+1}\big((\bar{\lambda}^{k+1}\boldsymbol{g}_p^{k+1} + \bar{\kappa}^{k+1}\hat{\boldsymbol{g}}_p^{k+1} + \bar{\nu}^{k+1}\check{\boldsymbol{g}}_p^{k+1}) \\ - (\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)\big).\end{aligned} \tag{4.158}$$

Proof Since the assumptions of Proposition 4.14 for applying (4.133) are satisfied—$\boldsymbol{x}_k \xrightarrow{K} \bar{\boldsymbol{x}}$ holds due to (4.145), $\sigma(\bar{\boldsymbol{x}}) > 0$ holds due to Lemma 4.10—applying (4.133) for $i = 1$ and $i = 0$ yields $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \xrightarrow{K} 0$. Due to (4.67) and (4.68), we obtain (4.154). Because of Lemma 4.3, (4.145) and the continuity of f, F, and G, we obtain (4.155). Due to (4.114) the assumptions of Proposition 4.10 are satisfied and therefore we estimate using (4.106), (4.148), (4.60), (4.108), (4.109), (4.110), (4.111), (4.148), and (4.61) that $\|G_p^{k+1}\| \leq C_G$, $\|\hat{G}_p^{k+1}\| \leq \hat{C}_G$, $\|\check{G}_p^{k+1}\| \leq \check{C}_G$. Due to (4.71), the Cauchy–Schwarz inequality and (4.152) resp. (4.73), the Cauchy–Schwarz inequality and (4.152) resp. (4.158), (4.77), (4.83), (4.46)–(4.48), (4.149) and the boundedness of $\{\bar{\kappa}^{k+1}\}$ and $\{\bar{\nu}^{k+1}\}$ (by assumption) and of $\{\bar{\lambda}^{k+1}\}$ (by implication), we obtain (4.156). □

Lemma 4.12 (Zero Sequence) *We have*

$$\left|\bar{\lambda}^{k+1}\left(\alpha_p^{k+1}-\tilde{\alpha}_p^k\right)+\bar{\kappa}^{k+1}\left(A_p^{k+1}-\tilde{A}_p^k+F(\boldsymbol{x}_k)-F(\boldsymbol{x}_{k+1})\right)\right.$$
$$\left.+\,\bar{v}^{k+1}\left(B_p^{k+1}-\tilde{B}_p^k\right)\right| \xrightarrow{K} 0.$$

Proof Because of $0 \leq \tilde{s}_p^k \leq \left(\frac{\tilde{\alpha}_p^k}{\gamma_1}\right)^{\frac{1}{\omega_1}}$ due to (4.49) and (4.50) and because of the boundedness of $\{\tilde{\alpha}_p^k\}$ due to (4.152), $\tilde{s}_p^k$ is bounded. Since the function $\xi \mapsto \xi^{\omega_1}$ with $\omega_1 \geq 1$ is Lipschitz continuous on every bounded subset of $\mathbb{R}_+$ there exists $c_L > 0$ with

$$|(s_p^{k+1})^{\omega_1} - (\tilde{s}_p^k)^{\omega_1}| \leq c_L |s_p^{k+1} - \tilde{s}_p^k|. \tag{4.159}$$

In the case $\bar{\kappa}^{k+1} = 0$, we have $\bar{\kappa}^{k+1}\tilde{\hat{s}}_p^k = 0$ due to (4.47) and (4.51). Now consider the case $\bar{\kappa}^{k+1} > 0$. Because of $0 \leq \bar{\kappa}^{k+1}\tilde{\hat{s}}_p^k \leq (\bar{\kappa}^{k+1})^{\omega_2}\left(\frac{\kappa^{k+1}\tilde{A}_p^k}{\gamma_2}\right)^{\frac{1}{\omega_2}}$ due to (4.51) and (4.52) and because of the boundedness of $\{\bar{\kappa}^{k+1}\}$ due to assumption and the boundedness of $\{\bar{\kappa}^{k+1}\tilde{A}_p^k\}$ due to (4.152), $\bar{\kappa}^{k+1}\tilde{\hat{s}}_p^k$ is bounded. Therefore, $\{\bar{\kappa}^{k+1}\tilde{\hat{s}}_p^k\}$ is bounded for all $\bar{\kappa}^{k+1} \geq 0$. Since the function $\xi \mapsto \xi^{\omega_2}$ with $\omega_2 \geq 1$ is Lipschitz continuous on every bounded subset of $\mathbb{R}_+$ there exists $\bar{c}_L > 0$ with $|(\bar{\kappa}^{k+1}\hat{s}_p^{k+1})^{\omega_2} - (\bar{\kappa}^{k+1}\tilde{\hat{s}}_p^k)^{\omega_2}| \leq \bar{c}_L\bar{\kappa}^{k+1}|\hat{s}_p^{k+1} - \tilde{\hat{s}}_p^k|$ and hence, using the assumption of the boundedness of $\{\bar{\kappa}^{k+1}\}$ and $\omega_2 \geq 1$ as well as setting $\hat{c}_L := \bar{c}_L \sup_{k\geq 1}(\bar{\kappa}^{k+1})^{1+\frac{1}{\omega_2}} < \infty$, we obtain

$$\bar{\kappa}^{k+1}|(\hat{s}_p^{k+1})^{\omega_2} - (\tilde{\hat{s}}_p^k)^{\omega_2}| \leq \hat{c}_L|\hat{s}_p^{k+1} - \tilde{\hat{s}}_p^k|. \tag{4.160}$$

Therefore, we have $|\alpha_p^{k+1} - \tilde{\alpha}_p^k| \xrightarrow{K} 0$ due to (4.39), (4.50), (4.159), (4.156), (4.155) and (4.154). Furthermore, due to (4.41) and (4.52), we obtain

$$|A_p^{k+1} - \tilde{A}_p^k| \leq |F_p^{k+1} - \tilde{F}_p^k| + |F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})| + \gamma_2|(\hat{s}_p^{k+1})^{\omega_2} - (\tilde{\hat{s}}_p^k)^{\omega_2}|.$$

Multiplying this last inequality with $\bar{\kappa}^{k+1} \geq 0$ (due to (4.47)) and using (4.160), the boundedness of $\{\bar{\kappa}^{k+1}\}$, (4.156), (4.155) and (4.154) yields $\bar{\kappa}^{k+1}|A_p^{k+1} - \tilde{A}_p^k| \xrightarrow{K} 0$ and $\bar{\kappa}^{k+1}|F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})| \xrightarrow{K} 0$. Therefore, using (4.47), we obtain the desired result.

Completely analogously we prove that

$$|B_p^{k+1} - \tilde{B}_p^k| \leq |G_p^{k+1} - \tilde{G}_p^k| + |G(\boldsymbol{x}_k) - G(\boldsymbol{x}_{k+1})| + \gamma_2|(\hat{s}_p^{k+1})^{\omega_3} - (\tilde{\hat{s}}_p^k)^{\omega_3}|$$

and this yields $\bar{v}^{k+1}|B_p^{k+1} - \tilde{B}_p^k| \xrightarrow{K} 0$. □

Lemma 4.13 (Estimates for Zero Sequences) *Assume $\sigma(\bar{\boldsymbol{x}}) > 0$. Then the constants*

$$\begin{aligned}
c &:= \sup_{k\geq 1}\Big(\|H_k \boldsymbol{g}_{k+1}^{k+1}\|, \|H_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\hat{\tilde{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\check{\tilde{\boldsymbol{g}}}_p^k)\|, \sqrt{\tau_k}\Big), \quad \delta := \tfrac{\sigma(\bar{\boldsymbol{x}})}{2},\\
\tilde{c} &:= \sup_{k\geq 1}(\|\boldsymbol{g}_{k+1}^{k+1}\| + \|\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\hat{\tilde{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\check{\tilde{\boldsymbol{g}}}_p^k\|), \quad \bar{c} := \delta\tfrac{1-m_R}{4c},\\
\widehat{C} &:= \tilde{c}C\max\{2c, 1, \tfrac{1}{2}\tilde{c}C\}
\end{aligned} \tag{4.161}$$

are finite and there exists $\bar{k} \geq 0$ such that

$$\begin{aligned}
\tfrac{1}{2}\bar{c}^2 &> 4c\|\boldsymbol{\Delta}_k\| + \tfrac{\|\boldsymbol{\Delta}_k\|^2}{2} + \big|\bar{\kappa}^{k+1}\big(A_p^{k+1} - \tilde{A}_p^k + F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})\big)\\
&\quad + \bar{\lambda}^{k+1}(\alpha_p^{k+1} - \tilde{\alpha}_p^k) + \bar{\nu}^{k+1}\big(B_p^{k+1} - \tilde{B}_p^k\big)\big|,\\
\tfrac{1}{2}\bar{c}^2 &> \widehat{C}(|\bar{\lambda}^{k+2} - \bar{\lambda}^{k+1}| + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}| + \|\boldsymbol{\Delta}_k\| \cdot |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}|\\
&\quad + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}|^2 + |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}| + \|\boldsymbol{\Delta}_k\| \cdot |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}|\\
&\quad + |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}|^2)
\end{aligned} \tag{4.162}$$

hold for all $k > \bar{k}$.

Proof We have that c is finite due to (4.161), (4.149), (4.150) and (4.152). Furthermore, we have $c > 0$ (If we had $c = 0$, then using (4.161), (4.151) and (4.56) would imply $w_k = 0$ for all $k \geq 1$, which is a contradiction to assumption (4.114)). Due to (4.161), $\sigma(\bar{\boldsymbol{x}}) > 0$ and $1 - m_R > 0$ (cf. the initialization of Algorithm 4.1, we have $\bar{c} = \frac{\sigma(\bar{\boldsymbol{x}})}{2} \cdot \frac{1-m_R}{4c}$, where $\sigma(\bar{\boldsymbol{x}}) > 0$ implies $\bar{c} > 0$, and Lemma 4.10 implies $\bar{c} < \infty$. Due to (4.161), (4.150), (4.152) and the assumption of the boundedness of $\{\bar{\kappa}^{k+1}\}$, $\{\bar{\nu}^{k+1}\}$, and $\tilde{c} \geq 0$ is bounded. Therefore, (4.161) and (4.140) imply $0 \leq \widehat{C}$. Since $4c\|\boldsymbol{\Delta}_k\| + \frac{\|\boldsymbol{\Delta}_k\|^2}{2} + \big|\bar{\lambda}^{k+1}(\alpha_p^{k+1} - \tilde{\alpha}_p^k) + \bar{\kappa}^{k+1}\big(A_p^{k+1} - \tilde{A}_p^k + F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})\big) + \bar{\nu}^{k+1}\big(B_p^{k+1} - \tilde{B}_p^k\big)\big| \xrightarrow{K} 0$ due to (4.156) and Lemma 4.12 and since $\widehat{C}(|\bar{\lambda}^{k+2} - \bar{\lambda}^{k+1}| + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}| + \|\boldsymbol{\Delta}_k\| \cdot |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}| + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}|^2 + |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}| + \|\boldsymbol{\Delta}_k\| \cdot |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}| + |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}|^2)| \xrightarrow{K} 0$ due to (4.146) and (4.105) there exists $\bar{k} \geq 0$ such that (4.162) holds for all $k > \bar{k}$. □

Lemma 4.14 (Estimate with Error Term) *We define for $k \geq 1$*

$$\begin{aligned}
\boldsymbol{q}_k &:= H_k \boldsymbol{g}_{k+1}^{k+1},\\
\boldsymbol{p}_k &:= H_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\hat{\tilde{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\check{\tilde{\boldsymbol{g}}}_p^k),\\
e_k &:= (2c + \|\boldsymbol{\Delta}_k\|)\tilde{c}\|E_k\| + \tfrac{1}{2}\tilde{c}^2\|E_k\|^2,\\
E_k &:= H_{k+1} - H_k.
\end{aligned} \tag{4.163}$$

Then we have for all $\xi \in [0,1]$ *and for all* $k \geq 1$

$$\begin{aligned} &\tfrac{1}{2}\|\xi H_{k+1}\boldsymbol{g}_{k+1}^{k+1}+(1-\xi)H_{k+1}(\bar{\lambda}^{k+1}\boldsymbol{g}_p^{k+1}+\bar{\kappa}^{k+1}\hat{\boldsymbol{g}}_p^{k+1}+\bar{\nu}^{k+1}\check{\boldsymbol{g}}_p^{k+1})\|^2 \\ &\qquad \leq \tfrac{1}{2}\|\xi \boldsymbol{q}_k+(1-\xi)(\boldsymbol{p}_k+\boldsymbol{\Delta}_k)\|^2+e_k. \end{aligned} \tag{4.164}$$

Proof Setting $z_k := E_k \boldsymbol{g}_{k+1}^{k+1}$, we obtain $H_{k+1}\boldsymbol{g}_{k+1}^{k+1} = \boldsymbol{q}_k + z_k$ due to (4.163). Setting $\hat{z}_k := E_k(\bar{\lambda}^{k+1}\tilde{\boldsymbol{g}}_p^k + \bar{\kappa}^{k+1}\tilde{\hat{\boldsymbol{g}}}_p^k + \bar{\nu}^{k+1}\tilde{\check{\boldsymbol{g}}}_p^k)$, we obtain $H_{k+1}(\bar{\lambda}^{k+1}\boldsymbol{g}_p^{k+1} + \bar{\kappa}^{k+1}\hat{\boldsymbol{g}}_p^{k+1} + \bar{\nu}^{k+1}\check{\boldsymbol{g}}_p^{k+1}) = \boldsymbol{p}_k + \boldsymbol{\Delta}_k + \hat{z}_k$ due to (4.158) and (4.163). Furthermore, we estimate for all $\xi \in [0,1]$

$$\big(\xi \boldsymbol{q}_k + (1-\xi)(\boldsymbol{p}_k+\boldsymbol{\Delta}_k)\big)^T\big(\xi z_k + (1-\xi)\hat{z}_k\big) \leq (2c + \|\boldsymbol{\Delta}_k\|)|\xi z_k + (1-\xi)\hat{z}_k|$$

due to the Cauchy–Schwarz inequality, (4.163) and (4.161) as well as $\|\xi z_k + (1-\xi)\hat{z}_k\| \leq \tilde{c}\|E_k\|$ due to (4.161). Hence we obtain (4.164) due to (4.163). □

Lemma 4.15 (Index Construction) *Assume* $\sigma(\bar{\boldsymbol{x}}) > 0$ *and define*

$$\hat{r} := \tfrac{3}{2}\cdot\frac{c^2}{\tilde{c}^2} + i_m, \quad r := i_l + \hat{r}. \tag{4.165}$$

Then there exists a finite index $k_0 \in K$ *such that*

$$w_k \geq \delta, \quad t_L^k < t_0^k, \tag{4.166}$$

$$i_n > i_l + i_m \tag{4.167}$$

hold for $k := k_0 + i_l + i$ *with* $i \in [i_m, \hat{r}] \cap \{0,1,\dots\}$.

Proof Completely analogous to the proof of [15, Lemma 14]. □

Lemma 4.16 (Error Estimate) *For* k *defined in Lemma 4.15 we have* $e_k < \frac{1}{2}\tilde{c}^2$.

Proof Since $i_n > i_l + i_m$ due to (4.167) and since i_n increases at most by one at each iteration due to (4.64) we have at iteration k at least $i_n \geq i_l + i_m$ and hence either the case (4.37) or (4.38) occurs (at iteration k). Furthermore, since $i_n > i_l + i_m$ due to (4.167) the cases (4.35) and (4.38) occur at iteration $k+1$. Therefore, combining these facts yields $E_k = Z_k(\bar{\lambda}^{k+2}, \bar{\kappa}^{k+2}, \bar{\nu}^{k+2}) - Z_k(\bar{\lambda}^{k+1}, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})$ due to (4.163), (4.45), (4.47), the fact that $\sum_{j\in J_k}\zeta_j^k + \zeta_p^k = 1 = \sum_{j\in J_{k+1}}\zeta_j^{k+1} + \zeta_p^{k+1}$ and $\sum_{j\in J_k}\eta_j^k + \eta_p^k = 1 = \sum_{j\in J_{k+1}}\eta_j^{k+1} + \eta_p^{k+1}$, and (4.139). Since $\{\bar{\kappa}^{k+1}\}$ and $\{\bar{\nu}^{k+1}\}$ are bounded and $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ is bounded as well as uniformly positive definite (by assumption), we can make use of Proposition 4.15 and hence we obtain $\|E_k\| \leq C(|\bar{\lambda}^{k+2} - \bar{\lambda}^{k+1}| + |\bar{\kappa}^{k+2} - \bar{\kappa}^{k+1}| + |\bar{\nu}^{k+2} - \bar{\nu}^{k+1}|)$ due to (4.140). Consequently, we obtain the desired estimate due to (4.163), (4.161) and (4.162). □

Lemma 4.17 (Termination Criterion Estimate) *For k defined in Lemma 4.15 a short or null step which changes the model of the objective function is executed and*

$$\begin{aligned} w_{k+1} \leq \big|&\bar{\lambda}^{k+1}(\alpha_p^{k+1} - \tilde{\alpha}_p^k) + \bar{\kappa}^{k+1}\big(A_p^{k+1} - \tilde{A}_p^k + F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})\big) \\ &+ \bar{\nu}^{k+1}\big(B_p^{k+1} - \tilde{B}_p^k\big)\big| + e_k + \min_{\xi\in[0,1]} \tfrac{1}{2}\|\xi \boldsymbol{q}_k + (1-\xi)(\boldsymbol{p}_k + \boldsymbol{\Delta}_k)\|^2 \\ &+ \xi\alpha_{k+1}^{k+1} + (1-\xi)\tau_k + \xi\sigma_k(H_k\tilde{\tilde{\boldsymbol{g}}}_p^k + B_p^{k+1})). \end{aligned}$$

Proof Combining (4.166) with step 6 (line search) of Algorithm 4.1 and considering the case $i_n > i_l + i_m \geq i_l$ due to (4.167) in the line search (Algorithm 4.2), we obtain that at iteration k a short or null step which changes the model of the objective function is executed. Furthermore, i_s is unchanged (since no serious step is executed), i.e. $i_s \leq i_r$ (no bundle reset) still holds (If $i_s > i_r$, then we would have had a serious step at iteration k, as a bundle reset can only occur after a serious step). Therefore, $(\boldsymbol{\lambda}, \lambda_p, \boldsymbol{\mu}, \mu_p, \boldsymbol{\nu}, \nu_p) \in \mathbb{R}^{3(|J_{k+1}|+1)}$ with

$$\begin{aligned} \lambda_j &:= \begin{cases} \xi, \text{ for } j = k+1, \\ 0, \text{ for } j \in J_{k+1} \setminus \{k+1\} \end{cases}, \\ \nu_p &:= \bar{\nu}^{k+1}\,,\ \lambda_p := 1-\xi,\ \nu_j := \mu_j := 0 \quad \text{for all } j \in J_{k+1}, \\ \mu_p &:= (1-\xi)\bar{\kappa}^{k+1}, \end{aligned} \tag{4.168}$$

where $\xi \in [0,1]$, is feasible for the $(k+1)$st (dual) problem (4.21) (note: This problem is written as a minimization problem) and, hence, due to (4.102), (4.101), (4.49), (4.51),(4.100), (4.47), inserting the feasible point from (4.168), (4.151), taking into account that $\xi \in [0,1]$ and (4.164), we estimate (note: $\hat{w}_{k+1}$ in (4.101) is the optimal function value of (4.21))

$$\begin{aligned} w_{k+1} \leq\ &\tfrac{1}{2}\|\xi \boldsymbol{q}_k + (1-\xi)(\boldsymbol{p}_k + \boldsymbol{\Delta}_k)\|^2 + e_k + \xi\alpha_{k+1}^{k+1} + (1-\xi)\tau_k \\ &+ \xi\sigma_k(H_k\tilde{\tilde{\boldsymbol{g}}}_p^k + B_p^{k+1}) \\ &+ \big|\bar{\lambda}^{k+1}(\alpha_p^{k+1} - \tilde{\alpha}_p^k) + \bar{\kappa}^{k+1}\big(A_p^{k+1} - \tilde{A}_p^k + F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})\big) \\ &+ \bar{\nu}^{k+1}\big(B_p^{k+1} - \tilde{B}_p^k\big)\big| \end{aligned}$$

and consequently, since $\xi \in [0,1]$ is arbitrary we obtain the desired estimate. □

Lemma 4.18 (Termination Criterion is Shrinking) *For k defined in Lemma 4.15 we have $w_{k+1} < w_k - \bar{c}^2$.*

Proof Since for $\boldsymbol{p} := \boldsymbol{p}_k$, $\boldsymbol{g} := \boldsymbol{q}_k$, $\boldsymbol{\Delta} := \boldsymbol{\Delta}_k$, $v := v_k - \frac{1}{2}\boldsymbol{d}_k^T\big(\sum_{j\in J_k}\lambda_j^k\bar{G}_j^k + \lambda_p^k\bar{G}^k + \mu_j^k\hat{\bar{G}}_j^k + \mu_p^k\hat{\bar{G}}^k + \nu_j^k\check{\bar{G}}_j^k + \nu_p^k\check{\bar{G}}^k\big)\boldsymbol{d}_k$, $w := w_k$, $\beta := \alpha_{k+1}^{k+1}$, $m := m_R$ and $\alpha := \tau_k$ the assumptions of [15, Proposition 19] are satisfied and since we have $\delta^2\frac{(1-m_R)^2}{8c^2} = 2\bar{c}^2$ due to (4.161) now applying [15, Proposition 19] yields the

desired estimate due to Lemma 4.17, (4.162), Lemmas 4.16 and (4.166), also using $\sigma_{k+1} \leq \sigma_k$. □

Lemma 4.19 (Contradiction) *For k_0 from Lemma 4.15 we have*

$$w_{k_0+n+1} < 0.$$

Proof Set $n := \max_{z \leq \hat{r}, z \in \{0,1,\dots\}} z$ (note that $\hat{r} > 0$ due to (4.165)), then we have $n + 1 > \hat{r}$ and hence (4.165) implies $-\bar{c}^2(n + 1 - i_m) < -\frac{3}{2}c^2$. Now, applying Lemma 4.18 $(n - i_m) + 1$ times as well as using (4.56), (4.151) and (4.161) yields

$$w_{k_0+n+1} < w_{k_0+i_m} - (n + 1 - i_m)\bar{c}^2 < w_{k_0+i_m} - \tfrac{3}{2}c^2 \leq \tfrac{1}{2}c^2 + c^2 - \tfrac{3}{2}c^2 = 0.$$

□

Theorem 4.4 *Let Assumption 4.2 be satisfied. Then there exist $\bar{\kappa}, \bar{\nu} \in \mathbb{R}_{\geq 0}$ such that (4.8) holds for $(\bar{\boldsymbol{x}}, \bar{\kappa}, \bar{\nu})$, i.e. each accumulation point of the sequence of iteration points $\{\boldsymbol{x}_k\}$ is stationary for the optimization problem (4.2).*

Proof (By Contradiction) Since $\{(\boldsymbol{x}_k, \bar{\kappa}^{k+1}, \bar{\nu}^{k+1})\}$ is bounded and $\{(\overline{W}_p^k)^{-\frac{1}{2}}\}$ is uniformly positive definite (both due to assumption) the statement follows from Proposition 4.12, if we can show $\sigma(\bar{\boldsymbol{x}}) = 0$. We suppose this is false, i.e. we have due to (4.117) $\sigma(\bar{\boldsymbol{x}}) > 0$ or $\sigma(\bar{\boldsymbol{x}}) = \infty$. Due to Assumption 4.2, we can make use of Lemmas 4.8–4.10, which implies that only the case $\sigma(\bar{\boldsymbol{x}}) > 0$ occurs. Therefore, we can use Lemmas 4.11–4.19, which yields a contradiction to the nonnegativity of w_k for all $k \geq 1$ due to (4.102). □

Remark 4.9 In examples that do not satisfy the nonsmooth constraint qualification (4.7), $\bar{\kappa}^{k+1}$ or $\bar{\nu}^{k+1}$ became very large in Algorithm 4.1 (note that Theorem 4.4 has in particular the assumption that $(\bar{\kappa}^{k+1}, \bar{\nu}^{k+1})$ is bounded).

The assumption (4.146) of Theorem 4.4 was satisfied in all numerical examples in [16] in which the termination criterion of Algorithm 4.1 was satisfied.

If t_0^k is only modified in, e.g., finitely many iterations of Algorithm 4.1, then (4.147) is satisfied (cf. Remark 4.7).

If we demand that all assumptions in the proof of convergence, which we imposed on $\overline{W}_p^k$, are satisfied for $\sum_{j \in J_k} \lambda_j^k \overline{G}_j^k + \lambda_p^k \overline{G}^k$, then the convergence result also holds in the case $\overline{W}_p^k = \mathbf{0}$. This is important, since first numerical results in the unconstrained case showed a better performance for the choice $\overline{W}_p^k = \mathbf{0}$, which is due to the fact that otherwise for a smooth, convex objective function f the Hessian information in the QCQP (4.20) is distorted—this can be seen by putting the constraints of the QCQP (4.20) into its objective function, which is then given by

$$\max_{j \in J_k} \left(-\alpha_j^k + \boldsymbol{d}^T \boldsymbol{g}_j^k + \tfrac{1}{2}\boldsymbol{d}^T \overline{G}_j^k \boldsymbol{d} \right) + \tfrac{1}{2}\boldsymbol{d}^T \overline{W}_p^k \boldsymbol{d}$$

$$= \max_{j \in J_k} \left(-\alpha_j^k + \boldsymbol{d}^T \boldsymbol{g}_j^k + \tfrac{1}{2}\boldsymbol{d}^T (\overline{G}_j^k + \overline{W}_p^k) \boldsymbol{d} \right).$$

The assumption $i_d \leq i_m$ throughout the algorithm excludes the case that the algorithm converges towards a local minimum in the constraint infeasibilty. This happens, e.g., when the feasible set is empty. A similar assumption was not necessary in [15], because the feasible second order bundle method assumes the existence of a strictly feasible starting point.

If the algorithm takes infinitely many steps with $i_d \leq i_m$, an infinite number of updates of σ_k is performed. Since every such update multiplies σ_k by a number in $[\tau_0, \tau_1] \subseteq (0, 1)$ in that case $\sigma_k \to 0$ automatically.

4.5 Conclusion

In this chapter we investigated the possibility of extending the second order bundle method developed in [15, 16] to nonsmooth nonlinearly constrained optimization problems, where we did not use a penalty function or a filter or an improvement function to handle the constraints. Instead we computed the search direction by solving a convex QCQP in the hope to obtain preferably feasible points that yield a good descent. Since the duality gap for such problems is zero, if the iteration point is strictly feasible, we were able to establish a global convergence result under certain assumptions. Furthermore, we discussed the presence of t_0^k in the line search, we explained why this should not be a problem when we use the solution of the QCQP as the search direction, and we refer to [16] that this turns out to be true in practice for at least many examples.

References

1. Bagirov, A.: A method for minimization of quasidifferentiable functions. Optim. Methods Softw. **17**, 31–60 (2002)
2. Bagirov, A.: Continuous subdifferential approximations and their applications. J. Math. Sci. **115**, 2567–2609 (2003)
3. Bagirov, A., Ganjehlou, A.: A quasisecant method for minimizing nonsmooth functions. Optim. Methods Softw. **25**(1), 3–18 (2010)
4. Bagirov, A., Beliakov, G., Mammadov, M., Rubinov, A.: Programming library GANSO (Global And Non-Smooth Optimization). Centre for Informatics and Applied Optimization, University of Ballarat, Australia (2005). http://www.ballarat.edu.au/ard/itms/CIAO/ganso/. Version 5.1.0.0
5. Bagirov, A., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**(2), 317–334 (2008)
6. Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization: Theoretical and Practical Aspects, 2nd edn. Springer, Berlin (2006)
7. Borwein, J., Lewis, A.: Convex Analysis and Nonlinear Optimization: Theory and Examples, 2nd edn. Springer, Berlin (2006)
8. Boţ, R. I., Csetnek, E. R.: An inertial Tseng's type proximal algorithm for nonsmooth and nonconvex optimization problems. J. Optim. Theory Appl. **171**(2), 600–616 (2016)

9. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
10. Burke, J., Lewis, A., Overton, M.: Approximating subdifferentials by random sampling of gradients. Math. Oper. Res. **27**(3), 567–584 (2002)
11. Burke, J., Lewis, A., Overton, M.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**(3), 751–779 (2005)
12. Byrd, R., Nocedal, J., Schnabel, R.: Representations of quasi-Newton matrices and their use in limited memory methods. Math. Program. **63**, 129–156 (1994)
13. Curtis, F., Overton, M.: A robust sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization. SIAM J. Optim. **22**(2), 474–500 (2012)
14. Eckstein, J., Bertsekas, D.P.: On the douglas–rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**(1–3), 293–318 (1992)
15. Fendl, H., Schichl, H.: A feasible second order bundle algorithm for nonsmooth, nonconvex optimization problems with inequality constraints: I. derivation and convergence. arXiv:1506.07937 (2015, preprint)
16. Fendl, H., Schichl, H.: A feasible second order bundle algorithm for nonsmooth nonconvex optimization problems with inequality constraints: II. implementation and numerical results. arXiv:1506.08021 (2015, preprint)
17. Fletcher, R., Leyffer, S.: A bundle filter method for nonsmooth nonlinear optimization. Numerical Analysis Report NA/195, University of Dundee, Department of Mathematics (1999)
18. Gill, P., Murray, W.: Newton-type methods for unconstrained and linearly constrained optimization. Math. Program. **28**, 311–350 (1974)
19. Gill, P., Murray, W., Saunders, M.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005)
20. Griewank, A., Corliss, G. (eds.): Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia (1991)
21. Haarala, M.: Large-scale nonsmooth optimization: variable metric bundle method with limited memory. Ph.D. Thesis, University of Jyväskylä, Department of Mathematical Information Technology (2004)
22. Haarala, M., Miettinen, M., Mäkelä, M.: New limited memory bundle method for large-scale nonsmooth optimization. Optim. Methods Softw. **19**(6), 673–692 (2004)
23. Haarala, N., Miettinen, M., Mäkelä, M.: Globally convergent limited memory bundle method for large-scale nonsmooth optimization. Math. Program. **109**(1), 181–205 (2007)
24. Herskovits, J.: Feasible direction interior-point technique for nonlinear optimization. J. Optim. Theory Appl. **99**(1), 121–146 (1998)
25. Herskovits, J., Santos, G.: On the computer implementation of feasible direction interior point algorithms for nonlinear optimization. Struct. Optim. **14**, 165–172 (1997)
26. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. J. Global Optim. **68**(3), 501–535 (2017)
27. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S.: Double bundle method for finding Clarke stationary points in nonsmooth DC programming. SIAM J. Optim. **28**(2), 1892–1919 (2018)
28. Jourani, A.: Constraint qualifications and lagrange multipliers in nondifferentiable programming problems. J. Optim. Theory Appl. **81**(3), 533–548 (1994)
29. Kappel, F., Kuntsevich, A.: An implementation of Shor's r-algorithm. Comput. Optim. Appl. **15**(2), 193–205 (2000). https://doi.org/10.1023/A:1008739111712
30. Karas, E., Ribeiro, A., Sagastizábal, C., Solodov, M.: A bundle-filter method for nonsmooth convex constrained optimization. Math. Program. **B**(116), 297–320 (2009)
31. Karmitsa, N.: Decision tree for nonsmooth optimization software. http://napsu.karmitsa.fi/solveromatic/
32. Karmitsa, N., Mäkelä, M.: Adaptive limited memory bundle method for bound constrained large-scale nonsmooth optimization. Optim. J. Math. Program. Oper. Res. **59**(6), 945–962 (2010)

33. Karmitsa, N., Mäkelä, M.: Limited memory bundle method for large bound constrained nonsmooth optimization: convergence analysis. Optim. Methods Softw. **25**(6), 895–916 (2010)
34. Karmitsa, N., Mäkelä, M., Ali, M.: Limited memory interior point bundle method for large inequality constrained nonsmooth minimization. Appl. Math. Comput. **198**(1), 382–400 (2008)
35. Karmitsa, N., Bagirov, A., Mäkelä, M.: Comparing different nonsmooth minimization methods and software. Optim. Methods Softw. **27**(1), 131–153 (2012)
36. Kiwiel, K.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics, vol. 1133. Springer, Berlin (1985)
37. Kiwiel, K.: A constraint linearization method for nondifferentiable convex minimization. Numer. Math. **51**, 395–414 (1987)
38. Kiwiel, K.: Restricted step and Levenberg-Marquardt techniques in proximal bundle methods for nonconvex nondifferentiable optimization. SIAM J. Optim. **6**(1), 227–249 (1996)
39. Kiwiel, K., Stachurski, A.: Issues of effectiveness arising in the design of a system of nondifferentiable optimization algorithms. In: Lewandowski, A., Wierzbicki, A. (eds.) Aspiration Based Decision Support Systems: Theory, Software and Applications. Lecture Notes in Economics and Mathematical Systems, vol. 331, pp. 180–192. Springer, Berlin (1989)
40. Kroshko, D.: `ralg`. http://openopt.org/ralg/. Software package
41. Lawrence, C., Tits, A.: A computationally efficient feasible sequential quadratic programming algorithm. SIAM J. Optim. **11**(4), 1092–1118 (2001)
42. Lewis, A., Overton, M.: Nonsmooth optimization via quasi–Newton methods. Math. Program. **141**, 135–163 (2013)
43. Lukšan, L.: An implementation of recursive quadratic programming variable metric methods for linearly constrained nonlinear minimax approximation. Kybernetika **21**(1), 22–40 (1985)
44. Lukšan, L., Vlček, J.: PBUN, PNEW – bundle-type algorithms for nonsmooth optimization. Technical report 718, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (1997). http://www.uivt.cas.cz/~luksan/subroutines.html
45. Lukšan, L., Vlček, J.: PMIN – a recursive quadratic programming variable metric algorithm for minimax optimization. Technical report 717, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (1997). http://www.uivt.cas.cz/~luksan/subroutines.html
46. Lukšan, L., Vlček, J.: A bundle-Newton method for nonsmooth unconstrained minimization. Math. Program. **83**, 373–391 (1998)
47. Lukšan, L., Vlček, J.: Globally convergent variable metric method for convex nonsmooth unconstrained minimization. J. Optim. Theory Appl. **102**(3), 593–613 (1999)
48. Lukšan, L., Vlček, J.: NDA: algorithms for nondifferentiable optimization. Technical report 797, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (2000). http://www.uivt.cas.cz/~luksan/subroutines.html
49. Mäkelä, M.: Multiobjective proximal bundle method for nonconvex nonsmooth optimization: FORTRAN subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B. Scientific computing, B 13/2003 University of Jyväskylä, Jyväskylä (2003). http://napsu.karmitsa.fi/proxbundle/
50. Mäkelä, M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific, Singapore (1992)
51. Mangasarian, O.L., Fromovitz, S.: The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. J. Math. Anal. Appl. **17**(1), 37–47 (1967)
52. Maratos, N.: Exact penalty function algorithms for finite dimensional and control optimization problems. Ph.D. Thesis, University of London (1978)
53. Merrill, O.: Applications and extensions of an algorithm that computes fixed points of certain upper semicontinuous point to set mappings. Ph.D. Thesis, University of Michigan, Ann Arbor (1972)
54. Mifflin, R.: An algorithm for constrained optimization with semismooth functions. Math. Oper. Res. **2**(2), 191–207 (1977)
55. Mifflin, R.: Semismooth and semiconvex functions in constrained optimization. SIAM J. Control Optim. **15**(6), 959–972 (1977)

56. Mifflin, R.: A modification and an extension of Lemarechal's algorithm for nonsmooth minimization. Math. Program. Study **17**, 77–90 (1982)
57. Mifflin, R., Sun, D., Qi, L.: Quasi-Newton bundle-type methods for nondifferentiable convex optimization. SIAM J. Optim. **8**(2), 583–603 (1998)
58. Nesterov, Y., Vial, J.P.: Homogeneous analytic center cutting plane methods for convex problems and variational inequalities. SIAM J. Optim. **9**, 707–728 (1999)
59. Nocedal, J., Wright, S.: Numerical Optimization. Springer Series in Operations Research and Financial Engineering, 2nd edn. Springer, New York (2006)
60. Oustry, F.: A second-order bundle method to minimize the maximum eigenvalue function. Math. Program. **89**(1), 1–33 (2000)
61. Overton, M.: Hanso: Hybrid algorithm for non-smooth optimization (2010). http://www.cs.nyu.edu/overton/software/hanso/. New York University, Department of Computer Science
62. Parikh, N., Boyd, S., et al.: Proximal algorithms. Found. Trends Optim. **1**(3), 127–239 (2014)
63. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM J. Control Optim. **14**(5), 877–898 (1976)
64. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**, 146–169 (2006)
65. Schramm, H.: Eine Kombination von Bundle- und Trust-Region-Verfahren zur Lösung nichtdifferenzierbarer Optimierungsprobleme. Ph.D. Thesis, Universität Bayreuth (1989)
66. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (1992)
67. Shor, N.: Minimization Methods for Non-Differentiable Functions. Springer, Berlin (1985)
68. Solodov, M.: On the sequential quadratically constrained quadratic programming methods. Math. Oper. Res. **29**(1), 1–90 (2004)
69. Tits, A.: Feasible sequential quadratic programming. In: Floudas, C., Pardalos, P. (eds.) Encyclopedia of Optimization, 2nd edn., pp. 1001–1005. Springer, Berlin (2009)
70. Vial, J.P., Sawhney, N.: OBOE User Guide Version 1.0 (2007). https://projects.coin-or.org/OBOE/
71. Vlček, J.: Bundle algorithms for nonsmooth unconstrained minimization. Research Report 608, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (1995)
72. Vlček, J., Lukšan, L.: Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. J. Optim. Theory Appl. **111**(2), 407–430 (2001)
73. Zowe, J.: The BT-algorithm for minimizing a nonsmooth functional subject to linear constraints. In: Clarke, F., Demyanov, V., Giannessi, F. (eds.) Nonsmooth Optimization and Related Topics, pp. 459–480. Plenum Press, New York (1989)

Chapter 5
Limited Memory Bundle Method and Its Variations for Large-Scale Nonsmooth Optimization

Napsu Karmitsa

Abstract There exist a vast variety of practical problems involving nonsmooth functions with large dimensions and nonconvex characteristics. Nevertheless, most nonsmooth solution methods have been designed to solve only small- or medium scale problems and they are heavily based on the convexity of the problem. In this chapter we describe three numerical methods for solving large-scale nonconvex NSO problems. Namely, the limited memory bundle algorithm (LMBM), the diagonal bundle method (D-BUNDLE), and the splitting metrics diagonal bundle method (SMDB). We also recall the convergence properties of these algorithms. To demonstrate the usability of the methods in large-scale settings, numerical experiments have been made using academic NSO problems with up to million variables.

5.1 Introduction

Nonsmooth optimization (NSO) problems arise in many application areas: for instance, in economics [33], mechanics [32], engineering [31], control theory [10], optimal shape design [16], machine learning [17], and data mining [3, 7] including cluster analysis [5, 11, 19, 20] and classification [1, 2, 6, 9]. Most of these problems are large-scale and nonconvex.

In this chapter, we consider solving the unconstrained NSO problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{5.1}$$

N. Karmitsa (✉)
University of Turku, Department of Mathematics and Statistics, Turku, Finland
e-mail: napsu@karmitsa.fi

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_5

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous (LLC) and the number of variables n is large. Neither differentiability nor convexity assumptions for the problem (5.1) are made.

The complexity of an optimization problem depends not only on the number of variables but also on the nature of the problem. That is, linearity, convexity, relaxation properties, possible structure etc. The most of the methods considered in this book are designed for small/medium-scale and/or convex problems: for example, in standard bundle methods (see Chap. 3) the computational demand often expands already with relatively small problems. One reason for this is that these methods need relatively large bundles (the number of stored subgradients $\sim n$) and, therefore, the size of the quadratic direction finding subproblem increases together with the number of variables. On the other hand, the basic subgradient methods (see Chap. 2) are guaranteed to work only in convex settings and they may converge slowly in larger cases. Gradient sampling methods (see Chap. 6) may have some potential but neither they have yet really proved themselves in large-scale settings.

In this chapter we recall three numerical methods for solving large-scale nonconvex NSO problems. Namely, the limited memory bundle algorithm (LMBM) [12–14], the diagonal bundle method (D-BUNDLE) [18], and the splitting metrics diagonal bundle method (SMDB) [23]. These methods and their slight modifications have been successfully applied in many practical problems, for example, clustering in large data sets [19, 20], regression analysis [22], missing data imputation [27], image denoising [30], and (coupled) data assimilation [15, 34]. In addition, constrained problems can be solved either by using the bound constrained [24, 25] or the inequality constrained [26] version of LMBM, or simply by using any of the methods with the exact penalty.

The LMBM is a crosbreed of the variable metric bundle methods [29, 35] and the limited memory variable metric methods (see e.g. [8]). Here, the variable metric bundle methods have been developed for small- and medium-scale NSO while the limited memory variable metric methods are designed for smooth large-scale optimization. Combining these two yields the LMBM for large-scale NSO. In its turn, the basic idea of the D-BUNDLE is to combine the LMBM with sparse matrix updating. Here the aim is to obtain a method for solving the problem (5.1) with the large number of variables, where the Hessian of the problem—if it exists—is sparse. The third variant, the SMDB, is a successor of the D-BUNDLE better capable of handling nonconvexity of problems.

All the methods considered in this chapter are characterized by the use of null steps together with the aggregation of subgradients. Moreover, the search direction is calculated using either the limited memory or the diagonal variable metric updates. Thus, the methods avoid solving the time-consuming quadratic direction finding subproblem appearing in the standard bundle methods as well as storing and manipulating large matrices as is the case in the variable metric bundle methods. Using null steps gives sufficient information about the nonsmooth objective function when the current search direction is not good enough and enables nonincreasing iterates as opposed to the standard subgradient methods. Finally, a simple aggregation procedure that uses only three subgradients guarantees the

convergence of the aggregate subgradients to zero and makes it possible to evaluate a termination criterion. These improvements make the LMBM and its successors suitable for large-scale NSO. Particularly, the number of operations needed for the calculation of the search direction and the aggregate values is only linearly dependent on the number of variables while, for example, with the variable metric bundle methods this dependence is quadratic.

This chapter is organized as follows. In Sects. 5.2, 5.3 and 5.4, we discuss the basic ideas of the LMBM, D-BUNDLE, and SMDB, respectively, and recall their convergence properties. In Sect. 5.5, we give the results of the numerical experiments and conclude the chapter.

Our notations are fairly standard: in addition, to standard Euclidean norm $\|\cdot\|$ and inner product $\langle\cdot,\cdot\rangle$, we denote by $\|A\|_F$ the Frobenius norm of the matrix $A \in \mathbb{R}^{n\times n}$ defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}^2}.$$

Further, we denote by $\text{diag}(\boldsymbol{a})$, for $\boldsymbol{a} \in \mathbb{R}^n$, the diagonal matrix such that $\text{diag}(\boldsymbol{a})_{i,i} = a_i$.

5.2 Limited Memory Bundle Method

In this section, we describe the LMBM for solving general, possibly nonconvex, large-scale NSO problems. To use the LMBM it is assumed that the objective function is LLC (see Definition 1.2) and at every point $\boldsymbol{x}$ both the value of the objective function $f(\boldsymbol{x})$ and one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential $\partial f(\boldsymbol{x})$ can be evaluated (see Definition 1.8 and Theorem 1.2). Now, we first describe in more detail the different components of the method—that is, the direction finding, serious and null steps, aggregation, matrix updating, and termination procedures—and then introduce the entire algorithm.

Direction Finding, Serious and Null Steps In the LMBM the *search direction* $\boldsymbol{d}_k$ is calculated using the classical limited memory variable metric scheme. Nevertheless, since the gradient does not need to exist for nonsmooth objective, the search direction is computed using (an aggregate) subgradient $\tilde{\boldsymbol{\xi}}_k$ instead of the usual gradient. That is,

$$\boldsymbol{d}_k = -D_k\tilde{\boldsymbol{\xi}}_k, \tag{5.2}$$

where D_k is the limited memory variable metric update which, in the smooth case, would represent the approximation of the inverse of the Hessian matrix. The matrix D_k is not formed explicitly but the search direction is calculated using the limited

memory approach (to be described later). The role of matrix D_k is to accumulate information about previous subgradients.

In NSO the search direction computed using a subgradient is not necessarily a descent one. Using so-called null steps gives more information about the nonsmooth objective if the current search direction is not "good enough". In order to determine a new step into the search direction $\boldsymbol{d}_k$, we use the *two steps line search procedure* (see Appendix for more details): a new iteration point $\boldsymbol{x}_{k+1}$ and a new auxiliary point $\boldsymbol{y}_{k+1}$ are produced such that

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k \qquad \text{and} \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k, \qquad \text{for } k \geq 1 \end{aligned} \tag{5.3}$$

with $\boldsymbol{y}_1 = \boldsymbol{x}_1$, where $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$ are step sizes, and $t_{max} > 1$ is the upper bound for the step size.

A necessary condition for a *serious step* to be taken is to have

$$t_R^k = t_L^k > 0 \qquad \text{and} \qquad f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L^k t_R^k w_k, \tag{5.4}$$

where $\varepsilon_L^k \in (0, 1/2)$ is a line search parameter, and $w_k > 0$ represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If the condition (5.4) is satisfied, we set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and a serious step is taken.

If condition (5.4) is not satisfied, a *null step* occurs. In null steps, we have

$$t_R^k > t_L^k = 0 \qquad \text{and} \qquad -\beta_{k+1} + \langle \boldsymbol{d}_k, \boldsymbol{\xi}_{k+1} \rangle \geq -\varepsilon_R^k w_k, \tag{5.5}$$

where $\varepsilon_R^k \in (\varepsilon_L^k, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, and β_{k+1} is the subgradient locality measure

$$\beta_{k+1} = \max\{|f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + \langle (\boldsymbol{y}_{k+1} - \boldsymbol{x}_k), \boldsymbol{\xi}_{k+1} \rangle|, \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|^2\}. \tag{5.6}$$

Here $\gamma \geq 0$ is a distance measure parameter supplied by the user ($\gamma > 0$ when f is nonconvex). In the case of a null step, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ but information about the objective function is increased because we store the auxiliary point $\boldsymbol{y}_{k+1}$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ and we use them to compute new aggregate values and the limited memory update matrix.

Under the semismoothness assumptions (see Definition 1.11) the line search procedure is guaranteed to find the step sizes t_L^k and t_R^k such that exactly one of the two possibilities—a serious step or a null step—occurs.

Aggregation The *aggregation procedure* used in the LMBM differs significantly from that usually used in bundle methods (see, e.g., [28]). Instead we use the procedure similar to the variable metric bundle methods [35], where only three subgradients and two locality measures are needed. The procedure is carried out by

determining multipliers λ_i^k satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^{3} \lambda_i^k = 1$ that minimize the function

$$\varphi(\lambda_1, \lambda_2, \lambda_3) = \langle \lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k \, , D_k[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k \,] \rangle + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k). \tag{5.7}$$

Here $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$ is the current subgradient (m denotes the index of the iteration after the latest serious step, i.e. $\boldsymbol{x}_k = \boldsymbol{x}_m$), $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ is the auxiliary subgradient, and $\tilde{\boldsymbol{\xi}}_k$ is the current aggregate subgradient from the previous iteration ($\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$). In addition, β_{k+1} is the current subgradient locality measure and $\tilde{\beta}_k$ is the current aggregate subgradient locality measure ($\tilde{\beta}_1 = 0$). The optimal values λ_i^k, $i \in \{1, 2, 3\}$ are easy to calculate (see [35]).

The resulting *aggregate subgradient* $\tilde{\boldsymbol{\xi}}_{k+1}$ and *aggregate subgradient locality measure* $\tilde{\beta}_{k+1}$ are computed by the formulae

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \qquad \tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{5.8}$$

The aggregation procedure gives us a possibility to retain the global convergence without solving the rather complicated quadratic direction finding subproblem appearing in standard bundle methods. Moreover, only one trial point $\boldsymbol{y}_{k+1}$ and the corresponding subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ need to be stored instead of $n + 3$ subgradients typically stored in standard bundle methods. Finally, it is worth of noting that the aggregate values need to be computed only if the last step was a null step. Otherwise, we just set $\tilde{\boldsymbol{\xi}}_{k+1} = \boldsymbol{\xi}_{k+1}$ and $\tilde{\beta}_{k+1} = 0$.

Matrix Updating The idea in limited memory matrix updating is that instead of storing and manipulating large $n \times n$ matrices D_k, one stores a small number of the so-called *correction vectors* obtained at the previous iterations of the algorithm, and uses these vectors to implicitly define the variable metric matrices. In the LMBM we use at most $\hat{m}_c$ correction vectors to compute updates for the matrix D_k. These correction vectors are slightly modified from those in classical limited memory variable metric methods for smooth optimization. That is, the correction vectors are given by $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$. Due to usage of null steps we may have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and thus, we use here the auxiliary point $\boldsymbol{y}_{k+1}$ instead of $\boldsymbol{x}_{k+1}$. In addition, since the gradient needs not to exist for nonsmooth objective, the correction vectors $\boldsymbol{u}_k$ are computed using subgradients.

Let us denote by $\hat{m}_c$ the user-specified maximum number of stored correction vectors ($3 \leq \hat{m}_c$) and by $\hat{m}_k = \min\{k - 1, \hat{m}_c\}$ the current number of stored correction vectors. Then the $n \times \hat{m}_k$ dimensional *correction matrices* S_k and U_k are defined by

$$S_k = \left[\boldsymbol{s}_{k-\hat{m}_k} \ldots \boldsymbol{s}_{k-1}\right] \qquad \text{and} \qquad U_k = \left[\boldsymbol{u}_{k-\hat{m}_k} \ldots \boldsymbol{u}_{k-1}\right]. \tag{5.9}$$

In the LMBM both the limited memory BFGS (L-BFGS) and the limited memory SR1 (L-SR1) update formulae [8] are used in calculations of the search direction and the aggregate values. In the case of a null step, the LMBM uses the L-SR1 update formula, since this formula allows us to preserve the boundedness and some other properties of generated matrices which guarantee the global convergence of the method. The *inverse L-SR1 update* is defined by

$$D_k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k {U_k}^T U_k - R_k - R_k^T + C_k)^{-1}(\vartheta_k U_k - S_k)^T,$$

where R_k is an upper triangular matrix of order $\hat{m}_k$ given by

$$(R_k)_{ij} = \begin{cases} \langle \boldsymbol{s}_{k-\hat{m}_k-1+i}, \boldsymbol{u}_{k-\hat{m}_k-1+j} \rangle, & \text{if } i \leq j, \\ 0, & \text{otherwise,} \end{cases}$$

C_k is a diagonal matrix of order $\hat{m}_k$ such that

$$C_k = \text{diag}\,[\langle \boldsymbol{s}_{k-\hat{m}_k}, \boldsymbol{u}_{k-\hat{m}_k} \rangle, \ldots, \langle \boldsymbol{s}_{k-1}, \boldsymbol{u}_{k-1} \rangle],$$

and ϑ_k is a positive scaling parameter.

Otherwise, since these additional properties are not required after a serious step, the more efficient L-BFGS update is employed. The *inverse L-BFGS update* is defined by the formula

$$D_k = \vartheta_k I + \begin{bmatrix} S_k & \vartheta_k U_k \end{bmatrix} \begin{bmatrix} (R_k^{-1})^T C_k + \vartheta_k U_k^T U_k R_k^{-1} & -(R_k^{-1})^T \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} {S_k}^T \\ \vartheta_k U_k^T \end{bmatrix}.$$

Note that we never compute the matrix D_k but only the product $D_k \boldsymbol{v}$ where $\boldsymbol{v}$ is equal to $\boldsymbol{\xi}_m$, $\boldsymbol{\xi}_{k+1}$ or $\tilde{\boldsymbol{\xi}}_k$. This way the number of operations needed for the calculation of the search direction and the aggregate values is only linearly dependent on the number of variables and we do not need to store large $n \times n$ matrices.

Stopping Criterion For smooth functions, a necessary condition for a local minimum is that the gradient has to be zero. By continuity a norm of the gradient becomes small when we are close to an optimal point providing a good stopping criterion for algorithms. This is no longer true when we replace the gradient with an arbitrary subgradient. In the LMBM the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$ provides a better approximation to the gradient but the direct test $\|\tilde{\boldsymbol{\xi}}_k\| < \varepsilon$, for some $\varepsilon > 0$, is still too uncertain as a stopping criterion. Therefore, the term $\langle \tilde{\boldsymbol{\xi}}_k, D_k \tilde{\boldsymbol{\xi}}_k \rangle = -\langle \tilde{\boldsymbol{\xi}}_k, \boldsymbol{d}_k \rangle$ and the aggregate subgradient locality measure $\tilde{\beta}_k$ are used to improve the accuracy of the sole norm $\|\tilde{\boldsymbol{\xi}}_k\|$. The *stopping parameter* w_k at iteration k is defined by

$$w_k = -\langle \tilde{\boldsymbol{\xi}}_k, \boldsymbol{d}_k \rangle + 2\tilde{\beta}_k \tag{5.10}$$

and the algorithm stops if $w_k \leq \varepsilon$ for a user specified tolerance $\varepsilon > 0$. In addition, the parameter w_k is used during the line search procedure to represent the desirable amount of descent.

Algorithm Now we present the algorithm for the LMBM.

Algorithm 5.1: LMBM

Data: The final accuracy tolerance $\varepsilon > 0$, initial line search parameters $\varepsilon_L^I \in (0, 1/2)$ and $\varepsilon_R^I \in (\varepsilon_L^I, 1/2)$, lower and upper bounds $t_{min} \in (0, 1)$ and $t_{max} > 1$ for serious steps, the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if f is convex), a control parameter $C > 0$ for the length of the direction vector, a correction parameter $\varrho \in (0, 1/2)$, and the number of stored corrections $\hat{m}_c \geq 3$.

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Set $\boldsymbol{y}_1 = \boldsymbol{x}_1$ and $\beta_1 = 0$. Compute $f_1 = f(\boldsymbol{x}_1)$ and $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$. Set $i_C = 0$ (a correction indicator) and $k = 1$.

Step 1. (*Serious step initialization.*) Set $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k$ and $\tilde{\beta}_k = 0$. Set $i_{CN} = 0$ (a correction indicator for consecutive null steps) and $m = k$.

Step 2. (*Direction finding*) Compute

$$\boldsymbol{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k$$

by the L-BFGS update if $m = k$ (use at most $\hat{m}_c$ correction vectors in U_k and S_k) and by the L-SR1 update, otherwise. If $k = 1$, set $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$.

Step 3. (*Correction*) If $-\langle \tilde{\boldsymbol{\xi}}_k, \boldsymbol{d}_k \rangle < \varrho \|\tilde{\boldsymbol{\xi}}_k\|^2$ or $i_{CN} = 1$, set

$$\boldsymbol{d}_k = \boldsymbol{d}_k - \varrho \tilde{\boldsymbol{\xi}}_k, \tag{5.11}$$

(i.e., $D_k = D_k + \varrho I$) and $i_C = 1$. Otherwise, set $i_C = 0$.
If $i_C = 1$ and $m < k$, then set $i_{CN} = 1$.

Step 4. (*Stopping criterion*) Compute w_k by (5.10). If $w_k < \varepsilon$, then **stop** with $\boldsymbol{x}_k$ as the final solution.

Step 5. (*Line search*) Set the scaling parameter for the length of the direction vector and for the line search

$$\theta_k = \min\{\, 1, C/\|\boldsymbol{d}_k\| \,\}.$$

Determine the step sizes $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$. Set the corresponding values

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \theta_k \boldsymbol{d}_k, \\
\boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \theta_k \boldsymbol{d}_k, \\
f_{k+1} &= f(\boldsymbol{x}_{k+1}), \qquad \text{and} \\
\boldsymbol{\xi}_{k+1} &\in \partial f(\boldsymbol{y}_{k+1}).
\end{aligned}$$

Set $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k = t_R^k \theta_k \boldsymbol{d}_k$ and append these values to U_k and S_k, respectively. If condition (5.4) is valid (i.e., we take a serious step),

then set $\beta_{k+1} = 0$, $k = k + 1$, and go to Step 1. Otherwise, calculate the locality measure β_{k+1} by (5.6).

Step 6. (*Aggregation.*) Determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^{3} \lambda_i^k = 1$ that minimize the function (5.7) where D_k is calculated by the same updating formula as in Step 2 and $D_k = D_k + \varrho I$ if $i_C = 1$. Compute $\tilde{\boldsymbol{\xi}}_{k+1}$ and $\tilde{\beta}_{k+1}$ as in (5.8). Set $k = k + 1$ and go to Step 2.

Remark 5.1 The boundedness of the matrices D_i^{-1} is required to guarantee the global convergence of the LMBM. We say that a matrix is bounded if its eigenvalues lie in the compact interval that does not contain zero. The utilization of correction (5.11) is equivalent to adding a positive definite matrix ϱI to the matrix D_k.

Remark 5.2 In order to guarantee the global convergence of the LMBM, the sequence $\{w_k\}$ (see (5.10)) has to be nonincreasing in consecutive null steps. That is, $w_k \leq w_{k-1}$ if $i_{null} = k - m > 1$. Therefore, the condition

$$\langle \tilde{\boldsymbol{\xi}}_k, (D_k - D_{k-1})\tilde{\boldsymbol{\xi}}_k \rangle \leq 0 \tag{5.12}$$

has to be satisfied each time there occurs more than one consecutive null step. In addition, the condition

$$-\langle \boldsymbol{d}_i, \boldsymbol{u}_i \rangle - \langle \tilde{\boldsymbol{\xi}}_i, \boldsymbol{s}_i \rangle < 0 \qquad \text{for all } i = 1, \ldots, k-1 \tag{5.13}$$

assures the positive definiteness of the matrices obtained (both L-SR1 and L-BFGS). In the LMBM, the individual updates that would violate either condition (5.12) or condition (5.13) are skipped.

Global Convergence of LMBM We now recall the convergence properties of the LMBM. But first, we give the assumptions needed.

Assumption 5.1 *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC (see Definition 1.2).*

Assumption 5.2 *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is upper semismooth (see Definition 1.11).*

Assumption 5.3 *The level set $\{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\}$ is bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$.*

Lemma 5.1 *Each execution of the line search procedure is finite.*

Proof See [35]. □

The optimality condition $\mathbf{0} \in \partial f(\boldsymbol{x})$ is sufficient if f is convex. Since the convexity of the function f is not assumed, we can only prove that the LMBM converges to a stationary point. We start by proving that if the LMBM terminates after a finite number of iterations, say at iteration k, then the point $\boldsymbol{x}_k$ is a stationary point of the problem (5.1). Then we prove that for an infinite sequence $\{\boldsymbol{x}_k\}$ every

accumulation point $\bar{\boldsymbol{x}}$ generated by the LMBM is a stationary point of the objective function. In order to do this, we assume that the final accuracy tolerance ε is equal to zero.

Remark 5.3 The sequence $\{\boldsymbol{x}_k\}$ generated by Algorithm 5.1 is bounded by assumption and the monotonicity of the sequence $\{f_k\}$ which, in turn, is obtained due to the condition (5.4) being satisfied for serious steps and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps. The sequence $\{\boldsymbol{y}_k\}$ is also bounded, since $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ for serious steps and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| \leq t_{max} C$ for null steps by (5.3), and due to the fact that we use the scaled direction vector $\theta_k \boldsymbol{d}_k$ with $\theta_k = \min\{1, C/\|\boldsymbol{d}_k\|\}$ and predefined $C > 0$ in the line search. By the local boundedness and the upper semi-continuity of the subdifferential, we obtain the boundedness of subgradients $\boldsymbol{\xi}_k$ as well as their convex combinations.

Lemma 5.2 *At the k-th iteration of Algorithm 5.1, we have*

$$w_k = \langle \tilde{\boldsymbol{\xi}}_k, D_k \tilde{\boldsymbol{\xi}}_k \rangle + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \varrho \|\tilde{\boldsymbol{\xi}}_k\|^2, \tag{5.14}$$

and

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2. \tag{5.15}$$

Proof We have $\tilde{\beta}_k \geq 0$ for all k by (5.6), (5.8), and Step 1 in Algorithm 5.1. Thus, relations in (5.14) follow immediately from (5.2), (5.10), and (5.11). If the correction (5.11) is used, we have $D_k = D_k + \varrho I$ and, thus, the result is valid also in this case.

By (5.6) and since we have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps, and, on the other hand, we have $\beta_{k+1} = 0$ and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| = 0$ for serious steps, the condition (5.15) always holds for some $\gamma \geq 0$. □

Lemma 5.3 *If condition (5.13) is valid for $k = k + 1$, then*

$$\langle \boldsymbol{u}_k, D_k \boldsymbol{u}_k - \boldsymbol{s}_k \rangle > 0. \tag{5.16}$$

Proof If condition (5.13) is valid for k replaced with $k+1$, then $\tilde{\boldsymbol{\xi}}_k \neq \boldsymbol{0}$. Otherwise, we would have $-\langle \boldsymbol{d}_k, \boldsymbol{u}_k \rangle - \langle \tilde{\boldsymbol{\xi}}_k, \boldsymbol{s}_k \rangle = 0$ that violates the condition. Furthermore, we have

$$\langle \boldsymbol{d}_k, \boldsymbol{u}_k \rangle > -\langle \tilde{\boldsymbol{\xi}}_k, \boldsymbol{s}_k \rangle = t_R^k \theta_k \langle \tilde{\boldsymbol{\xi}}_k, D_k \tilde{\boldsymbol{\xi}}_k \rangle, \tag{5.17}$$

with $t_R^k > 0$ and $\theta_k \in (0, 1]$. Using Cauchy's inequality and the positiveness of $\langle \boldsymbol{u}_k, \boldsymbol{s}_k \rangle$ (provided by the positive definiteness of D_k in (5.17) and the fact that

$\boldsymbol{s}_k = t_R^k \theta_k \boldsymbol{d}_k$) we obtain

$$\begin{aligned}\langle \boldsymbol{u}_k, \boldsymbol{s}_k\rangle^2 &= (t_R^k \theta_k \langle \tilde{\boldsymbol{\xi}}_k, D_k \boldsymbol{u}_k\rangle)^2 \\ &\leq (t_R^k \theta_k)^2 \langle \tilde{\boldsymbol{\xi}}_k, D_k \tilde{\boldsymbol{\xi}}_k\rangle \langle \boldsymbol{u}_k, D_k \boldsymbol{u}_k\rangle \\ &= t_R^k \theta_k \langle \boldsymbol{u}_k, D_k \boldsymbol{u}_k\rangle \langle -\boldsymbol{s}_k, \tilde{\boldsymbol{\xi}}_k\rangle \\ &< t_R^k \theta_k \langle \boldsymbol{u}_k, D_k \boldsymbol{u}_k\rangle \langle \boldsymbol{d}_k, \boldsymbol{u}_k\rangle = \langle \boldsymbol{u}_k, D_k \boldsymbol{u}_k\rangle \langle \boldsymbol{u}_k, \boldsymbol{s}_k\rangle.\end{aligned}$$

Therefore, we have $\langle \boldsymbol{u}_k, \boldsymbol{s}_k\rangle < \langle \boldsymbol{u}_k, D_k \boldsymbol{u}_k\rangle$. □

Lemma 5.4 *Suppose that Algorithm 5.1 is not terminated before the k-th iteration. Then, there exist numbers $\lambda^{k,j} \geq 0$ for $j = 1, \ldots, k$ and $\tilde{\sigma}_k \geq 0$ such that*

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\sigma}_k) = \sum_{j=1}^{k} \lambda^{k,j} (\boldsymbol{\xi}_j, \|\boldsymbol{y}_j - \boldsymbol{x}_k\|), \qquad \sum_{j=1}^{k} \lambda^{k,j} = 1, \quad \textit{and} \quad \tilde{\beta}_k \geq \gamma \tilde{\sigma}_k^2.$$

Proof ([35]) Let m be an index of the iteration after the latest serious step defined at Step 1 of Algorithm 5.1 (that is, $\boldsymbol{x}_j = \boldsymbol{x}_m$ for all $j = m, \ldots, k$). First we prove that there exist numbers $\lambda^{k,j} \geq 0$ for $j = m, \ldots, k$, such that

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\beta}_k) = \sum_{j=m}^{k} \lambda^{k,j} (\boldsymbol{\xi}_j, \beta_j), \qquad \sum_{j=m}^{k} \lambda^{k,j} = 1. \tag{5.18}$$

We prove this by induction. Suppose that $k = m$. Then we set $\lambda^{m,m} = 1$, since $\tilde{\boldsymbol{\xi}}_m = \boldsymbol{\xi}_m$ and $\tilde{\beta}_m = 0$ at Step 1 of Algorithm 5.1 and we have set $\beta_m = 0$ at Step 5 at the previous iteration ($\beta_1 = 0$ due to initialization). Thus, the base case is valid. Now, suppose that $k > m$, let $i \in \{m, \ldots, k-1\}$, and assume that (5.18) is valid for k replaced with i. We define

$$\begin{aligned}&\lambda^{i+1,m} = \lambda_1^i + \lambda_3^i \lambda^{i,m}, \\ &\lambda^{i+1,j} = \lambda_3^i \lambda^{i,j} \quad \text{for } j = m+1, \ldots, i, \qquad \text{and} \\ &\lambda^{i+1,i+1} = \lambda_2^i,\end{aligned}$$

where $\lambda_l^i \geq 0$ for all $l \in \{1, 2, 3\}$ are obtained at Step 6 of Algorithm 5.1. Now, we have $\lambda^{i+1,j} \geq 0$ for all $j = m, \ldots, i+1$, and

$$\sum_{j=m}^{i+1} \lambda^{i+1,j} = \lambda_1^i + \lambda_3^i \left(\lambda^{i,m} + \sum_{j=m+1}^{i} \lambda^{i,j} \right) + \lambda_2^i = 1,$$

since $\sum_{j=m}^{i} \lambda^{i,j} = 1$ due to the assumption and $\sum_{l=1}^{3} \lambda_l^i = 1$ (see Step 6 of Algorithm 5.1). Using relations in (5.8), the result given above and the fact that $\beta_m = 0$, we obtain

$$\begin{aligned}(\tilde{\boldsymbol{\xi}}_{i+1}, \tilde{\beta}_{i+1}) &= \lambda_1^i(\boldsymbol{\xi}_m, 0) + \lambda_2^i(\boldsymbol{\xi}_{i+1}, \beta_{i+1}) + \sum_{j=m}^{i} \lambda_3^i \lambda^{i,j}(\boldsymbol{\xi}_j, \beta_j) \\ &= \sum_{j=m}^{i+1} \lambda^{i+1,j}(\boldsymbol{\xi}_j, \beta_j),\end{aligned}$$

and, thus, the condition (5.18) is valid for $i + 1$. We define

$$\lambda^{k,j} = 0 \quad \text{for } j = 1, \ldots, m-1, \qquad \text{and} \qquad \tilde{\sigma}_k = \sum_{j=1}^{k} \lambda^{k,j} \|\boldsymbol{y}_j - \boldsymbol{x}_k\|.$$

Since $\boldsymbol{x}_j = \boldsymbol{x}_k$ for $j = m, \ldots, k$, we obtain

$$\tilde{\sigma}_k = \sum_{j=m}^{k} \lambda^{k,j} \|\boldsymbol{y}_j - \boldsymbol{x}_j\|,$$

and, thus, by (5.18), Lemma 5.2, and the convexity of the function $g \to \gamma g^2$ on $\mathbb{R}_+$ for $\gamma \geq 0$, we have

$$\gamma \tilde{\sigma}_k^2 = \gamma \left(\sum_{j=m}^{k} \lambda^{k,j} \|\boldsymbol{y}_j - \boldsymbol{x}_j\| \right)^2 \leq \sum_{j=m}^{k} \lambda^{k,j} \gamma \|\boldsymbol{y}_j - \boldsymbol{x}_j\|^2 \leq \sum_{j=m}^{k} \lambda^{k,j} \beta_j = \tilde{\beta}_k.$$

□

Lemma 5.5 *Let $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ be given and suppose that there exist vectors $\bar{\boldsymbol{g}}$, $\bar{\boldsymbol{\xi}}_i$, $\bar{\boldsymbol{y}}_i$, and numbers $\bar{\lambda}_i \geq 0$ for $i = 1, \ldots, l$, $l \geq 1$, such that*

$$\begin{aligned}&(\bar{\boldsymbol{g}}, 0) = \sum_{i=1}^{l} \bar{\lambda}_i (\bar{\boldsymbol{\xi}}_i, \|\bar{\boldsymbol{y}}_i - \bar{\boldsymbol{x}}\|), \\ &\bar{\boldsymbol{\xi}}_i \in \partial f(\bar{\boldsymbol{y}}_i), \quad i = 1, \ldots, l, \qquad \textit{and} \\ &\sum_{i=1}^{l} \bar{\lambda}_i = 1.\end{aligned} \tag{5.19}$$

Then $\bar{\boldsymbol{g}} \in \partial f(\bar{\boldsymbol{x}})$.

Proof ***([35])*** Let $\mathcal{I} = \{ i \mid 1 \leq i \leq l,\ \bar{\lambda}_i > 0 \}$. By (5.19) we have $\bar{\boldsymbol{y}}_i = \bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{\xi}}_i \in \partial f(\bar{\boldsymbol{x}})$ for all $i \in \mathcal{I}$. Therefore,

$$\begin{aligned} &\bar{\boldsymbol{g}} = \sum_{i \in \mathcal{I}} \bar{\lambda}_i \bar{\boldsymbol{\xi}}_i, \\ &\bar{\lambda}_i > 0, \qquad \text{for } i \in \mathcal{I}, \qquad \text{and} \\ &\sum_{i \in \mathcal{I}} \bar{\lambda}_i = 1, \end{aligned}$$

and $\bar{\boldsymbol{g}} \in \partial f(\bar{\boldsymbol{x}})$ by the convexity of $\partial f(\bar{\boldsymbol{x}})$. □

Theorem 5.1 *If Algorithm 5.1 terminates at the k-th iteration, then the point $\boldsymbol{x}_k$ is stationary for f.*

Proof If Algorithm 5.1 terminates at Step 4, then the fact $\varepsilon = 0$ implies that $w_k = 0$. Thus, $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{0}$ and $\tilde{\beta}_k = \tilde{\sigma}_k = 0$ by Lemma 5.2 and Lemma 5.4. Now, by Lemma 5.4 and using Lemma 5.5 with

$$\begin{aligned} &\bar{\boldsymbol{x}} = \boldsymbol{x}_k, \qquad l = k, \qquad \bar{\boldsymbol{g}} = \tilde{\boldsymbol{\xi}}_k, \\ &\bar{\boldsymbol{\xi}}_i = \boldsymbol{\xi}_i, \qquad \bar{\boldsymbol{y}}_i = \boldsymbol{y}_i, \quad \bar{\lambda}_i = \lambda^{k,i} \quad \text{for } i \leq k, \end{aligned}$$

we obtain $\boldsymbol{0} = \tilde{\boldsymbol{\xi}}_k \in \partial f(\boldsymbol{x}_k)$ and, thus, $\boldsymbol{x}_k$ is stationary for f. □

From now on, we suppose that Algorithm 5.1 does not terminate, that is, $w_k > 0$ for all k.

Lemma 5.6 *If there exist a point $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ and an infinite set $\mathcal{K} \subset \{1, 2, \ldots\}$ such that $\{\boldsymbol{x}_k\}_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$ and $\{w_k\}_{k \in \mathcal{K}} \to 0$, then $\boldsymbol{0} \in \partial f(\bar{\boldsymbol{x}})$.*

Proof ***([35].)*** Let $\mathcal{I} = \{1, \ldots, n+2\}$. Using the fact that $\boldsymbol{\xi}_k \in \partial f(\boldsymbol{y}_k)$ for all $k \geq 1$, Lemma 5.4, and Carathéodory's theorem, we deduce that there exist vectors $\boldsymbol{y}^{k,i}$, $\boldsymbol{\xi}^{k,i}$, and numbers $\lambda^{k,i} \geq 0$ and $\tilde{\sigma}_k$ for $i \in \mathcal{I}$ and $k \geq 1$, such that

$$\begin{aligned} &(\tilde{\boldsymbol{\xi}}_k, \tilde{\sigma}_k) = \sum_{i \in \mathcal{I}} \lambda^{k,i} (\boldsymbol{\xi}^{k,i}, \|\boldsymbol{y}^{k,i} - \boldsymbol{x}_k\|), \\ &\boldsymbol{\xi}^{k,i} \in \partial f(\boldsymbol{y}^{k,i}), \qquad \text{and} \\ &\sum_{i \in \mathcal{I}} \lambda^{k,i} = 1, \end{aligned} \tag{5.20}$$

with $(\boldsymbol{y}^{k,i}, \boldsymbol{\xi}^{k,i}) \in \{(\boldsymbol{y}_j, \boldsymbol{\xi}_j) \mid j = 1, \ldots, k\}$. From the boundedness of $\{\boldsymbol{y}_k\}$ (see Remark 5.3), we obtain the existence of points $\boldsymbol{y}_i^*$ $(i \in \mathcal{I})$, and an infinite set $\mathcal{K}_0 \subset \mathcal{K}$ satisfying $\{\boldsymbol{y}^{k,i}\}_{k \in \mathcal{K}_0} \to \boldsymbol{y}_i^*$ for $i \in \mathcal{I}$. The boundedness of $\{\boldsymbol{\xi}_k\}$ and $\{\lambda^{k,i}\}$ gives

us the existence of vectors $\boldsymbol{\xi}_i^* \in \partial f(\boldsymbol{y}_i^*)$, numbers λ_i^* for $i \in \mathcal{I}$, and an infinite set $\mathcal{K}_1 \subset \mathcal{K}_0$ satisfying $\{\boldsymbol{\xi}^{k,i}\}_{k\in\mathcal{K}_1} \to \boldsymbol{\xi}_i^*$ and $\{\lambda^{k,i}\}_{k\in\mathcal{K}_1} \to \lambda_i^*$ for $i \in \mathcal{I}$.

It can be seen from (5.20) that

$$\lambda_i^* \geq 0 \quad \text{for } i \in \mathcal{I}, \qquad \text{and} \qquad \sum_{i\in\mathcal{I}} \lambda_i^* = 1.$$

Since $\{w_k\}_{k\in\mathcal{K}} \to 0$, Lemmas 5.2 and 5.4 imply that

$$\{\tilde{\boldsymbol{\xi}}_k\}_{k\in\mathcal{K}} \to \mathbf{0}, \qquad \{\tilde{\beta}_k\}_{k\in\mathcal{K}} \to 0, \qquad \text{and} \qquad \{\tilde{\sigma}_k\}_{k\in\mathcal{K}} \to 0.$$

By letting $k \in \mathcal{K}_1$ approach infinity in (5.20) and using Lemma 5.5 with

$$\bar{\boldsymbol{\xi}}_i = \boldsymbol{\xi}_i^*, \qquad \bar{\boldsymbol{y}}_i = \boldsymbol{y}_i^*, \qquad \bar{\boldsymbol{g}} = \mathbf{0},$$
$$l = n+2, \qquad \text{and} \qquad \bar{\lambda}_i = \lambda_i^*,$$

we obtain $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$. □

Lemma 5.7 *Suppose that the number of serious steps is finite, and the last serious step occurs at the iteration $m-1$ of Algorithm 5.1. Then there exists a number $k^* \geq m$, such that*

$$\langle \tilde{\boldsymbol{\xi}}_{k+1}, D_{k+1}\tilde{\boldsymbol{\xi}}_{k+1}\rangle \leq \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_k\tilde{\boldsymbol{\xi}}_{k+1}\rangle \qquad \textit{and} \tag{5.21}$$

$$\operatorname{tr}(D_k) < \frac{3}{2}n \tag{5.22}$$

for all $k \geq k^*$, where $\operatorname{tr}(D_k)$ denotes the trace of the matrix D_k.

Proof First, we point out that since we are considering null steps the matrix D_{k+1} is computed using L-SR1 formula. Suppose that $i_{CN} = 0$ for all $k \geq m$, that is, the correction ϱI (see Algorithm 5.1, Step 3) is not added to any matrix D_k with $k \geq m$. If we skip the update (e.g., if the positive definiteness would be violated otherwise, see Remark 5.2), we have $D_{k+1} = D_k$, and the condition (5.21) is trivially satisfied with equalities. Otherwise, if $\hat{m}_k < \hat{m}_c$, the L-SR1 update is equal to the standard SR1 update and we have

$$D_{k+1} = D_k - \frac{(D_k\boldsymbol{u}_k - \boldsymbol{s}_k)(D_k\boldsymbol{u}_k - \boldsymbol{s}_k)^T}{\langle \boldsymbol{u}_k, D_k\boldsymbol{u}_k - \boldsymbol{s}_k\rangle},$$

where the denominator is greater than zero by Lemma 5.3 and the numerator is positive (semi)definite matrix. Thus, condition (5.21) is again valid. Finally, if $\hat{m}_k = \hat{m}_c$, we update the matrix only if $\tilde{\boldsymbol{\xi}}_{k+1}{}^T(D_{k+1} - D_k)\tilde{\boldsymbol{\xi}}_{k+1} \leq 0$ (see Remark 5.2). Since $i_{CN} = 0$ for all $k \geq m$, the correction ϱI is not added to the new matrix D_{k+1} and, thus, the condition (5.21) will be valid. Furthermore, in each case we have

$$\operatorname{tr}(D_k) - \frac{3}{2}n = \operatorname{tr}(D_k) - \operatorname{tr}(I) - \frac{1}{2}n = \operatorname{tr}(D_k - I) - \frac{1}{2}n < 0$$

for $k \geq m$, since the matrix $D_k - I$ is negative (semi)definite due to condition (5.13). Therefore, if $i_{CN} = 0$ for all $k \geq m$, conditions (5.21) and (5.22) are valid if we set $k^* = m$.

If $i_{CN} = 0$ does not hold for all $k \geq m$, then the correction ϱI, with $\varrho \in (0, 1/2)$, is added to all matrices D_k with $k \geq \bar{k}$ (see Algorithm 5.1, Step 3). Here $\bar{k}$ denotes the index $k \geq m$ of the iteration when $i_{CN} = 1$ occurred for the first time. Let us denote by $\hat{D}_k$ the matrix formed with the L-SR1 update and by D_k the corrected matrix, that is, $D_k = \hat{D}_k + \varrho I$ for all $k \geq \bar{k}$ (since we suppose $i_{CN} = 1$). Then, all the results given above are valid for $\hat{D}_k$ and $\hat{D}_{k+1}$. Since for all $k \geq \bar{k}$, we have $D_k = \hat{D}_k + \varrho I$ and we set $D_{k+1} = \hat{D}_{k+1} + \varrho I$, the condition (5.21) is valid for all $k \geq k^* = \bar{k}$. In addition, in each case we have

$$\begin{aligned} \mathrm{tr}(D_k) - \frac{3}{2}n &= \mathrm{tr}(\hat{D}_k + \varrho I) - \mathrm{tr}(I) - \frac{1}{2}n \\ &= \mathrm{tr}(\hat{D}_k - I) + \mathrm{tr}(\varrho I) - \frac{1}{2}n \\ &< \mathrm{tr}(\hat{D}_k - I) + \frac{1}{2}n - \frac{1}{2}n \leq 0 \end{aligned}$$

for $k \geq \bar{k}$, since the matrix $\hat{D}_k - I$ is negative (semi)definite and $\varrho \in (0, 1/2)$. Therefore, conditions (5.21) and (5.22) are valid for all $k \geq k^*$ with

$$k^* = \max\{\bar{k}, m\},$$

where $\bar{k} = 1$ if $i_{CN} = 0$. □

Lemma 5.8 *Suppose that the number of serious steps is finite, and the last serious step occurs at the iteration $m - 1$. Then, the point $\boldsymbol{x}_m$ is stationary for f.*

Proof From (5.7), (5.8), Lemmas 5.2 and 5.7 we obtain

$$\begin{aligned} w_{k+1} &= \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_{k+1}\tilde{\boldsymbol{\xi}}_{k+1} \rangle + 2\tilde{\beta}_{k+1} \\ &\leq \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_k\tilde{\boldsymbol{\xi}}_{k+1} \rangle + 2\tilde{\beta}_{k+1} \\ &= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) \\ &\leq \varphi(0, 0, 1) \\ &= \langle \tilde{\boldsymbol{\xi}}_k, D_k\tilde{\boldsymbol{\xi}}_k \rangle + 2\tilde{\beta}_k \\ &= w_k \end{aligned} \tag{5.23}$$

for $k \geq k^*$ with k^* defined in Lemma 5.7.

Let us denote by $D_k = {W_k}^T W_k$. Then, the function φ (see (5.7)) can be given in the form

$$\varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) = \|\lambda_1^k W_k \boldsymbol{\xi}_m + \lambda_2^k W_k \boldsymbol{\xi}_{k+1} + \lambda_3^k W_k \tilde{\boldsymbol{\xi}}_k\|^2 + 2(\lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k).$$

From (5.23) we obtain the boundedness of the sequences $\{w_k\}$, $\{W_k \tilde{\boldsymbol{\xi}}_k\}$, and $\{\tilde{\beta}_k\}$. Furthermore, Lemma 5.7 assures the boundedness of $\{D_k\}$ and $\{W_k\}$ while Remark 5.3, shows us the boundedness of $\{\boldsymbol{y}_k\}$, $\{\boldsymbol{\xi}_k\}$, and $\{W_k \boldsymbol{\xi}_{k+1}\}$.

Now, by noticing that we use the scaled direction vector $\theta_k \boldsymbol{d}_k$ (with $\theta_k = \min\{1, C/\|\boldsymbol{d}_k\|\}$ and the predefined $C > 0$) and the scaled line search parameter $\varepsilon_R^k = \theta_k \varepsilon_R^l$ in the line search (see Algorithm 5.1 Step 5 and Algorithm 5.4) the last part of the proof proceeds similar to the proof (part (ii)) of [35, Lemma 3.6]. □

Theorem 5.2 *Every accumulation point $\bar{\boldsymbol{x}}$ generated by Algorithm 5.1 is stationary for f.*

Proof Let $\bar{\boldsymbol{x}}$ be an accumulation point of $\{\boldsymbol{x}_k\}$, and let $\mathcal{K} \subset \{1, 2, \ldots\}$ be an infinite set such that $\{\boldsymbol{x}_k\}_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$. In view of Lemma 5.8, we can restrict our consideration to the case where the number of serious steps (with $t_L^k > 0$) is infinite. We denote

$$\mathcal{K}' = \{k \mid t_L^k > 0, \text{ there exists } i \in \mathcal{K},\ i \leq k \text{ such that } \boldsymbol{x}_i = \boldsymbol{x}_k\}.$$

Obviously, $\mathcal{K}'$ is infinite and $\{\boldsymbol{x}_k\}_{k \in \mathcal{K}'} \to \bar{\boldsymbol{x}}$. The continuity of f implies that $\{f_k\}_{k \in \mathcal{K}'} \to f(\bar{\boldsymbol{x}})$ and, thus, $f_k \downarrow f(\bar{\boldsymbol{x}})$ by the monotonicity of the sequence $\{f_k\}$ obtained due to the serious descent criterion (5.43) (see Appendix). Using the fact that $t_L^k \geq 0$ for all $k \geq 1$ and the condition (5.43), we obtain

$$0 \leq \varepsilon_L^k t_L^k w_k \leq f_k - f_{k+1} \to 0 \qquad \text{for } k \geq 1. \tag{5.24}$$

If the set $\mathcal{K}_1 = \{k \in \mathcal{K}' \mid t_L^k \geq t_{min}\}$ is infinite, then $\{w_k\}_{k \in \mathcal{K}_1} \to 0$ and $\{\boldsymbol{x}_k\}_{k \in \mathcal{K}_1} \to \bar{\boldsymbol{x}}$ by (5.24) and, thus, by Lemma 5.6 we have $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$.

If the set $\mathcal{K}_1$ is finite, then the set $\mathcal{K}_2 = \{k \in \mathcal{K}' \mid \beta_{k+1} > \varepsilon_A^k w_k\}$ has to be infinite (see Appendix, Algorithm 5.4, Step 2). To the contrary, let us assume that

$$w_k \geq \delta > 0 \qquad \text{for all } k \in \mathcal{K}_2.$$

From (5.24), we have $\{t_L^k\}_{k \in \mathcal{K}_2} \to 0$ and Step 5 in Algorithm 5.1 implies

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| = t_L^k \theta_k \|\boldsymbol{d}_k\| \leq t_L^k C$$

for all $k \geq 1$. Thus, we have $\{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|\}_{k \in \mathcal{K}_2} \to 0$. By (5.6), (5.24), and the boundedness of $\{\boldsymbol{\xi}_k\}$, and since $\boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1}$ for serious steps, we obtain $\{\beta_{k+1}\}_{k \in \mathcal{K}_2} \to 0$, which is in contradiction with

$$\varepsilon_A^k \delta \leq \varepsilon_A^k w_k < \beta_{k+1} \qquad k \in \mathcal{K}_2.$$

Therefore, there exists an infinite set $\mathcal{K}_3 \subset \mathcal{K}_2$ such that $\{w_k\}_{k\in\mathcal{K}_3} \to 0$, $\{\boldsymbol{x}_k\}_{k\in\mathcal{K}_3} \to \bar{\boldsymbol{x}}$, and $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$ by Lemma 5.6. □

Remark 5.4 The LMBM terminates in a finite number of steps if we choose $\varepsilon > 0$.

5.3 Diagonal Bundle Method

The classical variable metric techniques for nonlinear optimization construct a dense $n \times n$ -matrix to approximate the Hessian of the function. Then these techniques are required to store and manipulate this dense matrix, which becomes unmanageable in large-scale settings. In the limited memory variable metric methods the storage of this large matrix can be avoided, but still the formed approximation of the Hessian is dense. This is also true for the LMBM described in the previous section. Nevertheless, in many large-scale problems the real Hessian (if it exists) is sparse. In this section, we describe the diagonal bundle method (D-BUNDLE) that combines the LMBM with sparse matrix updating. We first describe in more details the different components of the method and then introduce the entire algorithm. As with the LMBM we assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC and we can compute $f(\boldsymbol{x})$ and $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ at every $\boldsymbol{x} \in \mathbb{R}^n$.

Matrix Updating and Direction Finding The D-BUNDLE applies the diagonal update formula to compute the *diagonal variable metric updates*. Similarly to the LMBM, the D-BUNDLE uses no more than $\hat{m}_c$ most recent correction vectors to compute the updates for the matrices. These vectors and matrices are defined as in (5.9). The diagonal approximation of the Hessian B_{k+1} is then defined by solving a problem

$$\begin{cases} \text{minimize} & \|B_{k+1}S_k - U_k\|_F^2 \\ \text{subject to} & (B_{k+1})_{i,j} = 0 \text{ for } i \neq j, \\ & (B_{k+1})_{i,i} \geq \mu \text{ for } i = 1, 2, \ldots, n \end{cases} \tag{5.25}$$

for some $\mu > 0$. Note that the check of positive definiteness is included as a constraint into the problem. The problem (5.25) has a solution

$$(B_{k+1})_{i,i} = \begin{cases} b_i/Q_{i,i}, & \text{if } b_i/Q_{i,i} > \mu, \\ \mu, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{b} = 2\sum_{i=k-\hat{m}_k}^{k-1} \text{diag}(\boldsymbol{s}_i)\boldsymbol{u}_i$ and $Q = 2\sum_{i=k-\hat{m}_k}^{k-1} [\text{diag}(\boldsymbol{s}_i)]^2$.

In the D-BUNDLE we use directly the inverse of this matrix, that is, $D_k = (B_k)^{-1}$, where the diagonal components of D_k are given by

$$(D_{k+1})_{i,i} = \begin{cases} \mu_{\texttt{min}}, & \text{if } Q_{i,i}/b_i < \mu_{\texttt{min}}, \\ Q_{i,i}/b_i, & \text{if } Q_{i,i}/b_i > \mu_{\texttt{min}} \text{ and } Q_{i,i}/b_i < \mu_{\texttt{max}}, \\ \mu_{\texttt{max}}, & \text{otherwise.} \end{cases} \tag{5.26}$$

Note that in addition to the upper bound $\mu_{\texttt{max}} = \frac{1}{\mu}$, we also use the lower bound $\mu_{\texttt{min}}$ ($0 < \mu_{\texttt{min}} < \mu_{\texttt{max}}$) for the components of the matrix. These bounds trivially guarantee the boundedness of matrices D_k ($k = 1, 2, \ldots$) needed in the convergence proof. The search direction is computed by the formula

$$\boldsymbol{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k, \tag{5.27}$$

where $\tilde{\boldsymbol{\xi}}_k$ is (an aggregate) subgradient. To ensure the global convergence of the D-BUNDLE, the condition (cf. (5.12))

$$\langle \tilde{\boldsymbol{\xi}}_k, D_k \tilde{\boldsymbol{\xi}}_k \rangle \leq \langle \tilde{\boldsymbol{\xi}}_k, D_{k-1} \tilde{\boldsymbol{\xi}}_k \rangle \tag{5.28}$$

has to be satisfied each time there occurs more than one consecutive null step. In the D-BUNDLE this is guaranteed simply by *skipping the updates* in consecutive null steps. That is, after a null step we set $D_{k+1} = D_k$, but the new aggregate values are computed.

Aggregation, Line Search and Stopping Criterion In order to guarantee the convergence of the method and to avoid the unbounded storage, the D-BUNDLE uses the aggregation procedure similar to the LMBM. That is, the convex combination of at most three subgradients is used to form a new aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$ and a new aggregate subgradient locality measure $\tilde{\beta}_{k+1}$ (cf. (5.7) and (5.8)). Naturally, the diagonal update matrix D_k is used in Eq. (5.7) instead of the limited memory update.

In addition, the D-BUNDLE uses the same line search procedure than the LMBM to determine new iteration and auxiliary points $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$. That is, the step sizes $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$ with $t_{max} > 1$ are computed such that either condition (5.4) for serious steps or condition (5.5) for null steps is satisfied.

Finally, the stopping criterion of the D-BUNDLE algorithm is similar to that of the LMBM (cf. (5.10)) and similarly to this method, the parameter w_k is used also during the line search procedure to represent the desirable amount of descent (cf. (5.4) and (5.5)).

Algorithm The algorithm for the D-BUNDLE proceeds as follows.

Algorithm 5.2: D-BUNDLE

Data: The final accuracy tolerance $\varepsilon > 0$, initial line search parameters $\varepsilon_L^I \in (0, 1/2)$ and $\varepsilon_R^I \in (\varepsilon_L^I, 1/2)$, lower and upper bounds $t_{min} \in (0, 1)$ and $t_{max} > 1$ for serious steps, the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if f is convex), a safeguard parameter $\mu_{\texttt{max}} > \mu_{\texttt{min}} > 0$, a control parameter $C > 0$ for the length of the direction vector, and the number of stored corrections $\hat{m}_c \geq 1$.

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Set $D_1 = I$, $\beta_1 = 0$ and $\boldsymbol{y}_1 = \boldsymbol{x}_1$. Compute $f_1 = f(\boldsymbol{x}_1)$ and $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$. Set $k = 1$ and $\hat{m}_k = 0$.

Step 1. (*Serious Step Initialization*) Set $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k$, $\tilde{\beta}_k = 0$, and $m = k$.

Step 2. (*Direction finding*) Compute $\boldsymbol{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k$.

Step 4. (*Stopping criterion*) Compute w_k by (5.10). If $w_k < \varepsilon$, then **stop** with $\boldsymbol{x}_k$ as the final solution.

Step 5. (*Line search*) Set the scaling parameter for the length of the direction vector and for the line search

$$\theta_k = \min\{\, 1, C/\|\boldsymbol{d}_k\| \,\}.$$

Determine the step sizes $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$. Set the corresponding values

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \theta_k \boldsymbol{d}_k, \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \theta_k \boldsymbol{d}_k, \\ f_{k+1} &= f(\boldsymbol{x}_{k+1}), \qquad \text{and} \\ \boldsymbol{\xi}_{k+1} &\in \partial f(\boldsymbol{y}_{k+1}). \end{aligned}$$

Set $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k = t_R^k \theta_k \boldsymbol{d}_k$ and append these values to U_k and S_k, respectively. Set $\hat{m}_k = \min\{k, \hat{m}_c\}$. If the condition (5.4) is valid (i.e., we take a serious step), then compute a new diagonal matrix D_{k+1} by (5.26), set $\beta_{k+1} = 0$, $k = k + 1$, and go to Step 1. Otherwise, calculate the locality measure β_{k+1} by (5.6).

Step 6. (*Aggregation*) Determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function (5.7) with the diagonal D_k. Compute $\tilde{\boldsymbol{\xi}}_{k+1}$ and $\tilde{\beta}_{k+1}$ as in (5.8). Set $D_{k+1} = D_k$, $k = k + 1$ and go to Step 2.

Global Convergence of D-BUNDLE We now recall the convergence properties of the D-BUNDLE. The assumptions needed are the same as those with the LMBM, that is, Assumptions 5.1–5.3. Under these assumptions, Lemma 5.1 is valid and Remark 5.3 holds for sequences generated by Algorithm 5.2. In addition, we want to remark that all the matrices generated by D-BUNDLE are bounded.

Remark 5.5 The diagonal matrix D_k is bounded for all $k \geq 1$ due to the fact that all its components are in the closed interval $[\mu_{\min}, \mu_{\max}]$ (see (5.26)).

The convergence analysis of the D-BUNDLE is very similar to that of the LMBM. In fact, all the results in the previous section are valid also for the D-BUNDLE except that, for the D-BUNDLE, we do not have—nor we need—the property $\langle \boldsymbol{u}_k, (D_k \boldsymbol{u}_k - \boldsymbol{s}_k) \rangle > 0$ given in Lemma 5.3. With the LMBM this property is used to guarantee that the condition (5.12) is valid in the case of consecutive null steps. However, with the D-BUNDLE the similar condition (5.28) is valid due to skipping of updates and, thus, the above mentioned property is not required. In addition, due to the safeguarded diagonal matrix updating we do not need the correction ϱI (see Algorithm 5.1, Step 3) and, thus, Lemma 5.2 takes the following form.

Lemma 5.9 *At the k-th iteration of Algorithm 5.2, we have*

$$w_k = \langle \tilde{\boldsymbol{\xi}}_k, D_k \tilde{\boldsymbol{\xi}}_k \rangle + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \mu_{min} \|\tilde{\boldsymbol{\xi}}_k\|^2,$$

and

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2.$$

Proof The proof is similar to the proof of Lemma 5.2 but the relation $w_k \geq \mu_{\min} \|\tilde{\boldsymbol{\xi}}_k\|^2$ that follows immediately from the lower bound $\mu_{\min}$ used for the matrices (see (5.26)). □

To prove the convergence of the D-BUNDLE, we assume $\varepsilon = 0$.

Theorem 5.3 *If Algorithm 5.2 terminates at the k-th iteration, then the point $\boldsymbol{x}_k$ is stationary for f.*

Proof The proof is similar to the proof of Theorem 5.1 if we replace Lemma 5.2 with Lemma 5.9. □

From now on, we suppose that Algorithm 5.2 does not terminate, that is, $w_k > 0$ for all k. Next we give a result that sharpens some parts of Lemmata 5.7 and 5.8.

Lemma 5.10 *Suppose that the number of serious steps is finite, and the last serious step occurs at the iteration $m - 1$. Then*

$$\langle \tilde{\boldsymbol{\xi}}_{k+1}, D_{k+1} \tilde{\boldsymbol{\xi}}_{k+1} \rangle = \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_k \tilde{\boldsymbol{\xi}}_{k+1} \rangle \qquad \textit{and} \tag{5.29}$$

$$\operatorname{tr}(D_k) \leq \mu_{\max} n \tag{5.30}$$

for all $k > m$. In addition, we have $w_{k+1} \leq w_k$.

Proof For all $k > m$ we have $D_{k+1} = D_k$ due to skipping of updates in consecutive null steps (see Algorithm 5.2 Step 6). Thus, the condition (5.29) is valid. Furthermore, we have

$$\begin{aligned}\mathrm{tr}(D_k) - \mu_{\max} n &= \mathrm{tr}(D_k) - \mu_{\max}\,\mathrm{tr}(I)\\ &= \mathrm{tr}(D_k) - \mathrm{tr}(\mu_{\max} I)\\ &= \mathrm{tr}(D_k - \mu_{\max} I) \leq 0\end{aligned}$$

for all k, since D_k is a diagonal matrix with the largest diagonal element less than or equal to $\mu_{\max}$. Therefore, the condition (5.30) is valid for all $k > m$.

Combining (5.29) with (5.7), (5.8), (5.10), and (5.27), we obtain

$$\begin{aligned}w_{k+1} &= \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_{k+1}\tilde{\boldsymbol{\xi}}_{k+1}\rangle + 2\tilde{\beta}_{k+1}\\ &= \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_k\tilde{\boldsymbol{\xi}}_{k+1}\rangle + 2\tilde{\beta}_{k+1}\\ &= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k)\\ &\leq \varphi(0, 0, 1)\\ &= \langle \tilde{\boldsymbol{\xi}}_k, D_k\tilde{\boldsymbol{\xi}}_k\rangle + 2\tilde{\beta}_k = w_k\end{aligned}$$

for all $k > m$. □

The convergence result of the D-BUNDLE is given in the next theorem.

Theorem 5.4 *Every accumulation point of an infinite sequence of solutions generated by Algorithm 5.2 is a stationary point of f.*

Proof The D-BUNDLE algorithm is essentially similar to the LMBM with the diagonal matrix updating instead of the limited memory matrix updating. If Algorithm 5.2 generates an infinite sequence of solutions, then we can replace Lemma 5.7 and the first part of the proof of Lemma 5.8 by Lemma 5.10 and all the remaining results of the previous section are valid also for the D-BUNDLE. □

To conclude, similarly to the LMBM, the D-BUNDLE either terminates at a stationary point of the objective function f or generates an infinite sequence $\{\boldsymbol{x}_k\}$ for which accumulation points are stationary for f. Moreover, if we choose $\varepsilon > 0$, the D-BUNDLE terminates in a finite number of steps.

5.4 Splitting Metrics Diagonal Bundle Method

NSO is traditionally based on convex analysis and most solution methods rely strongly on the convexity of the problem. After all, the convex model of the objective function is usually reasonably good for nonconvex problems as well, except in

some areas where there exists the so-called concave behaviour in the objective. In these cases the linearization error, used as a measure of the goodness of the current piecewise linear model, has negative values and the model is no longer an underestimate of the objective. The common way to deal with this difficulty is to do some downward shifting (e.g. to use the subgradient locality measures (5.6) instead of linearization errors), but the amount of this shifting may be more or less arbitrary. This is also the case in the LMBM and D-BUNDLE described in the previous sections. In this section we represent the splitting metrics diagonal bundle algorithm SMDB that combines the ideas of the D-BUNDLE to different usage of metrics depending on the convex or concave behaviour of the objective at the current iteration point. The usage of different metrics gives us a possibility to better deal with the nonconvexity of the problem than the sole downward shifting of the piecewise linear model does.

First we describe in more details the different components of the method and then introduce the entire algorithm. As before, we assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC and at every point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate $f(\boldsymbol{x})$ and $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$.

Linearization Error The SMDB uses the sign of the linearization error to detect the "convex" or "concave" behaviour of the objective. The *linearization error* α_{k+1} associated with the point $\boldsymbol{y}_{k+1}$ is given by

$$\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + \langle \boldsymbol{\xi}_{k+1}, \boldsymbol{d}_k \rangle,$$

where $\boldsymbol{x}_k$ is the current iteration point, $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$ is a new auxiliary point, $\boldsymbol{d}_k$ is the current search direction and $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$. Note that differently from previous sections no line search is used when generating $\boldsymbol{y}_{k+1}$.

Matrix Updating and Splitting of Data The diagonal update formula for updating the matrices is used in the SMDB, since for diagonal matrices it is easy to check and guarantee the positive (negative) definiteness. Moreover, using diagonal matrices requires minimum amount of the storage space and computations.

The correction vectors used in the SMDB are quite similar to those used in previous sections. However, instead of just one couple of correction matrices S_k and U_k used in the LMBM and D-BUNDLE (see (5.9)), we now use different corrections matrices depending on the sign of the linearization error. That is, we append $\boldsymbol{s}_k$ and $\boldsymbol{u}_k$ to S_k^+ and U_k^+ if $\alpha_k \geq 0$ and to S_k^- and U_k^-, otherwise. This means that the maximum number of correction vectors is $2m_c$ and in each matrix S_k^+ (U_k^+) and S_k^- (U_k^-) we have (at most) m_c correction vectors $\boldsymbol{s}_{\hat{i}}$ ($\boldsymbol{u}_{\hat{i}}$) with indices $\hat{i} \in \{1, 2, \ldots, k\}$. In addition, no $\hat{i}$ can be in both S_k^+ (U_k^+) and S_k^- (U_k^-). For simplicity, we denote these indices by $\hat{i} \in \{\hat{1}, \ldots, \hat{m}_c\}$ from now on. Note that $\hat{m}_c$ may be smaller than m_c and $\hat{m}_c$ may be different for S_k^+ (U_k^+) and S_k^- (U_k^-).

The approximation of the Hessian B_{k+1}^{+} (B_{k+1}^{-}) is a diagonal matrix and the check of the positive (negative) definiteness is included as a constraint into the problem. Thus, the update matrix B_{k+1}^{+} is defined by (cf. (5.25))

$$\begin{cases} \text{minimize} & \|B_{k+1}^{+}S_k^{+} - U_k^{+}\|_F^2 \\ \text{subject to} & (B_{k+1}^{+})_{i,j} = 0 \text{ for } i \neq j, \\ & (B_{k+1}^{+})_{i,i} \geq \mu, \ \ i = 1, 2, \ldots, n, \end{cases} \tag{5.31}$$

for some $\mu > 0$, with a solution

$$\left(B_{k+1}^{+}\right)_{i,i} = \begin{cases} b_i / Q_{i,i}, & \text{if } b_i / Q_{i,i} > \mu, \\ \mu, & \text{otherwise.} \end{cases}$$

Here $\boldsymbol{b} = 2\sum_{i=\hat{1}}^{\hat{m}_c} \text{diag}(\boldsymbol{s}_i)\boldsymbol{u}_i$ and $Q = 2\sum_{i=\hat{1}}^{\hat{m}_c}[\text{diag}(\boldsymbol{s}_i)]^2$ with $\boldsymbol{s}_i \in S_k^{+}$ and $\boldsymbol{u}_i \in U_k^{+}$. In our computations, we use the inverse of this matrix $D_k^{+} = \left(B_k^{+}\right)^{-1}$ and we call it the "*convex approximation*" of the Hessian. The diagonal components of D_k^{+} are given by

$$\left(D_{k+1}^{+}\right)_{i,i} = \begin{cases} \mu_{\texttt{min}}, & \text{if } Q_{i,i}/b_i < \mu_{\texttt{min}}, \\ Q_{i,i}/b_i, & \text{if } Q_{i,i}/b_i > \mu_{\texttt{min}} \text{ and } Q_{i,i}/b_i < \mu_{\texttt{max}}, \\ \mu_{\texttt{max}}, & \text{otherwise.} \end{cases}$$

Note that similarly to D-BUNDLE we use both the upper bound $\mu_{\texttt{max}} = \frac{1}{\mu}$, and the lower bound $\mu_{\texttt{min}}$ ($0 < \mu_{\texttt{min}} < \mu_{\texttt{max}}$) for the components of the matrix. The computation of the "*concave approximation*" D_k^{-} is analogous.

Direction Finding, Serious and Null Steps The SMDB uses the above mentioned diagonal approximations to compute the search direction. If the linearization error α_k is nonnegative or if the previous step was a serious step, we use the "convex approximation" . That is, we compute

$$\boldsymbol{d}_k = -D_k^{+}\tilde{\boldsymbol{\xi}}_k, \tag{5.32}$$

where, as before, $\tilde{\boldsymbol{\xi}}_k$ is an aggregate subgradient of the objective. Otherwise, we first compute the convex combination of the "convex and concave approximations" such that the combination still remains positive definite and then use this combination to compute the search direction. In other words, we compute the smallest $p_k \in [0, 1]$ such that $p_k D_k^{+} + (1 - p_k)D_k^{-}$ is positive definite. Note that, the computation of this value is very easy since both matrices D_k^{+} and D_k^{-} are diagonal. The search direction is then computed by the formula

$$\boldsymbol{d}_k = -(p_k D_k^{+} + (1 - p_k)D_k^{-})\tilde{\boldsymbol{\xi}}_k. \tag{5.33}$$

When the search direction is computed, we calculate a new auxiliary point: $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$. A necessary condition for a *serious step* to be taken is to have (cf. (5.4))

$$f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L w_k, \tag{5.34}$$

where $\varepsilon_L \in (0, 1/2)$ is a given descent parameter and $w_k > 0$ is a stopping parameter given by (cf. (5.10))

$$w_k = \langle \tilde{\boldsymbol{\xi}}_k, D_k^+ \tilde{\boldsymbol{\xi}}_k \rangle + 2\tilde{\beta}_k. \tag{5.35}$$

If (5.34) is satisfied, we take a serious step by setting $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$. In addition, we consider the current "convex approximation" to be good enough and continue with this metric even if the linearization error was negative.

If (5.34) is not satisfied, we first set $t = 1$ and check if $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$ satisfies the null step condition (cf. (5.5))

$$-\beta_{k+1}^t + \langle \boldsymbol{d}_k, \boldsymbol{\xi}_{k+1}^t \rangle \geq -\varepsilon_R w_k, \tag{5.36}$$

where $\varepsilon_R \in (\varepsilon_L, 1)$ is a given parameter and β_{k+1}^t is the subgradient locality measure similar to that in the previous sections (cf. (5.6)). That is,

$$\beta_{k+1}^t = \max \left\{ |f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\boldsymbol{d}_k) + t\langle \boldsymbol{\xi}_{k+1}^t, \boldsymbol{d}_k \rangle|, \gamma \|t\boldsymbol{d}_k\|^2 \right\}. \tag{5.37}$$

The condition (5.36) needs not to hold for $t = 1$. In that case we use the line search quite similar to the LMBM and D-BUNDLE (see Appendix for the line search algorithm). That is, we search for a step size $t \in (0, 1)$ such that either we have a descent condition

$$f(\boldsymbol{x}_k + t\boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \varepsilon_L t w_k \tag{5.38}$$

fulfilled or $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$ satisfies the null step condition (5.36). Whenever (5.36) holds we perform a null step by setting $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, but information about the objective function is increased because we utilize the auxiliary point $\boldsymbol{y}_{k+1}^t = \boldsymbol{x}_k + t\boldsymbol{d}_k$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{y}_{k+1}^t)$ in the computation of the next aggregate value. On the other hand, the fulfilment of condition (5.38) during the line search leads to a serious step with $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k$ as the new iteration point. For simplicity of the presentation we drop out the index t from $\boldsymbol{y}_k^t$, $\boldsymbol{\xi}_k^t$, and β_k^t even if $t \neq 1$, unless needed for the clarity.

Aggregation and Convergence Conditions The aggregation procedure used in the SMDB is similar to that of the LMBM with the limited memory update D_k replaced with the convex diagonal approximation D_k^+. More precisely, we determine multipliers $\lambda_i^k \geq 0$, $i = 1, 2, 3$, that minimize the function (5.7) with D_k^+, and we

set $\tilde{\boldsymbol{\xi}}_{k+1}$ and $\tilde{\beta}_{k+1}$ as in (5.8). In addition, the condition (cf. (5.12) and (5.28))

$$\langle \tilde{\boldsymbol{\xi}}_k, D_k^+ \tilde{\boldsymbol{\xi}}_k \rangle \leq \langle \tilde{\boldsymbol{\xi}}_k, D_{k-1}^+ \tilde{\boldsymbol{\xi}}_k \rangle \tag{5.39}$$

has to be satisfied each time there occurs more than one consecutive null step. In the SMDB this is guaranteed simply by *skipping the convex updates* if more than one consecutive null step occurs.

Algorithm The SMDB algorithm is as follows:

Algorithm 5.3: SMDB

Data: The final accuracy tolerance $\varepsilon > 0$, line search parameters $\varepsilon_L \in (0, 1/2)$ and $\varepsilon_R \in (\varepsilon_L, 1)$, the initial step size $t_I \in [0.5, 1)$, the distance measure parameter $\gamma \geq 0$, safeguard parameters $\mu_{\texttt{max}} > \mu_{\texttt{min}} > 0$, and the number of stored corrections $\hat{m}_c \geq 1$.

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Set $D_1^+ = I$, $\alpha_1 = 0$, $\beta_1 = 0$ and $\boldsymbol{y}_1 = \boldsymbol{x}_1$. Compute $f(\boldsymbol{x}_1)$ and $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$. Set $k = 1$.

Step 1. (*Serious step initialization*) Set $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k$, $\tilde{\beta}_k = 0$, and $m = k$.

Step 2. (*Convex direction*) Compute $\boldsymbol{d}_k = -D_k^+ \tilde{\boldsymbol{\xi}}_k$.

Step 3. (*Stopping criterion*) Compute w_k by (5.35). If $w_k < \varepsilon$, then **stop** with $\boldsymbol{x}_k$ as the final solution.

Step 4. (*Auxiliary step*) Evaluate

$$\begin{aligned} \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{d}_k, \\ \boldsymbol{\xi}_{k+1} &\in \partial f(\boldsymbol{y}_{k+1}), \text{ and} \\ \alpha_{k+1} &= f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + \langle \boldsymbol{\xi}_{k+1}, \boldsymbol{d}_k \rangle. \end{aligned}$$

Set $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_k = \boldsymbol{d}_k$. If $\alpha_{k+1} \geq 0$ append these values to U_k^+ and S_k^+, respectively. Otherwise, append them to S_k^- and U_k^- .

Step 5. (*Serious step without line search*) If (5.34) holds, compute D_{k+1}^+ using S_k^+ and U_k^+, set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$, $f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{y}_{k+1})$, $\alpha_{k+1} = 0$, $\beta_{k+1} = 0$, $k = k+1$, and go to Step 1.

Step 6. (*Null step test without line search*) Set $t = 1$ and compute β_{k+1} as in (5.37). If (5.36) holds, go to Step 8.

Step 7. (*Line search*) Find $t \in (0, t_I]$ for which either the null step condition (5.36) or the serious step condition (5.38) is valid. Set the corresponding values

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k, \quad \text{and} \quad \boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1}).$$

In the case of a serious step, compute D_{k+1}^+ using S_k^+ and U_k^+, set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$, $f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{y}_{k+1})$, $\alpha_{k+1} = 0$, $\beta_{k+1} = 0$, $k = k+1$, and go to Step 1. Otherwise, compute β_{k+1} as in (5.37).

Step 8. (*Aggregation*) Determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^{3} \lambda_i^k = 1$ that minimize the function (5.7) with the convex diagonal approximation D_k^+. Compute $\tilde{\boldsymbol{\xi}}_{k+1}$ and $\tilde{\beta}_{k+1}$ as in (5.8).

Step 9. (*Null step*) Three cases can occur:

a) (*First convex null step*) If $\alpha_{k+1} \geq 0$ and $m = k$, compute D_{k+1}^+ using S_k^+ and U_k^+. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k + 1$ and go to Step 2.

b) (*Consecutive convex null step*) If $\alpha_{k+1} \geq 0$ and $m < k$, set $D_{k+1}^+ = D_k^+$, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, and $k = k + 1$ and go to Step 2.

c) (*Concave null step*) If $\alpha_{k+1} < 0$, compute D_{k+1}^- using S_k^- and U_k^- and set $D_{k+1}^+ = D_k^+$. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k + 1$ and go to Step 10.

Step 10. (*Concave direction*) Compute the smallest $p \in (0, 1)$ such that the matrix $pD_k^+ + (1 - p)D_k^-$ remains positive definite. Compute

$$\boldsymbol{d}_k = -\left(pD_k^+ + (1 - p)D_k^-\right)\tilde{\boldsymbol{\xi}}_k$$

and go to Step 3.

Global Convergence of SMDB We now recall the convergence properties of the SMDB. The assumptions needed are the same as with the LMBM: that is, Assumptions 5.1–5.3. As before, we drop out the index t from $\boldsymbol{y}_k^t$, $\boldsymbol{\xi}_k^t$ and β_k^t even if $t \neq 1$.

Remark 5.6 The semismoothness assumption (Assumption 5.2) ensures

$$f'(\boldsymbol{x}, \boldsymbol{d}) = \lim_{t \downarrow 0} \langle \boldsymbol{\xi}(\boldsymbol{x} + t\boldsymbol{d}), \boldsymbol{d} \rangle$$

with $\boldsymbol{\xi}(\boldsymbol{x} + t\boldsymbol{d}) \in \partial f(\boldsymbol{x} + t\boldsymbol{d})$. This guarantees that Step 7 is well posed, that is for small values of t one of the two conditions (5.36) and (5.38) is satisfied.

Remark 5.7 The sequence $\{\boldsymbol{x}_k\}$ generated by Algorithm 5.3 is bounded by Assumption 5.3 and the monotonicity of the sequence $\{f_k\}$. The monotonicity of $\{f_k\}$ is guaranteed since either we have the condition (5.34) or the condition (5.38) satisfied for serious steps, while $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps.

By the local boundedness and the upper semi-continuity of the subdifferential, we obtain the boundedness of subgradients $\boldsymbol{\xi}_k$ and their convex combinations. The matrix D_k^+ (D_k^-) is bounded for all k since all its components are in the closed interval $[\mu_{\texttt{min}}, \mu_{\texttt{max}}]$ ($[-\mu_{\texttt{max}}, -\mu_{\texttt{min}}]$). Thus, the set of the search directions $\boldsymbol{d}_k$ and the sequence $\{\boldsymbol{y}_k\}$ are also bounded.

Lemma 5.11 *At the k-th iteration of Algorithm 5.3, we have*

$$w_k = \langle \tilde{\boldsymbol{\xi}}_k, D_k^+ \tilde{\boldsymbol{\xi}}_k \rangle + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \mu_{min} \|\tilde{\boldsymbol{\xi}}_k\|^2,$$

and

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2.$$

Proof The proof is similar to the proof of Lemma 5.9. □

In the following theorems and lemmata, we assume $\epsilon = 0$.

Theorem 5.5 *If Algorithm 5.3 terminates at the k-th iteration, then the point $\boldsymbol{x}_k$ is stationary for f.*

Proof The proof is similar to the proof of Theorem 5.1 if we replace Lemma 5.2 with Lemma 5.11. □

From now on, we suppose that Algorithm 5.3 does not terminate. Next we give a result that corresponds to Lemma 5.10.

Lemma 5.12 *Suppose that the number of serious steps is finite, and the last serious step occurs at the iteration $m-1$. Then*

$$\langle \tilde{\boldsymbol{\xi}}_{k+1}, D_{k+1}^{+} \tilde{\boldsymbol{\xi}}_{k+1} \rangle = \langle \tilde{\boldsymbol{\xi}}_{k+1}, D_{k}^{+} \tilde{\boldsymbol{\xi}}_{k+1} \rangle \qquad \textit{and} \qquad (5.40)$$
$$\mathrm{tr}(D_k^{+}) \leq \mu_{\max} n$$

for all $k > m$. In addition, we have $w_{k+1} \leq w_k$.

Proof For all $k > m$ we have $D_{k+1}^{+} = D_k^{+}$ due to the fact that we use either Step 9b or Step 9c of Algorithm 5.3 at null steps. Thus, the condition (5.40) is valid. The rest of the proof is similar to the proof of Lemma 5.10. □

Theorem 5.6 *Every accumulation point of an infinite sequence of solutions generated by Algorithm 5.3 is stationary for f.*

Proof Let $\bar{\boldsymbol{x}}$ be an accumulation point of $\{\boldsymbol{x}_k\}$, and let $\mathcal{K} \subset \{1, 2, \ldots\}$ be an infinite set such that $\{\boldsymbol{x}_k\}_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$. In view of the previous lemma combined with Lemma 5.8, we can restrict our consideration to the case where the number of serious steps is infinite. We denote

$$\mathcal{K}' = \{ k \mid \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k \text{ with } t > 0 \text{and}$$
$$\text{there exists } i \in \mathcal{K},\ i \leq k \text{ such that } \boldsymbol{x}_i = \boldsymbol{x}_k \}.$$

Obviously, $\mathcal{K}'$ is infinite and $\{\boldsymbol{x}_k\}_{k \in \mathcal{K}'} \to \bar{\boldsymbol{x}}$. The continuity of f implies that $\{f(\boldsymbol{x}_k)\}_{k \in \mathcal{K}'} \to f(\bar{\boldsymbol{x}})$, thus, $f(\boldsymbol{x}_k) \downarrow f(\bar{\boldsymbol{x}})$ by the monotonicity of the sequence $\{f(\boldsymbol{x}_k)\}$ obtained due to the descent step conditions (5.34) and (5.38). Using these conditions and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ in null steps, we obtain

$$0 \leq t\varepsilon_L w_k \leq f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \to 0 \qquad \text{for } k \geq 1. \qquad (5.41)$$

Thus, if the set $\mathcal{K}_1 = \{k \in \mathcal{K}' \mid t \geq t_{\texttt{min}}\}$ is infinite for some bound $t_{\texttt{min}} > 0$ then $(w_k)_{k\in\mathcal{K}'} \to 0$ and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$ by (5.41). Hence, by Lemma 5.6 we have $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$.

In the other case, where the set $\mathcal{K}_1$ is finite, the result is obtained the same way as in the proof of Theorem 5.2. □

Similarly to the LMBM and D-BUNDLE, Algorithm 5.3 terminates in a finite number of steps if we choose $\varepsilon > 0$.

5.5 Numerical Experiments and Discussion

To give an idea how the different methods described in this chapter work in practice we have tested them using large-scale nonsmooth minimization problems first introduced in [13]. These problems can be formulated with any number of variables. We have used here 1000, 10,000, 100,000 and one million variables. Problems numbered 1–5 are convex while problems 6–10 are nonconvex. Other numerical experiments comparing different NSO solvers including the LMBM, D-BUNDLE and SMDB can be found for instance in [4, 18, 21, 23].

Solvers and Parameters We now give a brief description of each software, the parameters used, and the references from which the code can be downloaded. The experiments were performed on an Intel® Core™ i5, 1.60 GHz. To compile the codes, we used `gfortran`, the GNU Fortran compiler.

`LMBM` is a Fortran77 implementation of the LMBM with the adaptive number of stored correction pairs (see [24] for the adaptive version of the code). The initial and maximum number of stored correction pairs used in our experiments were set to *seven* and 15, respectively. Otherwise, the default parameters of the code were used. The source code and the mex-driver (for MatLab users) are available for download at http://napsu.karmitsa.fi/lmbm/. In addition, the bound constrained version of the code [24, 25] is available.
`D-Bundle` and `SMDB` are a Fortran 95 implementations of the D-BUNDLE and SMDB, respectively. In both cases the maximum number of stored correction pairs was set to *seven* and for all the other parameters we used the default settings of the codes. The source codes of the methods are available for downloading at http://napsu.karmitsa.fi/dbundle/ and http://napsu.karmitsa.fi/smdb/.

In our previous experiments the line search was never needed with `SMDB` but the step size $t = 1$ was always accepted either as a serious or a null step. However, a simple acceptance or rejection of steps sometimes led to quite a large number of function and subgradient evaluations. Thus, we here use the version, which uses the nonmonotone Armijo-type line search. The idea is to seek for a suitable step size such that a serious step would occur. That is, we use at most *ten* previous function values obtained at serious steps to test the *modified serious step condition*

$$f(\boldsymbol{y}_{k+1}) \leq \max_{i\in M} f(\boldsymbol{x}_i) - \varepsilon_L w_k, \tag{5.42}$$

where $M \subseteq \{l \mid \boldsymbol{x}_{l+1} = \boldsymbol{x}_l + t_l \boldsymbol{d}_l\}$ such that M contains at most ten greatest indices l. The maximum number of Armijo step size searches within one iteration is set to 20. Only if none of Armijo step sizes satisfies (5.42) we do a null step. Note that no theoretical guarantee of the satisfaction of the null step condition (5.36) after this Armijo-type line search is given but, in our numerical experiments it was always satisfied.

We say that a solver finds the solution with respect to a tolerance $\varepsilon > 0$ if

$$\frac{f_{\texttt{best}} - f_{\texttt{opt}}}{1 + |f_{\texttt{opt}}|} \leq \varepsilon,$$

where $f_{\texttt{best}}$ is a solution obtained with the solver and $f_{\texttt{opt}}$ is the best known (or optimal) solution. We have *accepted the results* with respect to the tolerance $\varepsilon = 10^{-3}$. In addition, we say that the result is *inaccurate*, if a solver finds the solution with respect to a tolerance $\varepsilon = 10^{-2}$. Otherwise, we say that a solver *fails*. With the nonconvex problems it is not always easy to say whether the solution obtained is a local solution (or a stationary point) or not. Thus, we have only accepted the results that converge to the global minimum. Naturally, this leads us to report more "failures" than what really happens. In addition to the usual stopping criteria of the solvers, we terminated the experiments if the elapsed CPU time exceeded 2 h.

Results The results are summarized in Table 5.1. We have compared the efficiency of the solvers both in terms of the computational time (*cpu*) and the number of function and subgradient evaluations (nfg, evaluations for short). We have used bold-face text to emphasize the best results. An asterisk after a result means that the result obtained is inaccurate.

The overall performances of `LMBM`, `D-Bundle` and `SMDB` are quite similar, although, `D-Bundle` works best in convex settings while `LMBM` and `SMDB` solve nonconvex problems more efficiently. Note that `LMBM` and `SMDB` succeed in solving different nonconvex problems, but there are no such differences in the structures of these problems that explains this. In addition, `D-Bundle` and `SMDB` succeed better in solving Problem 1, in which `LMBM` is known to have difficulties due to the sparse structure of the problem and the dense approximation of Hessian used in `LMBM`. In the current settings none of the solvers could solve the piecewise linear problem 2. Nonetheless, Problem 2 is known to be very difficult to be solved by most NSO algorithms already in rather small dimensions [4] and we use here very large dimensions.

With one million variables the solvers succeed in solving only five or six problems out of 10. One could say that solving only about 50 or 60% of problems is not very convincing, but it is better than most nonsmooth algorithms can do [4, 18, 21]. Moreover, the solvers converged to the same (stationary) point also in Problem 6 and with all the solvers, some of the failures and inefficiencies could have been avoided if suitable parameters would have been chosen. However, we ran all our test cases with the same sets of parameters.

Table 5.1 Summary of the results

1 000 variables

P	LMBM		D-Bundle		SMDB	
	nfg	*cpu*	*nfg*	*cpu*	*nfg*	*cpu*
1	37 728*	0.97*	6 136	0.09	**5 999**	**0.06**
2	fail	–	fail	–	fail	–
3	3 292	0.03	3 751	0.03	**918**	**0.02**
4	3 450	0.04	6 917	0.08	**627**	**0.02**
5	**326**	**0.01**	1 388	**0.01**	719	0.03
6	1 138	**0.01**	1 075	0.02	**983**	0.04
7	5 690	0.45	13 319	0.84	**1225**	**0.14**
8	**6 020**	**0.05**	7 617	**0.05**	192 513	2.74
9	1 128	**0.01**	**1 106**	**0.01**	fail	–
10	11 282	0.10	17 377	0.11	**1 302**	**0.04**

10 000 variables

P	LMBM		D-Bundle		SMDB	
	nfg	*cpu*	*nfg*	*cpu*	*nfg*	*cpu*
1	fail	–	179 502	26.41	**60 003**	**6.40**
2	fail	–	fail	–	fail	–
3	6 082	0.52	3 669	0.27	**1 059**	**0.30**
4	7 080	0.81	**5 144**	**0.58**	9 007	2.68
5	**244**	**0.04**	307	**0.04**	792	0.27
6	10 108	1.55	10 104	2.03	**1 175**	**0.31**
7	8 888	6.26	49 823	32.95	**895**	**1.14**
8	**6 232**	**0.61**	11 261	0.88	361 347	48.87
9	fail	–	474*	0.06*	fail	–
10	fail	–	fail	–	**1 197**	**0.39**

100 000 variables

P	LMBM		D-Bundle		SMDB	
	nfg	*cpu*	*nfg*	*cpu*	*nfg*	*cpu*
1	fail	–	fail	–	**396 980**	**578.91**
2	fail	–	fail	–	fail	–
3	5 796	5.59	**144**	**0.13**	1 270	3.58
4	10 424	12.77	**584**	**0.82**	5 125	16.35
5	**438**	1.09	816	**0.84**	772	2.71
6	100 142	**166.46**	**100 100**	237.49	fail	–
7	fail	–	53 905	339.49	**2 251**	**25.18**
8	**2 100**	**4.05**	5 141	5.18	779 804	1 059.88
9	**1 400**	**3.87**	1 086*	2.30*	fail	–
10	34 630*	34.55*	fail	–	**1 757**	**5.78**

(continued)

Table 5.1 (continued)

One million variables						
P	LMBM		D-Bundle		SMDB	
	nfg	*cpu*	*nfg*	*cpu*	*nfg*	*cpu*
1	fail	–	fail	–	fail	–
2	fail	–	fail	–	fail	–
3	1 698	25.42	**367**	**4.67**	103 873	1 666.90
4	14 302	216.11	28 099	349.38	**3 698**	**64.92**
5	1 580	45.39	**772**	**9.08**	11 204	141.74
6	fail	–	fail	–	fail	–
7	fail	–	fail	–	**2 789**	**313.38**
8	**2 632**	**93.27**	19 727	195.34	119 455	1 765.93
9	**2748**	**107.13**	3 569*	79.64*	fail	–
10	fail	–	14 521*	173.89*	**349 915**	**7 200.22**

Acknowledgements The work was financially supported by the Academy of Finland (Project No. 289500).

Appendix

Line Search We now present a line search algorithm, which is used to determine the step sizes t_L^k and t_R^k in the LMBM, D-BUNDLE, and (if needed) in the SMDB. The line search procedure originates from [35]. However, in order to guarantee the global convergence of the LMBM, we use scaled line search parameters ε_L^k, ε_R^k, ε_A^k, and ε_T^k instead of fixed ones (scaled parameters are not needed with the D-BUNDLE and SMDB). Furthermore, in order to avoid many consecutive null steps, we have added an additional interpolation step (Step 3 in Algorithm 5.4). That is, we look for more suitable step sizes t_L^k and t_R^k by using an extra interpolation loop if necessary.

The additional interpolation step has no influence on the convergence properties but it has a significant effect on the efficiency of the method. The choice of the interpolation procedure (see Step 5 in Algorithm 5.4) has no effect on the convergence properties, either. We combine here the quadratic interpolation with the bisection procedure.

Algorithm 5.4: Line search

Data: Iteration point $\boldsymbol{x}_k$, search direction $\boldsymbol{d}_k$, scaling parameter $\theta_k \in (0, 1]$, initial line search parameters $\varepsilon_L^I \in (0, 1/2)$, $\varepsilon_R^I \in (\varepsilon_L^I, 1/2)$, $\varepsilon_A^I \in (0, \varepsilon_R^I - \varepsilon_L^I)$, and $\varepsilon_T^I \in (\varepsilon_L^I, \varepsilon_R^I - \varepsilon_A^I)$, lower and upper bounds for serious steps $t_{min} \in (0, 1)$

and $t_{max} > 1$, initial step size $t_I^k \in [t_{min}, t_{max})$, distance measure parameter $\gamma \geq 0$, desirable amount of descent w_k, maximum number of additional interpolations i_{max}, and the number of consecutive null steps $i_{null} \geq 0$.

Step 0. (*Initialization*) Set $t_A = 0$, $t = t_U = t_I^k$, and $i_I = 0$. Calculate the scaled line search parameters

$$\varepsilon_L^k = \theta_k \varepsilon_L^I, \quad \varepsilon_R^k = \theta_k \varepsilon_R^I, \quad \varepsilon_A^k = \theta_k \varepsilon_A^I, \quad \text{and} \quad \varepsilon_T^k = \theta_k \varepsilon_T^I$$

and the interpolation parameter

$$\kappa = 1 - \frac{1}{2(1 - \varepsilon_T^k)}.$$

Step 1. (*New values*) Compute $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k)$, $\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k)$, and

$$\beta = \max\{\, |f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) + t\theta_k \langle \boldsymbol{d}_k, \boldsymbol{\xi} \rangle|,\ \gamma\, (t\theta_k \|\boldsymbol{d}_k\|)^2 \,\}.$$

If $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \varepsilon_T^k t w_k$, then set $t_A = t$. Otherwise, set $t_U = t$.

Step 2. (*Serious step*) If

$$f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \varepsilon_L^k t w_k,$$

and either

$$t \geq t_{min} \qquad \text{or} \qquad \beta > \varepsilon_A^k w_k,$$

then set $t_R^k = t_L^k = t$ and **stop** with a serious step.

Step 3. (*Test for additional interpolation*) If $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) > f(\boldsymbol{x}_k)$, $i_{null} > 0$, and $i_I < i_{max}$, then set $i_I = i_I + 1$ and go to Step 5.

Step 4. (*Null step*) If

$$-\beta + \theta_k \langle \boldsymbol{d}_k, \boldsymbol{\xi} \rangle \geq -\varepsilon_R^k w_k,$$

then set $t_R^k = t$, $t_L^k = 0$ and **stop** with a null step.

Step 5. (*Interpolation*) If $t_A = 0$, then set

$$t = \max \left\{ \kappa t_U,\ \frac{-\frac{1}{2} t_U^2 w_k}{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) - t_U w_k} \right\}.$$

Otherwise, set $t = \frac{1}{2}(t_A + t_U)$. Go to Step 1.

Under the semismoothness assumptions (see Definition 1.11) Algorithm 5.4 is guaranteed to find the step sizes t_L^k and t_R^k such that exactly one of the two possibilities—a serious step or a null step—occurs [35]. In addition, on the output of Algorithm 5.4 (see Steps 2 and 4), the step sizes t_L^k and t_R^k satisfy the serious descent criterion

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\varepsilon_L^k t_L^k w_k \tag{5.43}$$

and, in case of $t_L^k = 0$ (null step), also the condition (5.5).

References

1. Astorino, A., Fuduli, A.: Nonsmooth optimization techniques for semi-supervised classification. IEEE Trans. Pattern Anal. Mach. Intell. **29**(12), 2135–2142 (2007)
2. Astorino, A., Fuduli, A., Gorgone, E.: Nonsmoothness in classification problems. Optim. Methods Softw. **23**(5), 675–688 (2008)
3. Äyrämö, S.: Knowledge mining using robust clustering. Ph.D. Thesis, University of Jyväskylä, Department of Mathematical Information Technology (2006)
4. Bagirov, A., Karmitsa, N., Mäkelä, M. M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, Berlin (2014)
5. Bagirov, A., Taheri, S., Ugon, J.: Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. Pattern Recogn. **53**, 12–24 (2016)
6. Bergeron, C., Moore, G., Zaretzki, J., Breneman, C., Bennett, K.: Fast bundle algorithm for multiple instance learning. IEEE Trans. Pattern Anal. Mach. Intell. **34**(6), 1068–1079 (2012)
7. Bradley, P.S., Fayyad, U.M., Mangasarian, O.L.: Mathematical programming for data mining: formulations and challenges. INFORMS J. Comput. **11**, 217–238 (1999)
8. Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-Newton matrices and their use in limited memory methods. Math. Program. **63**, 129–156 (1994)
9. Carrizosa, E., Romero Morales, D.: Supervised classification and mathematical optimization. Comput. Oper. Res. **40**(1), 150–165 (2013)
10. Clarke, F.H., Ledyaev, Y.S., Stern, R.J., Wolenski, P.R.: Nonsmooth Analysis and Control Theory. Springer, New York (1998)
11. Demyanov, V.F., Bagirov, A., Rubinov, A.: A method of truncated codifferential with application to some problems of cluster analysis. J. Global Optim. **23**(1), 63–80 (2002)
12. Haarala, M.: Large-scale nonsmooth optimization: variable metric bundle method with limited memory. Ph.D. Thesis, University of Jyväskylä, Department of Mathematical Information Technology (2004)
13. Haarala, M., Miettinen, K., Mäkelä, M.M.: New limited memory bundle method for large-scale nonsmooth optimization. Optim. Methods Softw. **19**(6), 673–692 (2004)
14. Haarala, N., Miettinen, K., Mäkelä, M.M.: Globally convergent limited memory bundle method for large-scale nonsmooth optimization. Math. Program. **109**(1), 181–205 (2007)
15. Han, G., Wu, X., Zhang, S., Liu, Z., Navon, I., Li, W.: A study of coupling parameter estimation implemented by 4D-Var and EnKF with a simple coupled system. Adv. Meteorol. **2015**, 530764, 16pp. (2015). https://doi.org/10.1155/2015/530764
16. Haslinger, J., Neittaanmäki, P.: Finite Element Approximation for Optimal Shape, Material and Topology Design, 2nd edn. Wiley, Chichester (1996)
17. Kärkkäinen, T., Heikkola, E.: Robust formulations for training multilayer perceptrons. Neural Comput. **16**, 837–862 (2004)

18. Karmitsa, N.: Diagonal bundle method for nonsmooth sparse optimization. J. Optim. Theory Appl. **166**(3), 889–905 (2015). https://doi.org/10.1007/s10957-014-0666-8
19. Karmitsa, N., Bagirov, A., Taheri, S.: New diagonal bundle method for clustering problems in large data sets. Eur. J. Oper. Res. **263**(2), 367–379 (2017). https://doi.org/10.1016/j.ejor.2017.06.010
20. Karmitsa, N., Bagirov, A., Taheri, S.: Clustering in large data sets with the limited memory bundle method. Pattern Recognit. **83**, 245–259 (2018)
21. Karmitsa, N., Bagirov, A., Mäkelä, M.M.: Comparing different nonsmooth optimization methods and software. Optim. Methods Softw. **27**(1), 131–153 (2012)
22. Karmitsa, N., Bagirov, A., Taheri, S.: Limited memory bundle method for solving large clusterwise linear regression problems. TUCS Technical Report, No. 1172, Turku Centre for Computer Science, Turku (2016). http://tucs.fi/publications/view/?pub_id=tKaBaTa16c.
23. Karmitsa, N., Gaudioso, M., Joki, K.: Diagonal bundle method with convex and concave updates for large-scale nonconvex and nonsmooth optimization. Optim. Methods Softw. **34**(2), 363–382 (2019). https://doi.org/10.1080/10556788.2017.1389941
24. Karmitsa, N., Mäkelä, M.M.: Adaptive limited memory bundle method for bound constrained large-scale nonsmooth optimization. Optim. J. Math. Program. Oper. Res. **59**(6), 945–962 (2010)
25. Karmitsa, N., Mäkelä, M.M.: Limited memory bundle method for large bound constrained nonsmooth optimization: convergence analysis. Optim. Methods Softw. **25**(6), 895–916 (2010)
26. Karmitsa, N., Mäkelä, M. M., Ali, M. M.: Limited memory interior point bundle method for large inequality constrained nonsmooth minimization. Appl. Math. Comput. **198**(1), 382–400 (2008)
27. Karmitsa, N., Taheri, S., Bagirov, A., Mäkinen, P.: Clusterwise linear regression based missing value imputation for data preprocessing. TUCS Technical Report, No. 1193, Turku Centre for Computer Science, Turku (2018). https://tucs.fi/publications/view/?pub_id=tKaTaBaMx18a
28. Kiwiel, K. C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics, vol. 1133. Springer, Berlin (1985)
29. Lukšan, L., Vlček, J.: Globally convergent variable metric method for convex nonsmooth unconstrained minimization. J. Optim. Theory Appl. **102**(3), 593–613 (1999)
30. Majava, K., Haarala, N., Kärkkäinen, T.: Solving variational image denoising problems using limited memory bundle method. In: Liu, W., Ng, M., Shi, Z.C. (eds.) Recent Progress in Scientific Computing. Proceedings of SCPDE05, pp. 319–332. Science Press, Beijing (2007)
31. Mistakidis, E.S., Stavroulakis, G.E.: Nonconvex Optimization in Mechanics. Smooth and Nonsmooth Algorithms, Heuristics and Engineering Applications by the F.E.M. Kluwert Academic Publishers, Dordrecht (1998)
32. Moreau, J., Panagiotopoulos, P.D., Strang, G. (eds.): Topics in Nonsmooth Mechanics. Birkhäuser Verlag, Basel (1988)
33. Outrata, J., Kočvara, M., Zowe, J.: Nonsmooth Approach to Optimization Problems with Equilibrium Constraints. Theory, Applications and Numerical Results. Kluwert Academic Publisher, Dordrecht (1998)
34. Steward, J., Navon, I., Zupanski, M., Karmitsa, N.: Impact of non-smooth observation operators on variational and sequential data assimilation for a limited-area shallow water equations model. Q. J. R. Meteorol. Soc. **138**(663), 323–339 (2012). Part B
35. Vlček, J., Lukšan, L.: Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. J. Optim. Theory Appl. **111**(2), 407–430 (2001)

Chapter 6
Gradient Sampling Methods for Nonsmooth Optimization

James V. Burke, Frank E. Curtis, Adrian S. Lewis, Michael L. Overton, and Lucas E. A. Simões

Dedicated to Krzysztof Kiwiel, in recognition of his fundamental work on algorithms for nonsmooth optimization

Abstract This article reviews the gradient sampling methodology for solving nonsmooth, nonconvex optimization problems. We state an intuitively straightforward gradient sampling algorithm and summarize its convergence properties. Throughout this discussion, we emphasize the simplicity of gradient sampling as an extension of the steepest descent method for minimizing smooth objectives. We provide an overview of various enhancements that have been proposed to improve practical performance, as well as an overview of several extensions that have been proposed in the literature, such as to solve constrained problems. We also clarify certain technical aspects of the analysis of gradient sampling algorithms, most notably related to the assumptions one needs to make about the set of points at which the objective is continuously differentiable. Finally, we discuss possible future research directions.

J. V. Burke
Department of Mathematics, University of Washington, Seattle, WA, USA

F. E. Curtis
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA
e-mail: frank.e.curtis@lehigh.edu

A. S. Lewis
School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA
e-mail: adrian.lewis@cornell.edu

M. L. Overton (✉)
Courant Institute of Mathematical Sciences, New York University, New York, NY, USA
e-mail: mo1@nyu.edu

L. E. A. Simões
Department of Applied Mathematics, University of Campinas, Campinas, Brazil

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_6

6.1 Introduction

The gradient sampling (GS) algorithm is a conceptually simple descent method for solving nonsmooth, nonconvex optimization problems, yet it is one that possesses a solid theoretical foundation and has been employed to substantial success in a wide variety of applications. Since the appearance of the fundamental algorithm and its analysis little over a dozen years ago, GS has matured into a comprehensive methodology. Various enhancements have been proposed that make it a competitive approach in many NSO contexts, and it has been extended in various interesting ways, such as for NSO on manifolds and for solving constrained problems. The purpose of this work is to provide background and motivation for the development of the GS method, discuss its theoretical guarantees, and provide an overview of the enhancements and extensions that have been the subject of research over recent years.

The underlying philosophy of GS is that virtually any nonsmooth objective function of interest is differentiable almost everywhere; in particular, this is true if the objective $f : \mathbb{R}^n \to \mathbb{R}$ is either LLC (see Theorem 1.5) or semialgebraic. In such cases, when f is evaluated at a randomly generated point $\boldsymbol{x} \in \mathbb{R}^n$, it is differentiable there with probability one. This means that an algorithm can rely on an ability to obtain the objective function value $f(\boldsymbol{x})$ and gradient $\nabla f(\boldsymbol{x})$, as when f is smooth, rather than require an oracle to compute a subgradient. In most interesting settings, f *is not* differentiable at its local minimizers, but, under reasonable assumptions, the carefully crafted mechanisms of the GS algorithm generate a sequence of iterates—at which f *is* differentiable—converging to stationarity.

At the heart of GS is a stabilized steepest descent approach. When f is differentiable at $\boldsymbol{x}$, the negative gradient $-\nabla f(\boldsymbol{x})$ is, of course, the traditional steepest descent direction for f at $\boldsymbol{x}$ in the 2-norm in that

$$-\frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|} = \underset{\|\boldsymbol{d}\|\leq 1}{\operatorname{argmin}} \nabla f(\boldsymbol{x})^T \boldsymbol{d}. \tag{6.1}$$

However, when $\boldsymbol{x}$ is near a point where f is not differentiable, it may be necessary to take a very short step along $-\nabla f(\boldsymbol{x})$ to obtain decrease in f. It is for this reason that the traditional steepest descent method may converge to nonstationary points when f is nonsmooth.[1] The GS algorithm stabilizes the choice of the search direction to avoid this issue. In each iteration, a descent direction from the current iterate $\boldsymbol{x}^k$ is obtained by supplementing the information provided by $\nabla f(\boldsymbol{x}^k)$ with gradients evaluated at randomly generated points $\{\boldsymbol{x}^{k,1}, \ldots, \boldsymbol{x}^{k,m}\} \subset \bar{B}(\boldsymbol{x}^k; \epsilon_k)$, which are *near* $\boldsymbol{x}^k$, and then computing the minimum-norm vector $\boldsymbol{g}^k$ in the convex hull of

[1] Although this fact has been known for decades, most of the examples that appear in the literature are rather artificial since they were designed with exact line searches in mind. Analyses showing that the steepest descent method with inexact line searches converges to nonstationary points of some simple convex nonsmooth functions have appeared recently in [1, 22].

these gradients. This choice can be motivated by the goal, using the ϵ-subdifferential $\partial_\epsilon^G f(\boldsymbol{x})$ given in Definition 1.10, that

$$-\frac{\boldsymbol{g}^k}{\|\boldsymbol{g}^k\|} \approx \operatorname*{argmin}_{\|\boldsymbol{d}\|\leq 1} \max_{\boldsymbol{g}\in\partial_\epsilon^G f(\boldsymbol{x})} \boldsymbol{g}^T\boldsymbol{d}; \tag{6.2}$$

i.e., $-\boldsymbol{g}^k$ can essentially be viewed as a steepest descent direction for f from $\boldsymbol{x}^k$ in a more "robust" sense. A line search is then used to find a positive stepsize t_k yielding decrease in f, i.e., $f(\boldsymbol{x}^k - t_k\boldsymbol{g}^k) < f(\boldsymbol{x}^k)$. The sampling radius ϵ_k that determines the meaning of "*near* $\boldsymbol{x}^k$" may either be fixed or adjusted dynamically.

A specific instance of the GS algorithm is presented in Sect. 6.2. Its convergence guarantees are summarized in Sect. 6.3. We then present various enhancements and extensions of the approach in Sects. 6.4 and 6.5, respectively, followed by a discussion of some successful applications of the GS methodology in Sect. 6.6. Throughout this work, our goal is to emphasize the *simplicity* of the fundamental GS strategy. We believe that this, in addition to its strong convergence properties for locally Lipschitz optimization, makes it an attractive choice when attempting to solve difficult types of NSO problems.

Although the first convergence analysis of a GS algorithm was given by Burke, Lewis, and Overton in [8], an earlier version of the method was presented by these authors in [7]. That algorithm, originally called a "gradient bundle" method, was applied to a function that was not only nonconvex and nonsmooth, but also non-locally-Lipschitz, namely, the spectral abscissa—i.e., the largest of the real parts of the eigenvalues—of a linear matrix function A mapping a parameter vector $\boldsymbol{x}$ to the space of nonsymmetric square matrices. The spectral abscissa is not locally Lipschitz at a matrix $\bar{X}$ when an eigenvalue of $\bar{X}$ with largest real part has multiplicity two or more [5], but it is semialgebraic and, hence, differentiable almost everywhere, so a GS algorithm was applicable. The method was surprisingly effective. As anticipated, in most cases the apparent local minimizers that were approximated had the property that the eigenvalues of A with largest real part had multiplicity two or more. An illustration that appeared in [7] is reproduced in Fig. 6.1; the extremely "steep" contours of the objective function indicate its non-Lipschitzness. Obtaining theoretical results for a GS algorithm applied to solve non-locally-Lipschitz problems seems very challenging; we discuss this issue further in Sect. 6.3.3, after describing the substantial body of theory that has been developed for the locally Lipschitz case in Sects. 6.3.1 and 6.3.2.

6.2 Algorithm GS

We now state a specific variant of the GS algorithm. We start by assuming only that the objective function f is locally Lipschitz over $\mathbb{R}^n$, which implies, by Rademacher's theorem (see Theorem 1.5), that f is differentiable almost every-

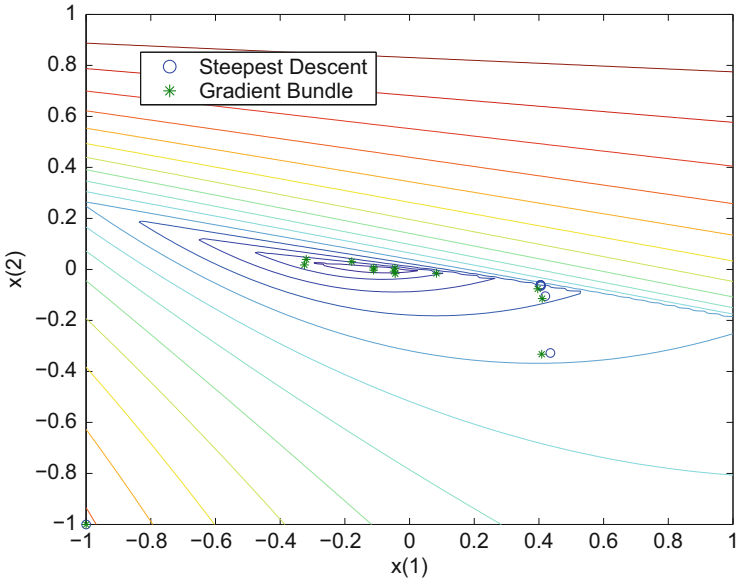

Fig. 6.1 Contours of the spectral abscissa of an affine matrix family given in [7]. Iterates of the ordinary gradient ("steepest descent") method with a line search are shown (small circles) along with those of the GS ("gradient bundle") algorithm (asterisks). Both start at $(-1, -1)$

where. As previously mentioned, at the heart of the algorithm is the computation of a descent direction by finding the minimum norm element of the convex hull of gradients obtained about each iterate. The remaining procedures relate to the line search to obtain decrease in f and the selection of a subsequent iterate so that f is differentiable at all elements of $\{\boldsymbol{x}^k\}$.

While the essence of the methods from [7] and [8] remains intact, Algorithm GS differs in subtle yet important ways from the methods presented in these papers, as we now explain.

1. Algorithm GS incorporates a key modification proposed by Kiwiel in [28, Algorithm 2.1], namely, the second inequality in (6.4); the version in [8] used ϵ_k instead of $\min\{t_k, \epsilon_k\}$. As Kiwiel explained, this minor change allowed him to drop the assumption in [8] that the level set $\{\boldsymbol{x} : f(\boldsymbol{x}) \leq f(\boldsymbol{x}^0)\}$ is compact, strengthening the convergence results for the algorithm.
2. A second change suggested in [28, Algorithm 2.1] is the introduction of the termination tolerances ν_{opt} and ϵ_{opt}. These were used in the computational experiments in [8], but not in the algorithm statement or analysis. Note that if ϵ_{opt} is set to zero, then Algorithm GS never terminates since ϵ_k can never be zero, though it may happen that one obtains $\|\boldsymbol{g}^k\| = 0$.
3. A third change, also suggested by Kiwiel, is the usage of the nonnormalized search direction $-\boldsymbol{g}^k$ (originally used in [7]) instead of the normalized search direction $-\boldsymbol{g}^k/\|\boldsymbol{g}^k\|$ (used in [8]). The resulting inequalities in (6.3) and (6.4) are

taken from [28, Section 4.1]. This choice does not affect the main conclusions of the convergence theory as in both cases it is established [8, 28] that the stepsize t_k can be determined by a finite process. However, since Theorem 6.1 below shows that a subsequence of $\{\boldsymbol{g}^k\}$ converges to zero under reasonable conditions, one expects that fewer function evaluations should be required by the line search asymptotically when using the nonnormalized search direction, whereas using the normalized direction may result in the number of function evaluations growing arbitrarily large [28, Section 4.1]. Our practical experience is consistent with this viewpoint.

4. Another aspect of Algorithm GS that is different in both [8] and [28] concerns the randomization procedure in Step 2. In the variants given in those papers, it was stated that the algorithm terminates if f is not *continuously* differentiable at the randomly sampled points $\{\boldsymbol{x}^{k,1}, \ldots, \boldsymbol{x}^{k,m}\}$. In the theorem stated in the next section, we require only that f is differentiable at the sampled points. Since by Rademacher's theorem and countable additivity of probabilities this holds for every sampled point with probability one, we do not include a termination condition here.
5. Finally, Steps 11–14 in Algorithm GS *do* require explicit checks that ensure that f is differentiable at $\boldsymbol{x}^{k+1}$, but unlike in the variants in [8] and [28], it is not required that f be *continuously* differentiable at $\boldsymbol{x}^{k+1}$. This differentiability requirement is included since it is not the case that f is differentiable at $\boldsymbol{x}^k - t_k \boldsymbol{g}^k$ with probability one, as is shown via an example in [24], discussed further in Sect. 6.4.2. For a precise procedure for implementing Step 14, see [28].

The computation in Step 3 of Algorithm GS requires solving a strongly convex quadratic optimization problem (QP) to compute the minimum-norm element of the convex hull of the current and sampled gradients, or, equivalently, to compute the 2-norm projection of the origin onto this convex hull. It is essentially the same operation required in every iteration of a bundle method. To see this, observe that solving the QP in Step 3 can be expressed, with

$$G_k := \left[\nabla f(\boldsymbol{x}^k)\ \nabla f(\boldsymbol{x}^{k,1}) \cdots \nabla f(\boldsymbol{x}^{k,m})\right],$$

as computing $(z_k, \boldsymbol{d}^k, \boldsymbol{\lambda}^k) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^{m+1}$ as the primal-dual solution of the QPs

$$\begin{cases} \text{minimize} & z + \frac{1}{2}\|\boldsymbol{d}\|^2 \\ \text{subject to} & G_k^T \boldsymbol{d} \le z\mathbf{1}, \\ & (z, \boldsymbol{d}) \in \mathbb{R} \times \mathbb{R}^n, \end{cases} \quad \text{and} \quad \begin{cases} \text{minimize} & \frac{1}{2}\|G_k \boldsymbol{\lambda}\|^2 \\ \text{subject to} & \mathbf{1}^T \boldsymbol{\lambda} = 1,\ \boldsymbol{\lambda} \ge \mathbf{0}, \\ & \boldsymbol{\lambda} \in \mathbb{R}^{m+1}. \end{cases} \tag{6.5}$$

The latter problem, yielding $G_k \boldsymbol{\lambda}^k = \boldsymbol{g}^k$, can easily be seen to be equivalent to solving the subproblem $\min_{\boldsymbol{g} \in \mathcal{G}^k} \frac{1}{2}\|\boldsymbol{g}\|^2$ stated in Step 3, whereas the former problem, yielding $\boldsymbol{d}^k = -\boldsymbol{g}^k$, can be seen to have the same form as the subproblems arising in bundle methods.

Algorithm GS: Gradient sampling

Data: initial point $\boldsymbol{x}^0$ at which f is differentiable, initial sampling radius $\epsilon_0 \in (0, \infty)$, initial stationarity target $\nu_0 \in [0, \infty)$, sample size $m \geq n + 1$, line search parameters $(\beta, \gamma) \in (0, 1) \times (0, 1)$, termination tolerances $(\epsilon_{\text{opt}}, \nu_{\text{opt}}) \in [0, \infty) \times [0, \infty)$, and reduction factors $(\theta_\epsilon, \theta_\nu) \in (0, 1] \times (0, 1]$;

for $k \in \mathbb{N}$ **do**

independently sample $\{\boldsymbol{x}^{k,1}, \ldots, \boldsymbol{x}^{k,m}\}$ uniformly from $\bar{B}(\boldsymbol{x}^k; \epsilon_k)$

compute $\boldsymbol{g}^k$ as the solution of $\min_{\boldsymbol{g} \in \mathcal{G}^k} \frac{1}{2}\|\boldsymbol{g}\|^2$, where

$$\mathcal{G}^k := \text{conv}\{\nabla f(\boldsymbol{x}^k), \nabla f(\boldsymbol{x}^{k,1}), \ldots, \nabla f(\boldsymbol{x}^{k,m})\};$$

if $\|\boldsymbol{g}^k\| \leq \nu_{\text{opt}}$ and $\epsilon_k \leq \epsilon_{\text{opt}}$, **then**

terminate;

else

if $\|\boldsymbol{g}^k\| \leq \nu_k$, **then**

set $\nu_{k+1} \leftarrow \theta_\nu \nu_k$, $\epsilon_{k+1} \leftarrow \theta_\epsilon \epsilon_k$, and $t_k \leftarrow 0$;

else

set $\nu_{k+1} \leftarrow \nu_k$, $\epsilon_{k+1} \leftarrow \epsilon_k$, and

$$t_k \leftarrow \max\left\{t \in \{1, \gamma, \gamma^2, \ldots\} : f(\boldsymbol{x}^k - t\boldsymbol{g}^k) < f(\boldsymbol{x}^k) - \beta t \|\boldsymbol{g}^k\|^2\right\}; \tag{6.3}$$

if f is differentiable at $\boldsymbol{x}^k - t_k \boldsymbol{g}^k$, **then**

set $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k - t_k \boldsymbol{g}^k$;

else

set $\boldsymbol{x}^{k+1}$ randomly as any point where f is differentiable such that

$$f(\boldsymbol{x}^{k+1}) < f(\boldsymbol{x}^k) - \beta t_k \|\boldsymbol{g}^k\|^2 \quad \text{and} \quad \|\boldsymbol{x}^k - t_k \boldsymbol{g}^k - \boldsymbol{x}^{k+1}\| \leq \min\{t_k, \epsilon_k\}\|\boldsymbol{g}^k\|; \tag{6.4}$$

Normally, the initial stationarity target ν_0 is chosen to be positive and the reduction factors θ_ν and θ_ϵ are chosen to be less than one so that the stationarity target and sampling radius are reduced every time the condition $\|\boldsymbol{g}^k\| \leq \nu_k$ is satisfied. However, it is also interesting to consider the variant with $\nu_0 = 0$ and $\theta_\epsilon = 1$, forcing the algorithm to run forever with ϵ fixed unless it terminates with $\boldsymbol{g}^k = \boldsymbol{0}$ for some $k \in \mathbb{N}$. We consider both of these variants in the global convergence theory given in the next section.

6.3 Convergence Theory for Algorithm GS

Any NSO method seeks to find a point $\boldsymbol{x}$ that is Clarke stationary (such that $\mathbf{0} \in \partial f(\boldsymbol{x})$; see (1.7), Definition 1.8 and Theorem 1.2) or at least ϵ-stationary (such that $\mathbf{0} \in \partial_\epsilon^G f(\boldsymbol{x})$; see Definition 1.10). For all practical purposes, one cannot generally evaluate ∂f (or $\partial_\epsilon^G f$) at (or near) any point where f is not differentiable. That said, Algorithm GS is based on the idea that one can approximate the minimum norm element in $\partial_\epsilon^G f(\boldsymbol{x}^k)$ through random sampling of gradients in the ball $\bar{B}(\boldsymbol{x}^k; \epsilon)$. To a large extent this idea is motivated by [6] which investigates how well the entire Clarke subdifferential $\partial f(\boldsymbol{x})$ can be approximated through random sampling. However, the results in [6] cannot be directly exploited in the analysis of the GS algorithm because the gradients are sampled only at a finite number of points near any given iterate.

One might wonder why Algorithm GS involves the use of $m \geq n+1$ gradients at randomly sampled points in addition to the gradient at the current iterate. Loosely speaking, this lower bound on the number of gradients is required in the analysis to ensure that one can use Carathéodory's theorem to argue that there is a sufficiently good chance that $-\boldsymbol{g}^k$ either (1) is a sufficiently good search direction from $\boldsymbol{x}^k$, or (2) is small enough to indicate that the current iterate is *approximately* ϵ^k-stationary. In the latter case, the analysis actually involves checking for ϵ^k-stationarity of a point *near* $\boldsymbol{x}^k$, not of $\boldsymbol{x}^k$ itself, which explains why $n+1$ gradients are needed *in addition to* $\nabla f(\boldsymbol{x}^k)$. It remains an open question whether it would suffice to sample gradients at just n randomly sampled points in addition to the gradient at $\boldsymbol{x}^k$; however, such a possibility is not of much consequence, especially if one employs the *adaptive sampling* enhancement discussed in Sect. 6.4.3.

6.3.1 Global Convergence Guarantees

A critical aspect of theoretical convergence guarantees for Algorithm GS concerns the set of points where f is *continuously* differentiable, which we denote by D. Consideration of D played a crucial role in the analysis in both [8] and [28], but there were some oversights concerning both the requirements of the algorithm with respect to D and the assumptions on D. Regarding the requirements of the algorithm with respect to D, there is actually no need, from a theoretical point of view, for either the iterates $\{\boldsymbol{x}^k\}$ or the randomly generated sampled points $\{\boldsymbol{x}^{k,j}\}$ to lie in D; all that is needed is that f is differentiable at these points. Most implementations of GS algorithms do not attempt to check any form of differentiability in any case, but if one were to attempt to implement such a check, it is certainly more tractable to check for differentiability than continuous differentiability. Regarding the assumptions on D, in the theorems that we state below, we assume that D is an open set with full measure in $\mathbb{R}^n$. In contrast, the relevant assumption stated in [8, 28] is weaker, namely, that D is an open dense subset of $\mathbb{R}^n$. However, the proofs

of convergence actually require the full measure assumption on D that we include below.[2]

There are three types of global convergence guarantees of interest for Algorithm GS: one when the input parameters ensure that $\{\epsilon_k\} \downarrow 0$, one when ϵ_k is repeatedly reduced but a positive stopping tolerance prevents it from converging to zero, and one when $\epsilon_k = \epsilon > 0$ for all k. These lead to different properties for the iterate sequence. The first theorem below relates to cases when the stationarity tolerance and sampling radius tolerance are both set to zero so that the algorithm can never terminate.

Theorem 6.1 *Suppose that f is locally Lipschitz on $\mathbb{R}^n$ and continuously differentiable on an open set D with full measure in $\mathbb{R}^n$. Suppose further that Algorithm GS is run with $\nu_0 > 0$, $\nu_{\text{opt}} = \epsilon_{\text{opt}} = 0$, and strict reduction factors $\theta_\nu < 1$ and $\theta_\epsilon < 1$. Then, with probability one, Algorithm GS is well defined in the sense that the gradients in Step 3 exist in every iteration, the algorithm does not terminate, and either*

(i) $f(\boldsymbol{x}^k) \downarrow -\infty$*; or*
(ii) $\nu_k \downarrow 0$, $\epsilon_k \downarrow 0$, *and every cluster point of $\{\boldsymbol{x}^k\}$ is Clarke stationary for f.*

Theorem 6.1 is essentially the same as [28, Theorem 3.3] (with the modifications given in [28, Section 4.1] for nonnormalized directions), except for two aspects:

1. The proof given in [28] implicitly assumes that D is an open set with full measure, as does the proof of [8, Theorem 3.4] on which Kiwiel's proof is based, although the relevant assumption on D in both papers is the weaker condition that D is an open dense set. Details are given in Appendix 1.
2. In the algorithms analyzed in [28] and [8], the iterates $\{\boldsymbol{x}^k\}$ and the randomly sampled points $\{\boldsymbol{x}^{k,j}\}$ were enforced to lie in the set D where f is continuously differentiable. We show in Appendix 2 that the theorem still holds when this requirement is relaxed to ensure only that f is differentiable at these points.

As Kiwiel argues, Theorem 6.1 is essentially the best result that could be expected. Furthermore, as pointed out in [28, Remark 3.7(ii)], it leads immediately to the following corollary.

Corollary 6.1 *Suppose that f is locally Lipschitz on $\mathbb{R}^n$ and continuously differentiable on an open set D with full measure in $\mathbb{R}^n$. Suppose further that Algorithm GS is run with $\nu_0 > \nu_{\text{opt}} > 0$, $\epsilon_0 > \epsilon_{\text{opt}} > 0$, and strict reduction factors $\theta_\nu < 1$ and $\theta_\epsilon < 1$. Then, with probability one, Algorithm GS is well defined in the sense that the gradients in Step 3 exist at every iteration, and either*

(i) $f(\boldsymbol{x}^k) \downarrow -\infty$*; or*
(ii) *Algorithm GS terminates by the stopping criteria in Step 4.*

[2] This oversight went unnoticed for 12 years until J. Portegies and T. Mitchell brought it to our attention recently.

The final result that we state concerns the case when the sampling radius is fixed. A proof of this result is essentially given by that of [28, Theorem 3.5], again taking into account the comments in the Appendices.

Theorem 6.2 *Suppose that f is locally Lipschitz on $\mathbb{R}^n$ and continuously differentiable on an open set D with full measure in $\mathbb{R}^n$. Suppose further that Algorithm GS is run with $\nu_0 = \nu_{\text{opt}} = 0$, $\epsilon_0 = \epsilon_{\text{opt}} = \epsilon > 0$, and $\theta_\epsilon = 1$. Then, with probability one, Algorithm GS is well defined in the sense that the gradients in Step 3 exist at every iteration, and one of the following occurs:*

(a) $f(\boldsymbol{x}^k) \downarrow -\infty$*; or*
(b) *Algorithm GS terminates for some $k \in \mathbb{N}$ with $\boldsymbol{g}^k = \mathbf{0}$; or*
(c) *there exists $\mathcal{K} \subseteq \mathbb{N}$ with $\{\boldsymbol{g}^k\}_{k\in\mathcal{K}} \to \mathbf{0}$ and every cluster point of $\{\boldsymbol{x}^k\}_{k\in\mathcal{K}}$ is ϵ-stationary for f.*

Of the five open questions regarding the convergence analysis for GS raised in [8], three were answered explicitly by Kiwiel in [28]. Another open question was: "Under what conditions can one guarantee that the GS algorithm terminates finitely?" This was posed in the context of a fixed sampling radius and therefore asks how one might know whether outcome (b) or (c) occurs in Theorem 6.2, assuming f is bounded below. This remains open, but Kiwiel's introduction of the termination tolerances in the GS algorithm statement led to Corollary 6.1 which guarantees that when the sampling radius is reduced dynamically and the tolerances are nonzero, Algorithm GS must terminate if f is bounded below. The only other open question concerns extending the convergence analysis to the non-Lipschitz case.

Overall, Algorithm GS has a very satisfactory convergence theory in the locally Lipschitz case. Its main weakness is its per-iteration cost, most notably due to the need to compute $m \geq n + 1$ gradients in every iteration and solve a corresponding QP. However, enhancements to the algorithm have been proposed that can vastly reduce this per-iteration cost while maintaining these guarantees. We discuss these and other enhancements in Sect. 6.4.

6.3.2 A Local Linear Convergence Result

Given the relationship between GS and a traditional steepest descent approach, one might ask if there are scenarios in which Algorithm GS can attain a linear rate of local convergence. The study in [25] answers this in the affirmative, at least in a certain probabilistic sense. If (1) the set of sampled points is *good* in a certain sense described in [25], (2) the objective function f belongs to a class of functions defined by the maximum of a finite number of smooth functions ("finite-max" functions), and (3) the input parameters are set appropriately, then Algorithm GS will produce a step yielding a reduction in f that is significant. This analysis involves $\mathcal{VU}$-decomposition ideas [31, 32, 36], where in particular it is shown that the reduction in f is comparable to that achieved by a steepest descent method restricted to the

smooth $\mathcal{U}$-space of f. This means that a linear rate of local convergence can be attained over any infinite subsequence of iterations in which the sets of sampled points are *good*.

6.3.3 The Non-Lipschitz Case

In the non-locally-Lipschitz case, the Clarke subdifferential ∂f is defined in [6, p. 573]; unlike in the Lipschitz case, this set may be unbounded, presenting obvious difficulties for approximating it through random sampling of gradients. In fact, more than half of [6] is devoted to investigating this issue, relying heavily on modern variational analysis as expounded in [41]. Some positive results were obtained, specifically in the case that f is "directionally Lipschitz" at $\bar{\boldsymbol{x}}$, which means that the "horizon cone" [6, p. 572] of f at $\bar{\boldsymbol{x}}$ is pointed, that is, it does not contain a line. For example, this excludes the function on $\mathbb{R}$ defined by $f(x) = |x|^{1/2}$ at $\bar{x} = 0$, but it applies to the case $f(x) = (\max\{0, x\})^{1/2}$ even at $\bar{x} = 0$. The discussion of the directionally Lipschitz case culminates with Corollary 6.1, which establishes that the Clarke subdifferential can indeed be approximated by convex hulls of gradients. On the more negative side, Example 7.2 shows that this approximation can fail badly in the general Lipschitz case. Motivated by these results, Burke and Lin have recently extended the GS convergence theory to the directionally Lipschitz case [4, 34]. However, it would seem difficult to extend these results to the more general non-Lipschitz case.

Suppose $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ is defined by

$$f(X) = \max\{\mathrm{Re}\lambda : \det(\lambda I - X) = 0\},$$

the spectral abscissa (maximum of the real parts of the eigenvalues) of X. Assume that $\bar{X}$ has the property that its only eigenvalues whose real parts coincide with f make up a zero eigenvalue with multiplicity q associated with a single Jordan block (the generic case). In this case the results in [5] tell us that the horizon cone of f is pointed at $\bar{X}$ if and only if the multiplicity $q \leq 2$; on the other hand f is locally Lipschitz at X if and only if $q = 1$. In the light of the previous paragraph, one might expect much greater practical success in applying GS to minimize the spectral abscissa of a parameterized matrix if the optimal multiplicities are limited to 1 or 2. However, this seems not to be the case. The results reported in [7] for unconstrained spectral abscissa minimization, as well as results for applying the algorithm of [12] (see Sect. 6.5.2 below) for constrained nonsmooth, nonconvex optimization to problems with spectral radius objective and constraints, as reported in [15, Section 4.2 and Appendix A.1], do not show any marked deterioration as the optimal multiplicities increase from 2 or 3, although certainly the problems are much more challenging for larger multiplicities. We view understanding the rather remarkably good behavior of the GS algorithm on such examples as a potentially rewarding, though certainly challenging, line of investigation.

6.4 Enhancements

As explained above, the statement of Algorithm GS differs in several ways from the algorithms stated in [7, 8], and [28]. Other variants of the strategy have also been proposed in recent years, in some cases to pose new solutions to theoretical obstacles (such as the need, in theory, to check for differentiability of f at each new iterate), and in others to enhance the practical performance of the approach. In this section, we discuss a few of these enhancements.

6.4.1 Restricting the Line Search to Within a Trust Region

Since the gradient information about f is obtained only within the ball $\bar{B}(\boldsymbol{x}^k; \epsilon_k)$ for all $k \in \mathbb{N}$, and since one might expect that smaller steps should be made when the sampling radius is small, an argument can be made that the algorithm might benefit by restricting the line search to within the ball $\bar{B}(\boldsymbol{x}^k; \epsilon_k)$ for all $k \in \mathbb{N}$. In [28, Section 4.2], such a variant is proposed where in place of $-\boldsymbol{g}^k$ the search direction is set as $-\epsilon_k \boldsymbol{g}^k / \|\boldsymbol{g}^k\|$. With minor corresponding changes to conditions (6.3) and (6.4), all of the theoretical convergence guarantees of the algorithm are maintained. Such a variant with the trust region radius defined as a positive multiple of the sampling radius ϵ_k for all $k \in \mathbb{N}$ would have similar properties. This variant might perform well in practice, especially in situations when otherwise setting the search direction as $-\boldsymbol{g}^k$ would lead to significant effort being spent in the line search.

6.4.2 Avoiding the Differentiability Check

The largest distraction from the fundamentally simple nature of Algorithm GS is the procedure for choosing a perturbed subsequent iterate if f is not differentiable at $\boldsymbol{x}^k - t_k \boldsymbol{g}^k$; see Steps 11–14. This procedure is necessary for the algorithm to be well defined since, to ensure that $-\boldsymbol{g}^k$ is a descent direction for all $k \in \mathbb{N}$, the algorithm relies on the existence of and ability to compute $-\nabla f(\boldsymbol{x}^k)$ for all $k \in \mathbb{N}$. One might hope that this procedure, while necessary for theoretical convergence guarantees, could be ignored in practice. However, due to the deterministic nature of the line search, situations exist in which landing on a point of nondifferentiability of f occurs with positive probability.

Example 6.1 Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(w, z) = \max\{0.5w^2 + 0.1z, w + 0.1z + 1, -w + 0.1z + 1, -0.05z - 50\};$$

see Fig. 6.2. As shown by Helou, Santos, and Simões in [24], if Algorithm GS is initialized at $\boldsymbol{x}^0 = (w_0, z_0)$ chosen anywhere in the unit ball centered at (10, 10), then there is a positive probability that the function f will not be differentiable at $\boldsymbol{x}^0 - t_0\boldsymbol{g}^0$. This can be explained as follows. At any point in the unit ball centered at (10, 10), the function f is continuously differentiable and $\nabla f(\boldsymbol{x}^0) = (w_0, 0.1)$. Moreover, there is a positive probability that the sampled points obtained at this first iteration will yield $\boldsymbol{g}^0 = \nabla f(\boldsymbol{x}^0)$. Therefore, given a reasonable value for the parameter β that appears in (6.3) (e.g., $\beta = 10^{-4}$), the sufficient decrease of the function value is attained with $t_0 = 1$. This guarantees that the function f will not be differentiable at the next iterate, since the first coordinate of $\boldsymbol{x}^1 = (w_1, z_1)$ will be zero.

The authors of [24] propose two strategies to avoid the issues highlighted by this example. The first is that, rather than perturb the iterate after the line search, one could perturb the search direction before the line search. It is shown that if the random perturbation of the search direction is sufficiently small such that, for one thing, the resulting direction is still one of descent for f, then f will be differentiable at all iterates with probability one. Their second proposed strategy involves the use of a nonmonotone line search. In particular, it is shown that if a strictly positive value Δ_k is added on the right-hand side of the sufficient decrease condition in (6.3) such

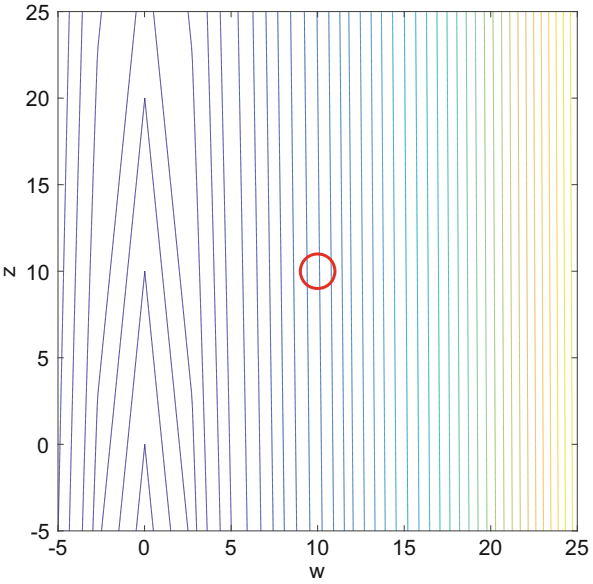

Fig. 6.2 Contours of a function illustrating the necessity of the differentiability check in Algorithm GS. Initialized uniformly within the illustrated ball, there is a positive probability that $\boldsymbol{x}^0 - t_0\boldsymbol{g}^0 \notin D$

that $\{\Delta_k\}$ is summable, then one can remove $\nabla f(\boldsymbol{x}^k)$ from the set $\mathcal{G}_k$ for all $k \in \mathbb{N}$ and maintain convergence guarantees even when f is *not* differentiable at $\{\boldsymbol{x}^k\}$. This can be shown by noticing that, due to the positive term Δ_k, the line search procedure continues to be well defined even if $-\boldsymbol{g}^k$ is not a descent direction for f at $\boldsymbol{x}^k$ (which may happen since $\nabla f(\boldsymbol{x}^k)$ is no longer involved in the computation of $\boldsymbol{g}^k$). However, the summability of $\{\Delta_k\}$ implies that the possible increases in f will be finite and, when the sampled points are good enough to describe the local behavior of f around the current iterate, the function value will necessarily decrease if the method has not reached (approximate) stationarity. Overall, the sufficient reductions in f achieved in certain iterations will ultimately exceed any increases.

Another proposal for avoiding the need to have f differentiable at $\boldsymbol{x}^k$ for all $k \in \mathbb{N}$ is given in [28, Section 4.3], wherein a technique is proposed for using a limited line search, potentially causing the algorithm to take null steps in some iterations. In fact, this idea of using a limited line search can also be used to avoid the need to sample a new set of $m \geq n+1$ gradients in each iteration, as we discuss next.

6.4.3 Adaptive Sampling

As previously mentioned, the main weakness of Algorithm GS is the cost of computing $m \geq n+1$ gradients in every iteration and solving a corresponding QP. Indeed, in many situations in practice, the sampling of such a large number of gradients in each iteration can lead to a significant amount of wasted computational effort; e.g., this is the case if a productive step would have been produced if fewer gradients were computed and used in the step computation. To take advantage of such situations, one can instead sample *adaptively*, attempting to search along directions computed using fewer gradients and proceeding as long as a sufficient reduction is attained.

In [13], Curtis and Que show how such an adaptive sampling strategy can be employed so that the convergence guarantees of Algorithm GS are maintained while only a constant number (independent of n) of gradients need to be sampled in each iteration. A key aspect that allows one to maintain these guarantees is the employment of a limited line search, as first proposed in [28], potentially leading to a null step when fewer than $n+1$ gradients are currently in hand and when the line search is not successful after a prescribed finite number of function evaluations. See also [14] for further development of these ideas, where it is shown that one might not need to sample *any* gradients as long as a sufficient reduction is attained.

The work in [13] also introduces the idea that, when adaptive sampling is employed, the algorithm can exploit a practical feature commonly used in bundle methods. This idea relates to warm-starting the algorithm for solving the QP subproblems. In particular, suppose that one has solved the primal-dual pair of QPs in (6.5) for some $m \in \mathbb{N}$ to obtain $(z_k, \boldsymbol{d}^k, \boldsymbol{\lambda}^k)$, yielding $\boldsymbol{g}^k = -\boldsymbol{d}^k$. If one were to

subsequently aim to solve the pair of QPs corresponding to the augmented matrix of gradients

$$\overline{G}_k = \left[G_k \ \nabla f(\boldsymbol{x}^{k,m+1}) \ \cdots \ \nabla f(\boldsymbol{x}^{k,m+p}) \right],$$

then one obtains a viable feasible starting point for the latter QP in (6.5) by augmenting the vector $\boldsymbol{\lambda}^k$ with p zeros. This can be exploited, e.g., in an active-set method for solving this QP; see [27].

As a further practical enhancement, the work in [13, 14] also proposes the natural idea that, after moving to a new iterate $\boldsymbol{x}^k$, gradients computed in previous iterations can be "reused" if they correspond to points that lie within $\bar{B}(\boldsymbol{x}^k; \epsilon_k)$. This may further reduce the number of sampled points needed in practice.

6.4.4 Second Order-Type Variants

The solution vector $\boldsymbol{d}^k$ of the QP in (6.5) can be viewed as the minimizer of the model of f at $\boldsymbol{x}^k$ given by

$$q_k(\boldsymbol{d}) = f(\boldsymbol{x}^k) + \max_{\boldsymbol{g} \in \mathcal{G}_k} \boldsymbol{g}^T \boldsymbol{d} + \tfrac{1}{2} \boldsymbol{d}^T H_k \boldsymbol{d}$$

with $H_k = I$, the identity matrix. As in other second order-type methods for nonlinear optimization, one might also consider algorithm variants where H_k is set to some other symmetric positive definite matrix. Ideas of this type have been explored in the literature. For example, in [13], two techniques are proposed: one in which H_k is set using a quasi-Newton updating strategy and one in which the matrix is set in an attempt to ensure that the model q_k represents an upper bounding model for f. The idea of employing a quasi-Newton approach, inspired by the success of quasi-Newton methods in practice for NSO (see [33]), has also been explored further in [14, 16].

Another approach, motivated by the encouraging results obtained when employing spectral gradient methods to solve smooth [3, 21] and nonsmooth [11] optimization problems, has been to employ a Barzilai-Borwein (BB) strategy for computing initial stepsizes in a GS approach; see [35] and the background in [2, 38, 39]. Using a BB strategy can be viewed as choosing $H_k = \alpha_k I$ for all $k \in \mathbb{N}$ where the scalar α_k is set according to iterate and gradient displacements in the latest iteration.

In all of these second order-type approaches, one is able to maintain convergence guarantees of the algorithm as long as the procedure for setting the matrix H_k is safeguarded during the optimization process. For example, one way to maintain guarantees is to restrict each H_k to the set of symmetric matrices whose eigenvalues are contained within a fixed positive interval. One might also attempt to exploit the self-correcting properties of BFGS updating; see [16].

6.5 Extensions

In this section, we discuss extensions to the GS methodology to solve classes of problems beyond unconstrained optimization on $\mathbb{R}^n$.

6.5.1 Riemannian GS for Optimization on Manifolds

Hosseini and Uschmajew in [26] have extended the GS methodology for minimizing a locally Lipschitz f over a set $\mathcal{M}$, where $\mathcal{M}$ is a complete Riemannian manifold of dimension n. The main idea of this extension is to employ the convex hull of gradients from tangent spaces at randomly sampled points *transported* to the tangent space of the current iterate. In this manner, the algorithm can be characterized as a generalization of the Riemannian steepest descent method just as GS is a generalization of traditional steepest descent. Assuming that the vector transport satisfies certain assumptions, including a *locking condition*, the algorithm attains convergence guarantees on par with those for Algorithm GS.

6.5.2 SQP-GS for Constrained Optimization

Curtis and Overton in [12] proposed a combination sequential quadratic programming (SQP) and GS method for solving constrained optimization problems involving potentially nonsmooth constraint functions, i.e.,

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{c}(\boldsymbol{x}) \leq \boldsymbol{0}, \\ & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{6.6}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{c} : \mathbb{R}^n \to \mathbb{R}^{n_c}$ are locally Lipschitz. A key aspect of the proposed SQP-GS approach is that sampled points for the objective and each individual constraint function are generated independently. With this important feature, it is shown that the algorithm, which follows a penalty-SQP strategy (e.g., see [20]), attains convergence guarantees for minimizing an exact penalty function that are similar to those in Sect. 6.3.1. Moreover, with the algorithm's penalty parameter updating strategy, it is shown that either the penalty function is driven to $-\infty$, the penalty parameter settles at a finite value and any limit point will be feasible for the constraints and stationary for the penalty function, or the penalty parameter will be driven to zero and any limit point of the algorithm will be stationary for a constraint violation measure. As for other exact penalty function methods for nonlinear optimization, one can translate between guarantees

for minimizing the exact penalty function and solving the constrained problem (6.6); in particular, if the problem is *calm* [10, 40], then at any local minimizer $\boldsymbol{x}_*$ of (6.6) there exists a threshold for the penalty parameter beyond which $\boldsymbol{x}_*$ will be a local minimizer of the penalty function.

Tang, Liu, Jian, and Li have also proposed in [42] a *feasible* variant of the SQP-GS method in which the iterates are forced to remain feasible for the constraints and the objective function is monotonically decreasing throughout the optimization process. This opens the door to employing a two-phase approach common for solving some optimization problems, where phase 1 is responsible for attaining a feasible point and phase 2 seeks optimality while maintaining feasibility.

6.5.3 Derivative-Free Optimization

Given its simple nature, GS has proved to be an attractive basis for the design of new algorithms even when gradient information cannot be computed explicitly. Indeed, there have been a few variants of *derivative-free* algorithms that have been inspired by GS.

The first algorithm for derivative-free optimization inspired by GS was proposed by Kiwiel in [29]. In short, in place of the gradients appearing in Algorithm GS, this approach employs Gupal's estimates of gradients of the Steklov averages of f. In this manner, function values only—specifically, $\mathcal{O}(mn)$ per iteration—are required for convergence guarantees. A less expensive *incremental* version is also proposed.

Another derivative-free variant of GS, proposed by Hare and Nutini in [23], is specifically designed for minimizing finite-max functions. This approach exploits knowledge about which of these functions are *almost active*—in terms of having value close to the objective function—at a particular point. In so doing, rather than attempt to approximate gradients at nearby points, as in done in [29], this approach only attempts to approximate gradients of almost active functions. The convergence guarantees proved for the algorithm are similar to those for GS methods, though the practical performance is improved by the algorithm's tailored gradient approximation strategy.

Finally, we mention the *manifold sampling* algorithm, proposed by Larson, Menickelly, and Wild in [30], for solving nonconvex problems where the objective function is the L_1-norm of a smooth vector function $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^r$. While this approach does not employ a straightforward GS methodology in that it does not randomly sample points, it does employ a GS-type approach in the way that the gradient of a model of the objective function is constructed by solving a QP of the type in (6.5). Random sampling can be avoided in this construction since the algorithm can exploit knowledge of the signs of the elements of $\boldsymbol{F}(\boldsymbol{x})$ at any $\boldsymbol{x} \in \mathbb{R}^n$ along with knowledge of $\partial \|\cdot\|_1$.

6.6 Applications

We mentioned in the introduction that the original GS paper [7] reported results for spectral abscissa optimization problems that had not been solved previously. The second GS paper [8] reported results for many more applications that again had not been solved previously: these included Chebyshev approximation by exponential sums, eigenvalue product minimization for symmetric matrices, spectral and pseudospectral abscissa minimization, maximization of the "distance to instability", and fixed order controller design by static output feedback.

Subsequently, the GS algorithm played a key role in the HANSO (hybrid algorithm for nonsmooth optimization)[3] and HIFOO (H-infinity fixed order optimization)[4] toolboxes. The former is a stand-alone code for unconstrained NSO while the latter is a more specialized code used for the design of low order controllers for linear dynamical systems with input and output, computing fixed order controllers by optimizing stability measures that are generally nonsmooth at local minimizers [9]. HIFOO calls HANSO to carry out the optimization. The use of "hybrid" in the expansion of the HANSO acronym indicated that, from its inception, HANSO combined the use of both a quasi-Newton algorithm (BFGS) and GS. The quasi-Newton method was used in an initial phase which, rather surprisingly, typically worked very effectively even in the presence of nonsmoothness, very often providing a fast way to approximate a local minimizer. This was followed by a GS phase to refine the approximation, typically verifying a loose measure of local optimality. The HIFOO toolbox has been used successfully in a wide variety of applications, including synchronization of heterogeneous multi-agent systems and networks, design of motorized gimbals that stabilize an angular motion of an optical payload around an axis, flight control via static output feedback, robust observer-based fault detection and isolation, influence of tire damping on control of quarter-car suspensions, flexible aircraft lateral flight dynamic control, optimal control of aircraft with a blended wing body, vibration control of a fluid/plate system, controller design of a nose landing gear steering system, bilateral teleoperation for minimally invasive surgery, design of an aircraft controller for improved gust alleviation and passenger comfort, robust controller design for a proton exchange membrane fuel cell system, design of power systems controllers, and design of winding systems for elastic web materials—for a full list of references, see [15].

The successful use of BFGS in HANSO and HIFOO led to papers on the use of quasi-Newton methods in the nonsmooth context, both for unconstrained [33] and constrained [15] optimization. The latter paper introduced a new BFGS-SQP method for nonsmooth constrained optimization and compared it with the SQP-GS method discussed in Sect. 6.5.2 on a suite of challenging static output feedback

[3] www.cs.nyu.edu/overton/software/hanso/.

[4] www.cs.nyu.edu/overton/software/hifoo/.

controller design problems, half of them non-Lipschitz (spectral radius minimization) and half of them locally Lipschitz (pseudospectral radius minimization). It was found that although the BFGS-SQP method was much faster than SQP-GS, nonetheless, if the latter method based on GS was allowed sufficient running time, it frequently found better approximate solutions than the former method based on BFGS in a well defined sense, evaluated using "relative minimization profiles". Interestingly, this was particularly pronounced on the non-Lipschitz problems, despite the fact that the GS convergence theory does not extend to this domain. See Sect. 6.3.3 for further discussion of this issue.

Finally, we mention an interesting application of GS to robot path planning [43]. This work is based on the observation that shortest paths generated through gradient descent on a value function have a tendency to chatter and/or require an unreasonable number of steps to synthesize. The authors demonstrate that the GS algorithm can largely alleviate this problem. For systems subject to state uncertainty whose state estimate is tracked using a particle filter, they proposed the GS with particle filter (GSPF) algorithm, which uses the particles as the locations in which to sample the gradient. At each step, the GSPF efficiently finds a consensus direction suitable for all particles or identifies the type of stationary point on which it is stuck. If the stationary point is a minimum, the system has reached its goal (to within the limits of the state uncertainty) and the algorithm terminates; otherwise, the authors propose two approaches to find a suitable descent direction. They illustrated the effectiveness of the GSPF on several examples using well known robot simulation environments. This work was recently extended and modified in [19], where the practical effectiveness of both the GSPF algorithm and the new modification was demonstrated on a Segway robotic mobility platform.

6.7 Conclusion and Future Directions

Gradient sampling is a conceptually straightforward approximate steepest descent method. With a solid convergence theory, the method has blossomed into a powerful methodology for solving nonsmooth minimization problems. The theme of our treatment of GS in this work has been to emphasize the fact that, even though the basic algorithm has been enhanced and extended in various ways, the foundation of the approach is fundamentally simple in nature.

We have also corrected an oversight in the original GS theory (i.e., that the convergence results depend on assuming that the set of points over which the Lipschitz function f is continuously differentiable has full measure, although we do not have a counterexample to convergence of GS in the absence of this assumption). At the same time we have loosened the requirements of the algorithm (showing that f need only be differentiable at the iterates and sampled points). An open question that still remains is whether one can extend the GS theory to broader function classes, such as the case where f is assumed to be semi-algebraic but not necessarily locally Lipschitz or directionally Lipschitz.

Opportunities for extending GS theory for broader function classes may include connecting the algorithm to other randomized/stochastic optimization methods. For example, one might view the algorithm as a stochastic-gradient-like method applied to a smoothed objective. (A similar philosophy underlies the analysis by Nesterov and Spokoiny in [37]). More precisely, given a locally Lipschitz objective f, consider a smoothing f_ϵ whose value at any point $\boldsymbol{x}$ is given by the mean value of f over the ball $\bar{B}(\boldsymbol{x};\epsilon)$. The GS algorithm uses gradients of f at uniformly distributed random points in this ball. Notice that each such gradient can also be viewed as a stochastic gradient for the smoothing f_ϵ in the sense that its expectation is the gradient of f_ϵ at $\boldsymbol{x}$. Thus, one might hope to prove convergence results for a GS algorithm (with predetermined stepsizes rather than line searches) that parallel convergence theory for stochastic gradient methods. Recent work by Davis, Drusvyatskiy, Kakade and Lee [18] gives convergence results for stochastic *subgradient* methods on a broad class of problems.

Another potentially interesting connection is with the work of Davis and Drusvyatskiy [17] on stochastic model-based optimization. Consider a GS variant that successively minimizes stochastic models of the objective function f, where we assume for simplicity that f is a globally Lipschitz convex function. In this variant, rather than moving along the direction $-\boldsymbol{g}^k$, consider instead the construction of a cutting plane approximation of f from its affine minorants at the current iterate $\boldsymbol{x}^k$ and the sampled points $\{\boldsymbol{x}^{k,i}\}$, augmented by the proximal term $\beta_k\|\boldsymbol{x}-\boldsymbol{x}^k\|^2$, where $\{\beta_k\}$ is a predetermined sequence. Suppose that the next iterate is chosen as the minimizer of this model; for a given k and with $\beta_k = 1$, by Eq. (6.5), this scheme and GS produce similar descent directions as the sampling radius tends to zero. It follows from the results of [17] that the expected norm of the gradient of the Moreau envelope [17, p. 6] is reduced below ϵ in $\mathcal{O}(\epsilon^{-4})$ iterations. In fact, the assumptions on f in [17] are substantially weaker than convexity, and do not require any property of the set on which f is continuously differentiable.

Connecting the convergence theory for GS to stochastic methods as suggested in the previous two paragraphs could be enlightening. However, while stochastic methods are often designed for settings in which it is intractable to compute function values exactly—a feature reflected in the fact that the analyses for such methods are based on using predetermined stepsize sequences—the GS methodology has so far been motivated by problems for which functions and gradients are tractable to compute. In such settings, the line search in Algorithm GS is an ingredient that is crucial to its practical success.

Acknowledgements The authors would like to acknowledge the following financial support. J.V. Burke was supported in part by the U.S. National Science Foundation grant DMS-1514559. F.E. Curtis was supported in part by the U.S. Department of Energy grant DE-SC0010615. A.S. Lewis was supported in part by the U.S. National Science Foundation grant DMS-1613996. M.L. Overton was supported in part by the U.S. National Science Foundation grant DMS-1620083. L.E.A. Simões was supported in part by the São Paulo Research Foundation (FAPESP), Brazil, under grants 2016/22989-2 and 2017/07265-0.

Appendix 1

This appendix is devoted to justifying the requirement that D, the set of points on which the locally Lipschitz function f is continuously differentiable, must be an open full-measure subset of $\mathbb{R}^n$, instead of the original assumption in [8] that D should be an open and dense set in $\mathbb{R}^n$.

There are two ways in which the analyses in [8, 28] actually depend on D having full measure:

1. The most obvious is that both papers require that the points sampled in each iteration should lie in D, and a statement is made in both papers that this occurs with probability one, but this is not the case if D is assumed only to be an open dense subset of $\mathbb{R}^n$. However, as already noted earlier and justified in Appendix 2, this requirement can be relaxed, as in Algorithm GS given in Sect. 6.2, to require only that f be differentiable at the sampled points.
2. The set D must have full measure for Property 6.1, stated below, to hold. The proofs in [8, 28] depend critically on this property, which follows from [6, Eq. (1.2)] (where it was stated without proof). For completeness we give a proof here, followed by an example that demonstrates the necessity of the full measure assumption.

Property 6.1 Assume that D has full measure and let

$$G_\epsilon(\boldsymbol{x}) := \operatorname{cl}\operatorname{conv}\nabla f\left(\bar{B}(\boldsymbol{x};\epsilon)\cap D\right).$$

For all $\epsilon > 0$ and all $\boldsymbol{x} \in \mathbb{R}^n$, one has $\partial f(\boldsymbol{x}) \subseteq G_\epsilon(\boldsymbol{x})$, where ∂f is the Clarke subdifferential set presented in Definition 1.8.

Proof of Property 6.1 Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \partial f(\boldsymbol{x})$. We have from [10, Theorem 2.5.1] that Theorem 1.2 can be stated in a more general manner. Indeed, for any set S with zero measure, and considering Ω_f to be the set of points at which f fails to be differentiable, the following holds:

$$\partial f(\boldsymbol{x}) = \operatorname{conv}\left\{\lim_j \nabla f(\boldsymbol{y}^j) : \boldsymbol{y}^j \to \boldsymbol{x} \text{ where } \boldsymbol{y}^j \notin S\cup\Omega_f \text{ for all } j\in\mathbb{N}\right\}.$$

In particular, since D has full measure and f is differentiable on D, it follows that

$$\partial f(\boldsymbol{x}) = \operatorname{conv}\left\{\lim_j \nabla f(\boldsymbol{y}^j) : \boldsymbol{y}^j \to \boldsymbol{x} \text{ with } \boldsymbol{y}^j \in D \text{ for all } j\in\mathbb{N}\right\}.$$

Considering this last relation and Carathéodory's theorem, it follows that $\boldsymbol{v} \in \operatorname{conv}\left\{\boldsymbol{\xi}^1,\ldots,\boldsymbol{\xi}^{n+1}\right\}$, where, for all $i\in\{1,\ldots,n+1\}$, one has $\boldsymbol{\xi}^i = \lim_j \nabla f(\boldsymbol{y}^{j,i})$

for some sequence $\{\boldsymbol{y}^{j,i}\}_{j\in\mathbb{N}} \subset D$ converging to x. Hence, there must exist a sufficiently large $j_i \in \mathbb{N}$ such that, for all $j \geq j_i$, one obtains

$$\boldsymbol{y}^{j,i} \in \bar{B}(\boldsymbol{x};\epsilon) \cap D \implies \nabla f(\boldsymbol{y}^{j,i}) \in \nabla f\left(\bar{B}(\boldsymbol{x};\epsilon) \cap D\right) \subseteq \operatorname{conv} \nabla f\left(\bar{B}(\boldsymbol{x};\epsilon) \cap D\right).$$

Recalling that $G_\epsilon(\boldsymbol{x})$ is the closure of $\operatorname{conv} \nabla f\left(\bar{B}(\boldsymbol{x};\epsilon) \cap D\right)$, it follows that $\boldsymbol{\xi}^i \in G_\epsilon(\boldsymbol{x})$ for all $i \in \{1,\ldots,n+1\}$. Moreover, since $G_\epsilon(\boldsymbol{x})$ is convex, we have $\boldsymbol{v} \in G_\epsilon(\boldsymbol{x})$. The result follows since $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \partial f(\boldsymbol{x})$ were arbitrarily chosen. □

With the assumption that D has full measure, Property 6.1 holds and hence the proofs of the results in [8, 28] are all valid. In particular, the proof of (ii) in [28, Lemma 3.2], which borrows from [8, Lemma 3.2], depends on Property 6.1. See also [8, the top of p. 762].

The following example shows that Property 6.1 might not hold if D is assumed only to be an open dense set, not necessarily of full measure.

Example 6.2 Let $\delta \in (0,1)$ and $\{q_k\}_{k\in\mathbb{N}}$ be the enumeration of the rational numbers in $(0,1)$. Define

$$D := \bigcup_{k=1}^{\infty} \mathcal{Q}_k, \text{ where } \mathcal{Q}_k := \left(q_k - \frac{\delta}{2^{k+1}}, q_k + \frac{\delta}{2^{k+1}}\right).$$

Clearly, its Lebesgue measure satisfies $0 < \lambda(D) \leq \delta$. Moreover, the set D is an open dense subset of $[0,1]$. Now, let $i_D : [0,1] \to \mathbb{R}$ be the indicator function of the set D,

$$i_D(x) = \begin{cases} 1, \text{ if } x \in D, \\ 0, \text{ if } x \notin D. \end{cases}$$

Then, considering the Lebesgue integral, we define the function $f : [0,1] \to \mathbb{R}$,

$$f(x) = \int_{[0,x]} i_D d\lambda.$$

Let us prove that f is a Lipschitz continuous function on $(0,1)$. To see this, note that given any $a, b \in (0,1)$ with $b > a$, it follows that

$$|f(b) - f(a)| = \left|\int_{[0,b]} i_D d\lambda - \int_{[0,a]} i_D d\lambda\right| = \left|\int_{(a,b]} i_D d\lambda\right| \leq \int_{(a,b]} 1 d\lambda = b - a,$$

which ensures that f is a Lipschitz continuous function on $(0,1)$. Consequently, the Clarke subdifferential set of f at any point in $(0,1)$ is well defined.

Moreover, we claim that, for all $k \in \mathbb{N}$, f is continuously differentiable at any point $q \in \mathcal{Q}_k$ and the following holds

$$f'(q) = i_D(q) = 1. \tag{6.7}$$

Indeed, given any $q \in \mathcal{Q}_k$, we have

$$f(q+t) - f(q) = \int_{[0,q+t]} i_D d\lambda - \int_{[0,q]} i_D d\lambda = \int_{(q,q+t]} i_D d\lambda, \text{ for } t > 0.$$

Since $\mathcal{Q}_k$ is an open set, we can find $\overline{t} > 0$ such that $[q, q+t] \subset \mathcal{Q}_k \subset D$, for all $t \leq \overline{t}$. Hence, given any $t \in (0, \overline{t}]$, it follows that

$$f(q+t) - f(q) = \int_{(q,q+t]} 1 d\lambda = t \implies \lim_{t \downarrow 0} \frac{f(q+t) - f(q)}{t} = 1 = i_D(q).$$

The same reasoning can be used to see that the left derivative of f at q exists and it is equal to $i_D(q)$. Consequently, we have $f'(q) = i_D(q) = 1$ for all $q \in \mathcal{Q}_k$, which yields that f is continuously differentiable on D.

By the Lebesgue differentiation theorem, we know that $f'(x) = i_D(x)$ almost everywhere. Since the set $[0, 1] \setminus D$ does not have measure zero, this means that there must exist $z \in [0, 1] \setminus D$ such that $f'(z) = i_D(z) = 0$. Defining $\epsilon := \min\{z, 1 - z\}/2$, we see, by (6.7), that the set

$$G_\epsilon(z) := \operatorname{cl} \operatorname{conv} \nabla f([z - \epsilon, z + \epsilon] \cap D)$$

is a singleton $G_\epsilon(z) = \{1\}$. However, since $f'(z) = 0$, it follows that $0 \in \partial f(z)$, which implies $\partial f(z) \not\subset G_\epsilon(z)$.

Note that it is stated on [8, p. 754] and [28, p. 381] that the following holds: for all $0 \leq \epsilon_1 < \epsilon_2$ and all $\boldsymbol{x} \in \mathbb{R}^n$, one has $\bar{\partial}_{\epsilon_1} f(\boldsymbol{x}) \subseteq G_{\epsilon_2}(\boldsymbol{x})$. Property 6.1 is a special case of this statement with $\epsilon_1 = 0$, and hence this statement too holds only under the full measure assumption.

Finally, it is worth mentioning that in practice, the full measure assumption on D usually holds. In particular, whenever a real-valued function is semi-algebraic (or, more generally, "tame")—in other words, for all practical purposes virtually always—it is continuously differentiable on an open set of full measure. Hence, the original proofs hold in such contexts.

Appendix 2

In this appendix, we summarize why it is not necessary that the iterates and sampled points of the algorithm lie in the set D in which f is continuously differentiable, and that rather it is sufficient to ensure that f is differentiable at these points, as in Algorithm GS. We do this by outlining how to modify the proofs in [28] to extend to this case.

1. That the gradients at the sampled points $\{\boldsymbol{x}^{k,j}\}$ exist follows with probability one from Rademacher's theorem, while the existence of the gradients at the iterates $\{\boldsymbol{x}^k\}$ is ensured by the statement of Algorithm GS. Notice that the proof of part (ii) of [28, Theorem 3.3] still holds in our setting with the statement that the components of the sampled points are "sampled independently and uniformly from $\bar{B}(\boldsymbol{x}^k;\epsilon) \cap D$" replaced with "sampled independently and uniformly from $\bar{B}(\boldsymbol{x}^k;\epsilon)$".
2. One needs to verify that f being differentiable at $\boldsymbol{x}^k$ is enough to ensure that the line search procedure presented in (6.3) terminates finitely. This is straightforward. Since $\nabla f(\boldsymbol{x}^k)$ exists, it follows that the directional derivative along any vector $\boldsymbol{d} \in \mathbb{R}^n \setminus \{0\}$ is given by $f'(\boldsymbol{x}^k;\boldsymbol{d}) = \nabla f(\boldsymbol{x}^k)^T\boldsymbol{d}$. Furthermore, since $-\nabla f(\boldsymbol{x}^k)^T\boldsymbol{g}^k \leq -\|\boldsymbol{g}^k\|^2$ (see [8, p. 756]), it follows, for any $\beta \in (0,1)$, that there exists $\bar{t} > 0$ such that

$$f(\boldsymbol{x}^k - t\boldsymbol{g}^k) < f(\boldsymbol{x}^k) - t\beta\|\boldsymbol{g}^k\|^2 \text{ for any } t \in (0,\bar{t}).$$

   This shows that the line search is well defined.
3. The only place where we actually need to modify the proof in [28] concerns item (ii) in Lemma 3.2, where it is stated that $\nabla f(\boldsymbol{x}^k) \in G_\epsilon(\bar{\boldsymbol{x}})$ (for a particular point $\bar{\boldsymbol{x}}$) because $\boldsymbol{x}^k \in \bar{B}(\bar{\boldsymbol{x}};\epsilon/3) \cap D$; the latter is not true if $\boldsymbol{x}^k \notin D$. However, using Property 6.1, we have

$$\nabla f(\boldsymbol{x}^k) \in \partial f(\boldsymbol{x}^k) \subset G_{\epsilon/3}(\boldsymbol{x}^k) \subset G_\epsilon(\bar{\boldsymbol{x}}) \text{ when } \boldsymbol{x}^k \in \bar{B}(\bar{\boldsymbol{x}};\epsilon/3),$$

   and therefore, $\nabla f(\boldsymbol{x}^k) \in G_\epsilon(\bar{\boldsymbol{x}})$ even when $\boldsymbol{x}^k \notin D$.

Finally, although it was convenient in Appendix 1 to state Property 1 in terms of D, it actually holds if D is replaced by any full measure set on which f is differentiable. Nonetheless, it is important to note that the proofs of the results in [8, 28] *do* require that f be *continuously* differentiable on D. This assumption is used in the proof of (i) in [28, Lemma 3.2].

References

1. Asl, A., Overton, M.L.: Analysis of the gradient method with an Armijo–Wolfe line search on a class of nonsmooth convex functions. Optim. Method Softw. (2017). https://doi.org/10.1080/10556788.2019.1673388

2. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA J. Numer. Anal. **8**(1), 141–148 (1988)
3. Birgin, E., Martinez, J., Raydan, M.: Spectral projected gradient methods: review and perspectives. J. Stat. Softw. **60**(3), 1–21 (2014)
4. Burke, J.V., Lin, Q.: The gradient sampling algorithm for directionally Lipschitzian functions (in preparation)
5. Burke, J.V., Overton, M.L.: Variational analysis of non-Lipschitz spectral functions. Math. Program. **90**(2, Ser. A), 317–351 (2001)
6. Burke, J.V., Lewis, A.S., Overton, M.L.: Approximating subdifferentials by random sampling of gradients. Math. Oper. Res. **27**(3), 567–584 (2002)
7. Burke, J.V., Lewis, A.S., Overton, M.L.: Two numerical methods for optimizing matrix stability. Linear Algebra Appl. **351/352**, 117–145 (2002)
8. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**(3), 751–779 (2005)
9. Burke, J.V., Henrion, D., Lewis, A.S., Overton, M.L.: HIFOO—a MATLAB package for fixed-order controller design and H_∞ optimization. In: Fifth IFAC Symposium on Robust Control Design, Toulouse (2006)
10. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983). Reprinted by SIAM, Philadelphia, 1990
11. Crema, A., Loreto, M., Raydan, M.: Spectral projected subgradient with a momentum term for the Lagrangean dual approach. Comput. Oper. Res. **34**(10), 3174–3186 (2007)
12. Curtis, F.E., Overton, M.L.: A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization. SIAM J. Optim. **22**(2), 474–500 (2012)
13. Curtis, F.E., Que, X.: An adaptive gradient sampling algorithm for nonsmooth optimization. Optim. Methods Softw. **28**(6), 1302–1324 (2013)
14. Curtis, F.E., Que, X.: A quasi-Newton algorithm for nonconvex, nonsmooth optimization with global convergence guarantees. Math. Program. Comput. **7**(4), 399–428 (2015)
15. Curtis, F.E., Mitchell, T., Overton, M.L.: A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles. Optim. Methods Softw. **32**(1), 148–181 (2017)
16. Curtis, F.E., Robinson, D.P., Zhou, B.: A self-correcting variable-metric algorithm framework for nonsmooth optimization. IMA J. Numer. Anal. (2019). https://doi.org/10.1093/imanum/drz008; https://academic.oup.com/imajna/advance-article/doi/10.1093/imanum/drz008/5369122?guestAccessKey=a7e5eee5-9ed6-4a95-9f6c-f305237d0849
17. Davis, D., Drusvyatskiy, D.: Stochastic model-based minimization of weakly convex functions. SIAM J. Optim. **29**(1), 207–239 (2019). https://doi.org/10.1137/18M1178244
18. Davis, D., Drusvyatskiy, D., Kakade, S., Lee, J.D.: Stochastic subgradient method converges on tame functions. Found. Comput. Math. (2019). https://doi.org/10.1007/s10208-018-09409-5
19. Estrada, A., Mitchell, I.M.: Control synthesis and classification for unicycle dynamics using the gradient and value sampling particle filters. In: Proceedings of the IFAC Conference on Analysis and Design of Hybrid Systems, pp. 108–114 (2018).
20. Fletcher, R.: Practical Methods of Optimization, 2nd edn. Wiley, New York (1987)
21. Fletcher, R.: On the Barzilai-Borwein method. In: Qi, L., Teo, K., Yang, X. (eds.) Optimization and Control with Applications, pp. 235–256. Springer, Boston (2005)
22. Guo, J., Lewis, A.S.: Nonsmooth variants of Powell's BFGS convergence theorem. SIAM J. Optim. **28**(2), 1301–1311 (2018). https://doi.org/10.1137/17M1121883
23. Hare, W., Nutini, J.: A derivative-free approximate gradient sampling algorithm for finite minimax problems. Comput. Optim. Appl. **56**(1), 1–38 (2013). https://doi.org/10.1007/s10589-013-9547-6
24. Helou, E.S., Santos, S.A., Simões, L.E.A.: On the differentiability check in gradient sampling methods. Optim. Methods Softw. **31**(5), 983–1007 (2016)
25. Helou, E.S., Santos, S.A., Simões, L.E.A.: On the local convergence analysis of the gradient sampling method for finite max-functions. J. Optim. Theory Appl. **175**(1), 137–157 (2017)

26. Hosseini, S., Uschmajew, A.: A Riemannian gradient sampling algorithm for nonsmooth optimization on manifolds. SIAM J. Optim. **27**(1), 173–189 (2017). https://doi.org/10.1137/16M1069298
27. Kiwiel, K.C.: A method for solving certain quadratic programming problems arising in nonsmooth optimization. IMA J. Numer. Anal. **6**(2), 137–152 (1986)
28. Kiwiel, K.C.: Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **18**(2), 379–388 (2007)
29. Kiwiel, K.C.: A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **20**(4), 1983–1994 (2010). https://doi.org/10.1137/090748408
30. Larson, J., Menickelly, M., Wild, S.M.: Manifold sampling for ℓ_1 nonconvex optimization. SIAM J. Optim. **26**(4), 2540–2563 (2016). https://doi.org/10.1137/15M1042097
31. Lemaréchal, C., Oustry, F., Sagastizábal, C.: The U-Lagrangian of a convex function. Trans. Am. Math. Soc. **352**(2), 711–729 (2000)
32. Lewis, A.S.: Active sets, nonsmoothness, and sensitivity. SIAM J. Optim. **13**(3), 702–725 (2002)
33. Lewis, A.S., Overton, M.L.: Nonsmooth optimization via quasi-Newton methods. Math. Program. **141**(1–2, Ser. A), 135–163 (2013). https://doi.org/10.1007/s10107-012-0514-2
34. Lin, Q.: Sparsity and nonconvex nonsmooth optimization. Ph.D. thesis, Department of Mathematics, University of Washington (2009)
35. Loreto, M., Aponte, H., Cores, D., Raydan, M.: Nonsmooth spectral gradient methods for unconstrained optimization. EURO J. Comput. Optim. **5**(4), 529–553 (2017)
36. Mifflin, R., Sagastizábal, C.: A VU-algorithm for convex minimization. Math. Program. **104**(2-3), 583–608 (2005)
37. Nesterov, Y., Spokoiny, V.: Random gradient-free minimization of convex functions. Found. Comput. Math. **17**(2), 527–566 (2017). https://doi.org/10.1007/s10208-015-9296-2
38. Raydan, M.: On the Barzilai and Borwein choice of steplength for the gradient method. IMA J. Numer. Anal. **13**(3), 321–326 (1993)
39. Raydan, M.: The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. SIAM J. Optim. **7**(1), 26–33 (1997)
40. Rockafellar, R.T.: Lagrange multipliers and subderivatives of optimal value functions in nonlinear programming. In: Sorensen, D.C., Wets, R.J.B. (eds.) Mathematical Programming Study, Mathematical Programming Studies, Chap. 3, pp. 28–66. North-Holland, Amsterdam (1982). http://www.springerlink.com/index/g03582565267714p.pdf
41. Rockafellar R.T., Wets, R.J.B.: Variational Analysis. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 317. Springer, Berlin (1998). https://doi.org/10.1007/978-3-642-02431-3
42. Tang, C.M., Liu, S., Jian, J.B., Li, J.L.: A feasible SQP-GS algorithm for nonconvex, nonsmooth constrained optimization. Numer. Algorithms **65**(1), 1–22 (2014). https://doi.org/10.1007/s11075-012-9692-5
43. Traft, N., Mitchell, I.M.: Improved action and path synthesis using gradient sampling. In: Proceedings of the IEEE Conference on Decision and Control, pp. 6016–6023 (2016)

Part II
Structure Exploiting Methods

Chapter 7
Local Search for Nonsmooth DC Optimization with DC Equality and Inequality Constraints

Alexander S. Strekalovsky

Abstract The chapter addresses the nonsmooth optimization problem with the objective function and equality and inequality constraints given by DC functions. First, the original problem is reduced to a problem without constraints by the exact penalization theory, so that the reduced (penalized) problem is also a DC minimization problem. Then, we develop a local search (LS) scheme which is based, first, on the linearization of the basic nonconvexity of the penalized problem and, second, on consecutive solutions of linearized (convex) problems. Convergence properties of the LS scheme are also investigated, which, in particular, yield that the sequence produced by LSM converges to a solution of the problem linearized at the limit point just. Moreover, the cluster point of the sequence is the KKT point for the original problem with the Lagrange multipliers provided by an auxiliary linearized problem. Finally, on the base of the developed theory several new stopping criteria are elaborated, which allow to transform the local search scheme into a local search algorithm.

7.1 Introduction

It is well-known in the optimization society that most of real-life problems are nonconvex. According to the renown authors (see, for instance, [39]), the challenges of the twenty-first century, such as hierarchical optimization problems and the search for equilibriums in competitions (conflict's situations and various games), generally lead to optimization problems with hidden nonconvexities. As a consequence, we can even say that nonconvexity itself is a challenge of the twenty-first century.

Furthermore, the specialists decided to split the optimization problems into two classes: convex and nonconvex problems. This is due to the fact that convex problems are computationally tractable under minimal computability assumptions

A. S. Strekalovsky (✉)
Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, Irkutsk, Russia
e-mail: strekal@icc.ru

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_7

[2, 9, 25, 38, 57]. It means that the computational efforts required to solve the problem to a given accuracy grow moderately with the dimension of the problem. In contrast, real-life nonconvex problems might have a huge number of local pitfalls and stationary vectors which are rather far from a global solution, and this makes numerical solution quite difficult [16, 17, 24, 26, 40, 56].

In addition, the classical optimization methods (conjugate gradients, Newton and quasi-Newton's methods, sequential quadratic programming, trust region and interior point methods etc.) turn out inoperative when it comes to finding a global solution to a nonconvex problem [2, 9, 14, 16, 17, 25, 26, 38, 56, 57], and too often cannot provide even a stationary (KKT) point [4, 35–37]. Besides, specialists in applied problems often do not take into account that it is incorrect to directly apply classical optimization methods [2, 9, 25, 38, 57] to nonconvex problems. Moreover, they interpret the numerical results only in the content aspect, i.e. in engineering, biology or economic etc. sense forgetting at the same time, that all classical optimization methods converge to a global solution only in convex problems [2, 38, 57].

On the other hand, we can observe the emergence of widespread and very popular approaches that completely neglect the classical optimization theory and methods [2, 7–10, 15, 25, 38, 43, 44, 57], and employ, for example, the B&B and cut's methodology. Recall that the latter algorithms usually suffer the so-called curse of dimensionality, when the volume of computations grows exponentially as the dimension of the problem increases.

Also, recall that application of the Lagrange duality theory and corresponding methods do not exactly provide a desirable result, since the theory is faulty and there exists a duality gap [2, 25, 38, 57] in nonconvex optimization with integer and mixed variables [2, 17, 26, 40, 45, 53], which can be represented in a continuous form [16, 17, 26, 31, 32, 45, 53, 56].

Thus, the current situation is not very promising or optimistic, even though now there is available a whole range of new powerful mathematical tools, such as convex analysis and variational analysis, [2, 7, 8, 10, 25, 42, 44, 57], the exact penalization theory [1–3, 5–8, 11–13, 15, 22, 25, 27, 28, 41, 59], developed by the great scientists, among which are R.T. Rockafellar, R.B. Wets, J.-B. Hiriart-Urruty, I.I. Eremin, W. Zangwill, J. Borwein. Nonetheless, the development of nonconvex optimization was continued due to the works, first of all, Tuy and Hiriart-Urruty [14, 23, 24, 56] and Toland [53–55]. In addition, the considerable contributions to nonconvex optimization by Le Thi Hoai An and Pham Dinh Tao, who develop the DC approach proposed by H. Tuy, are also widespread and well-known [29–34] and have many real-life applications.

We develop the part of nonconvex optimization founded on the *global optimality conditions* (GOCs) and numerical methods for nonconvex optimization related to the GOCs.

This chapter has three key objectives.

- First, we will develop a new version of the special local search method (LSM) for the general DC optimization problem with the objective function and equality

and inequality constraints given by DC functions, which are nonsmooth. This statement generalizes the smooth data of the problems considered in [18–21, 45–47, 49–56].

- Second, it seems to be necessary to combine the exact penalization theory with the technique of consecutive solutions of the convex problems linearized with respect to the basic nonconvexity of the original problem $(\mathcal{P})$.
- Finally, we will obtain convergency results that help us to construct new stopping criteria that would guarantee Stop at the iteration $k \geq 1$, when the iterate, for example, $\boldsymbol{x}^k$ (or $\boldsymbol{x}^{k+1}$) is close to a KKT point of the original problem $(\mathcal{P})$.

Thus, the chapter consists of the following sections: Sect. 7.2 presents the statement of the original problem $(\mathcal{P})$ while Sect. 7.3 introduces the penalty function $W(\boldsymbol{x})$, defined by (7.1), along with the penalized problem $(\mathcal{P}_\sigma)$. Furthermore, we prove that the goal function $\Theta_\sigma(\cdot)$ of $(\mathcal{P}_\sigma)$ is a DC function, whence it follows that the problem $(\mathcal{P}_\sigma)$ is a DC minimization problem over the convex and closed set $S \subset \mathbb{R}^n$. In Sect. 7.4, we develop a local search scheme based on the linearization of the basic nonconvexity of the penalized problem $(\mathcal{P}_\sigma)$, consider linearized problems $(\mathcal{P}_\sigma L(\boldsymbol{v}))$, $(\mathcal{P}L_k(\sigma))$, $(\mathcal{P}_k L)$, $(\mathcal{P}_k^* L)$, and $(\mathcal{A}\mathcal{P}_W L_k)$ and consecutive solutions of the linearized problems, as well as the penalty parameter update procedure.

Section 7.5 is devoted to investigation of some convergency properties of the proposed numerical Scheme 1 (Algorithm 7.1). In particular, in Proposition 7.1 we establish the convergence of the number sequence $\{\Theta_k(\boldsymbol{x}^k)\}$ of the values of the goal function $\Theta_k(\cdot)$ at the iterate $\boldsymbol{x}^k$, which immediately entails the convergence of the sequence $\{\boldsymbol{x}^k\}$ under the assumption of strong convexity of a part of the data. In addition, the convergence proof of the dual (to $\{\boldsymbol{x}^k\}$ in the Toland's sense) sequence $\{\boldsymbol{y}^k\}$ is also briefly presented. Employing these convergence results, we prove the principal result of the section, which states that the limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ produced by Scheme 1 turns out to be a solution to the "limit" linearized problem $(\mathcal{P}L_*)$.

In Sect. 7.6 we move on to numerical solution of the problem $(\mathcal{P}_\sigma)$ and study supplementary convergency properties of the cluster point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$, using some reduction theorem to replace linearized nonsmooth problems by other linearized convex problems with supplementary number parameters and only the inequality constraints. First, we perform this reduction for the "limit" linearized convex problem $(\mathcal{P}L_*)$ and obtain a rather unexpected result that the limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ is not only a solution to the linearized problem $(\mathcal{P}L_*)$, but a KKT-point to the original problem $(\mathcal{P})$. However, our most unexpected discovery is the fact that the Lagrange multipliers λ_i of $\boldsymbol{x}_*$ are completely defined by the corresponding Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ of the linearized problem (7.45), equivalent to $(\mathcal{P}L_*)$. Moreover, the latter Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ together with the limit penalty parameter $\sigma_* > 0$ satisfy some equations, so that all the parameters of computational process defined by Scheme 1 fulfill some shared constraints. Furthermore, repeating the similar reasonings for the approximate solution to the current linearized problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$, we get the same conclusions, and also some stopping criterions (7.64)–(7.69). Finally, we propose a modification

of Scheme 1 using the combination of the developed stopping criteria and advertise the properties of the point produced by Scheme 2.

7.2 Problem's Statement

In this chapter we consider the following optimization problem

$$\begin{cases} \text{minimize} & f_0(\boldsymbol{x}) := g_0(\boldsymbol{x}) - h_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) := g_i(\boldsymbol{x}) - h_i(\boldsymbol{x}) \le 0,\ i \in \mathcal{I} = \{1, \ldots, m\}, \\ & f_j(\boldsymbol{x}) := g_j(\boldsymbol{x}) - h_j(\boldsymbol{x}) = 0,\ j \in \mathcal{E} = \{m+1, \ldots, l\}, \\ & \boldsymbol{x} \in S, \end{cases} \tag{$\mathcal{P}$}$$

where the functions $g_i(\cdot), h_i(\cdot),\ i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$, are convex and finite on $\mathbb{R}^n$, so that the functions $f_i(\cdot)$ are the DC functions represented as a difference of two convex functions [25, 38, 42, 44, 57]. One can address a more general problem where $g_i, h_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are proper convex functions such that

$$\emptyset \neq S \subset \operatorname{int}(\operatorname{dom} g_i) \cap \operatorname{int}(\operatorname{dom} h_i), \quad i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}.$$

Below, we assume that the set $S \subset \mathbb{R}^n$ is convex and closed, while the functions $f_i(\cdot),\ i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$, are nonsmooth on $\mathbb{R}^n$, as well as the functions $g_i(\cdot),\ h_i(\cdot)$. Finally, suppose that the feasible set

$$\mathcal{F} := \{\boldsymbol{x} \in S : f_i(\boldsymbol{x}) \le 0,\ i \in \mathcal{I},\ f_j(\boldsymbol{x}) = 0,\ j \in \mathcal{E}\}$$

of the problem $(\mathcal{P})$ is nonempty, and the optimal value of the problem $(\mathcal{P})$ is finite:

$$\mathcal{V}(\mathcal{P}) := \inf(f_0, \mathcal{F}) := \inf_{\boldsymbol{x}}\{f_0(\boldsymbol{x}) |\ \boldsymbol{x} \in \mathcal{F}\} > -\infty.$$

7.3 The Penalized Problem

Introduce the penalty function as follows

$$W(\boldsymbol{x}) := \max\{0,\ f_1(\boldsymbol{x}),\ \ldots,\ f_m(\boldsymbol{x})\} + \sum_{j \in \mathcal{E}} |f_j(\boldsymbol{x})|, \tag{7.1}$$

and together with $(\mathcal{P})$ consider the penalized problem without equality and inequality constraints

$$\begin{cases} \text{minimize} & \Theta_\sigma(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \tag{$\mathcal{P}_\sigma$}$$

where $\sigma \geq 0$, and the function

$$\Theta_\sigma(\boldsymbol{x}) := f_0(\boldsymbol{x}) + \sigma W(\boldsymbol{x}) \tag{7.2}$$

is the cost function of the problem $(\mathcal{P}_\sigma)$.

Let us look at the relations between problems $(\mathcal{P})$ and $(\mathcal{P}_\sigma)$. As well-known, if $z \in Sol(\mathcal{P}_\sigma)$ and z is feasible in $(\mathcal{P})$, $z \in \mathcal{F}$, then $z \in Sol(\mathcal{P})$. Unfortunately, the inverse implication does not hold, in general.

Our approach is based on theory of exact penalization (EP) [1–3, 11–13, 22, 27, 28, 41, 58], which implies that for the penalty parameter $\sigma \geq 0$ there exists a threshold value $\sigma_* \geq 0$ such that for all $\sigma > \sigma_*$ problems $(\mathcal{P})$ and $(\mathcal{P}_\sigma)$ are equivalent in the sense that $Sol(\mathcal{P}) = Sol(\mathcal{P}_\sigma)$ and their optimal values coincide. On the other hand, the existence of the threshold value $\sigma_* \geq 0$ of the penalty parameter allows us to solve a single unconstrained problem instead of solving a sequence of unconstrained problems when the penalty parameter tends to infinity $(\sigma_k \uparrow \infty)$.

Since this chapter addresses only local search methods which provide, in general, only local solutions or even stationary (KKT) vectors, one can employ the well-known and widespread results on the existence of the threshold value $\sigma_* \geq 0$ of penalty parameters for a nonsmooth case [1–3, 11–13, 22, 27, 28, 41, 59]. When talking about this subject, we should highlight the results of Demyanov and his school [8], Kruger [27, 28], and Zaslavski [59], Burke [3], Clarke [7], Di Pillo et.al. [11–13] and many others as well.

Furthermore, let us demonstrate that the auxiliary problem $(\mathcal{P}_\sigma)$ is also a DC optimization problem, more precisely, the goal function $\Theta_\sigma(\boldsymbol{x})$ is a DC function [23, 24, 26, 56]. Indeed, since for $f(\cdot) = g(\cdot) - h(\cdot)$, where $g(\cdot)$, $h(\cdot)$ are convex functions, we have [23, 45, 56]

$$\begin{aligned} |f(\boldsymbol{x})| &= \max\{g(\boldsymbol{x}) - h(\boldsymbol{x}), h(\boldsymbol{x}) - g(\boldsymbol{x})\} \pm [g(\boldsymbol{x}) + h(\boldsymbol{x})] \\ &= 2\max\{g(\boldsymbol{x}), h(\boldsymbol{x})\} - [g(\boldsymbol{x}) + h(\boldsymbol{x})], \end{aligned}$$

and we can readily see that

$$\begin{aligned} \Theta_\sigma(\boldsymbol{x}) &\triangleq f_0(\boldsymbol{x}) + \sigma \max\{0,\ f_i(\boldsymbol{x}),\ i \in \mathcal{I}\} + \sum_{j \in \mathcal{E}} |f_j(\boldsymbol{x})| \\ &= G_\sigma(\boldsymbol{x}) - H_\sigma(\boldsymbol{x}), \end{aligned} \tag{7.3}$$

where

$$H_\sigma(\boldsymbol{x}) := h_0(\boldsymbol{x}) + \sigma[\sum_{i\in\mathcal{I}} h_i(\boldsymbol{x}) + \sum_{j\in\mathcal{E}}(g_j(\boldsymbol{x}) + h_j(\boldsymbol{x}))], \tag{7.4}$$

$$\begin{aligned} G_\sigma(\boldsymbol{x}) &:= \Theta_\sigma(\boldsymbol{x}) + H_\sigma(\boldsymbol{x}) \\ &= g_0(\boldsymbol{x}) + \sigma \max\{\sum_{p\in\mathcal{I}} h_p(\boldsymbol{x}), [g_i(\boldsymbol{x}) + \sum_{\substack{p\in\mathcal{I}\\ p\neq i}} h_p(\boldsymbol{x})],\ i \in \mathcal{I}\} \\ &\quad + 2\sigma \sum_{j\in\mathcal{E}} \max\{g_j(\boldsymbol{x}), h_j(\boldsymbol{x})\}. \end{aligned} \tag{7.5}$$

It can be readily seen that both functions $G_\sigma(\cdot)$ and $H_\sigma(\cdot)$ are convex [8, 9, 25], so that the goal function $\Theta_\sigma(\cdot)$ turns out to be a DC function. As a result, when we address the penalized problem $(\mathcal{P}_\sigma)$, we should be aware that we deal with a DC minimization problem [16, 17, 23, 26, 56].

7.4 Local Search Method

In this section for the penalized problem $(\mathcal{P}_\sigma)$ we develop a numerical scheme, some convergence features of which are in question. For this purpose, consider the linearized problem as follows ($\boldsymbol{v} \in \mathbb{R}^n$)

$$\begin{cases} \text{minimize} & \Phi_{\boldsymbol{v}}(\boldsymbol{x}) := G_\sigma(\boldsymbol{x}) - \langle H'_\sigma(\boldsymbol{v}), \boldsymbol{x}\rangle \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \tag{$\mathcal{P}_\sigma L(\boldsymbol{v})$}$$

where $H'_\sigma(\boldsymbol{v})$ is a subgradient (in the classical sense of the convex analysis) [7–9, 25, 38, 42, 44, 57] of the convex function $H_\sigma(\cdot)$ at the point $\boldsymbol{v} \in \mathbb{R}^n$, $H'_\sigma(\boldsymbol{v}) \in \partial_c H_\sigma(\boldsymbol{v})$. Due to (7.4) and in virtue of Moreau-Rockafellar theorem [25, 42], $H'_\sigma(\boldsymbol{v})$ has the following form

$$H'_\sigma(\boldsymbol{v}) = h'_0(\boldsymbol{v}) + \sigma\Big[\sum_{i\in\mathcal{I}} h'_i(\boldsymbol{v}) + \sum_{j\in\mathcal{E}}(g'_j(\boldsymbol{v}) + h'_j(\boldsymbol{v}))\Big], \tag{7.6}$$

where $h'_i(\boldsymbol{v}) \in \partial_c h_i(\boldsymbol{v}),\ i \in \{0\}\cup\mathcal{I}\cup\mathcal{E},\ g'_j(\boldsymbol{v}) \in \partial_c g_j(\boldsymbol{v}),\ j \in \mathcal{E}$, are subgradients of the corresponding functions.

It is worth noting that the problem $(\mathcal{P}_\sigma L(\boldsymbol{v}))$ is convex, and, therefore, can be solved by suitable classical optimization methods [2, 9, 38, 57] (for the case

when all the functions defining $G_\sigma(\cdot)$ are all smooth, see the next section) and optimization software packages (CPLEX, Xpress-MP, Gurobi, etc), since in the convex optimization any local ε-solution or τ-stationary (KKT) point turns out to be a global approximate solution [2, 8, 9, 25, 38, 57]. Besides, it is clear that the linearization is applied to the function $H_\sigma(\cdot)$ which accumulates all the nonconvexities of the problems $(\mathcal{P})$ and $(\mathcal{P}_\sigma)$.

Furthermore, let there be given a starting point $\boldsymbol{x}_0 \in S$, an initial value $\sigma_0 > 0$ of the penalty parameter σ, a current iterate $\boldsymbol{x}^k \in S$ and a current value $\sigma_k \geq \sigma_0$ of the penalty parameter $\sigma \geq 0$. Introduce the following notations

$$H_k(\cdot) := H_{\sigma_k}(\cdot), \;\; G_k(\cdot) := G_{\sigma_k}(\cdot), \;\; \boldsymbol{y} \in \partial_c H_k(\boldsymbol{x}^k), \tag{7.7}$$

where, on account of (7.6), $\boldsymbol{y}$ has the form

$$\begin{cases} \boldsymbol{y}(\sigma) & = \boldsymbol{y}(\sigma, \boldsymbol{x}^k) := h'_{0k} + \sigma[\sum\limits_{i\in\mathcal{I}} h'_{ik} + \sum\limits_{j\in\mathcal{E}} (g'_{jk} + h'_{jk})] \in \partial_c H_\sigma(\boldsymbol{x}^k), \\ h'_{ik} & := h'_i(\boldsymbol{x}^k) \in \partial_c h_i(\boldsymbol{x}^k), \quad i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}, \\ g'_{jk} & = g'_j(\boldsymbol{x}^k) \in \partial_c g_j(\boldsymbol{x}^k), \quad j \in \mathcal{E}. \end{cases} \tag{7.8}$$

Now consider the following linearized problem

$$\begin{cases} \text{minimize} & \Phi_\sigma(\boldsymbol{x}) := G_\sigma(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma), \boldsymbol{x} \rangle \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \tag{$\mathcal{P}L_k(\sigma)$}$$

which, due to $(\mathcal{P}_\sigma L(\boldsymbol{v}))$, is convex, and hence, can be solved by modern optimization methods and software [2, 8, 9, 25, 38, 57].

As a particular case of $(\mathcal{P}L_k(\sigma))$, consider the following linearized problem $(\mathcal{P}_k L) = (\mathcal{P}L_k(\sigma_k))$

$$\begin{cases} \text{minimize} & \Phi_k(\boldsymbol{x}) := G_k(\boldsymbol{x}) - \langle \boldsymbol{y}^k, \boldsymbol{x} \rangle \\ \text{subject to} & \boldsymbol{x} \in S. \end{cases} \tag{$\mathcal{P}_k L$}$$

The choice of $\boldsymbol{y}^k$ will be described below.

Since problems $(\mathcal{P}L_k(\sigma))$ and $(\mathcal{P}_k L)$ are convex, then the necessary and sufficient conditions for $\boldsymbol{x}(\sigma) \in Sol(\mathcal{P}L_k(\sigma))$ and $\boldsymbol{x}^k \in Sol(\mathcal{P}_k L)$ are, correspondingly, the inclusions

$$\boldsymbol{y}(\sigma) \in \partial_c G_\sigma(\boldsymbol{x}(\sigma)) + N_S(\boldsymbol{x}(\sigma)), \quad \boldsymbol{y}^k \in \partial_c G_k(\boldsymbol{x}^k) + N_S(\boldsymbol{x}^k), \tag{7.9}$$

or, which is the same [2, 23, 24, 42],

$$\boldsymbol{x}(\sigma) \in \partial_c[G_\sigma + \mathcal{X}_S]^*(\boldsymbol{y}(\sigma)), \quad \boldsymbol{x}^k \in \partial_c[G_k + \mathcal{X}_S]^*(\boldsymbol{y}^k). \tag{7.10}$$

Here $N_S(\boldsymbol{x})$ and $\mathcal{X}_S(\boldsymbol{x})$ are the normal cone to the set S at the point $\boldsymbol{x}$

$$N_S(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{x}' - \boldsymbol{x}\rangle \leq 0 \quad \text{for all } \boldsymbol{x}' \in S\},$$

and the indicator function of the set S, respectively,

$$\mathcal{X}_S(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \boldsymbol{x} \in S \\ +\infty, & \text{if } \boldsymbol{x} \notin S. \end{cases}$$

Besides, $\partial_c \mathcal{X}_s(\boldsymbol{x}) = N_S(\boldsymbol{x})$, $\boldsymbol{x} \in S$ [23–25, 42]. Thus, we have the following equalities

$$G_\sigma(\boldsymbol{x}(\sigma)) - \langle \boldsymbol{y}(\sigma), \boldsymbol{x}(\sigma)\rangle = \mathcal{V}(\sigma) := \inf_{\boldsymbol{x}}\{G_\sigma(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma), \boldsymbol{x}\rangle : \boldsymbol{x} \in S\}, \tag{7.11}$$

where $\mathcal{V}(\sigma)$ is the optimal value of the problem $(\mathcal{P}L_k(\sigma))$.

Finally, $[G_\sigma + \mathcal{X}_S]^*(\cdot)$ is the conjugate function to the function $[G_\sigma + \mathcal{X}_S](\cdot)$. Recall that, if $\psi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, then [23–25, 42, 44] by definition

$$\psi^*(\boldsymbol{y}) := \sup_{\boldsymbol{x}}\{\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \psi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^n\}.$$

Together with problems $(\mathcal{P}L_k(\sigma))$ and $(\mathcal{P}_k L)$ consider the following dual problem

$$\begin{cases} \text{minimize} & \varphi_k(\boldsymbol{y}) := H_k^*(\boldsymbol{y}) - \langle \boldsymbol{x}^k, \boldsymbol{y}\rangle \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n, \end{cases} \tag{$\mathcal{P}_k^* L$}$$

where $H_k^*(\cdot)$ is the conjugate of the function $H_k(\cdot)$, as above.

Since the problem $(\mathcal{P}_k^* L)$ is also convex, one can find its solution $\boldsymbol{y}^k \in Sol(\mathcal{P}_k^* L)$, which satisfies the necessary and sufficient condition as follows

$$\boldsymbol{x}^k \in \partial_c H_k^*(\boldsymbol{y}^k) \text{ or } \boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k), \tag{7.12}$$

if $H_k(\cdot)$ is convex and closed (i.e. lower semicontinuous), which is the case under assumptions of Sect. 7.3. Besides, we have

$$H_k^*(\boldsymbol{y}^k) - \langle \boldsymbol{x}^k, \boldsymbol{y}^k\rangle = \inf_{\boldsymbol{y}}\{\varphi_k(\boldsymbol{y}) : \boldsymbol{y} \in \mathbb{R}^n\}.$$

Thus, if we know $\boldsymbol{x}^k \in S$, we can find $\boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}_k)$ by solving the dual problem $(\mathcal{P}_k^* L)$.

Furthermore, it can be readily seen (see (7.3)–(7.5)) that the penalty function $W(\boldsymbol{x})$ is also a DC function, since it has the DC representation as follows $W(\boldsymbol{x}) = G_W(\boldsymbol{x}) - H_W(\boldsymbol{x})$, where (cf. (7.3)–(7.5))

$$\begin{cases} G_W(\boldsymbol{x}) & := \sigma^{-1}[G_\sigma(\boldsymbol{x}) - g_0(\boldsymbol{x})] = 2\sum\limits_{j\in\mathcal{E}} \max\{g_j(\boldsymbol{x}), h_j(\boldsymbol{x})\} \\ & \quad + \max\{\sum\limits_{p\in\mathcal{I}} h_p(\boldsymbol{x}), [g(\boldsymbol{x}) + \sum\limits_{\substack{p\in\mathcal{I}\\ p\neq i}} h_p(\boldsymbol{x})],\ i \in \mathcal{I}\}, \\ H_W(\boldsymbol{x}) & := \sigma^{-1}[H_\sigma(\boldsymbol{x}) - h_0(\boldsymbol{x})] = \sum\limits_{i\in\mathcal{I}} h_i(\boldsymbol{x}) + \sum\limits_{j\in\mathcal{E}} [g_j(\boldsymbol{x}) + h_j(\boldsymbol{x})]. \end{cases} \tag{7.13}$$

In addition, it can be easily shown that (see (7.6) and (7.8)) if $\boldsymbol{y}(\sigma) \in \partial_c H_\sigma(\boldsymbol{x}^k)$, then we have

$$\begin{aligned} \boldsymbol{y}_W^k := \sigma^{-1}[\boldsymbol{y}(\sigma) - h'_{0k}] &= \sum_{i\in\mathcal{I}} h'_{ik}(\boldsymbol{x}^k) + \sum_{j\in\mathcal{E}} (g'_{jk}(\boldsymbol{x}^k) + h'_{jk}(\boldsymbol{x}^k)) \\ &=: H'_W(\boldsymbol{x}^k) \in \partial_c H_W(\boldsymbol{x}^k). \end{aligned} \tag{7.14}$$

Now introduce the following auxiliary linearized problem

$$\begin{cases} \text{minimize} & \Phi_W(\boldsymbol{x}) := G_W(\boldsymbol{x}) - \langle \boldsymbol{y}_W^k, \boldsymbol{x}\rangle \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \tag{$\mathcal{AP}_W L_k$}$$

which is obviously convex, as $(\mathcal{P}L_k(\sigma))$, $(\mathcal{P}_k L)$ and $(\mathcal{P}_k^* L)$, and hence, can be solved by modern optimization methods and software [2, 25, 38, 57].

On the other hand, it is clear, that $(\mathcal{AP}_W L_k)$ is related to the minimization of the penalty function $W(\boldsymbol{x})$, i.e. to the following nonconvex problem

$$\begin{cases} \text{minimize} & W(\boldsymbol{x}) \triangleq G_W(\boldsymbol{x}) - H_W(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in S. \end{cases} \tag{$\mathcal{P}_W$}$$

More precisely, the problem $(\mathcal{AP}_W L_k)$ turns out to be the linearized (with respect to the function $H_W(\cdot)$) problem generated by the problem $(\mathcal{P}_W)$.

Now we are going to present a local search scheme for the problem $(\mathcal{P})$, which incorporates a penalty parameter update procedure. Let there be given a starting point $\boldsymbol{x}_0 \in S$, an initial value $\sigma^0 > 0$ of the penalty parameter $\sigma \geq 0$ together with two parameters $\eta_1, \eta_2 \in]0, 1[$ of the scheme. Then the first LS Scheme can be described as follows.

Algorithm 7.1: Scheme 1

Step 0. Set $k := 0,\ \boldsymbol{x}^k := \boldsymbol{x}_0,\ \sigma_k := \sigma^0$.

Step 1. Solve the dual subproblem $(\mathcal{P}_k^* L)$ to obtain $\boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k)$ (see (7.12)).

Step 2. Solve the subproblem $(\mathcal{P}_k L)$ to get $\boldsymbol{x}(\sigma_k) \in Sol(\mathcal{P}_k L)$ (see (7.9)–(7.11)) with $\boldsymbol{y}(\sigma_k) = \boldsymbol{y}^k$.

Step 3. If $W(\boldsymbol{x}(\sigma_k)) = 0$, then set $\sigma_+ := \sigma_k,\ \boldsymbol{x}(\sigma_+) := \boldsymbol{x}(\sigma_k)$, and go to Step 8.

Step 4. (Else, $W(\boldsymbol{x}(\sigma_k)) > 0$). By solving the linearized problems

$$\begin{cases} \text{minimize} & \varphi_{\sigma_k}(\boldsymbol{y}) := H_k^*(\boldsymbol{y}) - \langle \boldsymbol{x}(\sigma_k), \boldsymbol{y} \rangle \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n, \end{cases} \qquad (\mathcal{P}L^*(\sigma_k))$$

and $(\mathcal{AP}_W L_k)$ find $\hat{\boldsymbol{y}}^k \in \partial H_k(\boldsymbol{x}(\sigma_k))$, $\hat{\boldsymbol{y}}^k \in Sol(\mathcal{P}L^*(\sigma_k))$ and $\boldsymbol{x}_W^k \in Sol(\mathcal{AP}_W L_k)$ respectively. Further, set $\boldsymbol{x}^k := \boldsymbol{x}(\sigma_k),\quad \boldsymbol{y}^k := \hat{\boldsymbol{y}}^k$.

Step 5. If $W(\boldsymbol{x}_W^k) = 0$, then, starting at $\boldsymbol{x}_W^k$, and by consecutively (by increasing, if necessary, the value $\sigma > 0$ of penalty parameter $\sigma > \sigma_k$) solving a few problems $(\mathcal{P}L_k(\sigma))$, with $\boldsymbol{y}(\sigma) := h'_{0k} + \sigma \boldsymbol{y}_W^k \in \partial_c H_\sigma(\boldsymbol{x}^k)$ (see (7.14)), find $\sigma_+ > \sigma_k$ such that $W(\boldsymbol{x}(\sigma_+)) = 0,\ \boldsymbol{x}(\sigma_+) \in Sol(\mathcal{P}L_k(\sigma_+))$, and go to Step 8.

Step 6. If $W(\boldsymbol{x}_W^k) > 0$, or if a feasible $\sigma_+ > \sigma_k$, such that $W(\boldsymbol{x}(\sigma_+)) = 0$, is not found (on Step 5), find $\sigma_+ > \sigma_k$ satisfying the inequality

$$W(\boldsymbol{x}^k) - W(\boldsymbol{x}(\sigma_+)) \geq \eta_1 [W(\boldsymbol{x}^k) - W(\boldsymbol{x}_W^k)]. \qquad (7.15)$$

Step 7. Increase σ_+, if necessary, to fulfil the inequality

$$\Phi_k(\boldsymbol{x}^k) - \Phi_{\sigma_+}(\boldsymbol{x}(\sigma_+)) \geq \eta_2 \sigma_+ [W(\boldsymbol{x}^k) - W(\boldsymbol{x}(\sigma_+))]. \qquad (7.16)$$

Step 8. Set $k := k + 1,\ \sigma_{k+1} := \sigma_+,\ \boldsymbol{x}^{k+1} := \boldsymbol{x}(\sigma_+)$ and go to Step 1.

7.5 Convergence Properties of Scheme 1

It is clear that the above scheme is not yet to become a proper algorithm, because, in particular, convergence properties of the scheme have not been investigated. Therefore, it is impossible to propose any stopping criterion. In order to handle these issues, let us give a few preliminary remarks.

Case 1 On Step 3 of Scheme 1, when $W(\boldsymbol{x}(\sigma_k)) = 0$, we go to Step 8 and set $\sigma_{k+1} := \sigma_+ = \sigma_k,\ \boldsymbol{x}^{k+1} := \boldsymbol{x}(\sigma_k)$, besides, $\boldsymbol{y}(\sigma_+) := \boldsymbol{y}(\sigma_k) = \boldsymbol{y}^k,\ \boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k)$

and $\boldsymbol{x}(\sigma_k) \neq \boldsymbol{x}^k$, in general. Then, with the help of (7.11), we have

$$0 = \inf_{\boldsymbol{x}}\{G_k(\boldsymbol{x}) - G_k(\boldsymbol{x}(\sigma_k)) + \langle \boldsymbol{y}^k, \boldsymbol{x}(\sigma_k) - \boldsymbol{x}\rangle : \ \boldsymbol{x} \in S\}$$
$$\leq G_k(\boldsymbol{x}^k) - G_k(\boldsymbol{x}(\sigma_k)) + \langle \boldsymbol{y}^k, \boldsymbol{x}(\sigma_k) - \boldsymbol{x}^k\rangle = \Phi_k(\boldsymbol{x}^k) - \Phi_k(\boldsymbol{x}(\sigma_k)).$$

Now, due to convexity of $H_k(\cdot) \triangleq H_{\sigma_k}(\cdot)$ and because $\boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k)$, $\boldsymbol{x}^{k+1} = \boldsymbol{x}(\sigma_k)$, we derive

$$0 \leq G_k(\boldsymbol{x}^k) - G_k(\boldsymbol{x}(\sigma_k)) + H_k(\boldsymbol{x}(\sigma_k)) - H_k(\boldsymbol{x}^k)$$
$$= \Theta_k(\boldsymbol{x}^k) - \Theta_k(\boldsymbol{x}^{k+1}) \tag{7.17}$$

i.e.

$$\Theta_k(\boldsymbol{x}^{k+1}) \leq \Theta_k(\boldsymbol{x}^k). \tag{7.18}$$

$$\Theta_{k+1}(\boldsymbol{x}^{k+1}) \triangleq f_0(\boldsymbol{x}^{k+1}) + \sigma_{k+1} W(\boldsymbol{x}^{k+1})$$
$$= f_0(\boldsymbol{x}^{k+1}) + \sigma_k W(\boldsymbol{x}^{k+1}) \triangleq \Theta_k(\boldsymbol{x}^{k+1})$$

and, in the same manner, we obtain

$$\Theta_k(\boldsymbol{x}^k) = f_0(\boldsymbol{x}^k) + \sigma_k W(\boldsymbol{x}^k) = f_0(\boldsymbol{x}^k) + \sigma_{k+1} W(\boldsymbol{x}^k) = \Theta_{k+1}(\boldsymbol{x}^k).$$

The two last chains and (7.18) then yield

$$\Theta_{k+1}(\boldsymbol{x}^{k+1}) \leq \Theta_{k+1}(\boldsymbol{x}^k). \tag{7.19}$$

Case 2 On Step 5 one finds a sufficiently large $\sigma_+ > \sigma_k$, such that $W(\boldsymbol{x}(\sigma_+)) = 0$, where $\boldsymbol{x}(\sigma_+) \in Sol(\mathcal{P}L_k(\sigma_+))$ and $\boldsymbol{y}(\sigma_+) = h'_{0k} + \sigma_+ y^k_W \in \partial_c H_{\sigma_+}(\boldsymbol{x}^k)$, $\boldsymbol{y}(\sigma_{k+1}) := \boldsymbol{y}(\sigma_+)$. After that, we set $\sigma_{k+1} := \sigma_+$, $\boldsymbol{x}^{k+1} := \boldsymbol{x}(\sigma_+)$ and, as above, according to (7.11), we obtain

$$0 \leq G_{k+1}(\boldsymbol{x}^k) - G_{k+1}(\boldsymbol{x}^{k+1}) + \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\rangle \tag{7.20a}$$
$$= \Phi_{k+1}(\boldsymbol{x}^k) - \Phi_{k+1}(\boldsymbol{x}^{k+1}). \tag{7.20b}$$

Further, with the help of convexity of $H_{k+1}(\cdot)$ and the inclusion $\boldsymbol{y}(\sigma_{k+1}) \in \partial_c H_{k+1}(\boldsymbol{x}^k)$ we derive, as above, that

$$0 \leq G_{k+1}(\boldsymbol{x}^k) - G_{k+1}(\boldsymbol{x}^{k+1}) + H_{k+1}(\boldsymbol{x}^{k+1}) - H_{k+1}(\boldsymbol{x}^k) \tag{7.20c}$$
$$= \Theta_{k+1}(\boldsymbol{x}^k) - \Theta_{k+1}(\boldsymbol{x}^{k+1}), \tag{7.20d}$$

whence, the inequality (7.19) again follows.

Case 3 On Steps 6 and 7 we find $\sigma_+ \geq \sigma_k$ to satisfy the inequalities (7.15) and (7.16) consecutively. Finally, on Step 8 we set $\sigma_{k+1} := \sigma_+$, $\boldsymbol{x}^{k+1} := \boldsymbol{x}(\sigma_+)$, $\boldsymbol{y}(\sigma_{k+1}) := \boldsymbol{y}(\sigma_+) = h'_{0k} + \sigma_+ \boldsymbol{y}^k_W$. It can be readily seen that, as before, we obtain the chains (7.20a) and (7.20c), and, hence, the inequality (7.19). Thus, one can conclude that in all cases Scheme 1 produces the sequence $\{\boldsymbol{x}^k\} \subset S$ satisfying the inequality (7.19). Furthermore, with the help of the obvious chain of equalities

$$\begin{aligned} \Theta_{k+1}(\boldsymbol{x}^k) &= \Theta_k(\boldsymbol{x}^k) + [\Theta_{k+1}(\boldsymbol{x}^k) - \Theta_k(\boldsymbol{x}^k)] \\ &= \Theta_k(\boldsymbol{x}^k) + (\sigma_{k+1} - \sigma_k) W(\boldsymbol{x}^k) = \Theta_k(\boldsymbol{x}^k) + \xi_k, \text{ and} \qquad (7.21) \\ \xi_k &:= (\sigma_{k+1} - \sigma_k) W(\boldsymbol{x}^k), \qquad k = 0, 1, 2, \ldots. \qquad (7.22) \end{aligned}$$

The inequality (7.19) can be transformed into the following one

$$\Theta_{k+1}(\boldsymbol{x}^{k+1}) \leq \Theta_k(\boldsymbol{x}^k) + \xi_k, \qquad k = 0, 1, 2, \ldots. \qquad (7.23)$$

Using the denotation $\vartheta_k := \Theta_k(\boldsymbol{x}^k)$, we rewrite (7.23) as

$$\vartheta_{k+1} \leq \vartheta_k + \xi_k, \qquad k = 0, 1, 2, \ldots.$$

Introduce now the following assumption:

$$\begin{aligned} &(a) \quad \sum_{k=0}^{\infty} \xi_k \triangleq \sum_{k=0}^{\infty} (\sigma_{k+1} - \sigma_k) W(\boldsymbol{x}^k) < +\infty, \\ &(b) \quad \xi_k \triangleq (\sigma_{k+1} - \sigma_k) W(\boldsymbol{x}^k) \geq 0, \qquad k = 0, 1, 2, \ldots. \end{aligned} \qquad (\mathcal{H}_W)$$

It follows from $(\mathcal{H}_W)(b)$ that

$$\sigma_{k+1} \geq \sigma_k \geq 0. \qquad (7.24)$$

Then the number sequence $\{\vartheta_k = \Theta_k(\boldsymbol{x}^k)\}$ produced by Scheme 1, $(\mathcal{P}L_k(\sigma))$–$(\mathcal{AP}_W L_k)$ turns out to be (almost) descreasing (see (7.23)), and hence converging in virtue of the following result from [57, Lemma 2, pp. 105].

Lemma 7.1 *Let a number sequence $\{a_k\}$ satisfy the conditions*

$$a_{k+1} \leq a_k + \varepsilon_k, \ \varepsilon_k \geq 0, \ k = 0, 1, \ldots, \ \sum_{k=0}^{\infty} \varepsilon_k < \infty.$$

Then, there exists $\lim\limits_{k \to \infty} a_k < +\infty$. If, in addition, $\{a_k\}$ is bounded from below, then $\lim\limits_{k \to \infty} a_k$ is finite.

Henceforth, assume that the function $f_0(\cdot)$ is lower bounded over S, such that

$$\inf_{\boldsymbol{x}}\{f_0(\boldsymbol{x}) : \boldsymbol{x} \in S\} > -\infty. \qquad (\mathcal{H}_0)$$

Then it can be readily seen, that the goal function $\Theta_\sigma(\cdot)$ of the problem $(\mathcal{P}_\sigma)$ also satisfies the same condition

$$\inf\{\Theta_\sigma(\boldsymbol{x}) : \boldsymbol{x} \in S\} > -\infty,$$

due to the obvious inequalities:

$$\sigma \geq 0, \quad W(\boldsymbol{x}) \geq 0, \quad f_0(\boldsymbol{x}) \leq f_0(\boldsymbol{x}) + \sigma W(\boldsymbol{x}) \triangleq \Theta_\sigma(\boldsymbol{x}).$$

The next result provides the first convergence properties of Scheme 1.

Proposition 7.1 *Let the assumptions $(\mathcal{H}_W)$, $(\mathcal{H}_0)$ be fulfilled. Then, the sequence $\{\boldsymbol{x}^k\} \subset S$ produced by the local search Scheme 1 satisfies the following conditions. The number sequences $\{\vartheta_k \triangleq \Theta_k(\boldsymbol{x}^k)\}$ and $\{\Delta\Phi_{k+1}\}$, where $\Delta\Phi_{k+1} := \Phi_{k+1}(\boldsymbol{x}^k) - \Phi_{k+1}(\boldsymbol{x}^{k+1}) \triangleq G_{k+1}(\boldsymbol{x}^k) - G_{k+1}(\boldsymbol{x}^{k+1}) + \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\rangle$, converge so that*

$$(i) \qquad \lim_{k\to\infty} \vartheta_k = \lim_{k\to\infty} \Theta_k(\boldsymbol{x}^k) =: \Theta_* > -\infty; \qquad \text{and} \qquad (7.25)$$

$$(ii) \qquad \lim_{k\to\infty} \Delta\Phi_{k+1} = 0. \qquad (7.26)$$

Proof (i) As it has been shown above, in all situations which can happen (Cases 1, 2, 3) the inequality (7.23) holds with ξ_k defined in (7.22). In addition, under the assumptions $(\mathcal{H}_W)$ and $(\mathcal{H}_0)$ the sequences $\{\xi_k\}, \{\vartheta_k\}$ satisfy all the conditions of Lemma 7.1. Hence, there exists a finite limit

$$\Theta_* = \lim_{k\to\infty} \Theta_k(\boldsymbol{x}^k) = \lim_{k\to\infty} \vartheta_k > -\infty,$$

and (7.25) is proven.

(ii) From (7.20a) and (7.20c) and (7.21), it follows

$$\begin{aligned} 0 \leq \Delta\Phi_{k+1} &\triangleq \Phi_{k+1}(\boldsymbol{x}^k) - \Phi_{k+1}(\boldsymbol{x}^{k+1}) \qquad (7.27)\\ &\leq \Theta_{k+1}(\boldsymbol{x}^k) - \Theta_{k+1}(\boldsymbol{x}^{k+1}) = [\Theta_k(\boldsymbol{x}^k) - \Theta_{k+1}(\boldsymbol{x}^{k+1})] + \xi_k. \end{aligned}$$

The right-hand side of this chain, due to $(\mathcal{H}_W)$ and (7.24), tends to zero and therefore, (7.26) is also proven. □

Recall now that a function $H(\cdot)$ on $\mathbb{R}^n$ is said to be strongly convex on S, if for any $\boldsymbol{y} \in \partial_c H(\boldsymbol{x}_0)$ we have

$$H(\boldsymbol{x}) - H(\boldsymbol{x}_0) \geq \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{x}_0\rangle + \frac{\rho}{2}||\boldsymbol{x} - \boldsymbol{x}_0||^2 \ \text{ for all } \boldsymbol{x},\ \boldsymbol{x}_0 \in S, \qquad (7.28)$$

where $\rho > 0$ is a constant of strong convexity of $H(\cdot)$.

It can be readily seen that, if the following assumption holds

$$\text{at least one of the functions } h_i(\cdot),\ i \in \mathcal{I} \cup \mathcal{E},\ g_j(\cdot),\ j \in \mathcal{E},\ \text{is strongly convex,} \tag{$\mathcal{H}_{str}$}$$

then the functions $H_\sigma(\cdot)$, $H_k(\cdot)$, $H_{k+1}(\cdot)$, $H_W(\cdot)$ turn out to be strongly convex. Moreover, it is well-known [7, 23–26, 56] that in a DC representation $f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x})$ of an arbitrary DC function $f(\cdot)$, the convex components $g(\cdot)$ and $h(\cdot)$ can be always constructed to be strongly convex.

Proposition 7.2 *Let Assumption* $(\mathcal{H}_{str})$ *be satisfied. Then, the sequence* $\{\boldsymbol{x}^k\}$ *produced by Scheme 1,* $(\mathcal{P}L_k(\sigma))$–$(\mathcal{A}\mathcal{P}_W L_k)$*, is the Cauchy sequence, i.e.*

$$\lim_{k\to\infty} ||\boldsymbol{x}^k - \boldsymbol{x}^{k+1}|| = 0. \tag{7.29}$$

Proof Applying the inequality (7.28) with $H(\cdot) := H_{k+1}(\cdot)$ and $\boldsymbol{y} = \boldsymbol{y}(\sigma_{k+1})$ to the right-hand side of (7.20a), we obtain

$$\begin{aligned}\frac{\rho}{2}||\boldsymbol{x}^k - \boldsymbol{x}^{k+1}||^2 &\le \Theta_{k+1}(\boldsymbol{x}^k) - \Theta_{k+1}(\boldsymbol{x}^{k+1}) \\ &= [\Theta_k(\boldsymbol{x}^k) - \Theta_{k+1}(\boldsymbol{x}^{k+1})] + \xi_k \end{aligned} \tag{7.30}$$

whence, in virtue of (7.25) and $(\mathcal{H}_W)$, (7.29) follows. □

It obviously follows from (7.29) that there exists some $\boldsymbol{x}_* \in \mathbb{R}^n$, such that

$$\lim_{k\to\infty} \boldsymbol{x}^k = \boldsymbol{x}_*.$$

Remark 7.1 Recall that by construction (see $(\mathcal{P}L_k(\sigma))$–$(\mathcal{A}\mathcal{P}_W L_k)$) and according to Scheme 1, we have

$$\boldsymbol{y}(\sigma_{k+1}) = \boldsymbol{y}(\sigma_+) = h'_{0k} + \sigma_+ \boldsymbol{y}^k_W \in \partial_c H_{\sigma_+}(\boldsymbol{x}^k),$$

meanwhile, $\boldsymbol{y}^{k+1} \in \partial_c H_{k+1}(\boldsymbol{x}^{k+1})$, $\boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k)$, so that the sequences $\{\boldsymbol{y}^k\}$ and $\{\boldsymbol{y}(\sigma_{k+1})\}$ are different, in general.

Let us now look at convergence properties of the sequence $\{\boldsymbol{y}^k\}$ produced by Scheme 1. First, recall that, due to $(\mathcal{H}_W)$(b), we have $\sigma_{k+1} \ge \sigma_k$ (see (7.24)). Suppose now that the following assumption holds

$$\text{there exists } \sigma_{up} \in \mathbb{R} : \sigma_{up} \ge \sigma_k, \qquad k = 0, 1, 2, \ldots. \tag{$\mathcal{H}_\sigma$}$$

It is clear that from the practical viewpoint $(\mathcal{H}_\sigma)$ looks too natural, and, on the other hand, it is related to existence of the threshold value $\sigma_* \ge 0$ of the penalty

parameter $\sigma \geq 0$, i.e. to the exact penalty theory [2, 3, 11, 15, 22, 59]. Furthermore, using (7.24) and $(\mathcal{H}_\sigma)$, we derive that there exists $\sigma_* > 0$, such that

$$\lim_{k\to\infty} \sigma_k = \sigma_*. \tag{7.31}$$

On the other hand, by construction we have

$$\begin{aligned} \boldsymbol{y}(\sigma_{k+1}) = \boldsymbol{y}(\sigma_+) \in \partial_c H_{k+1}(\boldsymbol{x}^k), \quad \boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k), \\ \boldsymbol{y}^k = h'_{0k} + \sigma_k \boldsymbol{y}^k_W, \quad \boldsymbol{y}(\sigma_{k+1}) = h'_{0k} + \sigma_{k+1}\boldsymbol{y}^k_W, \\ \boldsymbol{y}^k_W = \sum_{i\in\mathcal{I}} h'_{ik} + \sum_{j\in\mathcal{E}} (g'_{jk} + h'_{jk}) \in \partial_c H_W(\boldsymbol{x}^k). \end{aligned}$$

Therefore, we obtain ($\boldsymbol{y}_+ := \boldsymbol{y}(\sigma_+) = \boldsymbol{y}(\sigma_{k+1})$)

$$||\boldsymbol{y}_+ - \boldsymbol{y}^k|| = ||\boldsymbol{y}(\sigma_{k+1}) - \boldsymbol{y}^k|| = (\sigma_{k+1} - \sigma_k)||\boldsymbol{y}^k_W||. \tag{7.32}$$

In addition, since $\boldsymbol{y}^k_W \in \partial_c H_W(\boldsymbol{x}^k)$, the set-value mapping $\boldsymbol{x} \to \partial_c H_W(\boldsymbol{x})$ is bounded [7, 9, 25, 42, 44], besides, $\boldsymbol{x}^k \to \boldsymbol{x}_*$ (so that $\{\boldsymbol{x}^k\}$ is a bounded set) we conclude that $||\boldsymbol{y}^k_W|| \leq C_W$, $0 < C_W < +\infty$.

Then, it follows from (7.31) and (7.32) that

$$||\boldsymbol{y}(\sigma_{k+1}) - \boldsymbol{y}^k|| \leq (\sigma_{k+1} - \sigma_k)C_W \downarrow 0 \quad (k \to \infty).$$

Hence,

$$\lim_{k\to\infty} ||\boldsymbol{y}(\sigma_{k+1}) - \boldsymbol{y}^k|| = 0. \tag{7.33}$$

Our next objective is the result below.

Proposition 7.3 *Let the assumptions of Propositions 7.1 and 7.2 and Assumption* $(\mathcal{H}_\sigma)$ *hold. Then*

$$\lim_{k\to\infty} ||\boldsymbol{y}^{k+1} - \boldsymbol{y}^k|| = 0. \tag{7.34}$$

Proof We have to prove only the relation

$$\lim_{k\to\infty} ||\boldsymbol{y}^{k+1} - \boldsymbol{y}(\sigma_{k+1})|| = 0, \tag{7.35}$$

since then, applying the triangle inequality

$$||\boldsymbol{y}^{k+1} - \boldsymbol{y}^k|| \leq ||\boldsymbol{y}^{k+1} - \boldsymbol{y}(\sigma_{k+1})|| + ||\boldsymbol{y}(\sigma_{k+1}) - \boldsymbol{y}^k||,$$

we get (7.34), as has been claimed.

Indeed, recall that, according to Scheme 1, $\boldsymbol{x}^{k+1} := \boldsymbol{x}(\sigma_+) \in Sol(\mathcal{P}L_k(\sigma_+))$, where

$$\begin{cases} \text{minimize} & G_{k+1}(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x} \rangle \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \qquad (\mathcal{P}L_k(\sigma_+))$$

so that $\boldsymbol{y}_+ := \boldsymbol{y}(\sigma_{k+1}) \in \partial_c[G_{k+1} + \mathcal{X}_s](\boldsymbol{x}^{k+1})$, i.e. $\boldsymbol{x}^{k+1} \in \partial_c[G_{k+1} + \mathcal{X}_s]^*(\boldsymbol{y}_+)$, which, in particular, implies

$$[G_{k+1} + \mathcal{X}_s]^*(\boldsymbol{y}^{k+1}) - [G_{k+1} + \mathcal{X}_s]^*(\boldsymbol{y}_+) \geq \langle \boldsymbol{x}^{k+1}, \boldsymbol{y}^{k+1} - \boldsymbol{y}_+ \rangle. \qquad (7.36a)$$

Furthermore, $\boldsymbol{y}^{k+1}$ is the solution of $(\mathcal{P}^*_{k+1}L)$ with $\boldsymbol{x}^{k+1}$, so that $\boldsymbol{x}^{k+1} \in \partial_c H^*_{k+1}(\boldsymbol{y}^{k+1})$, i.e.

$$H^*_{k+1}(\boldsymbol{y}_+) - H^*_{k+1}(\boldsymbol{y}^{k+1}) \geq \langle \boldsymbol{x}^{k+1}, \boldsymbol{y}_+ - \boldsymbol{y}^{k+1} \rangle + \frac{\rho}{2}||\boldsymbol{y}^{k+1} - \boldsymbol{y}_+||^2, \qquad (7.36b)$$

due to the strong convexity of $H_{k+1}(\cdot)$, and, as a consequence, of $H^*_{k+1}(\cdot)$, $\rho > 0$. By adding (7.36a) and (7.36b), we obtain

$$F_{k+1}(\boldsymbol{y}_+) - F_{k+1}(\boldsymbol{y}^{k+1}) \geq \frac{\rho}{2}||\boldsymbol{y}^{k+1} - \boldsymbol{y}_+||^2 > 0, \qquad (7.37)$$

where

$$F_{k+1}(\boldsymbol{y}) := H^*_{k+1}(\boldsymbol{y}) - [G_{k+1} + \mathcal{X}_s]^*(\boldsymbol{y}), \ \boldsymbol{y} \in \mathbb{R}^n. \qquad (7.38)$$

It is worth noting that the problem

$$\begin{cases} \text{minimize} & F_{k+1}(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n \end{cases} \qquad (\mathcal{P}^*_{k+1})$$

is dual to the problem $(\mathcal{P}_{k+1}) = (\mathcal{P}(\sigma_{k+1}))$, where $G_{k+1} := G_{\sigma_{k+1}}$, $H_{k+1} := H_{\sigma_{k+1}}$, and

$$\begin{cases} \text{minimize} & G_{k+1}(\boldsymbol{x}) - H_{k+1}(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \qquad (\mathcal{P}_{k+1})$$

in the Toland's sense [53–55], which implies, in particular, $\mathcal{V}(\mathcal{P}^*_{k+1}) = \mathcal{V}(\mathcal{P}_{k+1}) > -\infty$. Moreover, since (see (7.37))

$$F_{k+1}(\boldsymbol{y}^{k+1}) < F_{k+1}(\boldsymbol{y}_+) = F_{k+1}(\boldsymbol{y}(\sigma_{k+1})),$$

one can show (as it has been proven above in Proposition 7.1) that the number sequence $\{F_k(\boldsymbol{v}^k)\}$ also converges, where $\boldsymbol{v}^k := \boldsymbol{y}(\sigma_{k+1}) =: \boldsymbol{y}_+$, $\boldsymbol{v}^{k+1} := \boldsymbol{y}^{k+1}$, $k = 0, 1, 2, \ldots$ Then, in virtue of (7.37), the equality (7.35) holds, which completes the proof. □

Further, it obviously follows from (7.34) that there exists a limit point $\boldsymbol{y}_* \in \mathbb{R}^n$, such that

$$\lim_{k\to\infty} \boldsymbol{y}^k = \boldsymbol{y}_*. \tag{7.39}$$

Moreover, it can be readily seen that

$$\lim_{k\to\infty} ||\boldsymbol{y}(\sigma_{k+1}) - \boldsymbol{y}_*|| = 0.$$

On the other hand, since $\boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k)$, $\boldsymbol{x}^k \to \boldsymbol{x}_*$, $\boldsymbol{y}^k \to \boldsymbol{y}_*$, $H_k(\boldsymbol{x}) = h_0(\boldsymbol{x}) + \sigma_k H_W(\boldsymbol{x})$ and $\lim\limits_{k\to\infty} \sigma_k = \sigma_*$, the set-valued mapping $\partial_c H_k(\cdot)$ is compact and convex valued, as well as upper semicontinuous (u.s.c.), we obtain (see [25, Chapter VI, Propositions 6.2.1, 6.2.2 and Theorems 6.2.4, 6.2.7, pp. 282–284])

$$\boldsymbol{y}_* \in \partial_c H_*(\boldsymbol{x}_*), \tag{7.40}$$

where $H_*(\boldsymbol{x}) = h_0(\boldsymbol{x}) + \sigma_* H_W(\boldsymbol{x})$, $\boldsymbol{y}_* = h_0'(\boldsymbol{x}_*) + \sigma_* H_W'(\boldsymbol{x}_*)$.

Remark 7.2 Note that the relations between problems $(\mathcal{P})$ and $(\mathcal{P}^*)$, as well as between $(\mathcal{P}L_k)$ and $(\mathcal{P}^*L_k)$ were initially studied in the works of Hiriart-Urruty [23, 24], Pham Dinh Tao and Le Thi Hoai An [29–34], and Toland [53–55]. Besides, the successful development of the so-called DCA and its generalizations for the presence of inequality constraints and a number of applied problems was carried out in [29–34].

Furthermore, employing the convergence of the sequences $\{\boldsymbol{x}^k\}$ and $\{\boldsymbol{y}^k\}$, one immediately gets the following result.

Theorem 7.1 *Let the assumptions $(\mathcal{H}_W)$, $(\mathcal{H}_0)$, $(\mathcal{H}_{str})$ and $(\mathcal{H}_\sigma)$ be fulfilled. Then, the limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ produced by Scheme 1 is a solution to the following convex problem with $G_*(\boldsymbol{x}) := G_{\sigma_*}(\boldsymbol{x}) \stackrel{\triangle}{=} g_0(\boldsymbol{x}) + \sigma_* G_W(\boldsymbol{x})$ and $\sigma_* = \lim\limits_{k\to\infty} \sigma_k$:*

$$\begin{cases} \textit{minimize} & \Phi_*(\boldsymbol{x}) := G_*(\boldsymbol{x}) - \langle \boldsymbol{y}_*, \boldsymbol{x} \rangle \\ \textit{subject to} & \boldsymbol{x} \in S. \end{cases} \tag{$\mathcal{P}L_*$}$$

Proof By construction, due to Scheme 1, we have

$$\mathcal{V}(\mathcal{P}L_k(\sigma_{k+1})) \stackrel{\triangle}{=} \inf_{\boldsymbol{x}}\{G_{k+1}(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}\rangle \ : \ \boldsymbol{x} \in S\}$$
$$= G_{k+1}(\boldsymbol{x}^{k+1}) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}^{k+1}\rangle \leq G_{k+1}(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}\rangle$$

for all $\boldsymbol{x} \in S$ and $G_{k+1}(\boldsymbol{x}) = g_0(\boldsymbol{x}) + \sigma_{k+1}G_W(\boldsymbol{x})$.

When $k \to \infty$, we have $\boldsymbol{x}^{k+1} \to \boldsymbol{x}_*$, $\boldsymbol{y}(\sigma_{k+1}) \to \boldsymbol{y}_*$, and $G_{k+1}(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}\rangle$ tends to $G_*(\boldsymbol{x}) - \langle \boldsymbol{y}_*, \boldsymbol{x}\rangle$ for each $\boldsymbol{x} \in S$. Hence, in virtue of continuity of $g_0(\cdot)$, $G_W(\cdot)$ and the inner product $\langle \cdot, \cdot \rangle$, we obtain

$$G_*(\boldsymbol{x}_*) - \langle \boldsymbol{y}_*, \boldsymbol{x}_*\rangle \leq G_*(\boldsymbol{x}) - \langle \boldsymbol{y}_*, \boldsymbol{x}\rangle \ \text{ for all } \boldsymbol{x} \in S,$$

which completes the proof. □

Remark 7.3 Let now point out that the assumptions $(\mathcal{H}_W)$, $(\mathcal{H}_\sigma)$ on the sequence $\{\sigma_k\}$ of the penalty parameter are neither restrictive nor artificial and excessive. Indeed, from the formal view-point, Assumption $(\mathcal{H}_W)$ together with (7.23) has the objective to satisfy the conditions of Lemma 7.1 [57] by setting $a_k := \Theta_k(\boldsymbol{x}^k)$, $\varepsilon_k := \xi_k$, $k = 0, 1, 2, \ldots$. Since $\varepsilon_k \geq 0$, $\sum \varepsilon_k < +\infty$, then Assumption $(\mathcal{H}_W)$ follows. But, from the point of view of the sequence $\{\sigma_k\}$ of the penalty parameter, we obtain (7.24) $\sigma_{k+1} \geq \sigma_k \geq 0$ that completely corresponds to the ideology and the methodology of the penalty parameter. In addition, it fits the description of Scheme 1, where we have to increase the value $\sigma_k > 0$ of the penalty parameter, if, for example, $W(\boldsymbol{x}^k) > 0$. Besides, it can be readily seen that together with Assumption $(\mathcal{H}_\sigma)$ it leads us to the convergence of $\{\sigma_k\}$ (see (7.31)) which, in turn, produces the existence (in fact) $\sigma_* > 0$, which can be viewed as some analogue of the exact penalty parameter value, for the objectives of the local search. Therefore, Theorem 7.1 states that $\boldsymbol{x}_*$ is a solution to the linearized problem $(\mathcal{P}L_*) = (\mathcal{P}L(\boldsymbol{x}_*, \sigma_*))$ with $\sigma_* > 0$.

Remark 7.4 According to Theorem 7.1, any cluster point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ generated by Scheme 1 is a solution to the linearized problem $(\mathcal{P}L_*)$ (linearized just by $\boldsymbol{y}^* \in \partial_c H_*(\boldsymbol{x}_*)$ at the point $\boldsymbol{x}_*$ just with the limit value $\sigma_* = \lim_{k\to\infty} \sigma_k$ of the corresponding sequence $\{\sigma_k\}$ of the penalty parameter). Since $(\mathcal{P}L_*)$ is obviously convex, the vector $\boldsymbol{x}_* \in S$ satisfies the optimality conditions

$$\boldsymbol{y}_* \in \partial_c G_*(\boldsymbol{x}_*) + N_S(\boldsymbol{x}_*) = \partial_c[G_* + \mathcal{X}_S](\boldsymbol{x}_*). \tag{7.41}$$

In what follows, we will get back to the relations with classical optimization theory and properties of the sequences $\{\boldsymbol{x}^k\}$ and $\{\boldsymbol{y}^k\}$.

However, note that the point $\boldsymbol{x}_*$ is not only a stationary point to the problem $(\mathcal{P})$ (see (7.40) and (7.41)) but, in addition, is a solution to the linearized problem $(\mathcal{P}L_*)$. Without doubts, this supplementary property improves our abilities to find a global solution to $(\mathcal{P})$.

Remark 7.5 (Stopping criteria) With the help of the chains (7.20a), (7.20c)), and (7.23), we can conclude that the following inequalities

$$\Theta_{k+1}(\boldsymbol{x}^k) - \Theta_{k+1}(\boldsymbol{x}^{k+1}) \le \tau, \tag{7.42}$$

$$\Theta_k(\boldsymbol{x}^k) + \xi_k - \Theta_{k+1}(\boldsymbol{x}^{k+1}) \le \tau, \tag{7.43}$$

$$\Phi_{k+1}(\boldsymbol{x}^k) - \Phi_{k+1}(\boldsymbol{x}^{k+1}) \le \tau, \tag{7.44}$$

can be used as stopping criteria for the LSM described by Scheme 1 and ($\mathcal{P}L_k(\sigma)$)–($\mathcal{AP}_W L_k$). For instance, with the help of (7.11), (7.20a), (7.23) and (7.44) we obtain

$$\begin{aligned} G_{k+1}(\boldsymbol{x}^k) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}^k \rangle &\le \tau + G_{k+1}(\boldsymbol{x}^{k+1}) - \langle \boldsymbol{y}(\sigma_{k+1}), \boldsymbol{x}^{k+1} \rangle \\ &= \mathcal{V}(\mathcal{P}L_k(\sigma_{k+1})) + \tau. \end{aligned}$$

Hence, $\boldsymbol{x}^k$ turns out to be a τ-solution to ($\mathcal{P}L_k(\sigma_{k+1})$), which is quite satisfactory for a local search method in the problem ($\mathcal{P}$). However, we will get back to this question in the next section.

7.6 Exact Penalty and Lagrange Multipliers

It has been shown above that the limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ produced by Scheme 1 is a solution to the linearized problem ($\mathcal{P}L_*$) and satisfies the optimality conditions (7.41). One can easily point out the drawback of (7.41), where the function $G_*(\cdot)$ is not smooth (see (7.5)), even when the entries of ($\mathcal{P}$) can be differentiable.

In order to avoid the difficulties produced by nonsmoothness, apply [48, Lemma 4.1] (see also [2, 24, 38]), which states the equivalence of the problem ($\mathcal{P}L_*$) and the following problem with parameters $(\gamma, t_{m+1}, \ldots, t_l)$:

$$\begin{cases} \text{minimize} & g_0(\boldsymbol{x}) - \langle \boldsymbol{y}_*, \boldsymbol{x} \rangle + \sigma_* \gamma + 2\sigma_* \sum\limits_{j \in \mathcal{E}} t_j \\ \text{subject to} & g_i(\boldsymbol{x}) + \sum\limits_{\substack{j \in \mathcal{I} \\ j \ne i}} h_j(\boldsymbol{x}) \le \gamma, \ i \in \mathcal{I}; \quad \sum\limits_{j \in \mathcal{I}} h_j(\boldsymbol{x}) \le \gamma, \\ & g_p(\boldsymbol{x}) \le t_p, \ h_p(\boldsymbol{x}) \le t_p, \ p \in \mathcal{E}, \\ & \boldsymbol{x} \in S, \ \gamma \in \mathbb{R}, \ \boldsymbol{t} = (t_{m+1}, \ldots, t_l)^T \in \mathbb{R}^{l-m}. \end{cases} \tag{7.45}$$

It can be readily seen that the problem (7.45) is a convex optimization problem with the variables $(\boldsymbol{x}, \gamma, \boldsymbol{t}) \in \mathbb{R}^{n+1} \times \mathbb{R}^{l-m}$, and, in contrast to the problem ($\mathcal{P}$), it has only $(m+1)+2(l-m)$ of inequalities, whereas ($\mathcal{P}$) includes equality and inequality constraints. On the other hand, if the entries of the problem ($\mathcal{P}$) are smooth, the problem (7.45) also stays smooth.

In addition, it is not difficult to show that in (7.45) Slater's condition holds. Indeed, if we look at the feasible set of (7.45), it is easy to understand, that for every $\boldsymbol{x} \in S$ we can find a number $\gamma = \gamma(\boldsymbol{x}) \in \mathbb{R}$ and a vector $\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{x}) = (t_{m+1}, \ldots, t_l)^T \in \mathbb{R}^{l-m}$ such that the inequality constraints in (7.45) are strongly fulfilled (in particular, $g_p(\boldsymbol{x}) < t_p(\boldsymbol{x}),\ h_p(\boldsymbol{x}) < t_p(\boldsymbol{x}),\ p \in \mathcal{E}$). That is what was claimed.

Hence, we have $\mu_0 = 1$ in the corresponding to (7.45) Lagrange function. Moreover, the KKT-conditions for (7.45) become necessary and sufficient for a triple $(\boldsymbol{x}_*, \gamma_*, \boldsymbol{t}_*)$ to belong to $Sol(7.45)$, since $\boldsymbol{x}_*$ is the solution to $(\mathcal{P}L_*)$. For the sake of simplicity of presentation, let us consider the case when $S = \mathbb{R}^n$, and the Lagrange function for the problem (7.45) can be written as follows [42]

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{x}, \gamma, \boldsymbol{t}; \mu_1, \ldots, \mu_m, \mu_{m+1}; \eta_{m+1}, \ldots, \eta_l, \nu_{m+1}, \ldots, \nu_l) \\
&= g_0(\boldsymbol{x}) - \langle \boldsymbol{y}_*, \boldsymbol{x} \rangle + \sigma_* \gamma + 2\sigma_* \sum_{p \in \mathcal{E}} t_p + \sum_{i \in \mathcal{I}} \mu_i [g_i(\boldsymbol{x}) + \sum_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j(\boldsymbol{x}) - \gamma] \\
&+ \mu_{m+1} [\sum_{j \in \mathcal{E}} h_j(\boldsymbol{x}) - \gamma] + \sum_{p \in \mathcal{E}} \{\eta_p [g_p(\boldsymbol{x}) - t_p] + \nu_p [h_p(\boldsymbol{x}) - t_p]\}.
\end{aligned} \tag{7.46}$$

Hence, the KKT system for $(\boldsymbol{x}_*, \gamma_*, \boldsymbol{t}_*) \in Sol(7.45)$ with (see [48, Lemma 4.1])

$$\begin{cases}
\gamma_* = & \max\{\sum\limits_{j \in \mathcal{I}} h_j(\boldsymbol{x}_*), [g_i(\boldsymbol{x}_*) + \sum\limits_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j(\boldsymbol{x}_*)],\ i \in \mathcal{I}\}, \\
t_{p*} = & \max\{g_p(\boldsymbol{x}_*), h_p(\boldsymbol{x}_*)\},\ p \in \mathcal{E},
\end{cases} \tag{7.47}$$

can be represented in the following form.

There exists a vector $(\mu_1, \ldots, \mu_m, \mu_{m+1}, \eta_{m+1}, \ldots, \eta_l, \nu_{m+1}, \ldots, \nu_l) \in \mathbb{R}_+^{m+1} \times \mathbb{R}_+^{2(l-m)}$, such that, first, the linear complementarity conditions hold:

$$\begin{aligned}
&(a) \quad \mu_i [g_i(\boldsymbol{x}_*) + \sum_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j(\boldsymbol{x}_*) - \gamma_*] = 0, && \mu_i \geq 0,\ i \in \mathcal{I}; \\
&(b) \quad \mu_{m+1} [\sum_{j \in \mathcal{I}} h_j(\boldsymbol{x}_*) - \gamma_*] = 0, && \mu_{m+1} \geq 0; \\
&(c) \quad \eta_p [g_p(\boldsymbol{x}_*) - t_{k*}] = 0 = \nu_p [h_p(\boldsymbol{x}_*) - t_{p*}], && \eta_p, \nu_p \geq 0,\ p \in \mathcal{E}.
\end{aligned} \tag{7.48}$$

In addition, the KKT equations with respect to γ and t_p take place

$$\frac{\partial \mathcal{L}(\boldsymbol{x}_*, \gamma_*, \boldsymbol{t}_*)}{\partial \gamma} = \sigma_* - \mu_{m+1} - \sum_{i \in \mathcal{I}} \mu_i = 0, \text{ i.e. } \mu_{m+1} + \sum_{i \in \mathcal{I}} \mu_i = \sigma_*, \tag{7.49}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{x}_*, \gamma_*, \boldsymbol{t}_*)}{\partial t_p} = 2\sigma_* - \eta_p - \nu_p = 0, \text{ i.e. } \eta_p + \nu_p = 2\sigma_*, \quad p \in \mathcal{E}. \tag{7.50}$$

Besides, in virtue of the theorem of Moreau-Rockafellar, it follows from the inclusion $\mathbf{0} \in \partial_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_*, \gamma_*, \boldsymbol{t}_*)$ that

$$g_0'(\boldsymbol{x}_*) - \boldsymbol{y}_* + \mu_{m+1} \sum_{i \in \mathcal{I}} h_i'(\boldsymbol{x}_*) + \sum_{i \in \mathcal{I}} \mu_i [g_i'(\boldsymbol{x}_*) + \sum_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j'(\boldsymbol{x}_*)]$$

$$+ \sum_{p \in \mathcal{E}} [\eta_p g_p'(\boldsymbol{x}_*) + \nu_p h_p'(\boldsymbol{x}_*)] = \mathbf{0} \in \mathbb{R}^n. \tag{7.51}$$

for some collection of subgradients $g_i'(\boldsymbol{x}_*) \in \partial_c g_i(\boldsymbol{x}_*)$, $h_i'(\boldsymbol{x}_*) \in \partial_c h_i(\boldsymbol{x}_*)$, $i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$, of the corresponding functions $g_i(\cdot)$, $h_i(\cdot)$ at the point $\boldsymbol{x}_*$. Furthermore, from the inclusion (7.40) with the help of the theorem of Moreau-Rockafellar, we derive that

$$\partial_c H_*(\boldsymbol{x}_*) \ni \boldsymbol{y}_* = h_0'(\boldsymbol{x}_*) + \sigma_* \left[\sum_{i \in \mathcal{I}} h_i'(\boldsymbol{x}_*) + \sum_{p \in \mathcal{E}} \left(g_p'(\boldsymbol{x}_*) + h_p'(\boldsymbol{x}_*) \right) \right]. \tag{7.52}$$

Then, on account of (7.52), the equality (7.51) yields

$$\mathbf{0} = g_0'(\boldsymbol{x}_*) - h_0'(\boldsymbol{x}_*) + (\mu_{m+1} - \sigma_*) \sum_{i \in \mathcal{I}} h_i'(\boldsymbol{x}_*) \pm \sum_{i \in \mathcal{I}} \mu_i \sum_{j \in \mathcal{I}} h_j'(\boldsymbol{x}_*)$$

$$+ \sum_{i \in \mathcal{I}} \mu_i \left[g_i'(\boldsymbol{x}_*) + \sum_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j'(\boldsymbol{x}_*) \right] + \sum_{p \in \mathcal{E}} [\eta_p g_p'(\boldsymbol{x}_*) + \nu_p h_p'(\boldsymbol{x}_*)]$$

$$- \sigma_* \sum_{p \in \mathcal{E}} [g_p'(\boldsymbol{x}_*) + h_p'(\boldsymbol{x}_*)]. \tag{7.53}$$

For our local goals, let us introduce the difference $[g'(\boldsymbol{x}) - h'(\boldsymbol{x})]$ of two subgradients $g'(\boldsymbol{x}) \in \partial_c g(\boldsymbol{x})$ and $h'(\boldsymbol{x}) \in \partial_c h(\boldsymbol{x})$, which will be referred to as the DC subgradient of the DC function $f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x})$ at the point $\boldsymbol{x}$. Besides, denote $f'(\boldsymbol{x}) := g'(\boldsymbol{x}) - h'(\boldsymbol{x})$. It is clear that, in the smooth case, the definition of DC subgradient of $f(\cdot)$ entails the classical relation with the gradients $\nabla f(\boldsymbol{x}) = \nabla g(\boldsymbol{x}) - \nabla h(\boldsymbol{x})$, so that the new definition perfectly fits the classical analysis.

With these new notations $f_i'(\boldsymbol{x}_*) = g_i'(\boldsymbol{x}_*) - h_i'(\boldsymbol{x}_*), \quad i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$, the equality (7.51) on account of (7.49) and (7.50), $\nu_p = 2\sigma_* - \eta_p$, $p \in \mathcal{E}$, takes the following form

$$\mathbf{0} = f_0'(\boldsymbol{x}_*) + (\mu_{m+1} + \sum_{i\in\mathcal{I}} \mu_i - \sigma_*) \sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_*) + \sum_{i\in\mathcal{I}} \mu_i [g_i'(\boldsymbol{x}_*) - h_i'(\boldsymbol{x}_*)]$$
$$+ \sum_{p\in\mathcal{E}} [(\eta_p - \sigma_*) g_p'(\boldsymbol{x}_*) + (\sigma_* - \eta_p) h_p'(\boldsymbol{x}_*)].$$

Finally, due to (7.49), the latter equality amounts to the following one

$$\mathbf{0} = f_0'(\boldsymbol{x}_*) + \sum_{i\in\mathcal{I}} \mu_i f_i'(\boldsymbol{x}_*) + \sum_{p\in\mathcal{E}} [(\eta_p - \sigma_*) f_p'(\boldsymbol{x}_*)]. \tag{7.54}$$

Clearly, it is the principal equation of the KKT system for the original problem $(\mathcal{P})$ at the point $\boldsymbol{x}_*$ with the Lagrange multipliers $\lambda_0 = 1$, and $\lambda_i, \ i \in \mathcal{I} \cup \mathcal{E}$, satisfying the conditions

$$\lambda_i = \mu_i \geq 0, \ i \in \mathcal{I}, \quad \lambda_p = \eta_p - \sigma_*, \ \eta_p \geq 0, \ p \in \mathcal{E}. \tag{7.55}$$

It is worth noting that for the smooth problem $(\mathcal{P})$, the KKT equation (7.54) naturally takes the classical form [2, 15, 25, 38, 57], where $f_i'(\boldsymbol{x}_*) = \nabla f_i(\boldsymbol{x}_*), \quad i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$. Furthermore, it can be readily shown that, if the vector $\boldsymbol{x}_*$ is feasible in the problem $(\mathcal{P})$, i.e. $W(\boldsymbol{x}_*) = 0$, then the complementarity conditions in the problem $(\mathcal{P})$ are also fulfilled with $\lambda_i \geq 0$, $i \in \mathcal{I}$, defined in (7.55). Indeed, on account of (7.47) and (7.48), we obtain

$$\mu_i \Big[g_i(\boldsymbol{x}_*) + \sum_{\substack{j\in\mathcal{I} \\ j\neq i}} h_j(\boldsymbol{x}_*) - \max\Big\{ \sum_{s\in\mathcal{I}} h_s(\boldsymbol{x}_*), \Big[g_p(\boldsymbol{x}_*) + \sum_{\substack{j\in\mathcal{I} \\ j\neq p}} h_j(\boldsymbol{x}_*) \Big], \ p \in \mathcal{I} \Big\} \Big]$$
$$= 0, \quad i \in \mathcal{I},$$

which obviously amounts to $\mu_i \left[f_i(\boldsymbol{x}_*) - \max\left\{0, \ f_p(\boldsymbol{x}_*), \ p \in \mathcal{I}\right\}\right] = 0, \quad i \in \mathcal{I}$.

Because of the assumption that $W(\boldsymbol{x}_*) = 0$, i.e. $f_p(\boldsymbol{x}_*) \leq 0, \ p \in \mathcal{I}$, it follows from the latter equalities that

$$\mu_i f_i(\boldsymbol{x}_*) = 0, \quad i \in \mathcal{I}, \tag{7.56}$$

as was claimed above.

Thus, the Lagrange multipliers at the limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ produced by Scheme 1 are completely defined by the Lagrange multipliers $(\mu_1, \ldots, \mu_m, \mu_{m+1}, \eta_{m+1}, \ldots, \eta_l, \nu_{m+1}, \ldots, \nu_l)$ of the auxiliary convex problem (7.45) at the point $\boldsymbol{x}_*$ and the penalty parameter $\sigma_* > 0$.

Besides, it turns out that the feasibility of the cluster point $\boldsymbol{x}_*$ can be proven under now natural conditions of Proposition 7.1.

Proposition 7.4 *Let Assumption* $(\mathcal{H}_W)$ *and the following condition hold*

$$\text{If } W(\boldsymbol{x}^k) > 0, \text{ then } \sigma_{k+1} \geq \sigma_k + \varkappa, \quad \varkappa > 0. \tag{$\mathcal{H}_\sigma 2$}$$

Then, the limit point $\boldsymbol{x}_*$ *of sequence* $\{\boldsymbol{x}^k\}$ *produced by Scheme 1, is feasible in the problem* $(\mathcal{P})$, *i.e.* $W(\boldsymbol{x}_*) = 0$.

Proof Let, by contradiction, $W(\boldsymbol{x}_*) = 2C > 0$. Then, in virtue of continuity of $W(\cdot)$ and the fact that $\boldsymbol{x}^k \to \boldsymbol{x}_*$, there exists an integer $K > 0$, such that for every $k \geq K$ we have $W(\boldsymbol{x}^k) \geq C > 0$.

Then, according to Scheme 1 and $(\mathcal{H}_\sigma 2)$ we obtain for all $k \geq K$

$$\xi_k \triangleq (\sigma_{k+1} - \sigma_k) W(\boldsymbol{x}^k) \geq \varkappa C > 0.$$

The latter implies that the series $\sum_{k=0}^{\infty} \xi_k$ does not converge, because the number sequence $\{S_m\}$ of the partial sums $S_m = \sum_{k=0}^{m} \xi_k$ does not converge. Hence, we fall in contradiction with Assumption $(\mathcal{H}_W)$(a). □

It can be readily seen that Assumption $(\mathcal{H}_\sigma 2)$ is neither restrictive nor artificial, but it is a natural, minimal, and practical assumption, suggested by Scheme 1.

But, we should not forgot that we just proved the following result.

Theorem 7.2 *The limit point* $\boldsymbol{x}_*$ *of the sequence* $\{\boldsymbol{x}^k\}$, *produced by Scheme 1, turns out to be a KKT-vector for the original problem* $(\mathcal{P})$ *with the Lagrange multipliers* $\lambda_0 = 1$, $\lambda_i \geq 0$, $i \in \mathcal{I}$, *and* $\lambda_j \in \mathbb{R}$, $j \in \mathcal{E}$, *which are completely defined by the Lagrange multipliers* $(\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\eta}) \in \mathbb{R}_+^{m+1} \times \mathbb{R}_+^{2(l-m)}$ *of the problem* (7.45) *and the penalty parameter* $\sigma_* > 0$, *all satisfying the conditions* (7.49), (7.50), (7.55), *and* (7.57).

It is worth noting that we now have the supplementary information on the relations between the Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_l)^T \in \mathbb{R}^l$, $(\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\eta}) \in \mathbb{R}_+^{m+1} \times \mathbb{R}_+^{2(l-m)}$, and the limit penalty parameter $\sigma_* > 0$ (see (7.49), (7.50), and (7.55))

$$\sum_{i \in \mathcal{I}} \lambda_i \leq \mu_{m+1} + \sum_{i \in \mathcal{I}} \mu_i = \sigma_*, \quad \eta_p + \nu_p = 2\sigma_*, \quad p \in \mathcal{E}. \tag{7.57}$$

It is clear that (7.55) and (7.57) is rather informative and helpful for computational treatment of problems $(\mathcal{P})$, $(\mathcal{P}_\sigma)$, $(\mathcal{P}_k L)$, $(\mathcal{P}L_k(\sigma))$ etc. and (7.45).

Remark 7.6 Thus, the cluster point $\boldsymbol{x}_*$ produced by Scheme 1, $(\mathcal{P}L_k(\sigma))$–$(\mathcal{A}\mathcal{P}_W L_k)$, is critical (i.e. the limit vector of the sequence $\{\boldsymbol{x}^k\}$) and also is the solution to the convex problem $(\mathcal{P}L_*)$ (at the same time being feasible in $(\mathcal{P})$, i.e. $\boldsymbol{x}_* \in \mathcal{F}$) with the limit penalty parameter $\sigma_* > 0$.

In addition, due to Theorem 7.2 it is also the KKT-point to the original problem $(\mathcal{P})$ (our principal objective). Hence, the vector $\boldsymbol{x}_*$ is considerably stronger and better than the usual stationary (KKT-) points provided by classical optimization methods [2, 25, 38, 57].

Furthermore, from the computational point of view we are interested to modify Scheme 1 so that one can perform a finite number of iterations to obtain at the final step (say) a suitable approximate solution to $(\mathcal{P}_k L)$ or $(\mathcal{P}L_k(\sigma_{k+1}))$, which is the approximate KKT-point in the problem $(\mathcal{P})$.

According to Scheme 1, at the iteration $k \in \mathbb{N}$, we have the point $\boldsymbol{x}^k \in S$ and the penalty parameter $\sigma_k \geq 0$ $(k = 0, 1, 2, \ldots)$. Suppose, for the sake of simplicity, that $S = \mathbb{R}^n$. Further, using the function $H_k(\boldsymbol{x}) = H_{\sigma_k}(\boldsymbol{x})$ defined in (7.4), we find $\boldsymbol{y}^k \in \partial_c H_k(\boldsymbol{x}^k)$ by solving $(\mathcal{P}^* L_k)$.

After that, we have to solve the following convex optimization problem (see (7.45)), which is equivalent to the linearized problem $(\mathcal{P}L_k(\sigma_+))$

$$\begin{cases} \text{minimize} & g_0(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma_+), \boldsymbol{x} \rangle + \sigma_+ \gamma + 2\sigma_+ \sum\limits_{j \in \mathcal{E}} t_j \\ \text{subject to} & g_i(\boldsymbol{x}) + \sum\limits_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j(\boldsymbol{x}) \leq \gamma, \ i \in \mathcal{I}, \ \sum\limits_{j \in \mathcal{I}} h_j(\boldsymbol{x}) \leq \gamma, \\ & g_p(\boldsymbol{x}) \leq t_p, \quad h_p(\boldsymbol{x}) \leq t_p, \quad p \in \mathcal{E}, \\ & \boldsymbol{x} \in S, \ \gamma \in \mathbb{R}, \ \boldsymbol{t} = (t_{m+1}, \ldots, t_l)^T \in \mathbb{R}^{l-m}, \end{cases} \qquad (\mathcal{P}L_k(\gamma, \boldsymbol{t}))$$

where

$$\boldsymbol{y}(\sigma_+) = \boldsymbol{y}_+ = h'_{0k} + \sigma_+ H'_W(\boldsymbol{x}^k) \in \partial_c H_{\sigma_+}(\boldsymbol{x}^k).$$

Note, that the feasible set of $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ is the same for every $k \geq 0$, which makes the computing easier. Moreover, it coincides with the feasible set of the problem (7.45). The difference resides only in the objective. Hence, Slater's condition holds and $\mu_0 = 1$. Recall that, since the problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ is convex, the KKT-system is a necessary and sufficient condition for $(\boldsymbol{x}_+, \gamma_+, \boldsymbol{t}_+)(= (\boldsymbol{x}^{k+1}, \gamma_{k+1}, \boldsymbol{t}^{k+1}))$ to be a solution to $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with

$$\begin{cases} \gamma_+ & = \max\{\sum\limits_{p \in \mathcal{I}} h_p(\boldsymbol{x}_+), \Big[g_i(\boldsymbol{x}_+) + \sum\limits_{\substack{j \in \mathcal{I} \\ j \neq i}} h_j(\boldsymbol{x}_+)\Big], \ i \in \mathcal{I}\}, \\ t_p^+ & = \max\{g_p(\boldsymbol{x}_+), h_p(\boldsymbol{x}_+)\}, \quad p \in \mathcal{E}. \end{cases} \qquad (7.58)$$

Besides, the Lagrange function has the form

$$\begin{aligned}
&\mathcal{L}_+(\boldsymbol{x},\gamma,\boldsymbol{t};\mu_1,\ldots,\mu_{m+1},\eta_{m+1},\ldots,\eta_l,\nu_{m+1},\ldots,\nu_l)\\
&= g_0(\boldsymbol{x}) - \langle \boldsymbol{y}(\sigma_+),\boldsymbol{x}\rangle + \sigma_+\gamma + 2\sigma_+\sum_{p\in\mathcal{E}} t_p + \sum_{i\in\mathcal{I}}\mu_i\Big[g_i(\boldsymbol{x}) + \sum_{\substack{j\in\mathcal{I}\\ j\neq i}} h_j(\boldsymbol{x}) - \gamma\Big]\\
&\quad + \mu_{m+1}\Big[\sum_{j\in\mathcal{I}} h_j(\boldsymbol{x}) - \gamma\Big] + \sum_{p\in\mathcal{E}}\Big\{\eta_p[g_p(\boldsymbol{x}) - t_p] + \nu_p[h_p(\boldsymbol{x}) - t_p]\Big\}.
\end{aligned} \tag{7.59}$$

It is easy to show that the linear complementarity conditions (7.48) do not change (apart from $\boldsymbol{x}_+$ replacing $\boldsymbol{x}_*$), and the conditions (7.49) and (7.50) are also similar. I.e.,

$$\begin{aligned}
&(a) \quad \sum_{i\in\mathcal{I}}\mu_i + \mu_{m+1} = \sigma_+,\\
&(b) \quad \eta_j + \nu_j = 2\sigma_+, \quad p\in\mathcal{E}.
\end{aligned} \tag{7.60}$$

In addition, as it was proven in (7.51), we obtain (denote $\boldsymbol{x}_+ := \boldsymbol{x}(\sigma_+)$ for the sake of simplicity of denotations)

$$\begin{aligned}
&g_0'(\boldsymbol{x}_+) - \boldsymbol{y}(\sigma_+) + \mu_{m+1}\sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+) + \sum_{i\in\mathcal{I}}\mu_i\Big[g_i'(\boldsymbol{x}_+) + \sum_{\substack{j\in\mathcal{I}\\ j\neq i}} h_j'(\boldsymbol{x}_+)\Big]\\
&\qquad + \sum_{p\in\mathcal{E}}[\eta_p g_p'(\boldsymbol{x}_+) + \nu_p h_p'(\boldsymbol{x}_+)] = \mathbf{0}\in\mathbb{R}^n,
\end{aligned} \tag{7.61}$$

for some collection of subgradients $g_i'(\boldsymbol{x}_+)\in\partial_c g_i(\boldsymbol{x}_+)$, $h_i'(\boldsymbol{x}_+)\in\partial_c h_i(\boldsymbol{x}_+)$ of the corresponding functions $g_i(\cdot)$, $h_i(\cdot)$, $i\in\{0\}\cup\mathcal{I}\cup\mathcal{E}$ at the point $\boldsymbol{x}_+$.

On the other hand, since by construction (see (7.6), (7.8), and (7.14))

$$\boldsymbol{y}(\sigma_+) = h_{0k}' + \sigma_+\boldsymbol{y}_W^k = h_{0k}' + \sigma_+ H_W'(\boldsymbol{x}^k)\in\partial_c H_{k+1}(\boldsymbol{x}^k),$$

where $H_{k+1}(\boldsymbol{x}) = h_0(\boldsymbol{x}) + \sigma_+ H_W(\boldsymbol{x})$, we derive from (7.61) (compare with (7.51))

$$\begin{aligned}
\mathbf{0} &= g_0'(\boldsymbol{x}_+) - h_0'(\boldsymbol{x}^k) - \sigma_+ H_W'(\boldsymbol{x}^k) \pm h_0'(\boldsymbol{x}_+)\\
&\quad + \sum_{i\in\mathcal{I}}\mu_i\Big[g_i'(\boldsymbol{x}_+) + \sum_{\substack{j\in\mathcal{I}\\ j\neq i}} h_j'(\boldsymbol{x}_+)\Big] \pm \sum_{i\in\mathcal{I}}\mu_i\Big[\sum_{j\in\mathcal{I}} h_j'(\boldsymbol{x}_+)\Big] \pm \sigma_+\sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+)\\
&\quad + \mu_{m+1}\sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+) + \sum_{p\in\mathcal{E}}\Big[\eta_p g_p'(\boldsymbol{x}_+) + \nu_p h_p'(\boldsymbol{x}_+)\Big]\\
&\quad \pm \sigma_+\sum_{i\in\mathcal{I}}\Big[g_p'(\boldsymbol{x}_+) + h_p'(\boldsymbol{x}_+)\Big].
\end{aligned}$$

Further, we will use the denotations $f_i'(\boldsymbol{x})$ introduced above for DC subgradients of the DC function $f_i(\boldsymbol{x}) = g_i(\boldsymbol{x}) - h_i(\boldsymbol{x})$ (see (7.51), (7.53), and (7.54)).

Then, the latter equality yields

$$\begin{aligned}
\mathbf{0} = {} & f_0'(\boldsymbol{x}_+) - \sigma_+ H_W'(\boldsymbol{x}^k) + [h_0'(\boldsymbol{x}_+) - h_0'(\boldsymbol{x}^k)] \\
& + \sum_{i\in\mathcal{I}} \mu_i [g_i'(\boldsymbol{x}_+) - h_i'(\boldsymbol{x}_+)] + \sum_{i\in\mathcal{I}} \mu_i \Big[\sum_{j\in\mathcal{I}} h_j'(\boldsymbol{x}_+)\Big] \\
& + \sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+)[\mu_{m+1} - \sigma_+] + \sigma_+ \sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+) \\
& + \sum_{p\in\mathcal{E}} \Big[(\eta_p - \sigma_+) g_p'(\boldsymbol{x}_+) + (2\sigma_+ - \eta_p - \sigma_+) h_p'(\boldsymbol{x}_+)\Big] \\
& + \sigma_+ \sum_{p\in\mathcal{E}} [g_p'(\boldsymbol{x}_+) + h_p'(\boldsymbol{x}_+)] \\
= {} & f_0'(\boldsymbol{x}_+) + \sum_{i\in\mathcal{I}} \mu_i f_i'(\boldsymbol{x}_+) + \sum_{j\in\mathcal{E}} (\eta_j - \sigma_+) f_j'(\boldsymbol{x}_+) + \Delta_k, \qquad (7.62)
\end{aligned}$$

where

$$\begin{aligned}
\Delta_k = {} & h_0'(\boldsymbol{x}_+) - h_0'(\boldsymbol{x}^k) - \sigma_+ H_W'(\boldsymbol{x}^k) + \Big(\sum_{i\in\mathcal{I}} \mu_i + \mu_{m+1} - \sigma_+\Big) \sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+) \\
& + \sigma_+ \Big[\sum_{i\in\mathcal{I}} h_i'(\boldsymbol{x}_+) + \sum_{j\in\mathcal{E}} \big(h_j'(\boldsymbol{x}_+) + h_j'(\boldsymbol{x}_+)\big)\Big] \\
= {} & h_0'(\boldsymbol{x}_+) + \sigma_+ H_W'(\boldsymbol{x}_+) - h_0'(\boldsymbol{x}^k) - \sigma_+ H_W'(\boldsymbol{x}^k) \\
= {} & H_{k+1}'(\boldsymbol{x}_+) - H_{k+1}'(\boldsymbol{x}^k). \qquad (7.63)
\end{aligned}$$

The latter equality takes place in virtue of (7.60)(a), while the equality (7.62) holds due to (7.60)(b).

Let us now get back to $\boldsymbol{x}^{k+1}$ instead of $\boldsymbol{x}_+$ and to σ_{k+1} instead of σ_+, respectively. Then, Eqs. (7.62) and (7.63) suggest the new stopping criteria for the iterate $\boldsymbol{x}^{k+1} = \boldsymbol{x}_+$ to be an approximate KKT-point (in the sense of (7.54) and (7.56)) in the original problem $(\mathcal{P})$:

$$\| f_0'(\boldsymbol{x}^{k+1}) + \sum_{i\in\mathcal{I}} \mu_i f_i'(\boldsymbol{x}^{k+1}) + \sum_{j\in\mathcal{E}} (\eta_j - \sigma_{k+1}) f_j'(\boldsymbol{x}^{k+1}) \| \le \frac{\tau}{2}, \qquad (7.64)$$

and

$$\|\Delta_k\| = \|H_{k+1}'(\boldsymbol{x}^{k+1}) - H_{k+1}'(\boldsymbol{x}^k)\| \le \frac{\tau}{2}. \qquad (7.65)$$

Indeed, if (7.64) and (7.65) are satisfied, then we obtain the estimation, as follows. There exists a (convex) subgradient $\mathcal{L}'_{\boldsymbol{x}}(\boldsymbol{x}^{k+1})$ of the convex Lagrange function $\mathcal{L}(\boldsymbol{x}, \gamma, \boldsymbol{t}; \boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ defined in (7.51), such that

$$\|\mathcal{L}'_{\boldsymbol{x}}(\boldsymbol{x}^{k+1})\| \leq \tau. \tag{7.66}$$

It means, that $\boldsymbol{x}^{k+1}$ can be considered as an approximate solution to the linearized (convex) problem $(\mathcal{P}L_k(\sigma_{k+1}))$, which is rather suitable and satisfactory for the local search Scheme 1.

On the other hand, the inequality (7.64) yields that the iterate $\boldsymbol{x}^{k+1}$ is the approximate KKT-point in the nonconvex original problem $(\mathcal{P})$, when $\boldsymbol{x}^{k+1}$ is feasible in $(\mathcal{P})$.

In other words, the feasibility of $\boldsymbol{x}^{k+1}$ in $(\mathcal{P})$, i.e. $W(\boldsymbol{x}^{k+1}) = 0$, together with the complementarity conditions (7.48)(a) at $\boldsymbol{x}^{k+1}$, and the satisfaction of (7.64) imply that the iterate $\boldsymbol{x}^{k+1}$ is an approximate KKT-point in the original problem $(\mathcal{P})$ with the Lagrange multipliers

$$\boldsymbol{\lambda} = (\lambda_i = \mu_i,\ i \in \mathcal{I}, \quad \lambda_j = \eta_j - \sigma_{k+1},\ \eta_j \geq 0,\ j \in \mathcal{E}) \tag{7.67}$$

defined by the Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ of the convex problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with $\boldsymbol{y}(\sigma_{k+1}) \triangleq \boldsymbol{y}(\sigma_+) = h'_{0k} + \sigma_{k+1} H'_W(\boldsymbol{x}^k)$ and the current value σ_{k+1} of penalty parameter. Thus, we just proved the following result.

Theorem 7.3 *Suppose that the solution $\boldsymbol{x}^{k+1}$ of the linearized problem $(\mathcal{P}L_k(\sigma_{k+1}))$ (i.e. $(\boldsymbol{x}^{k+1}, \gamma_{k+1}, \boldsymbol{t}^{k+1})$ is a solution to the problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with $\sigma_+ = \sigma_{k+1}$ and $(\gamma_{k+1}, \boldsymbol{t}^{k+1})$ satisfying (7.59) is feasible in the problem $(\mathcal{P})$, $W(\boldsymbol{x}^{k+1}) = 0$. Let, in addition, the criteria (7.64) and (7.65) be satisfied. Then, the vector $\boldsymbol{x}^{k+1}$ turns out to be an approximate KKT-point for the original problem $(\mathcal{P})$ with the Lagrange multipliers $\lambda_0 = 1$, λ_i, $i \in \mathcal{I} \cup \mathcal{E}$, completely defined by the equalities (7.67) with the Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ of the convex problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with $\sigma_+ = \sigma_{k+1}$ at the point $(\boldsymbol{x}^{k+1}, \gamma_{k+1}, \boldsymbol{t}^{k+1})$.*

7.7 About Stopping Criteria

It is well-known that the choice of a stopping criterion is a rather difficult task that influences results of computational experiments as well as the solution accuracy of an applied problem. However, this issue is even more challenging in new applied areas or new research directions, such as the nonconvex optimization, where we do not have many mathematical tools and theoretical results that would allow us to assess an approximation to a desired objective provided by several iterations of a new method.

Only profound mathematical investigations can help evaluate the properties of the final results of the computational process. In this regard, it would be reasonable

to propose a few combinations of different stopping criteria for Scheme 1 of local search on the base of Theorems 7.1–7.3 and Propositions 7.1–7.4. To this end, we will use the criterions (7.42), (7.43), (7.44), and (7.64)–(7.65) and well-known and obvious condition as follows ($\boldsymbol{x}_+ := \boldsymbol{x}(\sigma_+)$)

$$W(\boldsymbol{x}_+) = 0 \quad (\text{or } W(\boldsymbol{x}_+) \leq \varepsilon), \tag{7.68}$$

and

$$\|\boldsymbol{x}_+ - \boldsymbol{x}^k\| \leq \varepsilon. \tag{7.69}$$

To begin with, let us briefly discuss various properties of those criteria.

When it comes to the criterion (7.68), which has partially been always included to Scheme 1, it is quite obvious that if (7.68) is fulfilled, it only means that $\boldsymbol{x}_+$ is feasible in the problem $(\mathcal{P})$ and nothing else can be added to that fact. However, it does not suite our objectives. On the other hand, the inequality (7.69) implies that the process of convergence: $\|\boldsymbol{x}^k - \boldsymbol{x}_+\| \downarrow 0$ described in (7.29),might be closed to or situated at the terminal stage. We should keep in mind that this is only a hypothesis, which might not be accomplished.

Therefore, it is clear that a single criterion cannot provide grounds for a conclusion about the properties of the iterates $\boldsymbol{x}^k$ or $\boldsymbol{x}^{k+1}$ in question. The similar things can be said about the inequalities (7.42), (7.43), and (7.44).

In addition, it can be readily seen, that it is rather difficult to compute the values of $\Theta_{k+1}(\boldsymbol{x}^k)$ and $\Theta_{k+1}(\boldsymbol{x}^{k+1})$, because of the formulas (7.3), (7.4), (7.5), (7.17), (7.20c), and (7.21) etc. Besides, it is obvious that those computations of the values of $\Theta_{k+1}(\boldsymbol{x}^k)$ and $\Theta_{k+1}(\boldsymbol{x}^{k+1})$ are additional efforts, which are not implemented within Scheme 1. Meanwhile, the inequality (7.44) is more feasible in this sense, because, firstly, one computes $\Phi_{k+1}(\boldsymbol{x}(\sigma_+)) = G_{k+1}(\boldsymbol{x}(\sigma_+)) - \langle \boldsymbol{y}(\sigma_+), \boldsymbol{x}(\sigma_+)\rangle$ at Step 5 of Scheme 1 by solving $(\mathcal{P}L_k(\sigma_+))$. Second, $\Phi_{k+1}(\boldsymbol{x})$ is apparently easier to compute, than, for instance, $\Theta_{k+1}(\boldsymbol{x})$. Hence, the inequality (7.44) can be considered as a member of a stopping criteria system that provides some desirable properties for the iterate. In what follows, we denote $\boldsymbol{x}_+ = \boldsymbol{x}(\sigma_+)$ denoted earlier by $\boldsymbol{x}^{k+1}$.

Furthermore, as it has been said above, the inequalities (7.64) and (7.65) entail the validity of (7.66), which guarantees that $\boldsymbol{x}^{k+1}$ is an approximate solution to the problem $(\mathcal{P}L_k(\sigma_{k+1}))$, because of convexity of the latter one.

At the same time, the inequality (7.64), separately, is very important, since, together with the feasibility of $\boldsymbol{x}_+$ in $(\mathcal{P})$ and the complementarity conditions (7.48)(a) (satisfied at $\boldsymbol{x}_+$ instead of $\boldsymbol{x}_*$), it entails that the vector $\boldsymbol{x}_+$ is an approximate KKT-point to the original problem $(\mathcal{P})$ with the Lagrange multipliers $\lambda_0 = 1, \lambda_i,\ i \in \mathcal{I} \cup \mathcal{E}$, satisfying (7.55).

On the other hand, the inequality (7.65) seems to be rather far from the logic of Scheme 1, it rather appears to be, in a sense, an auxiliary tool leading to

(7.66), which is very useful and important (see above). Nevertheless, due to the presentations ($\boldsymbol{x}_+ := \boldsymbol{x}(\sigma_+)$)

$$H'_{k+1}(\boldsymbol{x}_+) - H'_{k+1}(\boldsymbol{x}^k) = h'_0(\boldsymbol{x}_+) - h'_0(\boldsymbol{x}^k) + \sigma_+ \left[H'_W(\boldsymbol{x}_+) - H'_W(\boldsymbol{x}^k)\right]$$
$$= H'_{k+1}(\boldsymbol{x}_+) - \boldsymbol{y}(\sigma_+),$$

where $H'_{k+1}(\boldsymbol{x}_+) \in \partial_c H_{k+1}(\boldsymbol{x}_+)$, $\boldsymbol{x}_+ \in \partial_c H^*_{k+1}(\boldsymbol{y}^{k+1})$, i.e. $\boldsymbol{y}^{k+1} \in \partial_c H_{k+1}(\boldsymbol{x}_+)$, one might conjecture that the inequality (7.65) is closed to (see Proposition 7.3) $\|\boldsymbol{y}^{k+1} - \boldsymbol{y}(\sigma_+)\| \le \frac{\tau}{2}$ (although there is a number of counter-examples[2, 7–10, 42, 44, 57]) which testifies to the end of the computational process of Scheme 1.

To summarize, one can propose to finish Scheme 1 with replacing Step 8 by the following step:

Step 8'. Denote $\boldsymbol{x}_+ := \boldsymbol{x}(\sigma_+)$. If

$$(a) \quad W(\boldsymbol{x}_+) = 0, \quad (\text{or } W(\boldsymbol{x}_+) \le \tau),$$
$$(b) \quad \Phi_{k+1}(\boldsymbol{x}^k) - \Phi_{k+1}(\boldsymbol{x}_+) \le \tau,$$
$$(c) \quad \|f'_0(\boldsymbol{x}_+) + \sum_{i\in\mathcal{I}} \mu_i f'_i(\boldsymbol{x}_+) + \sum_{j\in\mathcal{E}} [(\eta_j - \sigma_+) f'_j(\boldsymbol{x}_+)\| \le \frac{\tau}{2}, \tag{7.70}$$
$$(d) \quad \|\Delta_k\| = \| H'_{k+1}(\boldsymbol{x}_+) - H'_{k+1}(\boldsymbol{x}^k)\| \le \frac{\tau}{2},$$

then **stop**, $\boldsymbol{x}_+ = \boldsymbol{x}(\sigma_+)$ is the point elaborated by Scheme 1. Here the Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ are produced by solving the problem $(\mathcal{P}L_k(\sigma_+))$ (more precisely $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with $\boldsymbol{y}(\sigma_+)$ and $f'_i(\boldsymbol{x}_+) = g'_i(\boldsymbol{x}_+) - h'_i(\boldsymbol{x}_+), i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$. Otherwise, set $k := k+1$, $\sigma_{k+1} := \sigma_+$, $\boldsymbol{x}^{k+1} := \boldsymbol{x}(\sigma_+) = \boldsymbol{x}_+$ and go to Step 1.

Let us point out that, as a result, the modified Scheme 1 (i.e. with the stopping criteria (7.70) (a), (b), (c), (d)) produces the point $\boldsymbol{x}_+$, which is

(a) a feasible point in the original problem $(\mathcal{P})$;
(b) an approximate solution to the linearized problem $(\mathcal{P}L_k(\sigma_+))$, i.e. the trial $(\boldsymbol{x}_+, \gamma_+, \boldsymbol{t}_+)$ is a solution to the convex problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with $\sigma_+ > 0$ and the Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ satisfying (7.60) and (7.67);
(c) an approximate KKT-point in the original problem $(\mathcal{P})$ with the Lagrange multipliers $\lambda_0 = 1, \lambda_i,\ i \in \mathcal{I} \cup \mathcal{E}$, completely defined by the equalities

$$\lambda_i = \mu_i,\ i \in \mathcal{I}, \quad \lambda_j = \eta_j - \sigma_+,\ \eta_j \ge 0,\ j \in \mathcal{E}, \tag{7.71}$$

with the Lagrange multipliers $(\boldsymbol{\mu}, \boldsymbol{\eta}, \boldsymbol{\nu})$ of the convex problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ and the value $\sigma_+ > 0$ of the penalty parameter σ, defining the problem $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$.

Summing up, the modified Scheme 1 yields a rather strong vector $\boldsymbol{x}_+$, which is not only an approximate solution to the linearized problem $(\mathcal{P}L_k(\sigma_+))$, but an approximate KKT-point (and, off-causes, feasible) in the original problem $(\mathcal{P})$ with the Lagrange multipliers provided, in fact, by $(\mathcal{P}L_k(\gamma, \boldsymbol{t}))$ with $\sigma_+ > 0$ according to (7.71).

7.8 Conclusion

In this chapter we considered probably the most difficult nonsmooth optimization problem $(\mathcal{P})$ with the DC data, where the cost function $f_0(\cdot)$ and the equality and inequality constraints $f_i(\boldsymbol{x}) \leq 0, \quad i \in I, \quad f_j(\boldsymbol{x}) = 0, \quad j \in \mathcal{E}$, are given by DC functions. It is worth noting that any optimization problem of the kind with continuous data can be approximated by the problem $(\mathcal{P})$ with DC data at any desirable accuracy.

Furthermore, by applying the technique of the exact penalization theory [2, 3, 7, 8, 11, 22, 58, 59], the original problem $(\mathcal{P})$ is reduced to a penalized problem $(\mathcal{P}_\sigma)$ with the penalty parameter $\sigma > 0$ and the penalty function $W(\cdot)$ combining the L_∞ and L_1 reducing for the inequality and equality systems, respectively.

Let us now highlight several novelties and new results presented in the chapter.

1. The problem statement not only has the objective function and inequality constraints given by DC functions, but, in addition, includes a finite number of DC equality constraints.
2. We managed to represent the cost function $\Theta_\sigma(\cdot)$ of the penalized problem $(\mathcal{P}_\sigma)$ and the penalty function $W(\cdot)$, as well as the d.c. functions.
3. The new splitting procedures allow us to separate the convex and "anticonvex" parts in the penalized problem $(\mathcal{P}_\sigma)$ and, besides to apply the linearization with respect to the "anticonvex" part of $(\mathcal{P}_\sigma)$. In other words, we can reduce the solution of the original problem $(\mathcal{P})$ to a study of a family of convex (linearized) problems. After that, the idea of a consecutive solution of linearized (at every iteration) problems becomes obvious, which leads us to the new local search (LS) Scheme 1 with the updating of the penalty parameter value $\sigma > 0$.
4. The convergence properties of the developed LS Scheme 1 become the priority of the investigations. They produce Propositions 7.1–7.3 on the convergence of the number sequences $\{\Theta_k(\boldsymbol{x}^k)\}$ and $\{\Delta\Phi_k(\boldsymbol{x}^k)\}$, as well as the vector sequences $\{\boldsymbol{x}^k\}$ and $\{\boldsymbol{y}^k\}$ generated by Scheme 1. These convergence results allow us to prove Theorem 7.1, which states that any limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ turns out to be a solution to the convex (linearized at $\boldsymbol{x}_*$ just) problem $(\mathcal{P}L_*)$ with the limit value $\sigma_* = \lim_{k\to\infty} \sigma_k$ of the corresponding sequence $\{\sigma^k\}$ of the penalty parameter. We cannot find a similar result in the available references on nonconvex optimization (see [2, 4, 5, 15–17, 25, 27–38, 45, 50, 56–58] and the references therein). Thus, a cluster vector $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ produced by the LS Scheme 1 turns out to be obviously stronger than an usual stationary point of the problem $(\mathcal{P}_\sigma)$.

5. In order to avoid the difficulties related to the nonsmoothness of the linearized problem $(\mathcal{P}L_*)$, we introduced the convex auxiliary problem (7.45) equivalent to $(\mathcal{P}L_*)$ with $(l-m+1)$ supplementary parameters that can be used for numerical solution of problems $(\mathcal{P})$ and $(\mathcal{P}_\sigma)$. By applying the KKT-theorem to (7.45), we obtained the most unexpected and new results about the properties of the limit point $\boldsymbol{x}_*$ of the sequence $\{\boldsymbol{x}^k\}$ produced by the LS Scheme 1. It turned out that $\boldsymbol{x}_*$ is a KKT-vector for the original problem $(\mathcal{P})$ with the Lagrange multipliers which are completely defined by the Lagrange multipliers of the convex auxiliary problem (7.45) and the limit value $\sigma_* > 0$ of the sequence $\{\sigma_k\}$ of the penalty parameter $\sigma_k > 0,\ k = 0, 1, 2, \ldots$, all satisfying the conditions (7.49), (7.50), (7.55), and (7.57). Additionally, the principal KKT equation (7.54) is satisfied with d.c. subgradients $f_i'(\boldsymbol{x}_*) \triangleq g_i'(\boldsymbol{x}_*) - h_i'(\boldsymbol{x}_*)$ of the DC functions $f_i(\cdot)$, so that $g_i'(\boldsymbol{x}_*) \in \partial_c g_i(\boldsymbol{x}_*),\ h_i'(\boldsymbol{x}_*) \in \partial_c h_i(\boldsymbol{x}_*),\ i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$.

   Using the same approach at the iterate $\boldsymbol{x}^k,\ k \in \mathbb{N}$, we obtained the new stopping criteria (7.64) and (7.65) (whence (7.66) follows) , which guarantee that the iterate $\boldsymbol{x}^{k+1} \in S$ will be an approximate KKT-vector in the original problem $(\mathcal{P})$ with the Lagrange multipliers provided by the auxiliary problem (7.45) and the corresponding value σ_{k+1} of penalty parameter. Besides, the principal KKT equation (7.62) is fulfilled with the d.c. subgradients $f_i'(\boldsymbol{x}^{k+1}) \triangleq g_i'(\boldsymbol{x}^{k+1}) - h_i'(\boldsymbol{x}^{k+1})$ of the functions $f_i(\cdot)$, where $g_i'(\boldsymbol{x}^{k+1}),\ h_i'(\boldsymbol{x}^{k+1})$ are the subgradients of the convex functions $g_i(\cdot), h_i(\cdot)$ at the point $\boldsymbol{x}^{k+1} \in S,\ i \in \{0\} \cup \mathcal{I} \cup \mathcal{E}$.
6. Finally, after a discussion of the properties of various stopping criteria it was revealed that the system of four inequalities (7.70) (a), (b), (c), (d) guarantees that the iterate $\boldsymbol{x}_+$, satisfying this system, to be rather strong and competitive with respect to any other vectors provided by any local search methods or classical optimization methods.

Acknowledgement This research was partially supported by the Russian Science Foundation (project No. 15-11-20015).

References

1. Benson, H., Sen, A., Shanno, D., Vanderbei, R.: Interior-point algorithms, penalty methods and equilibrium problems. Comput. Optim. Appl. **34**(2), 155–182 (2006)
2. Bonnans, J.F., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Theoretical and Practical Aspects. Springer, Berlin (2006)
3. Burke, J.: An exact penalization viewpoint of constrained optimization. SIAM J. Control Optim. **29**(4), 968–998 (1991)
4. Byrd, R., Marazzi, M., Nocedal, J.: On the convergence of Newton iterations to non-stationary points. Math. Program. **99**(1), 127–148 (2004)
5. Byrd, R., Nocedal, J., Waltz, R.: Steering exact penalty methods for nonlinear programming. Optim. Methods Softw. **23**(2), 197–213 (2008)
6. Byrd, R., Lopez-Calva, G., Nocedal, J.: A line search exact penalty method using steering rules. Math. Program. **133**(1), 39–73 (2012)
7. Clarke, F.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)

8. Demyanov, V.: Extremum's conditions and variational calculus. High School Edition, Moscow (2005). (in Russian)
9. Demyanov, V., Vasiliev, L.: Nondifferentiable Optimization. Translation Series in Mathematics and Engineering. Springer, New York (1985)
10. Demyanov, V., Rubinov, A.: Constructive Nonsmooth Analysis. Peter Lang, Frankfurt (1995)
11. Di Pillo, G., Grippo, L.: Exact penalty functions in constrained optimization. SIAM J. Control Optim. **27**, 1333–1360 (1989)
12. Di Pillo, G., Lucidi, S., Rinaldi, F.: An approach to constrained global optimization based on exact penalty functions. J. Global Optim. **54**, 251–260 (2012)
13. Di Pillo, G., Lucidi, S., Rinaldi, F.: A derivative-free algorithm for constrained global optimization based on exact penalty functions. J. Optim. Theory Appl. **164**, 862–882 (2015)
14. Dür, M., Hiriart-Urruty, J.B.: Testing copositivity with the help of difference-of-convex optimization. Math. Program. **140**(Ser. B), 31–43 (2013)
15. Eremin, I.: The penalty method in convex programming. Soviet Math. Dokl. **8**, 459–462 (1966)
16. Floudas, C.: Deterministic Global Optimization: Theory, Methods and Application. Kluwer, Dordrecht (1999)
17. Floudas, C., Pardalos, P. (eds.): Frontiers in Global Optimization. Springer, Dordrecht (2004)
18. Gaudioso, M., Gruzdeva, T., Strekalovsky, A.: On numerical solving the spherical separability problem. J. Global Optim. **66**, 21–34 (2016)
19. Gruzdeva, T., Strekalovsky, A.: Local search in problems with nonconvex constraints. Comput. Math. Math. Phys. **47**(3), 381–396 (2007)
20. Gruzdeva, T., Strekalovsky, A.: Clique problems and nonconvex optimization. Nauka, Novosibirsk (2014). (in Russian)
21. Gruzdeva, T., Strekalovsky, A.: On solving the sum-of-ratios problem. Appl. Math. Comput. **318**, 260–269 (2018)
22. Han, S., Mangasarian, O.: Exact penalty functions in nonlinear programming. Math. Program. **17**(3), 251–269 (1979)
23. Hiriart-Urruty, J.B.: Generalized differentiability, duality and optimization for problems dealing with difference of convex functions. In: Ponstein, J. (ed.) Convexity and Duality in Optimization. Lecture Notes in Economics and Mathematical Systems, vol. 256, pp. 37–70. Springer, Berlin (1985)
24. Hiriart-Urruty, J.B.: Optimisation et Analyse Convexe. Presses Universitaires de France, Paris (1998). (in French)
25. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer, Berlin (1993)
26. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin (1996)
27. Kruger, A.: Error bounds and metric subregularity. Optimization **64**(1), 49–79 (2015)
28. Kruger, A.: Nonlinear metric subregularity. J. Optim. Theory Appl. **171**(3), 820–855 (2016)
29. Le Thi, H.A.: D.C. programming for solving a class of global optimization problems via reformulation by exact penalty. In: Bliek, C., Jermann, C., Neumaier, A. (eds.) Global Optimization and Constraint Satisfaction. Lecture Notes in Computer Science, vol. 2861, pp. 87–101. Springer, Berlin (2003)
30. Le Thi, H.A., Pham Dinh, T.: Solving a class of linearly constrained indefinite quadratic problems by DCA algorithm. J. Global Optim. **11**, 253–285 (1997)
31. Le Thi, H.A., Pham Dinh, T.: The DC programming and DCA revisited with DC model of real world nonconvex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
32. Le Thi, H.A., Pham Dinh, T.: DC programming in communication systems: challenging problems and methods. Vietnam J. Comput. Sci. **1**(1), 15–28 (2014)
33. Le Thi, H.A., Pham Dinh, T., Van Ngai, H.: Exact penalty and error bounds in DC programming. J. Global Optim. **52**(3), 509–535 (2012)
34. Le Thi, H.A., Ngai Huynh, V., Pham Dinh, T.: DC programming and DCA for general DC programs. In: Advanced Computational Methods for Knowledge Engineering, pp. 15–35. Springer, New York (2014)

35. Mascarenhas, W.: The BFGS methods with exact line search fails for nonconvex objective functions. Math. Program. **99**(1, Ser. A), 49–61 (2004)
36. Mascarenhas, W.: On the divergence of line search methods. Comput. Appl. Math. **26**(1), 129–169 (2007)
37. Mascarenhas, W.: Newton's iterates can converge to non-stationary points. Math. Program. **112**(2), 327–334 (2008)
38. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006)
39. Pang, J.S.: Three modelling paradigms in mathematical programming. Math. Program. **125**(2, Ser. B), 297–323 (2010)
40. Pang, J.S., Razaviyan, M., Alvarado, A.: Computing B-stationary points of nonsmooth DC programs. Math. Oper. Res. **42**(1), 95–118 (2016)
41. Robinson, S.: Stability theory for systems of inequalities, Part II: differentiable nonlinear systems. SIAM J. Numer. Anal. **13**(4), 497–513 (1976)
42. Rockafellar, R.: Convex Analysis. Princeton University Press, Princeton (1970)
43. Rockafellar, R.: Lagrange multipliers and optimality. SIAM Rev. **35**(2), 183–238 (1993)
44. Rockafellar, R., Wets, R.B.: Variational Analysis. Springer, New York (1998)
45. Strekalovsky, A.: Elements of Nonconvex Optimization. Nauka, Novosibirsk (2003). (in Russian)
46. Strekalovsky, A.: On solving optimization problems with hidden nonconvex structures. In: Rassias, T., Floudas, C., Butenko, S (eds.) Optimization in Science and Engineering, pp. 465–502. Springer, New York (2014)
47. Strekalovsky, A.: On local search in D.C. optimization problems. Appl. Math. Comput. **255**, 73–83 (2015)
48. Strekalovsky, A.: Global optimality conditions in nonconvex optimization. J. Optim. Theory Appl. **173**(3), 770–792 (2017)
49. Strekalovsky, A.: Global optimality conditions and exact penalization. Optim. Lett. **13**(3), 597–615 (2019). https://doi.org/0.1007/s11590-017-1214-x
50. Strekalovsky, A., Minarchenko, I.: A local search method for optimization problem with d.c. inequality constraints. Appl. Math. Model. **58**, 229–244 (2018)
51. Strekalovsky, A., Orlov, A.: Bimatrix Games and Bilinear Programming. FIZMATLIT, Moskow (2007). (in Russian)
52. Strekalovsky, A., Orlov, A., Malyshev, A.: On computational search for optimistic solutions in bilevel problems. J. Global Optim. **48**(1), 159–172 (2010)
53. Toland, J.: Duality in nonconvex optimization. J. Math. Anal. Appl. **66**, 399–415 (1978)
54. Toland, J.: A duality principle for nonconvex optimization and the calculus of variations. Arch. Ration. Mech. Anal. **71**, 41–61 (1979)
55. Toland, J.: On the stability of rotating heavy chains. J. Differ. Equ. **32**, 15–31 (1979)
56. Tuy, H.: D.C. optimization: Theory, methods and algorithms. In: Horst, R., Pardalos, P. (eds.) Handbook of Global optimization, pp. 149–216. Kluwer, Dordrecht (1995)
57. Vasiliev, F.: Optimization Methods. Factorial Press, Moscow (2002). (in Russian)
58. Zangwill, W.: Non-linear programming via penalty functions. Manag. Sci. **13**, 344–358 (1967)
59. Zaslavski, A.: Exact penalty property in optimization with mixed constraints via variational analysis. SIAM J. Optim. **23**(1), 170–187 (2013)

Chapter 8
Bundle Methods for Nonsmooth DC Optimization

Kaisa Joki and Adil M. Bagirov

Abstract This chapter is devoted to algorithms for solving nonsmooth unconstrained difference of convex optimization problems. Different types of stationarity conditions are discussed and the relationship between sets of different stationary points (critical, Clarke stationary and inf-stationary) is established. Bundle methods are developed based on a nonconvex piecewise linear model of the objective function and the convergence of these methods is studied. Numerical results are presented to demonstrate the performance of the methods.

8.1 Introduction

Nonsmooth functions represented as a difference of two convex (DC) functions constitute an important subclass of nonsmooth functions. DC functions preserve some important properties of convex functions. Many practical problems such as supervised data classification and clustering problems in machine learning [6], clusterwise linear regression [4], piecewise linear regression [7], image restoration problems in artificial intelligence [26, 27] and statistical estimation problems [9] can be formulated as an unconstrained or constrained DC optimization problem.

The first important problem when dealing with DC optimization is how to represent a given function as a DC function. In many situations, it is easy to find such representations without using any special techniques. For example, in all above mentioned application areas the objective functions can be represented as a DC function using simple operations. For some other cases, like polynomials, special algorithms have been developed to find DC representations (see e.g. [1, 11, 12]).

K. Joki (✉)
Department of Mathematics and Statistics, University of Turku, Turku, Finland
e-mail: kaisa.joki@utu.fi

A. M. Bagirov
School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, VIC, Australia
e-mail: a.bagirov@federation.edu.au

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_8

Traditionally, over the long period since 1980s, DC optimization has been considered as a branch of global optimization and various algorithms have been developed to globally solve these problems [37]. The first local search method, the DCA (DC Algorithm), for solving DC optimization problems was introduced in 1985 [33].

Over the recent years DC optimization problems, especially nonsmooth DC optimization problems, have attracted a significant attention. Several methods have been developed [2, 3, 16, 21, 23, 31, 32, 35]. Most successful among these methods are those based on the extension of the bundle method for minimizing convex functions. In this chapter, we present two versions of the bundle method for solving unconstrained DC optimization problems. The convergence properties of these methods are studied and they are tested using some academic test problems. In addition, we briefly discuss the relation between these methods and another bundle method for DC optimization.

The rest of the chapter is organized as follows. In Sect. 8.2, the unconstrained DC optimization problem is formulated and optimality conditions are discussed. The cutting plane model of convex functions is studied in Sect. 8.3. The proximal bundle method for DC optimization is presented in Sect. 8.4 and the double bundle method is given in Sect. 8.5. The piecewise concave bundle method is briefly discussed in Sect. 8.6. Numerical results are reported in Sect. 8.7. Section 8.8 concludes the chapter.

8.2 Optimality Conditions in DC Optimization

In this chapter, the interest is in unconstrained DC optimization. Therefore, we start with formally defining DC functions.

Definition 8.1 A function $f : \mathbb{R}^n \to \mathbb{R}$ is a *DC function* if it can be represented in the form

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}),$$

where functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ are convex and finite valued.

In this definition, the convex functions f_1 and f_2, used to define a DC function f, are called *DC components* whereas $f_1 - f_2$ is a *DC decomposition* of f. DC functions, defined on $\mathbb{R}^n$, are locally Lipschitz continuous (LLC) and they can be nonsmooth. From Definition 8.1, we notice that, even though DC functions are typically nonconvex, they have a structure separating their convex and concave behaviour.

The unconstrained DC optimization problem is of the form

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{8.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a DC function.

Definition 8.2 For the problem (8.1), a point $\boldsymbol{x}^* \in \mathbb{R}^n$ is a local minimizer if $f(\boldsymbol{x}^*)$ is finite and there exists $\varepsilon > 0$ such that

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \quad \text{for all } \boldsymbol{x} \in B(\boldsymbol{x}^*; \varepsilon).$$

Below we summarize commonly used necessary optimality conditions in DC optimization. More conditions are presented, for example, in [18, 26, 33].

Theorem 8.1 ([8, 18, 26, 36]) *Let functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be convex. If a point $\boldsymbol{x}^* \in \mathbb{R}^n$ is a local minimizer of a DC function $f = f_1 - f_2$, then the following conditions hold:*

$$\partial f_2(\boldsymbol{x}^*) \subseteq \partial f_1(\boldsymbol{x}^*), \tag{8.2}$$

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*) \quad \textit{and} \tag{8.3}$$

$$\partial f_1(\boldsymbol{x}^*) \cap \partial f_2(\boldsymbol{x}^*) \neq \emptyset. \tag{8.4}$$

From Theorem 8.1, we obtain definitions of three types of stationary points. First, the points satisfying the condition (8.2) are called *inf-stationary points*. Second, if a point fulfils the condition (8.3) then it is *Clarke stationary*. Finally, the condition (8.4) gives a definition for a *critical point*. The next proposition presents relationships between the sets of different stationary points, which are also illustrated in Fig. 8.1.

Proposition 8.1 ([16]) *Let S_{inf} be a set of inf-stationary points, S_{cl} be a set of Clarke stationary points and S_{cr} be a set of critical points of the DC function $f = f_1 - f_2$ with convex DC components $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$. Then*

(i) $S_{inf} \subseteq S_{cl} \subseteq S_{cr}$;
(ii) *if f_1 is differentiable in $\mathbb{R}^n$, then $S_{cl} = S_{cr}$;*
(iii) *if f_2 is differentiable in $\mathbb{R}^n$, then $S_{inf} = S_{cl} = S_{cr}$.*

Proposition 8.1 shows that inf-stationarity is the strongest condition among those presented in Theorem 8.1. Furthermore, this condition guarantees local optimality if the second DC component f_2 is a polyhedral convex function of the form

$$f_2(\boldsymbol{x}) = \max_{i=1,\ldots,m} \{\boldsymbol{a}_i^T \boldsymbol{x} + b_i\}, \tag{8.5}$$

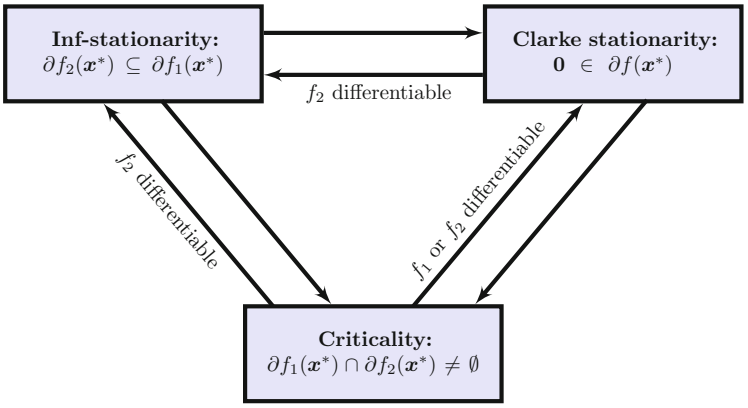

Fig. 8.1 Relationships between different stationary points

where $\boldsymbol{a}_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ and $m \in \mathbb{N}$. Therefore, it would be worthwhile to use inf-stationarity as a stopping condition to design algorithms. Unfortunately, this condition is hard to verify in practice as it requires the whole subdifferentials of DC components to be known. This is a strong requirement since the calculation of the whole subdifferential, in general, may be time consuming and in some cases even impossible. Most methods of nonsmooth optimization (NSO) typically require that only one arbitrary subgradient of an objective function can be calculated at any $\boldsymbol{x} \in \mathbb{R}^n$. In this chapter, this assumption appears in the form that at $\boldsymbol{x} \in \mathbb{R}^n$ at most one arbitrary subgradient for both DC components can be calculated and this information may not be sufficient to validate inf-stationarity.

Proposition 8.1 implies that Clarke stationarity is stronger than criticality. The former condition is widely used in algorithms for minimizing nonsmooth nonconvex functions. In the convex case, this condition is also sufficient and guarantees global optimality. Even though this condition is often utilized, it can be hard to verify for a DC function using subgradients of DC components. This follows from the subdifferential calculus rule yielding [5]

$$\partial f(\boldsymbol{x}) \subseteq \partial f_1(\boldsymbol{x}) - \partial f_2(\boldsymbol{x}). \tag{8.6}$$

Thus, the difference of the subdifferentials of DC components is an estimate for the subdifferential of f. If f_1 or f_2 is differentiable then the equality holds in (8.6) and the estimate coincides with the subdifferential of f. However, by selecting the DC decomposition in a special way this estimate can always be made as rough as desired [17]. Thus, subgradients of DC components cannot be used to verify Clarke stationarity without some extra assumptions.

The condition (8.4) defining criticality is a relaxation of inf-stationarity. Criticality is commonly used as a stopping condition in DC optimization. Unlike the conditions (8.2) and (8.3) it is quite easy to verify and it typically provides good solutions. However, one major drawback of a critical point is that it does not need to be a local optimizer or even a saddle point. In the worst case, the algorithm may stop in a point, where the original DC function f is differentiable and the direction opposite to the gradient of f would provide a descent direction decreasing the value of f [23].

As already said, inf-stationarity is the strongest condition presented in Theorem 8.1 and it always implies Clarke stationarity and criticality. In addition, a Clarke stationary point always satisfies the criticality condition. However, inverse relationships between these necessary conditions are not obtained in a general case. This means that criticality is the weakest condition. In the following, we provide examples, where the inverse relationships do not hold. We start with three examples showing that not all critical points of DC functions are Clarke stationary points. The final example gives an illustration of the case, where a Clarke stationary point is not inf-stationary.

Example 8.1 ([23]) Consider a linear function $f(x) = x,\ x \in \mathbb{R}$ for which one possible DC decomposition $f = f_1 - f_2$ can be stated by selecting

$$f_1(x) = \max\{-x, 2x\} \quad \text{and} \quad f_2(x) = \max\{-2x, x\}.$$

The subdifferentials of DC components at a point $x^* = 0$ are $\partial f_1(0) = [-1, 2]$ and $\partial f_2(0) = [-2, 1]$. From this we obtain $\partial f_1(0) \cap \partial f_2(0) = [-1, 1] \neq \emptyset$. This implies that the point x^* is a critical point. However, the function f is differentiable at $x^* = 0$ and $\partial f(0) = \{1\}$. Therefore, the point x^* is not a Clarke stationary point and does not provide any interesting feature for the function f.

Example 8.2 Consider the function $f(x) = f_1(x) - f_2(x),\ x \in \mathbb{R}$, where

$$f_1(x) = \max\{0, x\} \quad \text{and} \quad f_2(x) = \max\{0, -2x\}.$$

It is obvious that $f(x) = x$ if $x \geq 0$ and $f(x) = 2x$ if $x < 0$. Due to this the subdifferential of the function f at $x^* = 0$ is $\partial f(0) = [1, 2]$. This means that the point $x^* = 0$ is not Clarke stationary as $0 \notin \partial f(0)$. On the other hand, the subdifferentials of the DC components f_1 and f_2 at $x^* = 0$ are $\partial f_1(0) = [0, 1]$ and $\partial f_2(0) = [-2, 0]$. Since $\partial f_1(0) \cap \partial f_2(0) = \{0\} \neq \emptyset$, the point $x^* = 0$ is a critical point. However, it is not Clarke stationary.

Example 8.3 ([23]) Consider the function $f(x) = f_1(x) - f_2(x)$, $x \in \mathbb{R}$, where DC components are selected to be

$$f_1(x) = \max\{x^2, x\} \quad \text{and} \quad f_2(x) = \max\{0.5x^2, -x\}.$$

At the point $x^* = 0$, the DC components are not differentiable and their subdifferentials are $\partial f_1(0) = [0, 1]$ and $\partial f_2(0) = [-1, 0]$. Since $\partial f_1(0) \cap \partial f_2(0) = \{0\} \neq \emptyset$ the point $x^* = 0$ is a critical point. However, the original DC function f is differentiable at x^* and $\partial f(0) = \{1\}$. This shows that x^* is not Clarke stationary.

Example 8.4 Consider the function

$$f(x_1, x_2) = f_1(x_1, x_2) - f_2(x_1, x_2),$$

where $f_1(x_1, x_2) = |x_1|$ and $f_2(x_1, x_2) = |x_2|$. The subdifferentials of the DC components at $\boldsymbol{x}^* = \mathbf{0}$ are

$$\partial f_1(\mathbf{0}) = \text{conv}\{(1, 0)^T, (-1, 0)^T\} \quad \text{and} \quad \partial f_2(\mathbf{0}) = \text{conv}\{(0, 1)^T, (0, -1)^T\}.$$

However, the subdifferential of the function f at $\boldsymbol{x}^* = \mathbf{0}$ is

$$\partial f(\mathbf{0}) = \text{conv}\{(1, 1)^T, (-1, 1)^T, (1, -1)^T, (-1, -1)^T\}.$$

From this we easily deduce that $\partial f_2(\mathbf{0}) \not\subset \partial f_1(\mathbf{0})$. However, $\mathbf{0} \in \partial f(\mathbf{0})$ meaning that the point $\boldsymbol{x}^* = \mathbf{0}$ is Clarke stationary, but not inf-stationary.

It is worth noting, that instead of criticality it is also possible to test its generalization, the so-called *ε-criticality*, requiring that at a point $\boldsymbol{x}^* \in \mathbb{R}^n$ the condition

$$\partial_\varepsilon f_1(\boldsymbol{x}^*) \cap \partial_\varepsilon f_2(\boldsymbol{x}^*) \neq \emptyset$$

holds for $\varepsilon \geq 0$. The smaller the parameter $\varepsilon > 0$ is, the more accurate approximation of criticality is obtained. Naturally, ε-criticality coincides with criticality with the selection $\varepsilon = 0$.

Using ε-subdifferentials of DC components one can get the following necessary and sufficient condition for global optimality.

Theorem 8.2 ([18]) *Let functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be convex. A point $\boldsymbol{x}^* \in \mathbb{R}^n$ is a global minimizer of a DC function $f = f_1 - f_2$, if and only if*

$$\partial_\varepsilon f_2(\boldsymbol{x}^*) \subseteq \partial_\varepsilon f_1(\boldsymbol{x}^*) \quad \textit{for all } \varepsilon \geq 0.$$

Unfortunately, this condition is hard to utilize in practise, since it requires availability of the entire ε-subdifferentials of the DC components for all $\varepsilon \geq 0$. The calculation of the ε-subdifferential is a challenging task even for not very complex nonsmooth functions.

8.3 Convex Cutting Plane Model of a DC Component

Before presenting bundle methods for nonsmooth DC optimization, we introduce a classical convex cutting plane model used in bundle methods (see, e.g., [19, 24, 28, 30, 34]). This model serves as a basic tool for building an approximation of a DC function by treating DC components separately. This enables us to utilize explicitly the DC structure and to capture knowledge about both the convexity and concavity of the objective.

We require that at any point $\boldsymbol{x} \in \mathbb{R}^n$ one can calculate the values of the DC components f_1 and f_2. In addition, as already said, we assume that whenever necessary we can compute arbitrary subgradients $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{x})$ and $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{x})$ for both DC components or just for one of them.

In bundle methods, the basic idea is to approximate the subdifferential of the objective function by storing subgradients from the previous iterations into a bundle. We utilize a similar idea, but instead of one bundle, we construct separate bundles for both DC components at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$, namely, $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$. More precisely, these sets are defined in the form

$$\mathcal{B}_i^k = \left\{ (\boldsymbol{y}_j, f_i(\boldsymbol{y}_j), \boldsymbol{\xi}_{i,j}) \mid j \in J_i^k \right\} \quad \text{for } i = 1, 2,$$

where $\boldsymbol{y}_j \in \mathbb{R}^n$ is an auxiliary point from a previous iteration, $\boldsymbol{\xi}_{i,j} \in \partial f_i(\boldsymbol{y}_j)$ is an arbitrary subgradient and J_i^k is a nonempty set of indices. In general, the index sets J_1^k and J_2^k need not to be the same, but the current iteration point $\boldsymbol{x}_k$ has to be always included in both of them.

Elements of $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$ can be used to linearize the DC components f_1 and f_2, respectively. The classical *cutting plane model* for the function f_i, $i = 1, 2$ at the point $\boldsymbol{x}_k \in \mathbb{R}^n$ is given by

$$\hat{f}_i^k(\boldsymbol{x}) = \max_{j \in J_i^k} \left\{ f_i(\boldsymbol{y}_j) + \boldsymbol{\xi}_{i,j}^T (\boldsymbol{x} - \boldsymbol{y}_j) \right\} \quad \text{for } \boldsymbol{x} \in \mathbb{R}^n. \tag{8.7}$$

By introducing the variable $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}_k$, defining the *linearization error*

$$\alpha_{i,j}^k = f_i(\boldsymbol{x}_k) - f_i(\boldsymbol{y}_j) - \boldsymbol{\xi}_{i,j}^T (\boldsymbol{x}_k - \boldsymbol{y}_j) \quad \text{for all } j \in J_i^k \tag{8.8}$$

and denoting by

$$\Delta_i^k(\boldsymbol{d}) = \max_{j \in J_i^k} \left\{ \boldsymbol{\xi}_{i,j}^T \boldsymbol{d} - \alpha_{i,j}^k \right\}, \tag{8.9}$$

the model (8.7) can be rewritten as

$$\hat{f}_i^k(\boldsymbol{x}_k + \boldsymbol{d}) = \max_{j \in J_i^k} \left\{ f_i(\boldsymbol{x}_k) + \boldsymbol{\xi}_{i,j}^T \boldsymbol{d} - \alpha_{i,j}^k \right\} = f_i(\boldsymbol{x}_k) + \Delta_i^k(\boldsymbol{d}). \tag{8.10}$$

The important properties of the cutting plane model are presented in the next proposition.

Proposition 8.2 *Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then for any $\boldsymbol{d} \in \mathbb{R}^n$ and $j \in J_i^k$ the following properties hold:*

(i) $\hat{f}_i^k(\boldsymbol{x}_k + \boldsymbol{d}) \leq f_i(\boldsymbol{x}_k + \boldsymbol{d})$;
(ii) $\Delta_i^k(\boldsymbol{d}) \leq f_i(\boldsymbol{x}_k + \boldsymbol{d}) - f_i(\boldsymbol{x}_k)$;
(iii) $\hat{f}_i^k(\boldsymbol{y}_j) = f_i(\boldsymbol{y}_j)$ *and* $\alpha_{i,j}^k \geq 0$;
(iv) $\hat{f}_i^k$ *is convex.*

Proof The properties (i) and (iii) follow directly from Definition 1.7 of the subdifferential of f_i. By combining the property (i) with (8.10) we obtain the property (ii). It is obvious that a function represented as a maximum of finitely many linear functions is convex which implies convexity of the function $\hat{f}_i^k$. □

From the reformulation (8.10) of the cutting plane model, we see that the main ingredients to build the model are the subgradients and linearization errors. In addition, whenever a new point $\boldsymbol{x}_{k+1}$ differing from the current iteration point $\boldsymbol{x}_k$ is produced, linearization errors need to be updated. Such update can be performed easily using the formula

$$\alpha_{i,j}^{k+1} = \alpha_{i,j}^k + f_i(\boldsymbol{x}_{k+1}) - f_i(\boldsymbol{x}_k) - \boldsymbol{\xi}_{i,j}^T(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$$

requiring no knowledge about the auxiliary points $\boldsymbol{y}_j$ and function values $f_i(\boldsymbol{y}_j)$ and it is sufficient to store only subgradients and linearization errors. Therefore the bundle $\mathcal{B}_i^k$ is presented in the form

$$\mathcal{B}_i^k = \left\{ (\boldsymbol{\xi}_{i,j},\, \alpha_{i,j}^k) \mid j \in J_i^k \right\} \quad \text{for } i = 1, 2.$$

8.4 Proximal Bundle Method PBDC

In this section, we describe the proximal bundle method for unconstrained nonsmooth DC optimization (PBDC) originally introduced in [21] and also presented in [20]. This method guarantees approximate ε-criticality for the obtained solution.

The idea in the model construction is to build the convex cutting plane model for both DC components. Due to this, the original DC function is approximated with a nonconvex DC cutting plane model obtained by combining the separate approximations of the DC components. This way the model takes explicitly into account the DC structure of the objective and incorporates both the convex and concave behaviour.

Model and Direction Finding We start with introducing the model of the DC function f, where the idea is to substitute both DC components with their convex cutting plane models (8.7). Therefore, the *nonconvex DC cutting plane model* of f at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$ is given by

$$\tilde{f}^k(\boldsymbol{x}) = \hat{f}_1^k(\boldsymbol{x}) - \hat{f}_2^k(\boldsymbol{x}). \tag{8.11}$$

Even though this approximation is nonconvex, it is piecewise linear and models both convex and concave behaviour of the objective. Using the variable $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}_k$ and applying (8.10) we get

$$\tilde{f}^k(\boldsymbol{x}_k + \boldsymbol{d}) = f(\boldsymbol{x}_k) + \Delta_1^k(\boldsymbol{d}) - \Delta_2^k(\boldsymbol{d}).$$

One illustration of the model is given in Example 8.5.

Example 8.5 Consider the DC function $f(x) = f_1(x) - f_2(x)$, $x \in \mathbb{R}$ with

$$f_1(x) = \max\{0.85x^2 + x + 2, -x + 4.5\} \quad \text{and}$$
$$f_2(x) = \max\{0.5x^2, x + 1.5\}.$$

The convex cutting plane models of the DC components are constructed using the function values and subgradients of the DC components at $x = -6, -2$ and 2 and depicted in Fig. 8.2. The overall approximation of f is given in Fig. 8.3.

The model is used to determine a search direction. A quadratic stabilizing term is added into the model to keep it local enough and to guarantee the existence of the solution [28]. This leads to the following global optimization problem

$$\begin{cases} \text{minimize} & P^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) - \Delta_2^k(\boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{8.12}$$

where $t > 0$ is a proximity parameter. The solution to this problem is denoted by $\boldsymbol{d}_t^k \in \mathbb{R}^n$.

From Proposition 8.2(ii), we obtain that the terms $\Delta_1^k(\boldsymbol{d}_t^k)$ and $\Delta_2^k(\boldsymbol{d}_t^k)$ estimate the changes in the values of f_1 and f_2, respectively. In addition, we show in the next lemma that the term $\Delta_1^k(\boldsymbol{d}_t^k) - \Delta_2^k(\boldsymbol{d}_t^k)$ is always nonpositive and, thus, it can be seen as a predicted descent for the actual decrease in the objective function value.

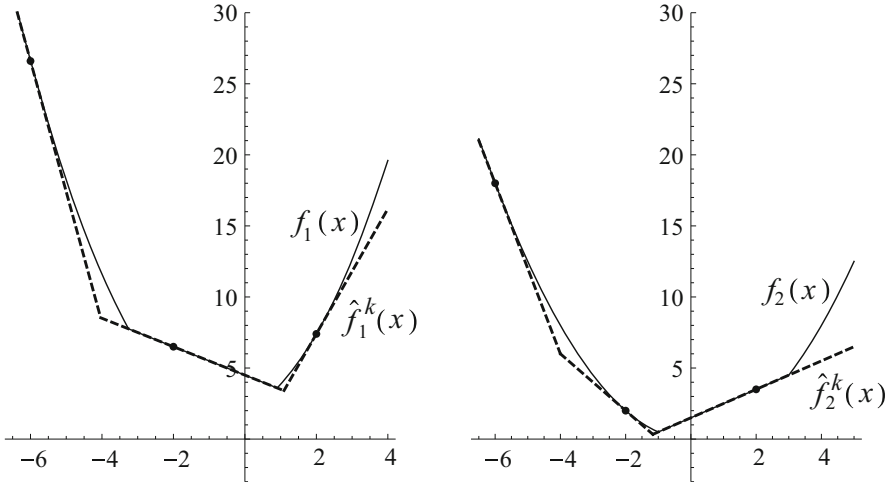

Fig. 8.2 The convex cutting plane models $\hat{f}_1^k(x)$ and $\hat{f}_2^k(x)$ of the DC components $f_1(x)$ and $f_2(x)$

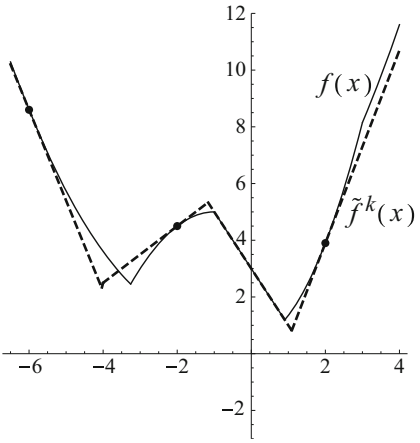

Fig. 8.3 The nonconvex DC cutting plane model $\tilde{f}^k(x)$ of the objective function $f(x)$

Lemma 8.1 *For any $t > 0$, we obtain $\Delta_1^k(\boldsymbol{d}_t^k) - \Delta_2^k(\boldsymbol{d}_t^k) \leq -\frac{1}{2t}\|\boldsymbol{d}_t^k\|^2 \leq 0$.*

Proof The direction $\boldsymbol{d}' = \mathbf{0}$ is a feasible solution for the problem (8.12). Since both bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$ contain a subgradient calculated at $\boldsymbol{x}_k$ it follows that among nonnegative linearization errors there is at least one whose value is zero. Thus, we get

$$P^k\left(\boldsymbol{d}'\right) = \Delta_1^k(\mathbf{0}) - \Delta_2^k(\mathbf{0}) + \frac{1}{2t}\|\mathbf{0}\|^2 = \max_{j \in J_1^k}\left\{-\alpha_{1,j}^k\right\} - \max_{j \in J_2^k}\left\{-\alpha_{2,j}^k\right\} = 0,$$

which implies that the global solution $\boldsymbol{d}_t^k$ of the problem (8.12) satisfies

$$P^k\left(\boldsymbol{d}_t^k\right) = \Delta_1^k\left(\boldsymbol{d}_t^k\right) - \Delta_2^k\left(\boldsymbol{d}_t^k\right) + \frac{1}{2t}\|\boldsymbol{d}_t^k\|^2 \leq P\left(\boldsymbol{d}'\right) = 0.$$

This proves the claim. □

Another useful property is that the search direction $\boldsymbol{d}_t^k$ is always bounded.

Lemma 8.2 *For any $t > 0$, it holds that*

$$\|\boldsymbol{d}_t^k\| \leq 2t\left(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|\right),$$

where $\boldsymbol{\xi}_1(\boldsymbol{x}_k) \in \partial f_1(\boldsymbol{x}_k)$ and $\|\boldsymbol{\xi}_{2,\max}\| = \max\limits_{j \in J_2^k}\{\|\boldsymbol{\xi}_{2,j}\|\}$.

Proof Using the Eq. (8.9), we deduce the following inequalities:

$$\Delta_2^k(\boldsymbol{d}) \leq \max_{j \in J_2^k} \boldsymbol{\xi}_{2,j}^T \boldsymbol{d} \leq \|\boldsymbol{\xi}_{2,\max}\|\|\boldsymbol{d}\| \quad \text{for all } \boldsymbol{d} \in \mathbb{R}^n$$

and

$$\Delta_1^k(\boldsymbol{d}) \geq \boldsymbol{\xi}_{1,j}^T \boldsymbol{d} - \alpha_{1,j}^k \quad \text{for all } j \in J_1^k.$$

By combining them and taking into account that the bundle $\mathcal{B}_1^k$ contains the element $\left(\boldsymbol{\xi}_1(\boldsymbol{x}_k), 0\right)$, where $\boldsymbol{\xi}_1(\boldsymbol{x}_k) \in \partial f_1(\boldsymbol{x}_k)$, we obtain

$$\Delta_1^k(\boldsymbol{d}) - \Delta_2^k(\boldsymbol{d}) \geq \boldsymbol{\xi}_1(\boldsymbol{x}_k)^T \boldsymbol{d} - \|\boldsymbol{\xi}_{2,\max}\|\|\boldsymbol{d}\| \geq -(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)\|\boldsymbol{d}\|$$

for all $\boldsymbol{d} \in \mathbb{R}^n$. Then Lemma 8.1 implies that

$$-\frac{1}{2t}\|\boldsymbol{d}_t^k\|^2 \geq \Delta_1^k(\boldsymbol{d}_t^k) - \Delta_2^k(\boldsymbol{d}_t^k) \geq -(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)\|\boldsymbol{d}_t^k\|.$$

This completes the proof. □

The problem (8.12) is nonconvex when $|J_2^k| \geq 2$ and it is not straightforward to distinguish its global solution among the local ones. However, the objective function P^k is still DC with the DC components $\Delta_1^k(\boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2$ and $\Delta_2^k(\boldsymbol{d})$ and the second DC component is polyhedral convex. This enables us to use a specific approach utilized in [25, 26, 33] to find the global solution.

The main observation in the approach is that the objective P^k can be rewritten in the form

$$P^k(\boldsymbol{d}) = \min_{i \in J_2^k}\left\{P_i^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) - \boldsymbol{\xi}_{2,i}^T \boldsymbol{d} + \alpha_{2,i}^k + \frac{1}{2t}\|\boldsymbol{d}\|^2\right\}$$

and, thus, we obtain

$$\min_{\boldsymbol{d}\in\mathbb{R}^n} \min_{i\in J_2^k} \left\{ P_i^k(\boldsymbol{d}) \right\} = \min_{i\in J_2^k} \min_{\boldsymbol{d}\in\mathbb{R}^n} \left\{ P_i^k(\boldsymbol{d}) \right\}$$

showing that the original nonconvex problem can be replaced by a collection of convex subproblems. Due to this, for each $i \in J_2^k$ we solve a convex nonsmooth subproblem

$$\begin{cases} \text{minimize} & P_i^k(\boldsymbol{d}) = \Delta_1^k(\boldsymbol{d}) - \boldsymbol{\xi}_{2,i}^T \boldsymbol{d} + \alpha_{2,i}^k + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{8.13}$$

whose solution is denoted by $\boldsymbol{d}_t^k(i)$ and the global solution $\boldsymbol{d}_t^k$ of the problem (8.12) is obtained by setting

$$\boldsymbol{d}_t^k = \boldsymbol{d}_t^k(i^*), \qquad \text{where } i^* = \arg\min_{i\in J_2^k} \left\{ P_i^k\left(\boldsymbol{d}_t^k(i)\right) \right\}.$$

This means that to solve the problem (8.12) one needs to solve $|J_2^k|$ subproblems and select the best solution. The bigger the size of the bundle $\mathcal{B}_2^k$ is, the more time-consuming it is to obtain the search direction. In practice, the computational burden can be controlled by restricting the size of $\mathcal{B}_2^k$, since the only requirement is that $|J_2^k| \geq 1$.

For each $i \in J_2^k$ the subproblem (8.13) can be reformulated as a quadratic programming problem

$$\begin{cases} \text{minimize} & v + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & (\boldsymbol{\xi}_{1,j} - \boldsymbol{\xi}_{2,i})^T \boldsymbol{d} - (\alpha_{1,j}^k - \alpha_{2,i}^k) \leq v, \quad j \in J_1^k, \\ & v \in \mathbb{R},\ \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{8.14}$$

Therefore, the global solution $\boldsymbol{d}_t^k$ can be obtained by solving only smooth subproblems (8.14) or their dual counterparts

$$\begin{cases} \text{minimize} & \frac{1}{2}t\|\sum\limits_{j\in J_1^k} \lambda_j \boldsymbol{\xi}_{1,j} - \boldsymbol{\xi}_{2,i}\|^2 + \sum\limits_{j\in J_1^k} \lambda_j \alpha_{1,j}^k - \alpha_{2,i}^k \\ \text{subject to} & \sum\limits_{j\in J_1^k} \lambda_j = 1, \\ & \lambda_j \geq 0, \quad j \in J_1^k. \end{cases} \tag{8.15}$$

The relationship between the optimal solutions $(v_t^k(i),\ \boldsymbol{d}_t^k(i))$ and $(\lambda_{t,j}^k(i),\ j \in J_1^k)$ to the primal problem (8.14) and the dual problem (8.15), respectively, is given as [21]:

$$\boldsymbol{d}_t^k(i) = -t\Big(\sum_{j\in J_1^k} \lambda_{t,j}^k(i)\boldsymbol{\xi}_{1,j} - \boldsymbol{\xi}_{2,i}\Big), \tag{8.16}$$

$$v_t^k(i) = -\frac{1}{t}\|\boldsymbol{d}_t^k(i)\|^2 - \sum_{j\in J_1^k} \lambda_{t,j}^k(i)\alpha_{1,j}^k + \alpha_{2,i}^k. \tag{8.17}$$

Algorithm The PBDC method for unconstrained DC optimization is presented in Algorithm 8.1 and its basic structure contains all the characteristic steps used in standard bundle methods including the direction finding, descent test, serious steps and null steps.

To guarantee ε-criticality, two different stopping criteria are used. The first criterion is tested in Step 1 after each serious step and it is formulated using the difference of subgradients of DC components. If this difference is small enough, then the current iteration point $\boldsymbol{x}_k$ satisfies approximate criticality. The second stopping criterion in Step 4 compares the approximations of the ε-subdifferentials of the DC components and, if the difference between ε-subgradients is small enough, then approximate ε-criticality is achieved.

Algorithm 8.1: PBDC

Data: The stopping tolerance $\delta \in (0,1)$, the proximity measure $\varepsilon > 0$, the decrease parameters $r, c \in (0,1)$, the increase parameter $R > 1$, the descent parameter $m \in (0,1)$ and the (overestimated) Lipschitz constants $L_1 > 0$ and $L_2 > 0$ of f_1 and f_2, respectively.

Step 0. (*Initialization*) Select $\boldsymbol{x}_0 \in \mathbb{R}^n$ and set $\theta = \varepsilon/\max\{2L_1, 2L_2, 1\}$. Calculate $\boldsymbol{\xi}_1(\boldsymbol{x}_0) \in \partial f_1(\boldsymbol{x}_0)$ and $\boldsymbol{\xi}_2(\boldsymbol{x}_0) \in \partial f_2(\boldsymbol{x}_0)$. Initialize $\mathcal{B}_1^0 = \{(\boldsymbol{\xi}_1(\boldsymbol{x}_0), 0)\}$ and $\mathcal{B}_2^0 = \{(\boldsymbol{\xi}_2(\boldsymbol{x}_0), 0)\}$, $\boldsymbol{\xi}_{2,\max} = \mathbf{0}$, and $k = 0$.

Step 1. (*Criticality*) If $\|\boldsymbol{\xi}_1(\boldsymbol{x}_k) - \boldsymbol{\xi}_2(\boldsymbol{x}_k)\| < \delta$, then **stop** with $\boldsymbol{x}^* = \boldsymbol{x}_k$ as the final solution.

Step 2. (*Proximity parameter*) If $\|\boldsymbol{\xi}_2(\boldsymbol{x}_k)\| > \|\boldsymbol{\xi}_{2,\max}\|$, then $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_2(\boldsymbol{x}_k)$. Set

$$t_{\min} = \frac{r\theta}{2(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)}, \tag{8.18}$$

$t_{\max} = Rt_{\min}$ and $\eta = r\, t_{\min}\, \delta$. Choose $t \in [t_{\min}, t_{\max}]$.

Step 3. (*Search direction*) Calculate the search direction $\boldsymbol{d}_t^k$ as a solution of (8.12).

Step 4. (ε-*criticality*) If $\|\boldsymbol{d}_t^k\| < \eta$, then set

$$J_1^k = J_1^k \setminus \{j \in J_1^k \mid \alpha_{1,j}^k > \varepsilon\}, \quad J_2^k = J_2^k \setminus \{j \in J_2^k \mid \alpha_{2,j}^k > \varepsilon\}$$

and calculate values $\boldsymbol{\xi}_1^*$ and $\boldsymbol{\xi}_2^*$ such that

$$\|\boldsymbol{\xi}_1^* - \boldsymbol{\xi}_2^*\| = \begin{cases} \text{minimize} & \|\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2\| \\ \text{subject to} & \boldsymbol{\xi}_1 \in \operatorname{conv}\{\boldsymbol{\xi}_{1,j} \mid j \in J_1^k\}, \\ & \boldsymbol{\xi}_2 \in \operatorname{conv}\{\boldsymbol{\xi}_{2,j} \mid j \in J_2^k\}. \end{cases} \tag{8.19}$$

If $\|\boldsymbol{\xi}_1^* - \boldsymbol{\xi}_2^*\| < \delta$, then **stop** with $\boldsymbol{x}^* = \boldsymbol{x}_k$ as the final solution. Otherwise set $t_{\max} = t_{\max} - r\,(t_{\max} - t_{\min})$, select the value $t \in [t_{\min}, t_{\max}]$ and go back to Step 3.

Step 5. (*Descent test*) Set $\boldsymbol{y} = \boldsymbol{x}_k + \boldsymbol{d}_t^k$. If

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \leq m\big(\Delta_1^k(\boldsymbol{d}_t^k) - \Delta_2^k(\boldsymbol{d}_t^k)\big), \tag{8.20}$$

then set $\boldsymbol{x}_{k+1} = \boldsymbol{y}$ and go to Step 8.

Step 6. (*Parameter update*) If $f(\boldsymbol{y}) > f(\boldsymbol{x}_0)$ and $\|\boldsymbol{d}_t^k\| > \theta$, then set $t = t - r(t - t_{\min})$ and go to Step 3.

Step 7. (*Bundle update*) If

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \geq -m\big(\Delta_1^k(\boldsymbol{d}_t^k) - \Delta_2^k(\boldsymbol{d}_t^k)\big),$$

then set $t = t - c(t - t_{\min})$. Calculate $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$ and $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{y})$ together with $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t^k$ and $\alpha_2 = f_2(\boldsymbol{x}_k) - f_2(\boldsymbol{y}) + \boldsymbol{\xi}_2^T \boldsymbol{d}_t^k$. Update the bundle $\mathcal{B}_1^k = \mathcal{B}_1^k \cup \{(\boldsymbol{\xi}_1, \alpha_1)\}$. If $\Delta_2^k(\boldsymbol{d}_t^k) \leq 0$ then update the bundle $\mathcal{B}_2^k = \mathcal{B}_2^k \cup \{(\boldsymbol{\xi}_2, \alpha_2)\}$ and test if $\|\boldsymbol{\xi}_2\| > \|\boldsymbol{\xi}_{2,\max}\|$ in which case set $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_2$, update $t_{\min}$ using (8.18) and set $\eta = r t_{\min} \delta$. Go to Step 3.

Step 8. (*Serious step*) Select $\mathcal{B}_1^{k+1} \subseteq \mathcal{B}_1^k$ and $\mathcal{B}_2^{k+1} \subseteq \mathcal{B}_2^k$ and update the linearization errors in $\mathcal{B}_1^{k+1}$ and $\mathcal{B}_2^{k+1}$. Calculate $\boldsymbol{\xi}_1(\boldsymbol{x}_{k+1}) \in \partial f_1(\boldsymbol{x}_{k+1})$ and $\boldsymbol{\xi}_2(\boldsymbol{x}_{k+1}) \in \partial f_2(\boldsymbol{x}_{k+1})$. Set $\mathcal{B}_1^{k+1} = \mathcal{B}_1^{k+1} \cup \{(\boldsymbol{\xi}_1(\boldsymbol{x}_{k+1}), 0)\}$ and $\mathcal{B}_2^{k+1} = \mathcal{B}_2^{k+1} \cup \{(\boldsymbol{\xi}_2(\boldsymbol{x}_{k+1}), 0)\}$. Update $k = k + 1$ and go to Step 1.

Remark 8.1 In Algorithm 8.1, we first define the enlargement parameter $\theta > 0$ depending on the (overestimated) Lipschitz constants $L_1 > 0$ and $L_2 > 0$ of the DC components f_1 and f_2 on the set $\mathcal{F}_\varepsilon = \{\boldsymbol{x} \in \mathbb{R}^n \mid d(\boldsymbol{x}, \mathcal{F}_0) \leq \varepsilon\}$, respectively. Here $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$, $d(\boldsymbol{x}, \mathcal{F}_0) = \inf\{\|\boldsymbol{x} - \boldsymbol{s}\| \mid \boldsymbol{s} \in \mathcal{F}_0\}$ and $\boldsymbol{x}_0 \in \mathbb{R}^n$ is a starting point. The local parameter, the so-called local proximity measure $\eta > 0$, is updated each time Step 2 is entered and possibly also in Step 7. In addition, the proximity parameter $t \in [t_{\min}, t_{\max}]$ is selected anew after each serious step and decreased during null steps (Steps 4, 6 and 7) to improve the model. Overall, the parameter selections together with parameter updates are inspired by [13–15].

As we have already seen in the previous subsection, the size of the bundle $\mathcal{B}_2^k$ affects the complexity of the search direction problem (8.12). Therefore, the size

of $\mathcal{B}_2^k$ needs to be kept limited. The only requirement is that the current iteration point needs to always belong to this bundle. Thus, the size of $\mathcal{B}_2^k$ should satisfy the condition $|J_2^k| \geq 1$. In addition, in Step 8 of Algorithm 8.1 the bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$ can be chosen independently. This means that we can also decide to omit the previous information and continue with empty bundles. This is possible since before going back to Step 1 we always insert the element calculated at the new iteration point into both bundles.

Convergence Analysis Next, we prove the finite convergence of the PBDC method to a point satisfying the approximate ε-criticality. To show this the following assumptions are needed:

Assumption 8.1 *The set $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is compact.*

Assumption 8.2 *Lipschitz constants of f_1 and f_2 or their overestimates are known on the set $\mathcal{F}_\varepsilon = \{\boldsymbol{x} \in \mathbb{R}^n \mid d(\boldsymbol{x}, \mathcal{F}_0) \leq \varepsilon\}$, where $\varepsilon > 0$ is the proximity measure. These constants are denoted by $L_1 > 0$ and $L_2 > 0$, respectively.*

First, we prove the following two useful lemmas.

Lemma 8.3 *If the condition (8.20) in Step 5 of Algorithm 8.1 is not satisfied, then the new element $(\boldsymbol{\xi}_1, \alpha_1)$ of $\mathcal{B}_1^k$ computed in Step 7 satisfies*

$$\boldsymbol{\xi}_1^T \boldsymbol{d}_t^k - \alpha_1 > m\Delta_1^k(\boldsymbol{d}_t^k) + (1-m)\Delta_2^k(\boldsymbol{d}_t^k).$$

Proof Whenever the condition (8.20) is not satisfied, we obtain

$$\begin{aligned} f_1(\boldsymbol{y}) - f_1(\boldsymbol{x}_k) &> m\Big(\Delta_1(\boldsymbol{d}_t^k) - \Delta_2(\boldsymbol{d}_t^k)\Big) + \Big(f_2(\boldsymbol{y}) - f_2(\boldsymbol{x}_k)\Big) \\ &\geq m\Delta_1^k(\boldsymbol{d}_t^k) + (1-m)\Delta_2^k(\boldsymbol{d}_t^k), \end{aligned}$$

where the second inequality follows from Proposition 8.2(ii). The result can be obtained from this by noticing that $f_1(\boldsymbol{y}) - f_1(\boldsymbol{x}_k) = \boldsymbol{\xi}_1^T \boldsymbol{d}_t^k - \alpha_1$, where $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$ and $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t^k$. □

Lemma 8.4 *Let Assumption 8.1 be valid. During each iteration k of Algorithm 8.1*

(i) *$\boldsymbol{x}_k \in \mathcal{F}_\varepsilon$ and $\boldsymbol{y}_j \in \mathcal{F}_\varepsilon$ for all $j \in J_1^k \cup J_2^k$;*
(ii) *there exists $K > 0$ such that $\|\boldsymbol{x}_k - \boldsymbol{y}_j\| \leq K$ for all $j \in J_1^k \cup J_2^k$;*
(iii) *$\boldsymbol{\xi}_{i,j} \in \partial f_i(\boldsymbol{y}_j)$ and $\alpha_{i,j}^k$ for all $j \in J_i^k$ and $i = 1, 2$ are bounded;*
(iv) *t_{min} is monotonically decreasing and bounded from below with a positive threshold;*
(v) *$t_{\max}$ is bounded from above.*

Proof

(i) Since each new iteration point decreases the value of the objective it follows that points $\boldsymbol{x}_k$ belong to the set $\mathcal{F}_\varepsilon$. In addition, Step 6 of Algorithm 8.1

guarantees that each point $\boldsymbol{y}_j$ inserted into the bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$ belongs to $\mathcal{F}_\varepsilon$.

(ii) The claim follows from (i), since Assumption 8.1 ensures that the set $\mathcal{F}_\varepsilon$ is compact.

(iii) Since a DC component f_i is convex on $\mathbb{R}^n$ it is LLC and has a Lipschitz constant $L_i > 0$ on a compact set. By taking into account (i), this yields that $\|\boldsymbol{\xi}_{i,j}\| \leq L_i$ for each $j \in J_i^k$ and $i = 1, 2$. This together with the property (ii) shows the boundedness of linearization errors, since

$$|\alpha_{i,j}^k| \leq |f_i(\boldsymbol{x}_k) - f_i(\boldsymbol{y}_j)| + \|\boldsymbol{\xi}_{i,j}\|\|\boldsymbol{x}_k - \boldsymbol{y}_j\| \leq L_i\|\boldsymbol{x}_k - \boldsymbol{y}_j\| + L_i K \leq 2L_i K$$

for each $j \in J_i^k$ and $i = 1, 2$.

(iv) During the iteration k of Algorithm 8.1, parameter $t_{\min}$ is either decreased or kept the same and this shows the first property. From the proof of the property (iii) we can obtain that

$$t_{\min} \geq \bar{t}_{\min} = \frac{r\theta}{2L_1 + 2L_2} > 0 \tag{8.21}$$

yielding the positive lower bound.

(v) Whenever $t_{\max}$ is set in Step 2 of Algorithm 8.1, the stopping condition in Step 1 cannot hold. Therefore, we have that

$$\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\| \geq \|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_2(\boldsymbol{x}_k)\| \geq \|\boldsymbol{\xi}_1(\boldsymbol{x}_k) - \boldsymbol{\xi}_2(\boldsymbol{x}_k)\| \geq \delta.$$

From this we can deduce that

$$t_{\max} \leq \bar{t}_{\max} = \frac{Rr\theta}{2\delta} \tag{8.22}$$

which proves the claim. □

In order to prove the finite convergence of Algorithm 8.1, we need to show that there does not exist any infinite loop. Therefore, we start with showing that it is impossible to execute an infinite sequence of consecutive null steps during one iteration.

Proposition 8.3 *Let Assumptions 8.1 and 8.2 be valid. At any iteration k, for any $\delta > 0$ and $\varepsilon > 0$, Algorithm 8.1 can pass through Steps 3–7 only finitely many times before entering Step 8 or fulfilling the stopping condition in Step 4.*

Proof To prove the proposition we show that there does not exist an infinite sequence of consecutive null steps, containing Steps from 3 to 7, during the k-th iteration of Algorithm 8.1. We index by $i \in \mathcal{I}$ all the quantities obtained during the i-th null step. Next, we consider separately all cases, which could possibly generate the infinite sequence.

Case 1 Step 4 cannot be performed infinitely many times. Indeed, in this case according to Lemma 8.4(iv) the safeguard parameter $t_{\max}$ will be decreased infinitely many times and as a result will become arbitrarily close to $t_{\min}$. Thus, the proximity parameter t becomes arbitrarily close to $t_{\min}$, since $t \in [t_{\min}, t_{\max}]$, meaning that asymptotically t falls below the threshold

$$\rho = \frac{\theta}{2(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)}. \tag{8.23}$$

This together with Lemma 8.2 yields that from some point on we always have $\|\boldsymbol{d}_t^k\| \leq \theta$. Therefore, if Step 6 of Algorithm 8.1 is entered, it is not executed and we move to Step 7, where the obtained subgradients $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ belong to $\partial_\varepsilon f_1(\boldsymbol{x}_k)$ and $\partial_\varepsilon f_2(\boldsymbol{x}_k)$, respectively. This is due to the selection $\theta = \varepsilon/\max\{2L_1, 2L_2, 1\}$ and Theorem 1.3. Altogether, the above considerations mean that there exists an iteration after which the bundles $\mathcal{B}_1^k$ and $\mathcal{B}_2^k$ contain only ε-subgradients when Step 4 is entered. Therefore, no subgradients are removed from the bundles in Step 4. In addition, according to (8.16) the solution $\boldsymbol{d}_t$ of the problem (8.12) is always of the form

$$\boldsymbol{d}_t^k = -t\left(\sum_{j \in J_1^k} \lambda_{t,j}^k(i^*)\boldsymbol{\xi}_{1,j} - \boldsymbol{\xi}_{2,i^*}\right),$$

where $i^* \in J_2^k$ and $\sum_{j \in J_1^k} \lambda_{t,j}^k(i^*) = 1$. Thus, $\boldsymbol{d}_t^k/t$ is a feasible solution of the problem (8.19) and

$$\frac{1}{t}\|\boldsymbol{d}_t^k\| < \frac{\eta}{t} = \frac{t_{\min}}{t} r\delta < \delta$$

since $\|\boldsymbol{d}_t^k\| < \eta$ whenever Step 4 is executed. Therefore, Step 4 cannot be called infinitely many times.

Case 2 Step 6 cannot be executed infinitely many times. The proximity parameter t decreases at each execution of Step 6. Thus, t converges to $t_{\min}$ due to Lemma 8.4(iv). This means that after a finite number of steps t falls below the threshold ρ given in (8.23). According to Lemma 8.2, $\|\boldsymbol{d}_t^k\| < \theta$ when $t < \rho$. Thus, Step 6 can be executed only finitely many times.

Case 3 Step 7 cannot be performed infinitely many times. To simplify the notation let $\{\boldsymbol{d}_t^i\}$, $\{t_i\}$ and $\{\eta_i\}$ be the sequences obtained during null steps $i \in \mathcal{I}$. First, $\{t_i\}$ and $\{\eta_i\}$ converge to positive limits $\hat{t} > 0$ and $\hat{\eta} > 0$, since by Lemma 8.4(iv) both sequences are nonincreasing and bounded from below with a positive threshold. Second, Lemma 8.2 and Lemma 8.4(iii) imply that $\{\boldsymbol{d}_t^i\}$ is bounded and, thus, it has a convergent subsequence for $i \in \mathcal{I}' \subseteq \mathcal{I}$ converging to a limit $\hat{\boldsymbol{d}}$. In addition, we get that the sequences $\{\Delta_1^k(\boldsymbol{d}_t^i)\}$ and $\{\Delta_2^k(\boldsymbol{d}_t^i)\}$ are bounded. Therefore, they also have convergent subsequences for $i \in \mathcal{I}'' \subseteq \mathcal{I}'$ and these limits are denoted by $\hat{\Delta}_1^k$ and

$\hat{\Delta}_2^k$. As a consequence of Lemma 8.1 and $\|\boldsymbol{d}_t^i\| > \eta_i$, we get the inequality

$$\Delta_1^k(\boldsymbol{d}_t^i) - \Delta_2^k(\boldsymbol{d}_t^i) \leq -\frac{1}{2t_i}\|\boldsymbol{d}_t^i\|^2 \leq -\frac{\eta_i^2}{2t_i} < 0 \quad \text{for all } i \in \mathcal{I} \tag{8.24}$$

implying

$$\hat{\Delta}_1^k - \hat{\Delta}_2^k \leq -\frac{\hat{\eta}^2}{2\hat{t}} < 0. \tag{8.25}$$

Next, consider two successive indices $r, s \in \mathcal{I}''$. During the null step r we obtain a new element $(\boldsymbol{\xi}_1^r, \alpha_1^r)$ for the DC component f_1 and according to Lemma 8.1 it satisfies $(\boldsymbol{\xi}_1^r)^T\boldsymbol{d}_t^r - \alpha_1^r > m\Delta_1^k(\boldsymbol{d}_t^r) + (1-m)\Delta_2^k(\boldsymbol{d}_t^r)$. In addition, it follows from (8.9) that $\Delta_1^k(\boldsymbol{d}_t^s) \geq (\boldsymbol{\xi}_1^r)^T\boldsymbol{d}_t^s - \alpha_1^r$, which together with the previous inequality gives

$$\Delta_1^k(\boldsymbol{d}_t^s) - m\Delta_1^k(\boldsymbol{d}_t^r) + (m-1)\Delta_2^k(\boldsymbol{d}_t^r) > (\boldsymbol{\xi}_1^r)^T(\boldsymbol{d}_t^s - \boldsymbol{d}_t^r).$$

Passing to the limit yields $(1-m)(\hat{\Delta}_1^k - \hat{\Delta}_2^k) > 0$ and as $m \in (0,1)$, this contradicts (8.25). Thus, Step 7 cannot be entered infinitely many times. □

Now, we are ready to prove the finite convergence of the PBDC method which means that serious steps cannot be repeated infinitely many times.

Theorem 8.3 *Let Assumptions 8.1 and 8.2 be valid. For any $\delta > 0$ and $\varepsilon > 0$, Algorithm 8.1 terminates after a finite number of iterations at a point $\boldsymbol{x}^*$ satisfying the approximate ε-criticality condition*

$$\|\boldsymbol{\xi}_1^* - \boldsymbol{\xi}_2^*\| \leq \delta \ \textit{with}\ \boldsymbol{\xi}_1^* \in \partial_\varepsilon f_1(\boldsymbol{x}^*) \ \textit{and}\ \boldsymbol{\xi}_2^* \in \partial_\varepsilon f_2(\boldsymbol{x}^*).$$

Proof Algorithm 8.1 terminates when the stopping criteria either in Step 1 or in Step 4 are satisfied. Both conditions guarantee approximate ε-criticality. Assume the contrary, that is Algorithm 8.1 never terminates. Then, we have an infinite sequence $\{\boldsymbol{x}_k\}$ and due to Proposition 8.3 each iteration point $\boldsymbol{x}_k$ is obtained after a finite number of null steps. In addition, at each iteration the sufficient descent condition (8.20) is satisfied guaranteeing that

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq m\big(\Delta_1^k(\boldsymbol{d}_t^k) - \Delta_2^k(\boldsymbol{d}_t^k)\big).$$

By combining this with (8.24) and taking into account the bounds (8.21) and (8.22) we obtain

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\frac{m\eta_k^2}{2t_k} \leq -\frac{m(r\bar{t}_{\min}\delta)^2}{2\bar{t}_{\max}} < 0.$$

By summing up the first k of the above inequalities we get

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_0) \leq -k\sigma,$$

where $\sigma = m(r\bar{t}_{\min}\delta)^2/(2\bar{t}_{\max}) > 0$. Therefore, we can conclude

$$\lim_{k\to\infty} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_0) \leq -\infty.$$

This is a contradiction since local Lipschitz continuity of f and Assumption 8.1 guarantee that the objective f is bounded from below. □

8.5 Double Bundle Method DBDC

The PBDC method, described above, terminates at solutions satisfying the approximate ε-criticality condition. This condition is easily tested by utilizing the DC structure, but as we have already demonstrated critical points have some drawbacks. Therefore, the PBDC method may terminate at a point which is not a local minimizer or even a saddle point. To avoid this undesirable feature, we now recall the double bundle method for unconstrained DC optimization (DBDC) introduced in [23] and presented also in [20].

The DBDC method is the successor of the PBDC method. It is based on the same DC cutting plane model that is used in the PBDC and the main structure of the DBDC method resembles to that of the PBDC. A significant difference between these two methods is that the DBDC method involves a special escape procedure to find descent directions at critical points which are not Clarke stationary. The use of such a procedure allows one to get stronger convergence results than those obtained for the PBDC method.

Calculating a Subgradient of a DC Function When the DC structure is available, we typically want to use it through the whole algorithm. However, to guarantee Clarke stationarity we need to calculate subgradients of the original DC function. It follows from (8.6) that, in general, the subdifferentials of the DC components cannot be used for this purpose and, therefore, subgradients of the DC components cannot be utilized to verify Clarke stationarity. However, the careful selection of the subgradients of the DC components enables us to bypass this difficulty and to obtain subgradients of the DC function.

Finite valued convex functions defined on $\mathbb{R}^n$ are directionally differentiable. Under this assumption the DC function is also directionally differentiable. For a convex DC component f_i, the directional derivative at $\boldsymbol{x} \in \mathbb{R}^n$ in the direction $\boldsymbol{d} \in \mathbb{R}^n$ can be given in the form (see, e.g., [5])

$$f_i'(\boldsymbol{x}; \boldsymbol{d}) = \max \left\{ \boldsymbol{\xi}^T \boldsymbol{d} \mid \boldsymbol{\xi} \in \partial f_i(\boldsymbol{x}) \right\} \quad \text{for } i = 1, 2.$$

For any $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{d} \neq \mathbf{0}$, we define the set

$$G_i(\boldsymbol{x}; \boldsymbol{d}) = \left\{ \boldsymbol{\xi} \in \partial f_i(\boldsymbol{x}) \mid \boldsymbol{\xi}^T \boldsymbol{d} = f_i'(\boldsymbol{x}; \boldsymbol{d}) \right\} \quad \text{for } i = 1, 2,$$

which consists of subgradients yielding the directional derivative of f_i. Since the directional derivative of a finite valued convex function defined on $\mathbb{R}^n$ is LLC it is differentiable almost everywhere. Then at a point $\boldsymbol{x} \in \mathbb{R}^n$ there exists a set $T_{DC}(\boldsymbol{x})$ of full measure such that both sets $G_1(\boldsymbol{x}; \boldsymbol{d})$ and $G_2(\boldsymbol{x}; \boldsymbol{d})$ are singletons for all $\boldsymbol{d} \in T_{DC}(\boldsymbol{x})$. Next, we show that directions $\boldsymbol{d} \in T_{DC}(\boldsymbol{x})$ have a central role when the DC components are used to determine a subgradient of the DC function at a point $\boldsymbol{x}$.

Theorem 8.4 *Let $f = f_1 - f_2$ be a DC function, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{d} \in T_{DC}(\boldsymbol{x})$, $G_1(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}_1\}$ and $G_2(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}_2\}$. Then*

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x}).$$

Proof We start with defining the set

$$U_i(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi} \in \mathbb{R}^n \mid \text{there exists } \{\boldsymbol{v}_k\} \text{ and } \{t_k\}, \boldsymbol{v}_k \in \partial f_i(\boldsymbol{x} + t_k \boldsymbol{d}), \\ \boldsymbol{v}_k \to \boldsymbol{\xi} \text{ and } t_k \downarrow 0 \text{ as } k \to \infty\}$$

for $i = 1, 2$. Since both f_1 and f_2 are convex they are weakly semismooth. Then applying Definition 1.11 we get

$$U_i(\boldsymbol{x}; \boldsymbol{d}) \subseteq G_i(\boldsymbol{x}; \boldsymbol{d}) \quad \text{for } i = 1, 2.$$

By combining this with the definition of $U_i(\boldsymbol{x}; \boldsymbol{d})$ we obtain that for any $\varepsilon > 0$ there exists $t_0 > 0$ such that

$$\partial f_i(\boldsymbol{x} + t\boldsymbol{d}) \subset U_i(\boldsymbol{x}; \boldsymbol{d}) + B(\mathbf{0}; \varepsilon) \subseteq G_i(\boldsymbol{x}; \boldsymbol{d}) + B(\mathbf{0}; \varepsilon) = \{\boldsymbol{\xi}_i\} + B(\mathbf{0}; \varepsilon)$$

for $i = 1, 2$ and any $t \in (0, t_0)$. This implies that

$$\|\boldsymbol{v} - \boldsymbol{\xi}_1\| < \varepsilon \quad \text{and} \quad \|\boldsymbol{w} - \boldsymbol{\xi}_2\| < \varepsilon \tag{8.26}$$

for all $\boldsymbol{v} \in \partial f_1(\boldsymbol{x} + t\boldsymbol{d})$, $\boldsymbol{w} \in \partial f_2(\boldsymbol{x} + t\boldsymbol{d})$ and $t \in (0, t_0)$. In addition, any $\boldsymbol{\xi}_t \in \partial f(\boldsymbol{x} + t\boldsymbol{d})$ can be expressed as $\boldsymbol{\xi}_t = \boldsymbol{v}_t - \boldsymbol{w}_t$ where $\boldsymbol{v}_t \in \partial f_1(\boldsymbol{x} + t\boldsymbol{d})$ and $\boldsymbol{w}_t \in \partial f_2(\boldsymbol{x} + t\boldsymbol{d})$. Therefore, by taking into account inequalities (8.26) we obtain

$$\|\boldsymbol{\xi}_t - (\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)\| \leq \|\boldsymbol{v}_t - \boldsymbol{\xi}_1\| + \|\boldsymbol{w}_t - \boldsymbol{\xi}_2\| < 2\varepsilon$$

for all $\boldsymbol{\xi}_t \in \partial f(\boldsymbol{x} + t\boldsymbol{d})$ and $t \in (0, t_0)$. This shows that

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x} + t\boldsymbol{d}) + B(\mathbf{0}; 2\varepsilon) \quad \text{for all } t \in (0, t_0). \tag{8.27}$$

On the other hand, upper semicontinuity of ∂f [8] means that for any $\varepsilon > 0$ there exists $t_1 > 0$ such that

$$\partial f(\boldsymbol{x} + t\boldsymbol{d}) \subset \partial f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon) \quad \text{for all } t \in (0, t_1) \tag{8.28}$$

which together with (8.27) implies that for any $\varepsilon > 0$

$$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x}) + B(\mathbf{0}; 3\varepsilon).$$

This proves that $\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x})$. □

Theorem 8.4 shows that if the direction is selected with care, then a subgradient of the DC function can be obtained by using the subgradients of the DC components. The difficulty in applying Theorem 8.4 is that its claim does not necessarily hold for an arbitrary direction. Next, we show that at a point $\boldsymbol{x} \in \mathbb{R}^n$ for any direction $\boldsymbol{d} \in \mathbb{R}^n$ one can construct a direction $\bar{\boldsymbol{d}} \in T_{DC}(\boldsymbol{x})$ which is sufficiently close to $\boldsymbol{d}$.

Theorem 8.5 *Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{d} \in \mathbb{R}^n$ be any direction such that $\boldsymbol{d} \neq \mathbf{0}$, and assume that for a DC function $f = f_1 - f_2$ the subdifferentials $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$ are polytopes. Then for a given $\bar{\boldsymbol{g}} \in V = \{\boldsymbol{g} \in \mathbb{R}^n \mid \boldsymbol{g} = (g_1, \ldots, g_n),\ |g_i| = 1,\ \text{for all } i\}$, there exists $\alpha_0 \in (0, 1]$ such that for all $\alpha \in (0, \alpha_0]$:*

(i) $\bar{\boldsymbol{d}}(\alpha) = \boldsymbol{d} + \boldsymbol{e}^n(\alpha) \in T_{DC}(\boldsymbol{x})$, *where* $\boldsymbol{e}^n(\alpha) = (\alpha\bar{g}_1, \alpha^2\bar{g}_2, \ldots, \alpha^n\bar{g}_n)$;
(ii) $G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha)) \subseteq G_1(\boldsymbol{x}; \boldsymbol{d})$ *and* $G_2(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha)) \subseteq G_2(\boldsymbol{x}; \boldsymbol{d})$;
(iii) $f'(\boldsymbol{x}; \boldsymbol{d}) = (\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)^T\boldsymbol{d}$ *for* $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$ *and* $\boldsymbol{\xi}_2 \in G_2(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$;
(iv) $\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2 \in \partial f(\boldsymbol{x})$ *for* $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$ *and* $\boldsymbol{\xi}_2 \in G_2(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$.

Proof The proof of (i) and (ii) is technical and it is given in [23]. To prove the case (iii) notice that $f_1'(\boldsymbol{x}; \boldsymbol{d}) = \boldsymbol{\xi}_1^T\boldsymbol{d}$ for $\boldsymbol{\xi}_1 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$, $f_2'(\boldsymbol{x}; \boldsymbol{d}) = \boldsymbol{\xi}_2^T\boldsymbol{d}$ for $\boldsymbol{\xi}_2 \in G_1(\boldsymbol{x}; \bar{\boldsymbol{d}}(\alpha))$ and $f'(\boldsymbol{x}; \boldsymbol{d}) = f_1'(\boldsymbol{x}; \boldsymbol{d}) - f_2'(\boldsymbol{x}; \boldsymbol{d})$. The property (iv) is obtained directly from Theorem 8.4 by taking into account (i). □

The above theorem shows that at a point $\boldsymbol{x} \in \mathbb{R}^n$ we can always make a small controlled change into any direction $\boldsymbol{d}$ to obtain $\bar{\boldsymbol{d}} \in T_{DC}(\boldsymbol{x})$. Moreover, if for the altered direction $\bar{\boldsymbol{d}}$ the sets $G_1(\boldsymbol{x}; \bar{\boldsymbol{d}})$ and $G_2(\boldsymbol{x}; \bar{\boldsymbol{d}})$ are singletons, then the elements in those sets can be used to define a subgradient of f. In addition, the property (iii) of Theorem 8.5 guarantees that the calculated subgradients of the DC components define also the directional derivative of the DC function in the direction $\boldsymbol{d}$. Thus, the altered direction $\bar{\boldsymbol{d}}$ is sufficiently close to $\boldsymbol{d}$. We also notice that the previous theorem requires that at a point $\boldsymbol{x} \in \mathbb{R}^n$ the subdifferentials $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$ of the DC components are polytopes. This is not a very restrictive assumption and in practical applications it is nearly always fulfilled.

Algorithm The DBDC method is described in Algorithm 8.3 and this method resembles the PBDC method (Algorithm 8.1). The significant difference is the stopping condition, since whenever a candidate solution is obtained, we execute the escape procedure. This procedure either guarantees Clarke stationarity or produces a descent direction yielding a sufficient decrease in the value of the objective. Thus,

whenever necessary we are able to escape from non-Clarke stationary points of a DC function.

In the escape procedure, the goal is to identify whether the point $\boldsymbol{x} \in \mathbb{R}^n$ is Clarke stationary or not. For this reason, we need to approximate the Goldstein ε-subdifferential $\partial_\varepsilon^G f(\boldsymbol{x})$ of f (see Definition 1.10) for some small $\varepsilon > 0$. Moreover, the smaller the parameter ε is, the more accurate approximation of $\partial f(\boldsymbol{x})$ is obtained. Let U_k denote this approximation during iteration k and we have that $U_k \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for all $k \geq 1$. In order to detect Clarke stationarity, we need to solve the problem

$$\begin{cases} \text{minimize} & \frac{1}{2}\|\boldsymbol{u}\|^2 \\ \text{subject to} & \boldsymbol{u} \in U_k, \end{cases} \tag{8.29}$$

providing the solution $\boldsymbol{u}_k$. If the norm of $\boldsymbol{u}_k$ is smaller than the stopping tolerance $\delta > 0$, then the point $\boldsymbol{x}$ satisfies approximate Clarke stationarity. Otherwise, we need to continue the verification procedure. In addition, if the point $\boldsymbol{x}$ is non-Clarke stationary then the final solution $\boldsymbol{x}^+$ provided by Algorithm 8.2 decreases the value of f.

Algorithm 8.2: Escape procedure

Data: The point $\boldsymbol{x} \in \mathbb{R}^n$ under consideration, the stopping tolerance $\delta > 0$, the proximity measure $\varepsilon > 0$ and the descent parameter $\hat{m} \in (0, 1)$.

Step 0. (*Initialization*) Select a direction $\boldsymbol{d}_1 \in \{\boldsymbol{d} \in \mathbb{R}^n \mid \|\boldsymbol{d}\| = 1\}$. Set $\tilde{\boldsymbol{x}} = \boldsymbol{x}$, $U_0 = \emptyset$ and $k = 1$.

Step 1. (*New subgradient*) Find $\bar{\boldsymbol{d}}_k(\alpha) \in T_{DC}(\tilde{\boldsymbol{x}})$ using $\boldsymbol{d}_k$. Compute subgradients $\boldsymbol{\xi}_{1,k} \in \partial f_1(\tilde{\boldsymbol{x}})$ and $\boldsymbol{\xi}_{2,k} \in \partial f_2(\tilde{\boldsymbol{x}})$ such that $\boldsymbol{\xi}_{1,k} \in G_1(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_k(\alpha))$ and $\boldsymbol{\xi}_{2,k} \in G_2(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}_k(\alpha))$. Set $\boldsymbol{\xi}_k = \boldsymbol{\xi}_{1,k} - \boldsymbol{\xi}_{2,k}$ and $U_k = \operatorname{conv}\left\{U_{k-1} \bigcup \{\boldsymbol{\xi}_k\}\right\}$.

Step 2. (*Clarke stationarity*) Compute $\boldsymbol{u}_k$ by solving the problem (8.29). If

$$\|\boldsymbol{u}_k\| \leq \delta, \tag{8.30}$$

then **stop** with $\boldsymbol{x}^+ = \boldsymbol{x}$.

Step 3. (*Search direction*) Compute the search direction $\boldsymbol{d}_{k+1} = -\boldsymbol{u}_k / \|\boldsymbol{u}_k\|$.

Step 4. (*Descent test*) If

$$f'(\boldsymbol{x}; \boldsymbol{d}_{k+1}) > -\hat{m}\|\boldsymbol{u}_k\|, \tag{8.31}$$

then set $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ and $k = k + 1$ and go to Step 1.

Step 5. (*Step-length*) Calculate the step-length

$$\beta^* = \arg\max\left\{\beta > 0 \mid f(\boldsymbol{x} + \beta \boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) \leq -\hat{m}\beta\|\boldsymbol{u}_k\|\right\}. \tag{8.32}$$

If $\beta^* \geq \varepsilon$, then **stop** with $\boldsymbol{x}^+ = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$. Otherwise, set $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ and $k = k + 1$ and go to Step 1.

Algorithm 8.3: DBDC

Data: The stopping tolerance $\delta \in (0,1)$, the proximity measure $\varepsilon > 0$, the enlargement parameter $\theta > 0$, the decrease parameters $r, c \in (0,1)$, the increase parameter $R > 1$ and the descent parameters $m, \hat{m} \in (0,1)$.

Step 0. (*Initialization*) Select $\boldsymbol{x}_0 \in \mathbb{R}^n$. Calculate $\boldsymbol{\xi}_1(\boldsymbol{x}_0) \in \partial f_1(\boldsymbol{x}_0)$ and $\boldsymbol{\xi}_2(\boldsymbol{x}_0) \in \partial f_2(\boldsymbol{x}_0)$. Initialize $\mathcal{B}_1^0 = \{(\boldsymbol{\xi}_1(\boldsymbol{x}_0), 0)\}$ and $\mathcal{B}_2^0 = \{(\boldsymbol{\xi}_2(\boldsymbol{x}_0), 0)\}$, $\boldsymbol{\xi}_{2,\max} = \mathbf{0}$, and $k = 0$.

Step 1. (*Criticality*) If $\|\boldsymbol{\xi}_1(\boldsymbol{x}_k) - \boldsymbol{\xi}_2(\boldsymbol{x}_k)\| < \delta$, then $\boldsymbol{d}_t^k = \mathbf{0}$ and go to Step 4.

Step 2. (*Proximity parameter*) If $\|\boldsymbol{\xi}_2(\boldsymbol{x}_k)\| > \|\boldsymbol{\xi}_{2,\max}\|$, then $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_2(\boldsymbol{x}_k)$. Set

$$t_{\min} = \frac{r\theta}{2(\|\boldsymbol{\xi}_1(\boldsymbol{x}_k)\| + \|\boldsymbol{\xi}_{2,\max}\|)} \tag{8.33}$$

and $t_{\max} = Rt_{\min}$. Choose $t \in [t_{\min}, t_{\max}]$.

Step 3. (*Search direction*) Calculate the search direction $\boldsymbol{d}_t^k$ as a solution of (8.12).

Step 4. (*Escape procedure*) If $\|\boldsymbol{d}_t^k\| < \delta$, then execute the escape procedure Algorithm 8.2 for the point $\boldsymbol{x}_k$ to obtain $\boldsymbol{x}^+$. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}^+$ and go to Step 8.

Step 5. (*Descent test*) Set $\boldsymbol{y} = \boldsymbol{x}_k + \boldsymbol{d}_t^k$. If

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \leq m\big(\Delta_1(\boldsymbol{d}_t^k) - \Delta_2(\boldsymbol{d}_t^k)\big), \tag{8.34}$$

then set $\boldsymbol{x}_{k+1} = \boldsymbol{y}$ and go to Step 8.

Step 6. (*Parameter update*) If $f(\boldsymbol{y}) > f(\boldsymbol{x}_0)$ and $\|\boldsymbol{d}_t^k\| > \theta$, then set $t = t - r(t - t_{\min})$. Go to Step 3.

Step 7. (*Bundle update*) If

$$f(\boldsymbol{y}) - f(\boldsymbol{x}_k) \geq -m\big(\Delta_1(\boldsymbol{d}_t^k) - \Delta_2(\boldsymbol{d}_t^k)\big),$$

then set $t = t - c(t - t_{\min})$. Calculate $\boldsymbol{\xi}_1 \in \partial f_1(\boldsymbol{y})$ and $\boldsymbol{\xi}_2 \in \partial f_2(\boldsymbol{y})$ together with $\alpha_1 = f_1(\boldsymbol{x}_k) - f_1(\boldsymbol{y}) + \boldsymbol{\xi}_1^T \boldsymbol{d}_t^k$ and $\alpha_2 = f_2(\boldsymbol{x}_k) - f_2(\boldsymbol{y}) + \boldsymbol{\xi}_2^T \boldsymbol{d}_t^k$. Update the bundle $\mathcal{B}_1^k = \mathcal{B}_1^k \cup \{(\boldsymbol{\xi}_1, \alpha_1)\}$. If $\Delta_2(\boldsymbol{d}_t^k) \leq 0$ then update the bundle $\mathcal{B}_2^k = \mathcal{B}_2^k \cup \{(\boldsymbol{\xi}_2, \alpha_2)\}$ and test if $\|\boldsymbol{\xi}_2\| > \|\boldsymbol{\xi}_{2,\max}\|$ in which case set $\boldsymbol{\xi}_{2,\max} = \boldsymbol{\xi}_2$ and update $t_{\min}$ using (8.33). Go to Step 3.

Step 8. (*Clarke stationarity*) If $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, then Clarke stationarity is achieved: **stop** with $\boldsymbol{x}^* = \boldsymbol{x}_k$ as the final solution.

Step 9. (*Serious step*) Select $\mathcal{B}_1^{k+1} \subseteq \mathcal{B}_1^k$ and $\mathcal{B}_2^{k+1} \subseteq \mathcal{B}_2^k$ and update the linearization errors in the selected sets. Calculate $\boldsymbol{\xi}_1(\boldsymbol{x}_{k+1}) \in \partial f_1(\boldsymbol{x}_{k+1})$ and $\boldsymbol{\xi}_2(\boldsymbol{x}_{k+1}) \in \partial f_2(\boldsymbol{x}_{k+1})$. Set $\mathcal{B}_1^{k+1} = \mathcal{B}_1^{k+1} \cup \{(\boldsymbol{\xi}_1(\boldsymbol{x}_{k+1}), 0)\}$, $\mathcal{B}_2^{k+1} = \mathcal{B}_2^{k+1} \cup \{(\boldsymbol{\xi}_2(\boldsymbol{x}_{k+1}), 0)\}$ and $k = k + 1$. Go to Step 1.

Convergence Analysis Next, we prove that the DBDC method terminates after a finite number of steps at a point satisfying the approximate Clarke stationarity. In order to show this, we need Assumption 8.1 and also the following assumption:

Assumption 8.3 *The subdifferentials $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$ of the DC components f_1 and f_2 are polytopes at any $\boldsymbol{x} \in \mathbb{R}^n$.*

First, we show that the escape procedure in Algorithm 8.2 has a finite convergence. We start with the following useful auxiliary result.

Lemma 8.5 *Let Assumption 8.3 be valid and assume that the level set $\mathcal{F}_{\boldsymbol{x}} = \{\boldsymbol{y} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \leq f(\boldsymbol{x})\}$ is compact for $\boldsymbol{x} \in \mathbb{R}^n$. If at the k-th iteration Algorithm 8.2 does not terminate in Step 5, then*

$$f'(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}; \boldsymbol{d}_{k+1}) > -\bar{m}\|\boldsymbol{u}_k\| \quad \text{for all } \bar{m} > \hat{m}.$$

Proof The compactness of the set $\mathcal{F}_{\boldsymbol{x}}$ guarantees that the formula (8.32) for finding the step-length is well-defined and $\beta^* < \infty$. If Algorithm 8.2 does not terminate in Step 5, then $\beta^* < \varepsilon$ and

$$f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) \leq -\hat{m}\beta^* \|\boldsymbol{u}_k\|. \tag{8.35}$$

In addition, no $\beta > \beta^*$ can satisfy the inequality in (8.32). Assume the contrary, that is

$$f'(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}; \boldsymbol{d}_{k+1}) \leq -\bar{m}\|\boldsymbol{u}_k\| \quad \text{for some } \bar{m} > \hat{m}.$$

This inequality can be rewritten as

$$\lim_{t \downarrow 0} \frac{f(\boldsymbol{x} + (\beta^* + t)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1})}{t} \leq -\bar{m}\|\boldsymbol{u}_k\|.$$

By Definition 1.4 of the directional derivative this means that for any $\rho > 0$ there exits $t^* > 0$ such that

$$\frac{f(\boldsymbol{x} + (\beta^* + t^*)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1})}{t^*} \leq -\bar{m}\|\boldsymbol{u}_k\| + \rho.$$

Since $\bar{m} > \hat{m}$, we can select $\rho = (\bar{m} - \hat{m})\|\boldsymbol{u}_k\| > 0$. Then we obtain

$$f(\boldsymbol{x} + (\beta^* + t^*)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}) \leq -\hat{m}t^*\|\boldsymbol{u}_k\|.$$

This together with (8.35) yields

$$f(\boldsymbol{x} + (\beta^* + t^*)\boldsymbol{d}_{k+1}) - f(\boldsymbol{x}) \leq -\hat{m}(\beta^* + t^*)\|\boldsymbol{u}_k\|$$

showing that the inequality in (8.32) holds for $\beta = \beta^* + t^*$. This is a contradiction since $\beta^* + t^* > \beta^*$. □

Theorem 8.6 *Let Assumption 8.3 be valid and assume that the level set $\mathcal{F}_{\boldsymbol{x}} = \{\boldsymbol{y} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \leq f(\boldsymbol{x})\}$ is compact for $\boldsymbol{x} \in \mathbb{R}^n$. For any $\delta > 0$ and $\varepsilon > 0$, Algorithm 8.2 terminates after at most*

$$N_{max} = \left\lceil \frac{\ln(\delta^2/L^2)}{\ln\left(1 - \frac{(1-\hat{m})^2\delta^2}{8L^2}\right)} \right\rceil + 1$$

iterations, where $\lceil \cdot \rceil$ is a ceiling of a number, $\hat{m} \in (0,1)$ and $L > \delta$ is the Lipschitz constant of f at $\boldsymbol{x} \in \mathbb{R}^n$.

Proof Algorithm 8.2 terminates if a new better iteration point is found in Step 5 or the stopping condition (8.30) is fulfilled. Therefore, to prove the theorem it is sufficient to show that one of these two stopping criteria will be satisfied after a finite number of steps.

Assume that the escape procedure does not terminate at the k-th iteration. This means that we end up calculating a new subgradient $\boldsymbol{\xi}_{k+1}$ in Step 1 at the start of the $(k+1)$-th iteration. Next, we show that this subgradient does not belong to $U_k \subset \partial_\varepsilon^G f(\boldsymbol{x})$ and, therefore, in this case the approximation of the Goldstein ε-subdifferential is improved. The necessary and sufficient condition for the quadratic problem (8.29) implies that $\boldsymbol{u}_k^T \boldsymbol{u} \geq \|\boldsymbol{u}_k\|^2$ for all $\boldsymbol{u} \in U_k$ which yields that

$$\boldsymbol{u}^T \boldsymbol{d}_{k+1} \leq -\|\boldsymbol{u}_k\| \quad \text{for all } \boldsymbol{u} \in U_k. \tag{8.36}$$

Now the new point $\tilde{\boldsymbol{x}}$, where the subgradient $\boldsymbol{\xi}_{k+1}$ is calculated, is either $\boldsymbol{x}$ or $\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$. If the first case occurs, then $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ and the condition (8.31) is satisfied providing that

$$f'(\tilde{\boldsymbol{x}}; \boldsymbol{d}_{k+1}) = f'(\boldsymbol{x}; \boldsymbol{d}_{k+1}) > -\hat{m}\|\boldsymbol{u}_k\| > -\bar{m}\|\bar{\boldsymbol{u}}_k\| \tag{8.37}$$

for $\bar{m} \in (\hat{m}, 1)$. In the latter case, $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}$ for $\beta^* < \varepsilon$ and, thus this point can also be used to calculate a subgradient from the set $\partial_\varepsilon^G f(\boldsymbol{x})$. Furthermore, Lemma 8.5 guarantees that

$$f'(\tilde{\boldsymbol{x}}; \boldsymbol{d}_{k+1}) = f'(\boldsymbol{x} + \beta^* \boldsymbol{d}_{k+1}; \boldsymbol{d}_{k+1}) > -\bar{m}\|\boldsymbol{u}_k\| \tag{8.38}$$

for $\bar{m} \in (\hat{m}, 1)$. The inequalities (8.37), (8.38) and Theorem 8.5(iii), (iv) imply that

$$\boldsymbol{\xi}_{k+1}^T \boldsymbol{d}_{k+1} > -\bar{m}\|\boldsymbol{u}_k\| \quad \text{for all } \bar{m} \in (\hat{m}, 1). \tag{8.39}$$

This means that (8.36) cannot hold for $\boldsymbol{\xi}_{k+1}$. Thus, the approximation of $\partial_\varepsilon^G f(\boldsymbol{x})$ is significantly improved, since $\boldsymbol{\xi}_{k+1} \notin U_k$.

Next, we show that the stopping condition (8.30) is always fulfilled after a finite number of steps if a new better iteration point is never found. This proves the finite termination of Algorithm 8.2. First, we notice that $t\boldsymbol{\xi}_{k+1} + (1-t)\boldsymbol{u}_k \in U_{k+1}$ for any $t \in (0,1)$ meaning that

$$\begin{aligned}\|\boldsymbol{u}_{k+1}\|^2 &\leq \|t\boldsymbol{\xi}_{k+1} + (1-t)\boldsymbol{u}_k\|^2 = \|\boldsymbol{u}_k + t(\boldsymbol{\xi}_{k+1} - \boldsymbol{u}_k)\|^2 \\ &= \|\boldsymbol{u}_k\|^2 + 2t\boldsymbol{u}_k^T(\boldsymbol{\xi}_{k+1} - \boldsymbol{u}_k) + t^2\|\boldsymbol{\xi}_{k+1} - \bar{\boldsymbol{u}}_k\|^2.\end{aligned}$$

In addition, local Lipschitz continuity of a DC function guarantees that the Goldstein ε-subdifferential $\partial_\varepsilon^G f(\boldsymbol{x})$ is bounded at $\boldsymbol{x}$ with a Lipschitz constant $L > \delta$ determined at that point and thus $\|\boldsymbol{\xi}\| \leq L$ for all $\boldsymbol{\xi} \in \partial_\varepsilon^G f(\boldsymbol{x})$. On the other hand, the inequality (8.39) yields that $\boldsymbol{\xi}_{k+1}^T \boldsymbol{u}_k < \bar{m}\|\bar{\boldsymbol{u}}_k\|^2$ and we obtain

$$\begin{aligned}\|\boldsymbol{u}_{k+1}\|^2 &\leq \|\boldsymbol{u}_k\|^2 + 2t\boldsymbol{u}_k^T(\boldsymbol{\xi}_{k+1} - \boldsymbol{u}_k) + 4t^2L^2 \\ &< \|\bar{\boldsymbol{u}}_k\|^2 - 2t(1-\bar{m})\|\boldsymbol{u}_k\|^2 + 4t^2L^2\end{aligned}$$

for each $\bar{m} \in (\hat{m}, 1)$. Since $t \in (0,1)$, we can select $t = \frac{(1-\bar{m})\|\boldsymbol{u}_k\|^2}{4L^2}$ and this gives the approximation

$$\|\boldsymbol{u}_{k+1}\|^2 < \|\boldsymbol{u}_k\|^2\left(1 - \frac{(1-\bar{m})^2\|\boldsymbol{u}_k\|^2}{4L^2}\right) \quad \text{for all } \bar{m} \in (\hat{m}, 1).$$

In addition, there exists $\bar{m} \in (\hat{m}, 1)$ such that $2(1-\bar{m})^2 > (1-\hat{m})^2$. Therefore, by selecting such $\bar{m}$ and taking into account that $\|\boldsymbol{u}_k\| > \delta$ for each $k > 0$ we get

$$\|\boldsymbol{u}_{k+1}\|^2 < \|\boldsymbol{u}_k\|^2\left(1 - \frac{(1-\hat{m})^2\delta^2}{8L^2}\right).$$

From this and $\|\boldsymbol{u}_1\| \leq L$ we can conclude that

$$\|\boldsymbol{u}_k\|^2 < \|\boldsymbol{u}_1\|^2\left(1 - \frac{(1-\hat{m})^2\delta^2}{8L^2}\right)^{k-1} \leq L^2\left(1 - \frac{(1-\hat{m})^2\delta^2}{8L^2}\right)^{k-1}$$

showing that when

$$k \geq \left\lceil \frac{\ln(\delta^2/L^2)}{\ln\left(1 - \frac{(1-\hat{m})^2\delta^2}{8L^2}\right)} \right\rceil + 1$$

the stopping criterion $\|\boldsymbol{u}_k\| \leq \delta$ is satisfied. □

Next, similar to the PBDC method, we show that during one iteration of the DBDC method the number of null steps is finite.

Proposition 8.4 *Let Assumption 8.1 be valid. At any iteration k, for any $\delta > 0$, Algorithm 8.3 can pass through Steps 3–7 only finitely many times before entering Step 8 or the escape procedure in Step 4.*

Proof The proof is similar to that of Proposition 8.3. Nevertheless, we do not need to consider *Case 1*, since the execution of Step 4 terminates the iteration k. Thus, we consider only *Cases 2* and *3*. Furthermore, in *Case 3* we do not have the sequence $\{\eta_i\}$, since η_i is replaced everywhere with the parameter $\delta > 0$. This means that $\hat{\eta}$ is also replaced with this parameter. □

The final result establishes the finite convergence of the DBDC method.

Theorem 8.7 *Let Assumptions 8.1 and 8.3 be valid. For any $\delta > 0$ and $\varepsilon > 0$, Algorithm 8.3 terminates after a finite number of iterations at a point $\boldsymbol{x}^*$ satisfying the approximate Clarke stationarity condition*

$$\|\boldsymbol{\xi}^*\| \leq \delta \quad \textit{with} \quad \boldsymbol{\xi}^* \in \partial_{\varepsilon}^{G} f(\boldsymbol{x}^*).$$

Proof Algorithm 8.3 terminates when the stopping condition in Step 8 is satisfied and this condition guarantees approximate Clarke stationarity. Similarly to the proof of Theorem 8.3 we next show that if the DBDC method never terminates then the objective function f is not bounded from below. This is a contradiction, since the boundedness of f follows from LLC and Assumption 8.1.

First, notice that if the stopping condition is never fulfilled then Algorithm 8.3 generates an infinite sequence $\{\boldsymbol{x}_k\}$. Proposition 8.4 and Theorem 8.6 guarantee that a new iteration point $\boldsymbol{x}_{k+1}$ is always found after a finite number of null steps. Moreover, a new point is obtained either from Step 4 or Step 5. In the first case, Algorithm 8.2 produces this point and therefore the inequality

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\hat{m}\varepsilon\delta < 0$$

is always satisfied. In the latter case, the descent condition (8.34) holds and similarly to the proof of Theorem 8.3 we can show that

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\frac{m\delta^2}{2\bar{t}_{\max}},$$

by taking into account (8.22) and the fact that the parameter η_i is replaced everywhere with $\delta > 0$. Thus, after each iteration of the DBDC method $f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\sigma < 0$, where $\sigma = \min\{\hat{m}\varepsilon\delta, m\delta^2/(2\bar{t}_{\max})\}$. This enables us to write

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_0) \leq -k\sigma$$

and a contradiction follows by passing to the limit. □

8.6 Piecewise Concave Bundle Method DCPCA

In this section, we briefly describe the main idea of the piecewise concave bundle method for unconstrained DC optimization (DCPCA), which terminates at points satisfying approximate ε-criticality. We also compare the model of the DCPCA with the one used in the PBDC and DBDC methods. For details of the DCPCA, we refer to [16].

The idea in the model construction is to substitute the DC components with their convex cutting plane models (8.7). Thus, the *nonconvex DC cutting plane model* of a DC function f is given by

$$\tilde{f}^k(\boldsymbol{x}) = \hat{f}_1^k(\boldsymbol{x}) - \hat{f}_2^k(\boldsymbol{x})$$

and in this general form the model is the same as the one used in the PBDC and DBDC methods. However, the bundles of the DC components used in the DCPCA differ from the ones used in the previous methods. In the PBDC and DBDC methods, both bundles of the DC components refer to a global model, since the points included into the bundles are collected from the current and past iterations. Nevertheless, in the DCPCA the bundle of the first DC component f_1 only contains information from points which are close to the current iteration point $\boldsymbol{x}_k$, while the bundle of f_2 refers to the global model. Thus, during each iteration we maintain a local approximation for the DC component f_1.

In order to obtain a search direction in the DCPCA method, the nonconvex cutting plane model is rewritten as

$$\tilde{f}^k(\boldsymbol{x}) = \max_{j \in J_1^k} \left\{ f(\boldsymbol{x}_k) + \boldsymbol{\xi}_{1,j}^T \boldsymbol{d} - \alpha_{1,j}^k - \Delta_2^k(\boldsymbol{d}) \right\}. \tag{8.40}$$

Therefore, the model can be seen as a pointwise maximum of concave piecewise affine functions. Different from the previous bundle methods, the search direction problem constructed from the nonconvex model (8.40) is not solved exactly, but instead it is locally approximated with an auxiliary convex quadratic problem. In addition, DCPCA utilizes a supplementary convex quadratic problem to improve the search direction at points which are far away from the current iteration point $\boldsymbol{x}_k$.

After the search direction is obtained, a line search procedure is executed in the DCPCA method to decide whether a serious step or a null step is performed. If a null step occurs the current iteration point does not change and only the bundle $\mathcal{B}_1^k$ of the DC component f_1 is updated with a new bundle element improving the model. However, if a serious step occurs a new better iteration point $\boldsymbol{x}_{k+1}$ is obtained and both bundles are updated. Therefore, the bundle $\mathcal{B}_2^k$ needs to be updated only whenever a serious step is done. On the other hand, the bundle $\mathcal{B}_1^k$ is updated in every step. However, in every serious step we are able to reduce the bundle $\mathcal{B}_1^k$, since for the DC component f_1 we maintain a local approximation, and thus each bundle element of $\mathcal{B}_1^k$ far away from $\boldsymbol{x}_{k+1}$ can be removed at a serious step.

8.7 Numerical Results

In this section, we illustrate the performance of the PBDC and DBDC methods with nonsmooth DC test problems. We start with demonstration of the main difference between these two methods using an illustrative example. Then, we compare the performance of the PBDC and DBDC methods with that of the proximal bundle method MPBNGC (see Chap. 13) which is designed for a general nonsmooth objective function. With this comparison the goal is to justify the usage of the DC bundle methods when the DC structure of the problem is available.

Illustrative Example The PBDC method converges to a critical point and due to this the obtained solution may be located in an unfavourable place being unable to describe an interesting feature for the objective. However, this disadvantage is overcome in the DBDC method with the escape procedure which is illustrated in the following example.

Example 8.6 Consider the functions f and their DC components f_1 and f_2 from Examples 8.1–8.3. Note that in each of these examples the point $x^* = 0$ is a critical point but not Clarke stationary. Due to this, the PBDC method can stop at this critical point. On the other hand, in the DBDC method the fulfilment of the criticality condition at x^* leads to the escape procedure presented in Algorithm 8.2.

The escape procedure starts by calculating a subgardient ξ of f at $x^* = 0$ by using either a direction $d_1 = 1$ or $d_1 = -1$. In Examples 8.1 and 8.3, we always obtain $\xi = 1$ regardless of the selection of d_1 whereas in Example 8.2 the result is either $\xi = 1$ or $\xi = 2$ depending on whether we use $d_1 = 1$ or $d_1 = -1$, respectively. However, both $\xi = 1$ and $\xi = 2$ yield the search direction $d_2 = -1$. Furthermore, in each example we obtain the same value of the directional derivative of f, that is, $f'(x^*; d_2) = f(0; -1) = -1$. Thus, with the selection of $\hat{m} \in (0, 1/2)$ the condition (8.31) is not satisfied, since for both $\xi = 1$ and $\xi = 2$ we deduce that

$$f'(x^*; d_2) = -1 \leq -\hat{m}\|\xi\|.$$

This shows that in all three examples d_2 is the descent direction. Thus, the DBDC is able to generate a better iteration point and bypass the problematic critical point x^*.

Numerical Experiments The numerical experiments were carried out using 53 instances of 16 academic DC test problems from [23]. More specifically, Problems 1–10 are introduced in [21] whereas Problems 11–16 can be found from [22]. The selection of the input parameters of the PBDC and DBDC is shown in Table 8.1. In addition, in both DC bundle methods the size of $\mathcal{B}_1$ is set to $\min\{n + 5, 1000\}$ and the size of $\mathcal{B}_2$ is *three*. In Algorithm 8.2, the size of the set U_k is restricted to $2n$.

Table 8.1 The input parameters of the PBDC and DBDC methods

PBDC	DBDC
$\delta = \begin{cases} 0.005n, & \text{if } n < 150, \\ 0.015n, & \text{if } 150 \le n \le 200, \\ 0.05n, & \text{if } n > 200, \end{cases}$	$\delta = \begin{cases} 10^{-5}, & \text{if } n \le 200, \\ 10^{-4}, & \text{if } n > 200, \end{cases}$
$\varepsilon = 0.1,$	$\varepsilon = \begin{cases} 10^{-6}, & \text{if } n \le 50, \\ 10^{-5}, & \text{if } n > 50, \end{cases}$
$L_1 = 1000,$ $L_2 = 1000,$	$\hat{m} = 0.01,$ $\theta = 5 \cdot 10^{-5},$
The same input parameters in both methods	
$r = \begin{cases} 0.75, & \text{if } n < 10, \\ \text{the first two decimals of } n/(n+5), & \text{if } 10 \le n < 300 \\ 0.99, & \text{if } n \ge 300, \end{cases}$	
$R = 10^7,$ $c = 0.1,$ $m = 0.2.$	

Furthermore, in MPBNGC the bundle size is fixed to $\min\{n+3, 1000\}$ and the final accuracy is set to 10^{-10}. Other input parameters of the MPBNGC are selected as default values [29].

Both PBDC and DBDC methdos are implemented using double precision Fortran 95 whereas the MPBNGC is implemented in double precision Fortran 77. The numerical tests are performed on an Intel® Core™ i5-2400 CPU (3.10 GHz, 3.10 GHz) running under Windows 7 and `gfortran` is used as a compiler.

To illustrate the numerical results we use performance profiles [10] which are presented in Figs. 8.4, 8.5, 8.6, and 8.7. Figure 8.4 contains the performance profiles with all three solvers and Figs. 8.5, 8.6, and 8.7 present pairwise comparisons. In all

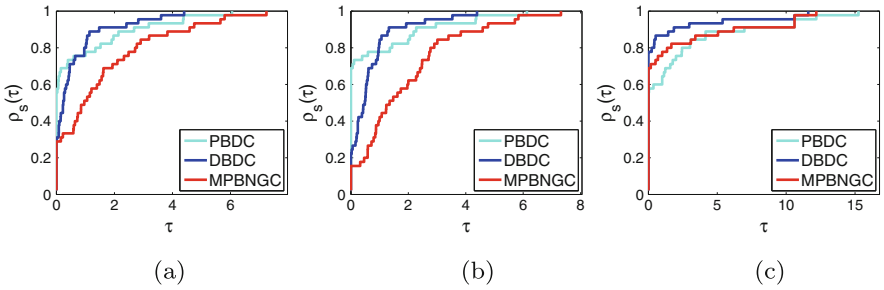

Fig. 8.4 The performance profiles for the PBDC, DBDC and MPBNGC with 45 instances. (**a**) function eval. (**b**) subgrad. eval. (**c**) CPU time

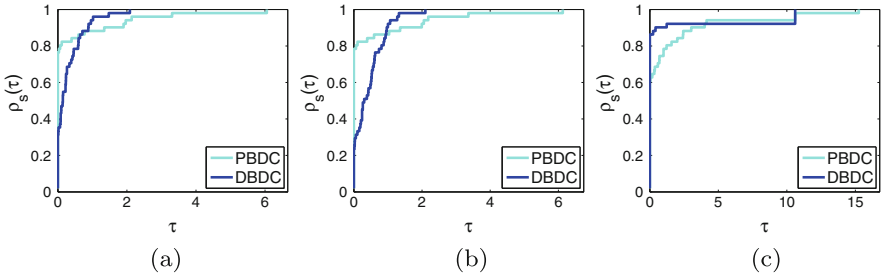

Fig. 8.5 The performance profiles for the PBDC and DBDC with 51 instances. (**a**) function eval. (**b**) subgrad. eval. (**c**) CPU time

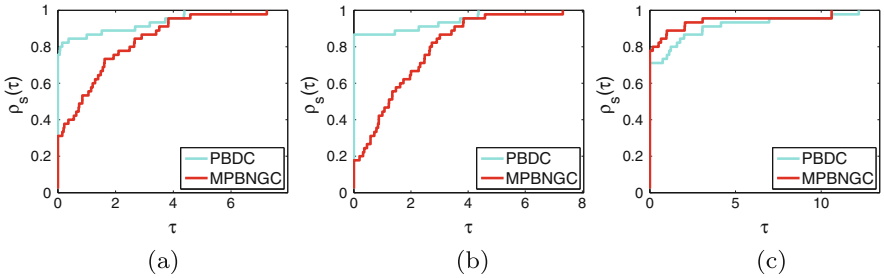

Fig. 8.6 The performance profiles for the PBDC and MPBNGC with 45 instances. (**a**) function eval. (**b**) subgrad. eval. (**c**) CPU time

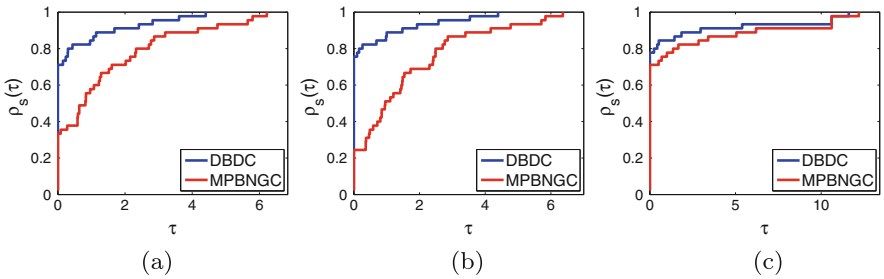

Fig. 8.7 The performance profiles for the DBDC and MPBNGC with 45 instances. (**a**) function eval. (**b**) subgrad. eval. (**c**) CPU time

cases the number of function evaluations, the number of subgradient evaluations and CPU time are used to draw performance profiles. Furthermore, in each performance profile we include from 53 instances of the test problems only those where the compared methods yield the same (local) solution. That is, 45 instances of the test problems were used for the performance profiles in Figs. 8.4, 8.6, and 8.7 whereas 51 instances were used in Fig. 8.5.

Results show that both the DC bundle methods PBDC and DBDC outperform the MPBNGC method in the sense of the number of function and subgradient evaluations. However, the pairwise comparison of the PBDC and DBDC methods,

presented in Fig. 8.5a, b, does not clearly show superiority of either of them. Figure 8.4c shows that the DBDC method performs slightly better than the other two methods in the sense of required CPU time. If we restrict our consideration to the pairwise comparison of the PBDC and DBDC methods (see Fig. 8.5c) then the differences in CPU time are not so significant. Furthermore, Fig. 8.6c demonstrates that the MPBNGC method is slightly better than the PBDC method in terms of CPU time.

One interesting feature of the DC bundle methods PBDC and DBDC is that they often find the global or best known solution for test problems, even though these methods are only local search methods. For example, in 48 out of 53 instances of the test problems both DC bundle methods find the best known solutions whereas the bundle method MPBNGC obtains the best known solutions only in 44 instances. This indicates that the DC cutting plane model of the DC objective function is able to capture some relevant information about the objective in order to avoid local optimizers.

8.8 Conclusions

In this chapter, we concentrate on the problem of unconstrained minimization of nonsmooth functions represented as the difference of two convex functions. Necessary and sufficient optimality conditions for such problems are presented and the relationship between sets of different stationary points are discussed in detail. Two different bundle methods are presented to solve DC problems together with their convergence analysis. In addition, the cutting plane model used in these methods is compared to a different DC model. The performance of the methods are demonstrated using nonsmooth test problems with DC objective functions. The results clearly show that the use of DC structure in NSO leads to the design of efficient and accurate methods in comparison with general purposed NSO methods. Furthermore, DC optimization methods based on the extension of the bundle methods for convex optimization are highly successful in finding global solutions to DC optimization problems.

Acknowledgements This work was financially supported by the University of Turku, the Academy of Finland (Projects 294002 and 319274) and the Australian Government through the Australian Research Council's Discovery funding scheme (Project No. DP190100580).

References

1. Ahmadi, A.A., Hall, G.: DC decomposition of nonconvex polynomials with algebraic techniques. Math. Program. **169**(1), 69–94 (2018)
2. Bagirov, A.M.: A method for minimization of quasidifferentiable functions. Optim. Methods Softw. **17**(1), 31–60 (2002)

3. Bagirov, A.M., Ugon, J.: Codifferential method for minimizing nonsmooth DC functions. J. Global Optim. **50**(1), 3–22 (2011)
4. Bagirov A.M., Ugon, J.: Nonsmooth DC programming approach to clusterwise linear regression: optimality conditions and algorithms. Optim. Methods Softw. **33**(1), 194–219 (2018)
5. Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, Cham, Heidelberg (2014)
6. Bagirov, A.M., Taheri, S., Ugon, J.: Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. Pattern Recognit. **53**(1), 12–24 (2016)
7. Bagirov, A.M., Taheri, S., Asadi, S.: A difference of convex optimization algorithm for piecewise linear regression. J. Ind. Manag. Optim. **15**(2), 909–932 (2019)
8. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)
9. Cui, Y., Pang, J.-S., Sen, B.: Composite difference-max programs for modern statistical estimation problems. SIAM J. Optim. **28**(4), 3344–3374 (2018)
10. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
11. Ferrer, A.: Representation of a polynomial function as a difference of convex polynomials, with an application. In: Hadjisavvas, N., Martínez-Legaz, J.E., Penot, J.P. (eds.) Generalized Convexity and Generalized Monotonicity, vol. 502, pp. 189–207. Springer, Berlin (2001)
12. Ferrer, A., Martínez-Legaz, J.E.: Improving the efficiency of DC global optimization methods by improving the DC representation of the objective function. J. Global Optim. **43**(4), 513–531 (2009)
13. Fuduli, A., Gaudioso, M., Giallombardo, G.: A DC piecewise affine model and a bundling technique in nonconvex nonsmooth minimization. Optim. Methods Softw. **19**(1), 89–102 (2004)
14. Fuduli, A., Gaudioso, M., Giallombardo, G.: Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. SIAM J. Optim. **14**(3), 743–756 (2004)
15. Fuduli, A., Gaudioso, M., Nurminski, E.A.: A splitting bundle approach for non-smooth nonconvex minimization. Optimization **64**(5), 1131–1151 (2015)
16. Gaudioso, M., Giallombardo, G., Miglionico, G., Bagirov, A.M.: Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. J. Global Optim. **71**(1), 37–55 (2018)
17. Hiriart-Urruty, J-.B.: Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. In: Ponstein, J. (ed.) Convexity and Duality in Optimization, vol. 256, pp. 37–70. Springer, Berlin (1985)
18. Hiriart-Urruty, J.-B.: From convex optimization to nonconvex optimization. Necessary and sufficient conditions for global optimality. In: Clarke, F.H., Dem'yanov, V.F., Giannessi, F. (eds.) Nonsmooth Optimization and Related Topics, Ettore Majorana International Sciences Series 43, pp. 219–239. Springer, Boston (1989)
19. Hou, L., Sun, W.: On the global convergence of a nonmonotone proximal bundle method for convex nonsmooth minimization. Optim. Methods Softw. **23**(2), 227–235 (2008)
20. Joki, K.: Bundle methods in nonsmooth DC optimization. PhD thesis, University of Turku (2018)
21. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. J. Global Optim. **68**(3), 501–535 (2017)
22. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S.: Double bundle method for nonsmooth DC optimization. TUCS Technical Report No. 1173, Turku Centre for Computer Science, Turku (2017)
23. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S.: Double bundle method for finding Clarke stationary points in nonsmooth DC programming. SIAM J. Optim. **28**(2), 1892–1919 (2018)
24. Kiwiel, K.C.: Proximity control in bundle methods for convex nondifferentiable minimization. Math. Program. **46**(1–3), 105–122 (1990)

25. Le Thi, H.A., Pham Dinh, T.: Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. J. Global Optim. **11**(3), 253–285 (1997)
26. Le Thi, H.A., Pham Dinh, T.: The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann. Oper. Res. **133**(1–4), 23–46 (2005)
27. Le Thi, H.A., Pham Dinh, T.: Difference of convex functions algorithms (DCA) for image restoration via a Markov random field model. Optim. Eng. **18**(4), 873–906 (2017)
28. Mäkelä, M.M.: Survey of bundle methods for nonsmooth optimization. Optim. Methods Softw. **17**(1), 1–29 (2002)
29. Mäkelä, M.M.: Multiobjective proximal bundle method for nonconvex nonsmooth optimization: Fortran subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing B 13/2003, University of Jyväskylä, Jyväskylä (2003)
30. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific, Singapore (1992)
31. Oliveira, W.: Proximal bundle methods for nonsmooth DC programming. J. Global Optim. (2019). https://doi.org/10.1007/s10898-019-00755-4
32. Pang, J.-S., Tao, M.: Decomposition methods for computing directional stationary solutions of a class of nonsmooth nonconvex optimization problems. SIAM J. Optim. **28**(2), 1640–1669 (2018)
33. Pham Dinh, T., Le Thi, H.A.: Convex analysis approach to D.C. programming: theory, algorithms and applications. Acta Math. Vietnam. **22**(1), 289–355 (1997)
34. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (1992)
35. Souza, J.C.O., Oliveira, P.R., Soubeyran, A.: Global convergence of a proximal linearized algorithm for difference of convex functions. Optim. Lett. **10**(7), 1529–1539 (2016)
36. Toland, J.F.: On subdifferential calculus and duality in nonconvex optimization. Memoires de la Société Mathématique de France **60**, 177–183 (1979)
37. Tuy, H.: Convex Analysis and Global Optimization, 1st edn. Kluwer, Dordrecht (1998)

Chapter 9
Beyond First Order: $\mathcal{VU}$-Decomposition Methods

Shuai Liu and Claudia Sagastizábal

Abstract When minimizing a nonsmooth convex function bundle methods are well known by their robustness and reliability. While such features are related to global convergence properties of the algorithm, speed of convergence is a different matter, as fast rates cannot be expected if the objective function lacks smoothness.

In the typical bundle dichotomy that splits iterates between null and serious steps, the latter subsequence converges with R-linear speed. Moving from the first-order method realm to the world of superlinear speed is possible when realizing that nonsmoothness often appears in a structured manner. This is the basis of the $\mathcal{VU}$-decomposition approach presented in this chapter. Thanks to this decomposition, it is possible to make a Newton-like move in certain $\mathcal{U}$-subspace, where all the function "smoothness" concentrates at least locally. On its orthogonal complement, the "sharp" $\mathcal{V}$-subspace, an intermediate iterate is defined such that the overall convergence is driven by the $\mathcal{U}$-step. As a result, the serious-step subsequence converges with superlinear speed.

By focusing on the proximal variants of bundle methods, this chapter introduces gradually the $\mathcal{VU}$-theory and the ingredients needed to build superlinearly convergent algorithms in nonsmooth optimization. For functions whose proximal points are easy to compute, as in machine learning, an implementable quasi-Newton $\mathcal{VU}$-algorithm is given. The bundle machinery is incorporated for functions whose proximal points are not readily available. In this case, superlinear speed can be achieved by solving two quadratic programming problems per iteration.

9.1 Introduction

This book deals with methods for solving optimization problems of the form (1.1). In this chapter we focus on *fast* variants for unconstrained convex problems, that

S. Liu · C. Sagastizábal (✉)
IMECC-UNICAMP, Campinas, SP, Brazil
e-mail: shuailiu@ime.unicamp.br; sagastiz@unicamp.br

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_9

is, when in (1.1) the objective function is convex and the feasible set is the whole space, that is, $G = \mathbb{R}^n$.

Classical forms of bundle algorithms, presented in Chap. 3, belong to the family of *first-order methods*. These are convergent algorithms defining a subsequence of iterates (the so-called serious steps) that eventually lead to a minimizer. Making an analogy with nonlinear programming, that deals with smooth functions, the serious step subsequence in bundle methods corresponds to iterates in gradient or steepest descent schemes. It is only natural that their speed of convergence is linear at best; we recall this result, due to [25], in Sect. 9.2.

Since the advent of quasi-Newton methods for nonlinear programming in the 1980s the nonsmooth optimization (NSO) community struggled to replicate those techniques and obtain convergence rates that are faster than linear. The quest was not an easy one, considering that passing from the concept of a gradient (a singleton) to a subgradient (a set) is already hard. When it comes to second-order objects the situation worsens, as one needs to deal with a set of matrices, instead of just one Hessian.

As explained in [21] (see also [27]), increasing convergence speed of NSO methods was a recurrent concern. For more than 20 years, the superlinearly convergent approach [9], a one-dimensional forerunner of the $\mathcal{VU}$-decomposition, enjoyed the rare status of unique (and perfect) specimen. For higher dimensions, that is for $n > 1$, a breakthrough was the variable metric proximal bundle method [2, 10], which exploited the variable metric of the Moreau-Yosida regularization of f and interpreted the subsequence of serious iterates as if generated by a preconditioned gradient scheme. The works [3, 16, 22, 24] explore further the equivalence between minimizing a function or its regularization (keeping in mind that computing the later at one point can prove as difficult a task as minimizing the former; see Remark 9.1).

The appeal of resorting to the Moreau-Yosida detour is that, having a Lipschitzian gradient, the regularization is prone to having some kind of Hessian that could be approximated by a quasi-Newton scheme and yield rapid convergence. However, in [12] it was shown that for the regularization to have a Hessian everywhere, the original function must be of class C^2. From the NSO perspective, the situation looked not too promising; fortunately [12] also points out the existence of a special second-order object, provided the nonsmooth function f is *restricted* to certain subspace. The bivariate function

$$f(\boldsymbol{x}) = f^1(x_1) + f^2(x_2) = |x_1| + \frac{a}{2}x_2^2,$$

depending on a positive parameter a and considered in Example 9.1, illustrates well the situation. Even though f fails to be differentiable when $x_1 = 0$, when restricted to the manifold $\mathcal{M} := \{\boldsymbol{x} = (0, x_2) : x_2 \in \mathbb{R}\}$ the function coincides with $\frac{a}{2}x_2^2$, looks smooth, and has a (one-dimensional) Hessian equal to a. As a result, performing a Newton move on $f|_{\mathcal{M}}$ drives the variable x_2 to zero. In the $\mathcal{VU}$-jargon, this corresponds to the $\mathcal{U}$-step. The $\mathcal{V}$-step acts on the first component, projecting

the iterate onto the subspace of smoothness. For this simple function, the minimizer is found by taking only one $\mathcal{U}$-step followed by one $\mathcal{V}$-step.

For a general convex function the situation is not as straightforward, because several approximations need to be suitably done. In particular, identifying the $\mathcal{V}$-subspace, that is, gathering at a given point all the nonsmoothness of f, proves to be a difficult task. Thanks to [18, Theorem 5.1] this can be done by computing the proximal point operator. This interpretation is a key to approximate the $\mathcal{V}$-step, as a bundle mechanism ending successive null steps with a serious step gives an implementable algorithm to approximate proximal points, [1, 4, 6]. The same mechanism provides a good approximation for the $\mathcal{U}$-Newton direction that drives the convergence speed of the serious subsequence in a superlinear manner.

This chapter is organized to introduce step by step the various layers of approximations needed to put in place an implementable $\mathcal{VU}$-bundle method that is superlinearly convergent. Throughout the presentation the emphasis is put on transmitting the main ideas and concepts, illustrating with simple examples all the presented material. Section 9.2 revises proximal bundle methods, seen from the perspective [4] related to the proximal point operator. A result by [25] stating R-linear convergence for the method is given in this section. The main elements of the $\mathcal{VU}$-theory are recalled and illustrated with examples in Sect. 9.3. This section revises the important notion of fast track, over which a partial second-order expansion for the function exists when parameterized in the $\mathcal{U}$-subspace. The superlinearly convergent, yet conceptual, $\mathcal{VU}$-algorithm [13] is the topic of Sect. 9.4. This scheme is made implementable gradually, in the subsequent sections. Section 9.5 links the $\mathcal{V}$-step with the proximal operator and explains how to make a dynamic approximation of the desired $\mathcal{V}$ and $\mathcal{U}$-subspaces. The new objects, depending on the $\mathcal{VU}$-decomposition around the current iterate should asymptotically make a good guess of the $\mathcal{VU}$-objects at the considered minimizer. An implementable quasi-Newton $\mathcal{VU}$ algorithm with exact prox-calculations is given in Sect. 9.6. The method is superlinearly convergent and fully implementable, for functions whose proximal points can be computed exactly without much effort as in many applications in machine learning. For functions whose proximal points are difficult to compute, the bundling machinery is incorporated in Sect. 9.7. In this case, the price to pay for the gain in accuracy provided by the superlinear speed is in the need of solving two quadratic programming problems per iteration. The chapter ends with final remarks given in Sect. 9.8.

9.2 Linear Convergence of Proximal Bundle Methods

The proximal point mapping of a function f is defined by

$$\boldsymbol{p}_\mu f(\cdot) := \operatorname*{argmin}_{\boldsymbol{y}\in\mathbb{R}^n} f(\boldsymbol{y}) + \frac{\mu}{2}\|\boldsymbol{y}-\cdot\|^2,$$

where μ is a positive parameter. Generally, computing the proximal points of the objective function is very difficult. Proximal bundle methods compute the proximal points of some simpler model of the objective function which makes the computations much easier. For example, for piecewise linear models the proximal bundle method needs to solve only a quadratic programming problem. After checking if the function value at the proximal point is sufficiently smaller than the current best functional value, these methods update the model through the manner of bundle methods, that is, by adding and/or deleting some affine planes from the bundle of information to improve the approximation of the model. Typical proximal bundle methods follow the structure in Algorithm 9.1 (cf. [4, Algorithm 4.1]).

Algorithm 9.1: Proximal bundle method

Data: starting point $\boldsymbol{x}^0$, a parameter $m \in (0, 1)$, a positive sequence μ^k.

1 Set $k \leftarrow 0$, iteration index for serious steps.
2 **for** $\ell = 0, 1, \ldots$ **do**
3 Choose a convex model function $\varphi^\ell : \mathbb{R}^n \to \mathbb{R}$.
4 Compute $\boldsymbol{p}^\ell := \boldsymbol{p}_{\mu^k}\varphi^\ell\left(\boldsymbol{x}^k\right)$ and $\boldsymbol{g}^\ell := \mu^k\left(\boldsymbol{x}^k - \boldsymbol{p}^\ell\right)$.
5 **if** $f\left(\boldsymbol{x}^k\right) - f\left(\boldsymbol{p}^\ell\right) \geq m\left[f\left(\boldsymbol{x}^k\right) - \varphi^\ell\left(\boldsymbol{p}^\ell\right)\right]$, **then**
6 set $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{p}^\ell$ and $k \leftarrow k + 1$.

In Algorithm 9.1, the inequality in line 5 is the descent test. It checks if the difference of the function values is at least a portion of that of the model values. Here, note that the model φ^ℓ is a lower approximation of f (thus $f\left(\boldsymbol{x}^k\right) \geq \varphi^\ell\left(\boldsymbol{x}^k\right)$) and $\varphi^\ell\left(\boldsymbol{x}^k\right) \geq \varphi^\ell\left(\boldsymbol{p}^\ell\right)$ by definition of proximal point. If line 6 is executed then the iteration is called a *serious step* or *descent step*. Otherwise, it is called a *null step*. Originally the convex function φ^ℓ was chosen as the cutting-plane model of f. The authors of [4] show that the convergence of this algorithm can be obtained for any sequence of models $\left\{\varphi^\ell\right\}$ satisfying the following conditions:

$$\varphi^\ell \leq f \text{ for } \ell = 0, 1, \ldots, \tag{9.1}$$

$$\left.\begin{aligned} \varphi^\ell\left(\boldsymbol{p}^\ell\right) + \left\langle \boldsymbol{g}^\ell, \cdot - \boldsymbol{p}^\ell \right\rangle &\leq \varphi^{\ell+1} \\ f\left(\boldsymbol{p}^\ell\right) + \left\langle \boldsymbol{s}^\ell, \cdot - \boldsymbol{p}^\ell \right\rangle &\leq \varphi^{\ell+1} \end{aligned}\right\} \text{ if the } \ell\text{-th iteration is a null step,}$$

where $\boldsymbol{s}^\ell \in \partial f\left(\boldsymbol{p}^\ell\right)$.

Example 9.1 Consider the simple function in Fig. 9.1, defined by

$$f(\boldsymbol{x}) = f^1(x_1) + f^2(x_2) = |x_1| + \frac{a}{2}x_2^2.$$

Set $\boldsymbol{x}^0 = (-1, 1)$ and compute a cutting plane at $\boldsymbol{x}^0$: $f\left(\boldsymbol{x}^0\right) + \left\langle \boldsymbol{s}^0, \boldsymbol{x} - \boldsymbol{x}^0 \right\rangle = -x_1 + ax_2 - \frac{a}{2}$, where $\boldsymbol{s}^0 \in \partial f\left(\boldsymbol{x}^0\right) = \{(-1, a)\}$. This cutting plane will be the first model φ^0. Simple calculations yield that

$$\boldsymbol{p}_\mu \varphi^0(\boldsymbol{x}) = \boldsymbol{x} - \frac{1}{\mu}(-1, a) = \left(x_1 + \frac{1}{\mu}, x_2 - \frac{a}{\mu}\right).$$

It will be interesting to know how close the proximal mapping of the model approximates that of the objective function. To give a basic idea, in this example we calculate the proximal mapping of f:

$$\boldsymbol{p}_\mu f(\boldsymbol{x}) = \left(p_\mu f^1(x_1), p_\mu f^2(x_2)\right)$$

$$\text{with } p_\mu f^1(x_1) = x_1 - P_T(x_1) \text{ and } p_\mu f^2(x_2) = \frac{\mu}{a+\mu}x_2,$$

where $P_T(x_1)$ is the projection of x_1 onto the interval $T = \left[-\frac{1}{\mu}, \frac{1}{\mu}\right]$. Due to the separability of $\boldsymbol{p}_\mu \varphi^0$ and $\boldsymbol{p}_\mu f$ we can compare their graphs. From Fig. 9.2 we can see that in the first component, the graphs of $\boldsymbol{p}_\mu \varphi^0$ and $\boldsymbol{p}_\mu f$ coincide locally around the point $x_1^0 = -1$ and have a small error around $x_2^0 = 1$.

The global convergence of proximal bundle methods was shown in the early stage of their developments; see for example [4, Theorem 4.4]. The local convergence, stating the speed of converge, is shown for the serious step subsequence by interpreting iterates as if generated by an epsilon-subgradient method for which R-linearly convergence is proven.

To see this interpretation, consider only the serious steps with sequences φ^k, $\boldsymbol{p}^k = \boldsymbol{x}^{k+1}$, and $\boldsymbol{g}^k = \mu^k\left(\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\right)$. The properties of the proximal point mapping entail that $\boldsymbol{p}^k = \boldsymbol{x}^k - \frac{1}{\mu^k}\boldsymbol{g}^k$ and $\boldsymbol{g}^k \in \partial\varphi^k\left(\boldsymbol{p}^k\right)$. In view of (9.1), we have for all $\boldsymbol{z} \in \mathbb{R}^n$ that

$$f(z) \geq \varphi^k(z) \geq \varphi^k\left(\boldsymbol{p}^k\right) + \left\langle \boldsymbol{g}^k, z - \boldsymbol{p}^k \right\rangle = f\left(\boldsymbol{x}^k\right) + \left\langle \boldsymbol{g}^k, z - \boldsymbol{x}^k \right\rangle - \varepsilon^k,$$

where

$$\varepsilon^k = f\left(\boldsymbol{x}^k\right) - \varphi^k\left(\boldsymbol{p}^k\right) - \left\langle \boldsymbol{g}^k, \boldsymbol{x}^k - \boldsymbol{p}^k \right\rangle \tag{9.2}$$

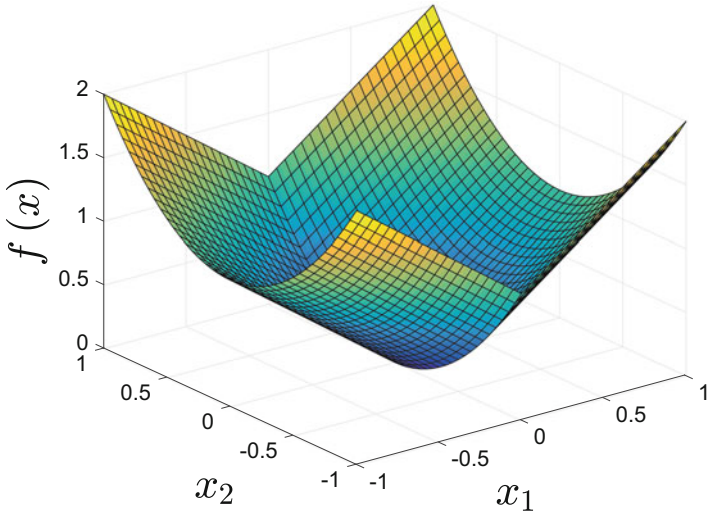

Fig. 9.1 Graph of function f in Example 9.1

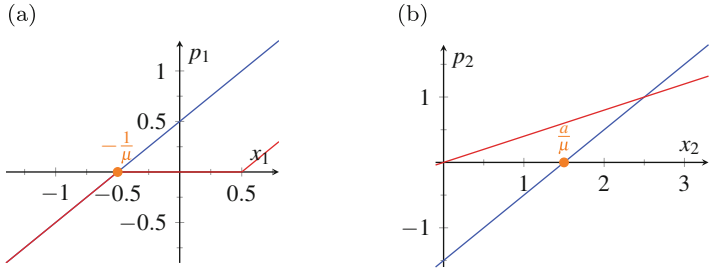

Fig. 9.2 The graphs of $\boldsymbol{p}_\mu\varphi^0$ (blue) and $\boldsymbol{p}_\mu f$ (red) for $a = 3$ and $\mu = 2$. (**a**) In the first component. (**b**) In the second component

is always nonnegative because $\boldsymbol{g}^k \in \partial\varphi^k\left(\boldsymbol{p}^k\right)$. Consequently, $\boldsymbol{g}^k \in \partial_{\varepsilon^k} f\left(\boldsymbol{x}^k\right)$. It follows that in Algorithm 9.1 the serious iterates satisfy the relation

$$\boldsymbol{x}^{k+1} = \boldsymbol{p}^k = \boldsymbol{x}^k - \frac{1}{\mu^k}\boldsymbol{g}^k, \qquad \boldsymbol{g}^k \in \partial_{\varepsilon^k} f\left(\boldsymbol{x}^k\right). \tag{9.3}$$

This is actually an epsilon-subgradient method with step size $\frac{1}{\mu^k}$. In [25] it is showed that the sequence $\{\boldsymbol{x}^k\}$ produced by (9.3) converges at least R-linearly to a minimum point of f (if such a minimum exists), under the following conditions.

Assumption 9.1 (Conditions for R-Linear Convergence) *Consider the problem of minimizing a convex function f with the formula (9.3). Suppose the following conditions hold:*

(i) *the stepsize $\frac{1}{\mu^k}$ is bounded away from 0 and from ∞;*
(ii) *there is a constant $m \in (0, 1]$ such that the inequality*

$$f\left(\boldsymbol{x}^{k+1}\right) \leq f\left(\boldsymbol{x}^{k}\right) + m\left(\left\langle \boldsymbol{g}^{k}, \boldsymbol{x}^{k+1} - \boldsymbol{x}^{k}\right\rangle - \varepsilon^{k}\right)$$

holds for each k;
(iii) *the* inverse growth condition with modulus $\alpha > 0$ *holds for f.*

A convex function f on $\mathbb{R}^n$ is said to satisfy the *inverse growth condition* if there exist a neighborhood $\mathcal{N}$ of the origin in $\mathbb{R}^n$ and a constant α such that for each $\boldsymbol{s} \in \mathcal{N}$, $\partial f^*(\boldsymbol{s}) \subset \partial f^*(\mathbf{0}) + \alpha\|\boldsymbol{s}\|\bar{B}(\mathbf{0}; 1)$, where f^* is the conjugate of f and $\bar{B}(\mathbf{0}; 1)$ is the unit Euclidean ball. With respect to f, the inverse growth condition means that locally f grows faster than a linear function when moving away from its minimum point. As a result, a convex function with non-empty set of minimizers X_* satisfies the inverse growth condition if and only if there exist $c \geq 0$ and $\delta > 0$ such that

$$f(\boldsymbol{x}) \geq \inf f + cd\left(\boldsymbol{x}, X_*\right), \text{ for any } \boldsymbol{x} \in X_* + \delta\bar{B}(\mathbf{0}; 1), \tag{9.4}$$

where $d\left(\boldsymbol{x}, X_*\right)$ denotes the distance function of $\boldsymbol{x}$ to X_* (cf. [29, Theorem 4.3]). For the function f in Example 9.1, the inverse growth condition holds because (9.4) can be satisfied with any $c \in \left(0, \frac{a}{2}\right]$ and $\delta = \frac{\sqrt{2}}{c}$.

In order to show the R-linear convergence, we can apply the Brøndsted-Rockafellar Theorem (see [25, Theorem 2]) on $\boldsymbol{x}^k$ and $\boldsymbol{g}^k$. Assumption 9.1 (iii), of inverse growth condition for a sequence $\boldsymbol{u}_k \in \partial f^*(\mathbf{0}) = X_*$, gives an upper bound for the quantity $\|\boldsymbol{x}^k - \boldsymbol{u}_k\|$ depending on $\boldsymbol{g}^k$ and ε^k. Combining this with the fact that $\boldsymbol{g}^k \in \partial_{\varepsilon^k} f\left(\boldsymbol{x}^k\right)$ and Assumption 9.1 (ii), we deduce $f(\boldsymbol{x}^k) - \inf f = f(\boldsymbol{x}^k) - f(\boldsymbol{u}_k) \leq \kappa\theta^{2n}$ for some constants $\kappa > 0$ and $\theta > 0$. Based on this, then we utilize (9.4) to get an upper bound for $\|\boldsymbol{x}^k - \boldsymbol{u}_k\|$: $\|\boldsymbol{x}^k - \boldsymbol{u}_k\| \leq \lambda\theta^k$ for some constant $\lambda > 0$ and an upper bound for $\|\boldsymbol{x}_* - \boldsymbol{u}_k\|$, where $\boldsymbol{x}_*$ is the limit point of the sequence $\{\boldsymbol{x}^k\}$, $\|\boldsymbol{x}_* - \boldsymbol{u}_k\| \leq \tau\theta^k$ for some constant $\tau > 0$. Finally, putting together the different bounds, the desired inequality $\|\boldsymbol{x}^k - \boldsymbol{x}_*\| \leq \|\boldsymbol{x}^k - \boldsymbol{u}_k\| + \|\boldsymbol{x}_* - \boldsymbol{u}_k\| \leq (\lambda + \tau)\,\theta^k$ holds.

Regarding Algorithm 9.1, the update of prox-parameters at serious steps can be done so that Assumption 9.1 (i) is satisfied. The descent test ensures that Assumption 9.1 (ii) holds for such subsequence, using (9.2) and recalling that at serious steps $f\left(\boldsymbol{x}^k\right) - f\left(\boldsymbol{x}^{k+1}\right) \geq m\left[f\left(\boldsymbol{x}^k\right) - \varphi^k\left(\boldsymbol{x}^{k+1}\right)\right]$. It follows that, under the inverse growth condition, proximal bundle methods are R-linearly convergent on the serious step subsequence.

Remark 9.1 (Exact Proximal Point Method) For some simple functions, the proximal point operator can be computed easily without resorting to a bundle mechanism and a model. This is the case for the L_1-norm, whose proximal mapping is called soft-thresholding operator.

The proximal point algorithm defines

$$\boldsymbol{x}^{k+1} = \boldsymbol{p}_{\mu^k} f(\boldsymbol{x}^k) = \boldsymbol{x}^k - \frac{1}{\mu^k}\boldsymbol{g}^k \quad \text{for } \boldsymbol{g}^k \in \partial f(\boldsymbol{x}^{k+1}). \tag{9.5}$$

By following a reasoning similar to the one in (9.3), the relation $\boldsymbol{g}^k \in \partial_{\varepsilon^k} f(\boldsymbol{x}^k)$ holds now for an error

$$\varepsilon^k = f\left(\boldsymbol{x}^k\right) - f\left(\boldsymbol{x}^{k+1}\right) - \left\langle \boldsymbol{g}^k, \boldsymbol{x}^k - \boldsymbol{x}^{k+1}\right\rangle$$

instead of (9.2). Therefore, Assumption 9.1 (ii) is in fact an equality, which holds with $m = 1$. As long as the prox-parameter sequence is kept bounded, the proximal point method is also R-linear convergent for functions satisfying the inverse growth condition.

An additional property of the proximal point operator is its relation with the Moreau-Yosida regularization of f:

$$F_\mu(\cdot) := \min_{\boldsymbol{y}\in\mathbb{R}^n} f(\boldsymbol{y}) + \frac{\mu}{2}\|\boldsymbol{y} - \cdot\|^2,$$

whose Lipschitzian gradient is given by $\nabla F_\mu(\boldsymbol{x}) = \mu\Big(\boldsymbol{x} - \boldsymbol{p}_\mu f(\boldsymbol{x})\Big)$. Since minimizing f is equivalent to minimizing F_μ, the proximal iteration (9.5) can be interpreted as a preconditioned gradient step to minimize the regularization:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \frac{1}{\mu^k}\nabla F_{\mu^k}(\boldsymbol{x}^k).$$

This interpretation sheds light on the role of the proximal parameter in terms of convergence speed. For the method to be more rapid, $1/\mu^k$ should contain information on F's "curvature" and approximate the Moreau-Yosida Hessian. For this to happen, the Hessian should exist. For Example 9.1 there is no curvature in the first component whenever $\mu|x_1| \leq 1$, as

$$\nabla F_\mu(\boldsymbol{x}) = \begin{pmatrix} 0 \\ \dfrac{a\mu}{a+\mu}x_2 \end{pmatrix}.$$

By contrast, on the second component a second-order object that could be exploited algorithmically does exists, namely $\dfrac{a\mu}{a+\mu}$.

As illustrated by Example 9.1, it is possible to decompose the space into components along which some kind of second-order object can be defined. This is the basis of the $\mathcal{VU}$-decomposition, introduced below.

9.3 Introduction of $\mathcal{VU}$-Analysis

The pursuit of superlinear convergent methods leads us to the study of second-order differentiability of a convex function, such as the second-order expansion used in Newton method for C^2 functions. In the nonsmooth case, we often find convex functions that, when restricted to a special trajectory, admit a second-order expansion at a certain point $\bar{\boldsymbol{x}}$ in the trajectory. Consider again the function in Example 9.1. Restricted to its only ridge of nondifferentiability $\mathcal{M} := \{\boldsymbol{x} \in \mathbb{R}^n : x_1 = 0\}$, the function is quadratic: $f|_{\mathcal{M}} = \frac{a}{2}x_2^2$. The function looks smooth along the manifold $\mathcal{M}$.

The trajectory of "apparent smoothness" is parametrized by a variable $\boldsymbol{u}$ in a subspace $\mathcal{U}$ of $\mathbb{R}^n$ associated with $\bar{\boldsymbol{x}}$. If such trajectory exists for f then it is called a *fast track* of f and the underlying second-order expansion is a *partial second-order expansion* of f. For the function f in Example 9.1, the fast track is just $\mathcal{M}$ and its partial second-order expansion is just $f|_{\mathcal{M}} = \frac{a}{2}x_2^2$.

In this section we give a basic introduction of $\mathcal{VU}$-analysis. We illustrate by examples that some convex nonsmooth functions have fast tracks over which a partial second-order expansion exists. We use the following notations:

$\operatorname{aff} C$	the affine hull of the set $C \subset \mathbb{R}^n$;
$\operatorname{ri} C$	the relative interior of C;
$\boldsymbol{x}_{\mathcal{S}}$	the projection of a point $\boldsymbol{x} \in \mathbb{R}^n$ onto a subspace $\mathcal{S} \subset \mathbb{R}^n$;
$N_D(z)$	the normal cone to the convex set D at $z \in D$;
$T_D(z)$	the tangent cone to the convex set D at $z \in D$;
$\mathcal{S}^{\perp}$	the orthogonal complement of the linear subspace $\mathcal{S}$;
$P_S(C)$	the projection of the points in set C onto set S;
$\|\cdot\|_{\mathcal{S}}$	the Euclidean norm of the linear subspace $\mathcal{S}$ induced by $\mathbb{R}^n$;
$\langle\cdot,\cdot\rangle_{\mathcal{S}}$	the inner product of the linear subspace $\mathcal{S}$ induced by $\mathbb{R}^n$.

We first give formal definitions of the notions in $\mathcal{VU}$-analysis.

Definition 9.1 ($\mathcal{VU}$-Objects) Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and a point $\bar{\boldsymbol{x}} \in \mathbb{R}^n$, let $\boldsymbol{g}$ be any subgradient in $\partial f(\bar{\boldsymbol{x}})$.

- The $\mathcal{VU}$-*decomposition* [13] of $\mathbb{R}^n$ at $\bar{\boldsymbol{x}}$ is defined by two orthogonal subspaces such that $\mathbb{R}^n = \mathcal{U}(\bar{\boldsymbol{x}}) \oplus \mathcal{V}(\bar{\boldsymbol{x}})$ where

$$\mathcal{V}(\bar{\boldsymbol{x}}) = \operatorname{span}(\partial f(\bar{\boldsymbol{x}}) - \boldsymbol{g}) = \operatorname{aff}(\partial f(\bar{\boldsymbol{x}})) - \boldsymbol{g}, \quad \mathcal{U}(\bar{\boldsymbol{x}}) = \mathcal{V}(\bar{\boldsymbol{x}})^{\perp}. \tag{9.6}$$

  Unless specified, we will use $\mathcal{U}$ and $\mathcal{V}$ to denote $\mathcal{U}(\bar{\boldsymbol{x}})$ and $\mathcal{V}(\bar{\boldsymbol{x}})$, respectively. Accordingly, each $\boldsymbol{x} \in \mathbb{R}^n$ can be decomposed into $\boldsymbol{x} = \boldsymbol{x}_{\mathcal{U}} \oplus \boldsymbol{x}_{\mathcal{V}}$.

- The $\mathcal{U}$-*Lagrangian* of f depending on the $\mathcal{V}$-component $\boldsymbol{g}_{\mathcal{V}}$ is defined by

$$\mathcal{U} \ni \boldsymbol{u} \mapsto L_{\mathcal{U}}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right) := \inf_{\boldsymbol{v} \in \mathcal{V}} \left\{ f(\bar{\boldsymbol{x}} + \boldsymbol{u} \oplus \boldsymbol{v}) - \left\langle \boldsymbol{g}_{\mathcal{V}}, \boldsymbol{v} \right\rangle_{\mathcal{V}} \right\},$$

and the associated set of $\mathcal{V}$-space minimizers is

$$W\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right) := \left\{ \boldsymbol{v} \in \mathcal{V} \colon L_{\mathcal{U}}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right) = f(\bar{\boldsymbol{x}} + \boldsymbol{u} \oplus \boldsymbol{v}) - \left\langle \boldsymbol{g}_{\mathcal{V}}, \boldsymbol{v} \right\rangle_{\mathcal{V}} \right\}.$$

From the definition, we see that the subspace $\mathcal{V}$ is vacuous wherever f is differentiable. When f is not differentiable, it is often smooth or "U-shaped" in $\bar{\boldsymbol{x}} + \mathcal{U}$, and "V-shaped" in $\bar{\boldsymbol{x}} + \mathcal{V}$. This can be seen from the graph of the function in Example 9.1. At any point $\bar{\boldsymbol{x}} = (0, b)$ we have $\partial f\left(\bar{\boldsymbol{x}}\right) = [-1, 1] \times \{ab\}$ and we can calculate easily $\mathcal{V} = \mathbb{R} \times \mathbf{0}$ with $\mathcal{U} = \mathbf{0} \times \mathbb{R}$. From Fig. 9.1, we see indeed that in the v-axis f is "V-shaped" and in the u-axis it is "U-shaped".

The next example, from [11], has a more complicated and interesting structure.

Example 9.2 Consider the bivariate function

$$f(\boldsymbol{x}) = \max\left\{ f_1(\boldsymbol{x}), f_2(\boldsymbol{x}) \right\} = \max\left\{ \frac{1}{2}\|\boldsymbol{x}\|^2 - \alpha \left\langle \boldsymbol{e}, \boldsymbol{x} \right\rangle, \left\langle \boldsymbol{e}, \boldsymbol{x} \right\rangle \right\},$$

where $\boldsymbol{e} = (0, 1)^T$, α is a nonnegative parameter and whose graph is shown in Fig. 9.3. The subdifferential at $\bar{\boldsymbol{x}} = (0, 0)$ is the segment $[-\alpha\boldsymbol{e}, \boldsymbol{e}]$, and thus f is minimized at $\bar{\boldsymbol{x}}$. From the definition of the $\mathcal{VU}$-decomposition it is easy to calculate here

$$\mathcal{V}\left(\bar{\boldsymbol{x}}\right) = \mathbf{0} \times \mathbb{R} \quad \text{and} \quad \mathcal{U}\left(\bar{\boldsymbol{x}}\right) = \mathbb{R} \times \mathbf{0}.$$

Actually, we can calculate the subspaces $\mathcal{V}(\boldsymbol{x})$ and $\mathcal{U}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^2$. Note that $\mathcal{V}(\boldsymbol{x})$ is the 0 subspace if f is differentiable at $\boldsymbol{x}$, in which case $\mathcal{U}(\boldsymbol{x}) = \mathbb{R}^2$. If $\boldsymbol{x}$ is such that $f_1(\boldsymbol{x}) = f_2(\boldsymbol{x})$, then $\partial f(\boldsymbol{x})$ is just the line segment $[\nabla f_1(\boldsymbol{x}), \nabla f_2(\boldsymbol{x})] = [\boldsymbol{x} - \alpha\boldsymbol{e}, \boldsymbol{e}]$. By definition, the subspace $\mathcal{V}(\boldsymbol{x})$ is the line spanned by the point $(k, 1)$ where $k = \frac{x_1}{x_2 - \alpha - 1}$ is the slope of the line segment. The subspace $\mathcal{U}(\boldsymbol{x})$ is therefore the line perpendicular to $\mathcal{V}(\boldsymbol{x})$, that is, the line spanned by $\left(-\frac{1}{k}, 1\right)$.

Definition 9.2 (Fast Track) Given $\boldsymbol{g} \in \operatorname{ri} \partial f\left(\bar{\boldsymbol{x}}\right)$, we say that the set

$$\mathcal{F} := \left\{ \bar{\boldsymbol{x}} + \boldsymbol{u} \oplus \boldsymbol{v}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right) : \boldsymbol{u} \in B\left(\mathbf{0}; \tau\right) \cap \mathcal{U} \right\}$$

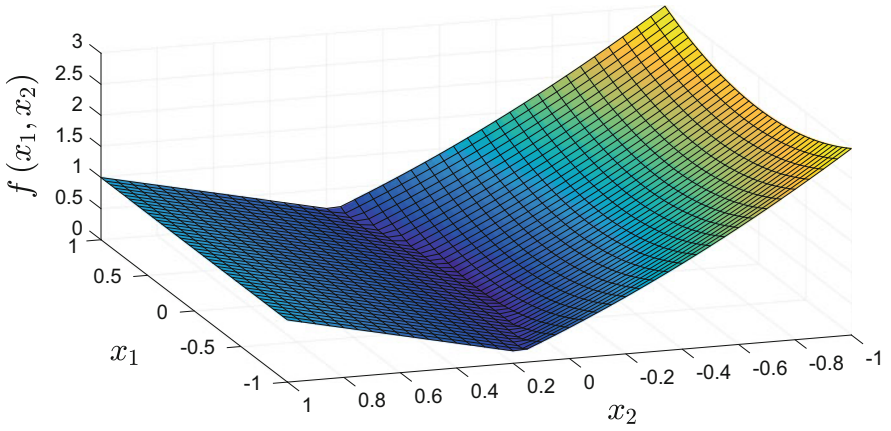

Fig. 9.3 Graph of function f in Example 9.2

is a *fast track* of f leading to a point $\bar{\boldsymbol{x}}$ associated with $\boldsymbol{g}$, if there exist $\tau > 0$ and a function $\boldsymbol{v} : \mathcal{U} \mapsto \mathcal{V}$ such that

(i) $\boldsymbol{v}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right)|_{B(\mathbf{0};\tau)\cap\mathcal{U}}$ is a C^2-function;
(ii) $\boldsymbol{v}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right)|_{B(\mathbf{0};\tau)\cap\mathcal{U}}$ satisfies $\boldsymbol{v}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right)|_{B(\mathbf{0};\tau)\cap\mathcal{U}} \in W\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right)$;
(iii) $L_{\mathcal{U}}\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right)|_{B(\mathbf{0};\tau)\cap\mathcal{U}}$ is a C^2-function.

Remark 9.2 The fast track definition above is different from that in [17, Definition 2.1] where the fast track is associated with a minimizer of f and is called a *fast track leading to a minimizer*. We remove this requirement and allow $\bar{\boldsymbol{x}}$ to be any point, because typically instead of a minimizer, one has an iterate, say $\boldsymbol{x}^k$. For the same reason, differently from [17], in condition (ii) the subgradient can be any, not just $\boldsymbol{g} = \mathbf{0}$. Another important difference is that in [17] the function $\boldsymbol{v}$ is *independent* of $\boldsymbol{g}_{\mathcal{V}}$, as it is supposed to be in the intersection of $W\left(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}\right)$ for all $\boldsymbol{g} \in \operatorname{ri} \partial f\left(\bar{\boldsymbol{x}}\right)$. The analysis below is developed without assuming such a restrictive condition which is hard to satisfy even for functions having second-order objects along a subspace. This is shown by the following example.

Example 9.3 Consider the bivariate function

$$f(v, u) = \max\left\{f^1(v),\, f^2(u)\right\} = \max\left\{|v|,\, \frac{a}{2}u^2\right\},$$

where a is a positive scalar. This function is differentiable everywhere except on the locus of the equation $|v| = \frac{a}{2}u^2$. The set defined by that locus is curved and not linear as the manifold $\mathcal{M}$ in Example 9.1.

The unique minimizer of our function is $\bar{\boldsymbol{x}} = (0, 0)$, where the subdifferential is $\partial f(\bar{\boldsymbol{x}}) = [-1, 1] \times \{\mathbf{0}\}$. From Fig. 9.4 we see that the graph of f is "U-shaped" along the u-axis and "V-shaped" along the v-axis.

Consider the two subspaces $\mathcal{V} = \mathbb{R} \times \mathbf{0}$ and $\mathcal{U} = \mathbf{0} \times \mathbb{R}$. We now construct a trajectory called fast track, over which f admits a second-order expansion parametrized by $\boldsymbol{u} \in \mathcal{U}$. Let γ be the $\mathcal{V}$-component of any element in $\operatorname{ri} \partial f(\bar{\boldsymbol{x}})$. Then as $\operatorname{ri} \partial f(\bar{\boldsymbol{x}}) = (-1, 1) \times \mathbf{0}$ we have $\gamma \in (-1, 1)$. Working out the calculations of three cases, we get

$$W(u; \gamma) = \begin{cases} \left\{\frac{a}{2}u^2\right\}, & \text{if } \gamma \in (0, 1), \\ \left\{v : |v| \leq \frac{a}{2}u^2\right\}, & \text{if } \gamma = 0, \\ \left\{-\frac{a}{2}u^2\right\}, & \text{if } \gamma \in (-1, 0). \end{cases} \tag{9.7}$$

Note that when it is clear that a point is in $\mathbb{R} \times \mathbf{0}$, we omit the 0 component. In view of the relationship between W and $L_{\mathcal{U}}$ we readily obtain

$$L_{\mathcal{U}}(\boldsymbol{u}; \gamma) = (1 - |\gamma|) \frac{a}{2}u^2. \tag{9.8}$$

Fix any $|\gamma| < 1$. We define a function $v : \mathcal{U} \mapsto \mathcal{V}$ as

$$v(u; \gamma) = \frac{a}{2} \operatorname{sign}(\gamma) u^2, \tag{9.9}$$

which belongs to the set $W(u; \gamma)$. Since both $v(u; \gamma)$ and $L_{\mathcal{U}}(u; \gamma)$ are C^2-functions, we have constructed the following fast-track for f,

$$\begin{aligned} \mathcal{F}_\gamma &= \{\bar{\boldsymbol{x}} + (v(u; \gamma), u) : u \in \mathcal{U}\} \\ &= \{\chi(u; \gamma) : u \in \mathcal{U}\}, \end{aligned}$$

where $\chi(u; \gamma) := \left(\frac{a}{2} \operatorname{sign}(\gamma) u^2,\ u\right)$. We call the set $\mathcal{F}_\gamma$ a fast track because f can be expanded up to second order along it:

$$f(\chi(u; \gamma)) = \frac{a}{2}u^2, \quad \nabla_{\mathcal{U}} f(\chi(u; \gamma)) = au, \quad \text{and} \quad \nabla^2_{\mathcal{U}} f(\chi(u; \gamma)) = a.$$

Notice that if we were to follow the definition of a fast track in [17, Definition 2.1], then the fast track for f would shrink to only $\left\{(0, 0)^T\right\}$, the minimizer itself. Thus the new definition of a fast track, parametrized by $\boldsymbol{g}_{\mathcal{V}}$, generalizing the original notion.

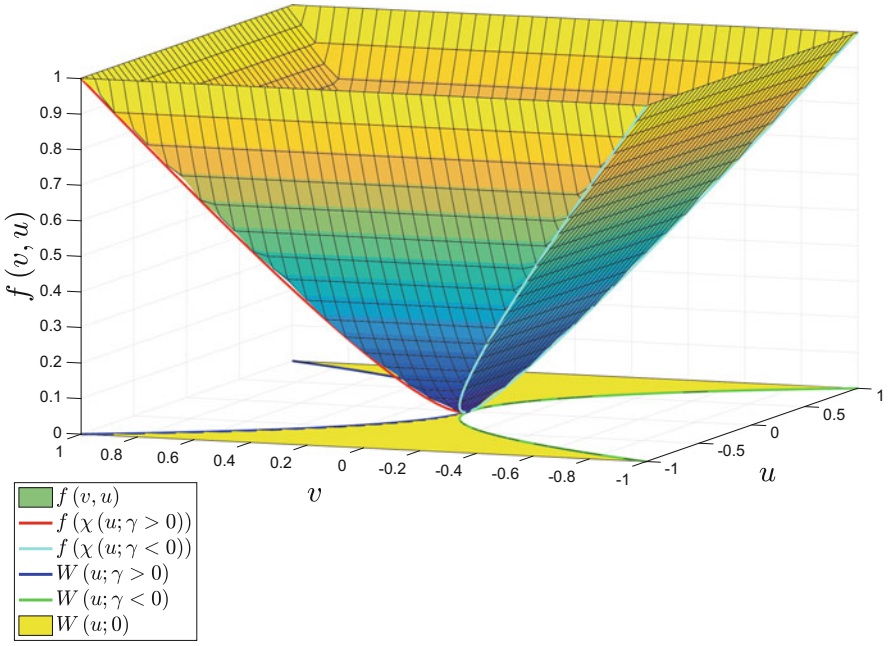

Fig. 9.4 Some objects of the function f in Example 9.3

Remark 9.3 A concept closely related to a fast track is *partial smoothness* [14, Definition 2.7]. A convex function h is said to be partly smooth at $\boldsymbol{x}$ relative to a set $\mathcal{M}$, if $\mathcal{M}$ is a manifold around $\boldsymbol{x}$ and the following three properties hold:

(i) (restricted smoothness) the restriction $h|_{\mathcal{M}}$ is smooth around $\boldsymbol{x}$;
(ii) (normals parallel to subdifferential) $N_{\mathcal{M}}(\boldsymbol{x}) = \mathcal{V}(\boldsymbol{x})$;
(iii) (subgradient continuity) the subdifferential ∂h is continuous at $\boldsymbol{x}$ relative to $\mathcal{M}$.

The second property above is equivalent to $T_{\mathcal{M}}(\boldsymbol{x}) = \mathcal{U}(\boldsymbol{x})$. As an exercise, the readers can verify that the function f in Example 9.3 is partly smooth at any point in $\mathcal{M} := \left\{(v, u) : v = \frac{a}{2}u^2\right\}$ relative to $\mathcal{M}$ itself.

The $\mathcal{VU}$-decomposition and the $\mathcal{U}$-Lagrangian satisfy the following properties which are crucial for the development of $\mathcal{VU}$-algorithms.

Proposition 9.1 ($\mathcal{VU}$-Relations) *Given any $\boldsymbol{g} \in \operatorname{ri} \partial f(\bar{\boldsymbol{x}})$, the subspaces $\mathcal{U}$ and $\mathcal{V}$ and the $\mathcal{U}$-Lagrangian satisfy*

$$\mathcal{U} = \left\{\boldsymbol{w} \in \mathbb{R}^n : f'(\bar{\boldsymbol{x}}; -\boldsymbol{w}) = -f'(\bar{\boldsymbol{x}}; \boldsymbol{w})\right\} = N_{\partial f(\bar{\boldsymbol{x}})}(\boldsymbol{g}), \tag{9.10}$$

$$\mathcal{V} = T_{\partial f(\bar{\boldsymbol{x}})}(\boldsymbol{g}),$$

$$W(\boldsymbol{u}; \boldsymbol{g}_{\mathcal{V}}) \neq \emptyset,\ W(\boldsymbol{0}; \boldsymbol{g}_{\mathcal{V}}) = \{\boldsymbol{0}\},\ L_{\mathcal{U}}(\boldsymbol{0}; \boldsymbol{g}_{\mathcal{V}}) = f(\bar{\boldsymbol{x}}), \tag{9.11}$$

$$\partial L_{\mathcal{U}}\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right)=\left\{\boldsymbol{g}'_{\mathcal{U}}\colon \boldsymbol{g}'_{\mathcal{U}}\oplus\boldsymbol{g}_{\mathcal{V}}\in\partial f\left(\overline{\boldsymbol{x}}+\boldsymbol{u}\oplus\boldsymbol{w}\right)\right\},\ \boldsymbol{w}\in W\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right), \tag{9.12}$$

$$L_{\mathcal{U}}\ \textit{is convex and is differentiable at}\ \mathbf{0},\ \textit{with}\ \nabla L_{\mathcal{U}}\left(\mathbf{0};\boldsymbol{g}_{\mathcal{V}}\right)=\boldsymbol{g}_{\mathcal{U}}, \tag{9.13}$$

$$W\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right)=o\left(\|\boldsymbol{u}\|_{\mathcal{U}}\right),\ \ \textit{for all}\ \boldsymbol{g}\in\operatorname{ri}\partial f\left(\overline{\boldsymbol{x}}\right), \tag{9.14}$$

$$\textit{the projection}\ P_{\mathcal{U}}\left(\partial f\left(\overline{\boldsymbol{x}}\right)\right)\ \textit{is a singleton,} \tag{9.15}$$

$$W\left((\boldsymbol{x}-\overline{\boldsymbol{x}})_{\mathcal{U}}\,;\boldsymbol{g}_{\mathcal{V}}\right)=T\left(\boldsymbol{x};\boldsymbol{g}_{\mathcal{V}}\right)+\left\{(\boldsymbol{x}-\overline{\boldsymbol{x}})_{\mathcal{V}}\right\}, \tag{9.16}$$

$$\partial L_{\mathcal{U}}\left((\boldsymbol{x}-\overline{\boldsymbol{x}})_{\mathcal{U}}\,;\boldsymbol{g}_{\mathcal{V}}\right)=\left\{\boldsymbol{g}'_{\mathcal{U}}\colon \boldsymbol{g}'_{\mathcal{U}}\oplus\boldsymbol{g}_{\mathcal{V}}\in\partial f\left(\boldsymbol{x}+\mathbf{0}\oplus\boldsymbol{v}^{*}\right)\right\}, \tag{9.17}$$

where $W\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right)=o\left(\|\boldsymbol{u}\|_{\mathcal{U}}\right)$ *is a short hand for*

$$\forall\,\epsilon>0\ \exists\,\delta>0\colon\ \|\boldsymbol{u}\|_{\mathcal{U}}\le\delta\Rightarrow\|\boldsymbol{w}\|_{\mathcal{V}}\le\epsilon\|\boldsymbol{u}\|_{\mathcal{U}}\ \textit{for any}\ \boldsymbol{w}\in W\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right),$$

$\boldsymbol{v}^{*}\in T\left(\boldsymbol{x};\boldsymbol{g}_{\mathcal{V}}\right)$ *and*

$$T\left(\boldsymbol{x};\boldsymbol{g}_{\mathcal{V}}\right):=\operatorname*{argmin}_{\boldsymbol{v}\in\mathcal{V}}\left\{f\left(\boldsymbol{x}+\mathbf{0}\oplus\boldsymbol{v}\right)-\left\langle\boldsymbol{g}_{\mathcal{V}},\boldsymbol{v}\right\rangle\right\}. \tag{9.18}$$

Proof Properties (9.10)–(9.14) are from Proposition 2.2, Theorems 3.2 and 3.3, and Corollary 3.5 of [13]. To see (9.15), observe from (9.6) that $\partial f\left(\overline{\boldsymbol{x}}\right)\subset\mathcal{V}+\boldsymbol{g}$ and the orthogonality of $\mathcal{V}$ and $\mathcal{U}$ gives $P_{\mathcal{U}}\left(\partial f\left(\overline{\boldsymbol{x}}\right)\right)\subset\boldsymbol{g}_{\mathcal{U}}$. To show (9.16), shorten the notation to $W:=W\left((\boldsymbol{x}-\overline{\boldsymbol{x}})_{\mathcal{U}}\,;\boldsymbol{g}_{\mathcal{V}}\right)$ write from its definition

$$\begin{aligned}
W&=\operatorname*{argmin}_{\boldsymbol{v}'\in\mathcal{V}}\left\{f\left(\overline{\boldsymbol{x}}+(\boldsymbol{x}-\overline{\boldsymbol{x}})_{\mathcal{U}}\oplus\boldsymbol{v}'\right)-\left\langle\boldsymbol{g}_{\mathcal{V}},\boldsymbol{v}'\right\rangle\right\}\\
&=\operatorname*{argmin}_{\boldsymbol{v}'\in\mathcal{V}}\left\{f\left(\boldsymbol{x}+\mathbf{0}\oplus\left(\boldsymbol{v}'+(\overline{\boldsymbol{x}}-\boldsymbol{x})_{\mathcal{V}}\right)\right)-\left\langle\boldsymbol{g}_{\mathcal{V}},\boldsymbol{v}'\right\rangle\right\}\\
&=\operatorname*{argmin}_{\boldsymbol{v}'\in\mathcal{V}}\left\{f\left(\boldsymbol{x}+\mathbf{0}\oplus\left(\boldsymbol{v}'+(\overline{\boldsymbol{x}}-\boldsymbol{x})_{\mathcal{V}}\right)\right)\right.\\
&\qquad\qquad\left.-\left\langle\boldsymbol{g}_{\mathcal{V}},\boldsymbol{v}'+(\overline{\boldsymbol{x}}-\boldsymbol{x})_{\mathcal{V}}\right\rangle+\left\langle\boldsymbol{g}_{\mathcal{V}},(\overline{\boldsymbol{x}}-\boldsymbol{x})_{\mathcal{V}}\right\rangle\right\}\\
&=\operatorname*{argmin}_{\boldsymbol{v}\in\mathcal{V}}\left\{f\left(\boldsymbol{x}+\mathbf{0}\oplus\boldsymbol{v}\right)-\left\langle\boldsymbol{g}_{\mathcal{V}},\boldsymbol{v}\right\rangle\right\}+\left\{(\boldsymbol{x}-\overline{\boldsymbol{x}})_{\mathcal{V}}\right\}.
\end{aligned} \tag{9.19}$$

By (9.19), combined with (9.12), we immediately get (9.17). □

In order to verify that (9.16) holds for the function f in Example 9.3, we decompose $\boldsymbol{x}$ into $x_{\mathcal{U}}\oplus x_{\mathcal{V}}=(u_x,v_x)$ and find the expression for the set $T\left(\boldsymbol{x};\gamma\right)=\operatorname{argmin}_v\left\{\max\left\{|v_x+v|,\frac{a}{2}u_x^2\right\}-\gamma v\right\}$. Simple calculations yield

$$T\left(\boldsymbol{x};\gamma\right)=\begin{cases}\left\{\frac{a}{2}u_x^2-v_x\right\}, & \text{if } \gamma\in(0,1),\\ \left\{v\colon |v+v_x|\le\frac{a}{2}u_x^2\right\}, & \text{if } \gamma=0,\\ \left\{-\frac{a}{2}u_x^2-v_x\right\}, & \text{if } \gamma\in(-1,0).\end{cases}$$

Omitting the 0 component, $(\boldsymbol{x}-\bar{\boldsymbol{x}})_{\mathcal{V}}=v_{\boldsymbol{x}}$ and thus, by (9.7),

$$T\left(\boldsymbol{x};\gamma\right)+(\boldsymbol{x}-\bar{\boldsymbol{x}})_{\mathcal{V}}=T\left(u_x;\gamma\right).$$

Finally, in view of (9.11) and (9.13),

$$L_{\mathcal{U}}\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right)=f\left(\bar{\boldsymbol{x}}\right)+\left\langle \boldsymbol{g}_{\mathcal{U}},\boldsymbol{u}\right\rangle+o\left(\|\boldsymbol{u}\|\right).$$

Essentially, the $\mathcal{U}$-Lagrangian can be considered as a first-order expansion of f in the $\mathcal{U}$ subspace. If f is minimized at $\bar{\boldsymbol{x}}$ then $\mathbf{0}\in\partial f\left(\bar{\boldsymbol{x}}\right)$. In view of (9.11) and (9.15), we get that $L_{\mathcal{U}}$ is minimized at $\mathbf{0}$. Moreover, the third equation of (9.11), implies that $L_{\mathcal{U}}$ has the same optimal value with f.

A convex function φ is said to have a *generalized Hessian* $H\varphi\left(\boldsymbol{x}_0\right)$ at a point $\boldsymbol{x}_0$ if the gradient $\nabla\varphi\left(\boldsymbol{x}_0\right)$ exists and there exists a symmetric positive semidefinite operator $H\varphi\left(\boldsymbol{x}_0\right)$ such that

$$\partial\varphi\left(\boldsymbol{x}_0+\boldsymbol{d}\right)\subset\nabla\varphi\left(\boldsymbol{x}_0\right)+H\varphi\left(\boldsymbol{x}_0\right)\boldsymbol{d}+B\left(\mathbf{0};o\left(\|\boldsymbol{d}\|\right)\right).$$

If $L_{\mathcal{U}}\left(\cdot,\boldsymbol{g}_{\mathcal{V}}\right)$ has a generalized Hessian at $\mathbf{0}$, then the Hessian is called a $\mathcal{U}$-Hessian of f at $\bar{\boldsymbol{x}}$ associated with $\boldsymbol{g}_{\mathcal{V}}$, and denoted by $\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}}f\left(\bar{\boldsymbol{x}}\right)$. Consequently, the existence of a $\mathcal{U}$-Hessian $\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}}f\left(\bar{\boldsymbol{x}}\right)$ implies

$$\partial L_{\mathcal{U}}\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right)\subset\boldsymbol{g}_{\mathcal{U}}+\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}}f\left(\bar{\boldsymbol{x}}\right)\boldsymbol{u}+B\left(\mathbf{0};o\left(\|\boldsymbol{u}\|_{\mathcal{U}}\right)\right). \tag{9.20}$$

In the case of Example 9.3, from (9.8) we see that $\nabla L_{\mathcal{U}}\left(\boldsymbol{u};\gamma\right)=\left(1-|\gamma|\right)a\boldsymbol{u}$ and $\nabla^2 L_{\mathcal{U}}\left(\boldsymbol{u};\gamma\right)=\left(1-|\gamma|\right)a$. Note that here $\mathrm{H}_{\mathcal{U}}^{\gamma}f\left(\bar{\boldsymbol{x}}\right)$ is a 2×2 matrix with all elements 0 except the lower-right corner which equals $\left(1-|\gamma|\right)a$.

9.4 A Conceptual $\mathcal{VU}$-Algorithm

Based on the second-order information of a function in the $\mathcal{U}$ subspace, a conceptual $\mathcal{VU}$-algorithm with superlinear convergence can be constructed [13]. Such an algorithm minimizes the $\mathcal{U}$-Lagrangian via utilizing the $\mathcal{U}$-Hessian so that a Newton step in the $\mathcal{U}$ subspace is possible.

Assumption 9.2 (On the $\mathcal{U}$-Hessian) *For a minimum point $\bar{\boldsymbol{x}}$ of f, $\mathbf{0}\in\mathrm{ri}\,\partial f\left(\bar{\boldsymbol{x}}\right)$ and there exists $\boldsymbol{g}\in\mathrm{ri}\,\partial f\left(\bar{\boldsymbol{x}}\right)$ such that the $\mathcal{U}$-Hessian $\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}}f\left(\bar{\boldsymbol{x}}\right)$ exists and is positive definite in $\mathcal{U}$.*

Under Assumption 9.2, the relation (9.15), combined with (9.13) and (9.20), implies that, for any minimum point $\bar{\boldsymbol{x}}$ and any $\boldsymbol{g}\in\mathrm{ri}\,\partial f\left(\bar{\boldsymbol{x}}\right)$,

$$\partial L_{\mathcal{U}}\left(\boldsymbol{u};\boldsymbol{g}_{\mathcal{V}}\right)\subset\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}}f\left(\bar{\boldsymbol{x}}\right)\boldsymbol{u}+B\left(\mathbf{0};o\left(\|\boldsymbol{u}\|\right)\right). \tag{9.21}$$

This condition implies a critical property useful for the convergence analysis. Fix any $\boldsymbol{g} \in \operatorname{ri} \partial f(\bar{\boldsymbol{x}})$. If there are two sequences $\boldsymbol{g}_{\mathcal{U}}^k$ and $\boldsymbol{u}_k$ in $\mathcal{U}$ satisfying $\boldsymbol{g}_{\mathcal{U}}^k \in \partial L_{\mathcal{U}}\left(\boldsymbol{u}_k; \boldsymbol{g}_{\mathcal{V}}\right)$ then

$$\boldsymbol{g}_{\mathcal{U}}^k = \mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}} f(\bar{\boldsymbol{x}})\, \boldsymbol{u}_k + o(\|\boldsymbol{u}_k\|). \tag{9.22}$$

For the function f in Example 9.3, this corresponds to the identity

$$(1-|\gamma|)\, a\boldsymbol{u}_k = \begin{bmatrix} 0 & 0 \\ 0 & (1-|\gamma|)\, a \end{bmatrix} \boldsymbol{u}_k.$$

The conceptual algorithm in [13], given below, exploits these relations, assuming perfect knowledge of the $\mathcal{VU}$-decomposition and the $\mathcal{U}$-Hessian at a minimizer.

Algorithm 9.2: Algorithm 4.5 of [13] with any fixed $\boldsymbol{g} \in \operatorname{ri} \partial f(\bar{\boldsymbol{x}})$

Data: starting point $\boldsymbol{x}^0$, the subspaces $\mathcal{U}$ and $\mathcal{V}$; the $\mathcal{V}$-gradient component $\boldsymbol{g}_{\mathcal{V}}$ and the $\mathcal{U}$-Hessian $\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}} f(\bar{\boldsymbol{x}})$ are known, where $\bar{\boldsymbol{x}}$ is a minimum point of f.

1 *$\mathcal{V}$-Step*. At iteration point $\boldsymbol{x}^k$, find an element

$$\boldsymbol{v}_k^* \in T\left(\boldsymbol{x}^k; \boldsymbol{g}_{\mathcal{V}}\right) = \operatorname*{argmin}_{\boldsymbol{v}\in\mathcal{V}} \left\{ f\left(\boldsymbol{x}^k + \boldsymbol{0} \oplus \boldsymbol{v}\right) - \left\langle \boldsymbol{g}_{\mathcal{V}}, \boldsymbol{v} \right\rangle \right\} \tag{9.23}$$

(cf. (9.18)). Set $\boldsymbol{p}^k := \boldsymbol{x}^k + \boldsymbol{0} \oplus \boldsymbol{v}_k^*$.

2 *$\mathcal{U}$-Step*. Take $\boldsymbol{g}^k \in \partial f\left(\boldsymbol{p}^k\right)$ such that $\boldsymbol{g}_{\mathcal{V}}^k = \boldsymbol{g}_{\mathcal{V}}$ so that $\boldsymbol{g}_{\mathcal{U}}^k \in \partial L_{\mathcal{U}}\left(\left(\boldsymbol{p}^k - \bar{\boldsymbol{x}}\right)_{\mathcal{U}}; \boldsymbol{g}_{\mathcal{V}}\right)$. Make a Newton step in $\boldsymbol{p}^k + \mathcal{U}$: compute the solution $\boldsymbol{u}_k^*$ of

$$\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}} f(\bar{\boldsymbol{x}})\, \boldsymbol{u} = -\boldsymbol{g}_{\mathcal{U}}^k\,. \tag{9.24}$$

3 *Update*. Set $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{p}^k + \boldsymbol{u}^* \oplus \boldsymbol{0} = \boldsymbol{x}^k + \boldsymbol{u}_k^* \oplus \boldsymbol{v}_k^*$. Set $k \leftarrow k+1$ and go to line 1.

The idea behind this algorithm is that first the $\mathcal{V}$-step generates some descent of f in $\mathcal{V}$ fixing the position of the iteration point in $\mathcal{U}$. Then the algorithm applies a Newton step as the $\mathcal{U}$-step on the $\mathcal{U}$-Lagrangian of f to produce a direction in $\mathcal{U}$. The next iteration point is a sum of the two steps.

Let us consider the function f in Example 9.3 and take $\boldsymbol{x}^0 = (2,3)^T$, $a = 2$ and $\boldsymbol{g}_{\mathcal{V}} = 0.5$. Then we can calculate $v_0^* = 7$, $\boldsymbol{p}^0 = (9,3)^T$, $g_{\mathcal{U}}^0 = 3$, $u_0^* = -3$, $\boldsymbol{x}^1 = (9,0)^T$, $v_1^* = -9$ and $\boldsymbol{p}^1 = (0,0)^T$. Notice that after the first $\mathcal{V}$-step, $\boldsymbol{p}^0$ falls into the fast track $\left\{\chi(u; 0.5) : \chi(u; 0.5) = \left(\frac{a}{2}u^2,\ u\right), u \in \mathcal{U}\right\}$. As illustrated by Fig. 9.5, once in the fast track, only one $\mathcal{U}$-step is needed to find the $\mathcal{U}$-component of the minimizer, because the $\mathcal{U}$-Lagrangian is quadratic along the fast track.

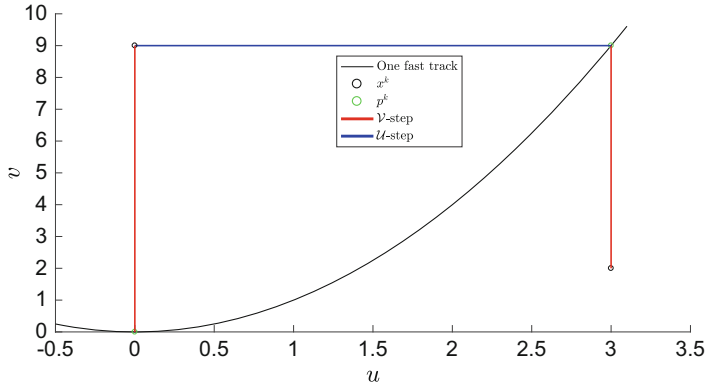

Fig. 9.5 Trajectory of the iterations of Algorithm 9.2 for f in Example 9.3 with $a = 2$ and $g_\gamma = 0.5$

The original algorithm in [13] requires the subgradient $\boldsymbol{g}$ to be $\mathbf{0}$. It turns out that this is not necessary, due to the property (9.21), any fixed $\boldsymbol{g} \in \operatorname{ri} \partial f(\overline{\boldsymbol{x}})$ can be used. This additional flexibility may have some importance when performing numerical results. The following theorem revises [13, Theorem 4.7] to take into account such modification.

Theorem 9.1 (Superlinear Convergence of Conceptual $\mathcal{VU}$-Scheme) *Under Assumption 9.2, the sequence $\boldsymbol{x}^k$ constructed by Algorithm 9.2 satisfies $\|\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\| = o\left(\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\|\right)$.*

Proof Denote $\boldsymbol{u}_k = \left(\boldsymbol{x}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{U}} = \left(\boldsymbol{p}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{U}}$. From line 3 of Algorithm 9.2 and (9.16), the next iterate $\mathcal{V}$-component satisfies the inclusion

$$\left(\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\right)_{\mathcal{V}} = \boldsymbol{v}_k^* + \left(\boldsymbol{x}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{V}} \in W\left(\boldsymbol{u}_k; \boldsymbol{g}_{\mathcal{V}}\right)$$

and (9.14) gives in the limit, $\lim_{k\to\infty} \frac{\|\left(\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\right)_{\mathcal{V}}\|}{\|\boldsymbol{u}_k\|_{\mathcal{U}}} = 0$. Since

$$0 \le \frac{\|\left(\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\right)_{\mathcal{V}}\|}{\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\|} \le \frac{\|\left(\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\right)_{\mathcal{V}}\|}{\|\boldsymbol{u}_k\|_{\mathcal{U}}} \to 0,$$

it holds that

$$\|\left(\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\right)_{\mathcal{V}}\|_{\mathcal{V}} = o\left(\|\boldsymbol{u}_k\|_{\mathcal{U}}\right) = o\left(\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\|\right). \tag{9.25}$$

Combining (9.17) and line 2 of Algorithm 9.2 gives (9.22), because the inclusion

$$\partial L_{\mathcal{U}}\left(\boldsymbol{u}_k; \boldsymbol{g}_{\mathcal{V}}\right) \ni \boldsymbol{g}_{\mathcal{U}}^k$$

holds. Replacing $\boldsymbol{u}$ in (9.24) with $\boldsymbol{u}_k^*$, yields that

$$\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}} f\left(\bar{\boldsymbol{x}}\right) \boldsymbol{u}_k^* = -\boldsymbol{g}_{\mathcal{U}}^k,$$

which, in view of (9.22), implies that

$$\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}} f\left(\bar{\boldsymbol{x}}\right) \left(\boldsymbol{u}_k + \boldsymbol{u}_k^*\right) = o\left(\|\boldsymbol{u}_k\|_{\mathcal{U}}\right).$$

The positive definiteness of $\mathrm{H}_{\mathcal{U}}^{\boldsymbol{g}_{\mathcal{V}}} f\left(\bar{\boldsymbol{x}}\right)$ yields $\|\boldsymbol{u}_k + \boldsymbol{u}_k^*\|_{\mathcal{U}} = o\left(\|\boldsymbol{u}_k\|_{\mathcal{U}}\right)$. By line 3 of Algorithm 9.2, the next iterate $\mathcal{U}$-component is $\left(\boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}\right)_{\mathcal{U}} = \boldsymbol{u}_k + \boldsymbol{u}_k^*$. Consequently, $\|\left(\boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}\right)_{\mathcal{U}}\|_{\mathcal{U}} = o\left(\|\boldsymbol{u}_k\|_{\mathcal{U}}\right) = o\left(\|\boldsymbol{x}^k - \bar{\boldsymbol{x}}\|\right)$. This together with (9.25) verifies the conclusion. □

The conceptual scheme shows a mechanism to obtain the desired superlinear convergence speed in nonsmooth optimization. The difficult question of how to pass from the conceptual algorithm to an implementable one is addressed in the next sections.

9.5 Towards Implementation

An obvious difficulty of implementing Algorithm 9.2 is in computing the required data $\mathcal{U}$, $\mathcal{V}$, and $\mathrm{H}_{\mathcal{U}} f\left(\bar{\boldsymbol{x}}\right)$ as they all depend on a minimum point $\bar{\boldsymbol{x}}$ that is unknown. In practice, those objects need to be properly approximated. For approximating the $\mathcal{U}$-Hessian, it is possible to use a quasi-Newton approach, which requires some second-order objects associated with the iterate $\boldsymbol{x}^k$. To be aligned with this approach, as we shall discuss later, the approximation of $\mathcal{U}$ and $\mathcal{V}$ also needs to be localized, that is, instead of $\mathcal{U}$ and $\mathcal{V}$, we need to use $\mathcal{U}\left(\boldsymbol{x}^k\right)$ and $\mathcal{V}\left(\boldsymbol{x}^k\right)$, the subspaces associated with $\boldsymbol{x}^k$.

Another major difficulty of implementing Algorithm 9.2 is the $\mathcal{V}$-step, that is, solving (9.23). In this section, we discuss in detail strategies for overcoming these difficulties.

9.5.1 Replacing the $\mathcal{V}$-Step by a Prox-Step

For implementation purposes, in Algorithm 9.2 the choice of a suitable element $\boldsymbol{g}$ is crucial.

- Regarding the $\mathcal{V}$-step, line 1 requires a solution $\boldsymbol{v}^* \in T\left(\boldsymbol{x}; \boldsymbol{g}_{\mathcal{V}}\right)$, or, equivalently, solving the optimality condition of (9.23), $\mathbf{0} \in P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{x} + \mathbf{0} \oplus \boldsymbol{v}^*\right)\right) - \boldsymbol{g}_{\mathcal{V}}$. Together with line 1 of Algorithm 9.2, this means that the output of the $\mathcal{V}$-step is a point $\boldsymbol{p}$ such that $\boldsymbol{g}_{\mathcal{V}} \in P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}\right)\right)$. A closer scrutiny suggests that the output

$\boldsymbol{p}$ is designed to fall on a trajectory approximating a fast track, namely

$$\left\{\overline{\boldsymbol{x}} + \boldsymbol{u}(\boldsymbol{x}) \oplus \boldsymbol{v}\left(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) : \boldsymbol{u}(\boldsymbol{x}) = (\boldsymbol{x} - \overline{\boldsymbol{x}})_{\mathcal{U}}\,, \boldsymbol{x} \to \overline{\boldsymbol{x}}\right\},$$

where $\boldsymbol{v}\left(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) = \boldsymbol{v}^* + (\boldsymbol{x} - \overline{\boldsymbol{x}})_{\mathcal{V}} \in W\left(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right)$ satisfies (9.14). To see this, apply (9.16) to the expression

$$\begin{aligned}\boldsymbol{p} &= \overline{\boldsymbol{x}} + (\boldsymbol{x} - \overline{\boldsymbol{x}})_{\mathcal{U}} \oplus (\boldsymbol{x} - \overline{\boldsymbol{x}})_{\mathcal{V}} + \boldsymbol{0} \oplus \boldsymbol{v}^* \\ &= \overline{\boldsymbol{x}} + (\boldsymbol{x} - \overline{\boldsymbol{x}})_{\mathcal{U}} \oplus (\boldsymbol{v}^* + (\boldsymbol{x} - \overline{\boldsymbol{x}})_{\mathcal{V}})\,.\end{aligned}$$

In the relations above, the fact that $\boldsymbol{v}\left(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) \in W\left(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right)$ satisfies $\boldsymbol{v}\left(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) = o\left(\|\boldsymbol{u}(\boldsymbol{x})\|\right)$ is fundamental (see Theorem 9.1 and (9.25)).

- Considering its usage in the $\mathcal{U}$-step, finding a proper $\boldsymbol{g}_{\mathcal{V}}$ is also of great significance. In [20, Theorem 4] it is shown that if a fast track leading to a minimizer $\overline{\boldsymbol{x}}$ of f exists, then proximal points of f at points close to $\overline{\boldsymbol{x}}$ fall on the fast track (see also [17, Theorem 5.2]). Under Assumption 9.2, one can take $\boldsymbol{g} = \boldsymbol{0}$. Even so, a point $\boldsymbol{p}$ such that $\boldsymbol{0} \in P_{\mathcal{V}}\left(\partial f(\boldsymbol{p})\right)$ is difficult to obtain.

To overcome these difficulties, the authors of the implementable $\mathcal{VU}$-algorithm [20] employed a special bundle subroutine that involves solving two quadratic programming (QP) subproblems in each iteration and approximates a pair of trajectories called *primal-dual tracks*. We will briefly discuss the approaches used in [20] in Sect. 9.7. Here we focus on strategies that do not require any bundle-type routines, by establishing an equivalence between the $\mathcal{V}$-step and the proximal point operator. The proposition below also establishes the connection with the fast track.

Proposition 9.2 (Proximal Points Are on the Fast Track) *Let $\overline{\boldsymbol{x}}$ be a minimizer of f. For the proximal point of f at an arbitrary $\boldsymbol{x}$ consider the $\mathcal{VU}$-components*

$$\boldsymbol{u}_\mu(\boldsymbol{x}) := \left(\boldsymbol{p}_\mu(\boldsymbol{x}) - \overline{\boldsymbol{x}}\right)_{\mathcal{U}} \quad \text{and} \quad \boldsymbol{v}_{\boldsymbol{p}_\mu}(\boldsymbol{x}) := \left(\boldsymbol{p}_\mu(\boldsymbol{x}) - \overline{\boldsymbol{x}}\right)_{\mathcal{V}}.$$

Let $\boldsymbol{g}$ be an arbitrary element in $\operatorname{ri}\partial f\left(\overline{\boldsymbol{x}}\right)$, *and $\boldsymbol{v}\left(\cdot; \boldsymbol{g}_{\mathcal{V}}\right) \in W\left(\cdot; \boldsymbol{g}_{\mathcal{V}}\right)$ be an arbitrary singleton selection from $\mathcal{U}$ to $\mathcal{V}$. The following statements are equivalent*

(i) $\boldsymbol{p}_\mu(\boldsymbol{x}) = \overline{\boldsymbol{x}} + \boldsymbol{u}_\mu(\boldsymbol{x}) \oplus \boldsymbol{v}\left(\boldsymbol{u}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right)$;
(ii) $\boldsymbol{g}_{\mathcal{V}} \in P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}_\mu(\boldsymbol{x})\right)\right)$; *and*
(iii) $\boldsymbol{0} \in T\left(\boldsymbol{p}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right)$.

Proof It is clear that $\boldsymbol{p}_\mu(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}_\mu(\boldsymbol{x}) \oplus \boldsymbol{v}\left(\boldsymbol{u}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right)$ if and only if $\boldsymbol{v}\left(\boldsymbol{u}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) = \boldsymbol{v}_{\boldsymbol{p}_\mu}(\boldsymbol{x})$. The equivalence between the three assertions result from applying (9.16) with $\boldsymbol{p}_\mu(\boldsymbol{x})$ replacing $\boldsymbol{x}$. Since

$$W\left(\boldsymbol{u}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) = T\left(\boldsymbol{p}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right) + \left\{\boldsymbol{v}_{\boldsymbol{p}_\mu}(\boldsymbol{x})\right\}$$

it suffices to show that $\boldsymbol{0} \in T\left(\boldsymbol{p}_\mu(\boldsymbol{x}); \boldsymbol{g}_{\mathcal{V}}\right)$. In turn, this is equivalent to $\boldsymbol{g}_{\mathcal{V}} \in P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}_\mu(\boldsymbol{x})\right)\right)$ because of the optimality condition of convex functions, and the result follows. □

In view of Proposition 9.2, let us consider the following scenario. Suppose an algorithm for minimizing f generates a sequence of iteration points $\boldsymbol{x}^k$ and proximal points $\boldsymbol{p}^k := \boldsymbol{p}_{\mu^k} f\left(\boldsymbol{x}^k\right)$ with the bounded parameter μ^k. If both $\boldsymbol{x}^k$ and $\boldsymbol{p}^k$ converge to $\bar{\boldsymbol{x}}$, then $\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right) \to \mathbf{0}$ (see Proposition 9.2 for the definition of $\boldsymbol{u}_{\mu^k}$) and (9.14) gives

$$\boldsymbol{v}\left(\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right); \boldsymbol{g}_{\mathcal{V}}\right) = o\left(\|\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right)\|\right) \tag{9.26}$$

for any fixed $\boldsymbol{g} \in \operatorname{ri} \partial f\left(\bar{\boldsymbol{x}}\right)$. Consequently, the proximal point $\boldsymbol{p}^k$ will fall on the following track:

$$\left\{\bar{\boldsymbol{x}} + \boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right) \oplus \boldsymbol{v}\left(\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right); \boldsymbol{g}_{\mathcal{V}}\right) : k \text{ is sufficiently large } \right\}$$

provided that $\boldsymbol{g}$ satisfies

$$\boldsymbol{g}_{\mathcal{V}} \in P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}^k\right)\right) \text{ for all sufficiently large } k. \tag{9.27}$$

However, finding such $\boldsymbol{g} \in \operatorname{ri} \partial f\left(\bar{\boldsymbol{x}}\right)$ can be very difficult and it may not exist at all. Consider the function in Example 9.1. We have shown that its $\mathcal{V} = \mathbb{R} \times \mathbf{0}$ and thus $P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{x}\right)\right) = \{1\}$, $\{-1\}$, or $[-1, 1]$. Let $k > N$ a large integer so that $\left\{\boldsymbol{p}^k\right\}_{k>N}$ is sufficiently close to the minimizer. The element $\boldsymbol{g}_{\mathcal{V}} \in P_{\mathcal{V}}\left(\operatorname{ri} \partial f(\bar{\boldsymbol{x}})\right) = (-1, 1)$ such that $\boldsymbol{g}_{\mathcal{V}} \in \cap_{k>N} P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}^k\right)\right)$ does not exist because $\cap_{k>N} P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}^k\right)\right) = \{1\}$, $\{-1\}$ or $\emptyset$.

In view of Assumption 9.2, a more sensible approach is to use a *subgradient sequence* $\{\boldsymbol{g}^k\}$ such that $\boldsymbol{g}^k_{\mathcal{V}} \to \mathbf{0}$. The definition of $\mathcal{V}$ implies that for any such sequence, $\boldsymbol{g}^k_{\mathcal{V}} \in P_{\mathcal{V}}\left(\partial f\left(\bar{\boldsymbol{x}}\right)\right)$ for all sufficiently large k, because $\mathbf{0} \in \operatorname{ri} \partial f\left(\bar{\boldsymbol{x}}\right)$. Additionally, for rapid convergence, the condition (9.27) should hold, that is, the $\mathcal{VU}$-algorithm should define subgradients such that

$$\boldsymbol{g}^k_{\mathcal{V}} \in P_{\mathcal{V}}\left(\partial f\left(\boldsymbol{p}^k\right)\right) \quad \text{with} \quad \boldsymbol{g}^k_{\mathcal{V}} \to \mathbf{0}\,.$$

At this point, it is useful to recall Remark 9.1: minimizing the Moreau-Yosida regularization is equivalent to minimizing the function. As the gradient of the former is $\mu(\boldsymbol{x} - \boldsymbol{p}_{\mu} f(\boldsymbol{x}))$, this means that a minimizer $\bar{\boldsymbol{x}}$ coincides with its proximal point. Since it also holds that $\mu^k\left(\boldsymbol{x}^k - \boldsymbol{p}^k\right) \in \partial f\left(\boldsymbol{p}^k\right)$, in order to satisfy the conditions above, a natural choice for the subgradients is to take

$$\boldsymbol{g}^k := \mu^k\left(\boldsymbol{x}^k - \boldsymbol{p}^k\right) \in \partial f\left(\boldsymbol{p}^k\right).$$

(In the next section, we'll see that this choice can be helpful in the implementation of $\mathcal{U}$-step too.) However, this subgradient choice prevents a direct application of (9.14), because of the varying $\boldsymbol{g}^k$. Instead, we assume that

$$\boldsymbol{v}\left(\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right); \boldsymbol{g}^k_{\mathcal{V}}\right) = o\left(\|\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right)\|\right). \tag{9.28}$$

Note that contrary to (9.26), the $\mathcal{V}$-element in (9.28) has a varying parameter $\boldsymbol{g}^k$. We can verify that the function f in Example 9.3 satisfies (9.28) because of (9.9). To see this write from (9.9), $\frac{\|\boldsymbol{v}(\boldsymbol{u};\gamma)\|}{\|\boldsymbol{u}\|} = \frac{a}{2}\operatorname{sign}(\gamma)\|\boldsymbol{u}\|$. For any sequence $\boldsymbol{u}_k \to \boldsymbol{0}$ with $\gamma^k \in (0,1)$, we have $\lim_{k\to\infty} \frac{\|\boldsymbol{v}(_k;\gamma^k)\|}{\|\boldsymbol{u}_k\|} = 0$.

In summary, we do not assume the existence of a fast track (as is the case in [20]). Instead, we require (9.28) to hold for

$$\boldsymbol{g}^k := \mu^k\left(\boldsymbol{x}^k - \boldsymbol{p}^k\right).$$

In this way, our $\mathcal{V}$-step will calculate the proximal point $\boldsymbol{p}^k$, which falls on the following track:

$$\left\{\overline{\boldsymbol{x}} + \boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right) \oplus \boldsymbol{v}\left(\boldsymbol{u}_{\mu^k}\left(\boldsymbol{x}^k\right); \boldsymbol{g}^k_{\mathcal{V}}\right) : k \text{ is sufficiently large}\right\}.$$

9.5.2 A Quasi-Newton $\mathcal{U}$-Step

After obtaining a proximal point $\boldsymbol{p}^k$ and a subgradient $\boldsymbol{g}^k = \mu^k\left(\boldsymbol{x}^k - \boldsymbol{p}^k\right)$, we continue from the $\mathcal{U}$-step in line 2 of Algorithm 9.2. Next, we see that in the $\mathcal{U}$-step of Algorithm 9.2 there is a specific requirement: $\boldsymbol{g}^k_{\mathcal{U}} \in \partial L_{\mathcal{U}}\left(\boldsymbol{u}_k; \boldsymbol{g}_{\mathcal{V}}\right)$ where $\boldsymbol{u}_k = \left(\boldsymbol{p}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{U}}$. This condition together with (9.21) is used to show (9.22), which holds for any fixed $\boldsymbol{g} \in \operatorname{ri}\partial f\left(\overline{\boldsymbol{x}}\right)$ as long as $\mathrm{H}^{\boldsymbol{g}_{\mathcal{V}}}_{\mathcal{U}} f\left(\overline{\boldsymbol{x}}\right)$ exists. In our case, we have chosen $\boldsymbol{p}^k$ and $\boldsymbol{g}^k$ such that they satisfy $\boldsymbol{g}^k \in \partial f\left(\boldsymbol{p}^k\right)$ and

$$\boldsymbol{g}^k_{\mathcal{U}} \in \partial L_{\mathcal{U}}\left(\boldsymbol{u}_k; \boldsymbol{g}^k_{\mathcal{V}}\right). \tag{9.29}$$

However, due to the varying $\boldsymbol{g}^k$, we may not be able to have the important condition (9.22) as it depends on a fixed $\boldsymbol{g}$. In order to use the property of the $\mathcal{U}$-Hessian in the convergence proof, we need to assume the following condition

$$\boldsymbol{g}^k_{\mathcal{U}} = \mathrm{H}^{\boldsymbol{g}^k_{\mathcal{V}}}_{\mathcal{U}} f\left(\overline{\boldsymbol{x}}\right)\boldsymbol{u}_k + o\left(\|\boldsymbol{u}_k\|_{\mathcal{U}}\right). \tag{9.30}$$

Compared with (9.22), the new assumption (9.30) requires the existence of $\mathcal{U}$-Hessians associated with $\boldsymbol{g}^k_{\mathcal{V}}$ for all k sufficiently large. Note that $\boldsymbol{g}^k_{\mathcal{V}}$ must be in $P_{\mathcal{V}}\left(\operatorname{ri}\partial f\left(\overline{\boldsymbol{x}}\right)\right)$ for k sufficiently large, which can be guaranteed as long as $\boldsymbol{g}^k \to \boldsymbol{0}$ and $\boldsymbol{0} \in \operatorname{ri}\partial f\left(\overline{\boldsymbol{x}}\right)$. One condition that yields (9.30) is

$$\partial L_{\mathcal{U}}\left(\boldsymbol{u}_k; \boldsymbol{g}^k_{\mathcal{V}}\right) \subset \mathrm{H}^{\boldsymbol{g}^k_{\mathcal{V}}}_{\mathcal{U}} f\left(\overline{\boldsymbol{x}}\right)\boldsymbol{u}_k + B\left(\boldsymbol{0}; o\left(\|\boldsymbol{u}_k\|_{\mathcal{U}}\right)\right).$$

Notice the similarity of this condition with (9.21).

For the function f in Example 9.3 and k sufficiently large, $\boldsymbol{g}^k_{\mathcal{V}}$ corresponds to some γ^k sufficiently close to 0. In view of (9.8) and (9.29),

$$\boldsymbol{g}^k_{\mathcal{U}} = \mu^k \left(\boldsymbol{x}^k - \boldsymbol{p}^k\right)_{\mathcal{U}} = \left(1 - |\gamma^k|\right) a\boldsymbol{u}_k.$$

Consequently, assumption (9.30) holds, since it amounts to the identity

$$\left(1 - |\gamma^k|\right) a\boldsymbol{u}_k = \begin{bmatrix} 0 & 0 \\ 0 & (1 - |\gamma|)\, a \end{bmatrix} \boldsymbol{u}_k.$$

Now, we can start constructing an implementable quasi-Newton step in our context. The subspaces $\mathcal{U}$ and $\mathcal{V}$ in general are difficult to compute because they are defined at $\bar{\boldsymbol{x}}$, a minimizer of f. To be aligned with the quasi-Newton approach and be able to use some localized second-order information of f, we need to use subspaces defined at iteration points $\boldsymbol{x}^k$, that is, $\mathcal{U}_k := \mathcal{U}\left(\boldsymbol{x}^k\right)$ and $\mathcal{V}_k := \mathcal{V}\left(\boldsymbol{x}^k\right)$. In this way, $\boldsymbol{g}^k_{\mathcal{U}}$ will not be used directly in the algorithm. Instead, it is approximated by $\boldsymbol{g}^k_{\mathcal{U}_k}$, which can be calculated by multiplying $\boldsymbol{g}^k$ on the left by the matrix U_k^T, whose columns approximate an orthonormal basis of $\mathcal{U}_k$. The dimension of U_k is $n \times n_k$ if the dimension of $\mathcal{U}_k$ is n_k.

To approximate the inverse $\mathcal{U}$-Hessian, we use a sequence of $n_k \times n_k$ symmetric positive semidefinite matrices Q_k. Consequently, the output of our quasi-Newton step will be a vector in $\mathbb{R}^{n_k}$, defined by

$$\boldsymbol{u}^*_k := -Q_k U_k^T \boldsymbol{g}^k. \tag{9.31}$$

Recall that the superlinear convergence of quasi-Newton methods for minimizing C^2 functions in general requires the Dennis-Moré condition:

$$\left(Q'_k - \nabla^2 f\left(\boldsymbol{x}^k\right)^{-1}\right) \nabla f\left(\boldsymbol{x}^k\right) = o\left(\|\nabla f\left(\boldsymbol{x}^k\right)\|\right), \tag{9.32}$$

where $Q'_k \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite. For the $\mathcal{U}$-Lagrangian, Eq. (9.32) is satisfied in a sense to be made precise below. We have $\boldsymbol{g}^k$ satisfying (9.29) and (9.30). In order to use the inverse of the $\mathcal{U}$-Hessian, we rewrite the $\mathcal{U}$-Lagrangian in the following form.

$$L_{\mathcal{U}}\left(\cdot; V^T \boldsymbol{g}^k\right) := \inf_{\boldsymbol{v}\in\mathbb{R}^{n-m}} \left\{ f\left(\bar{\boldsymbol{x}} + U\cdot + V\boldsymbol{v}\right) - \left\langle V^T \boldsymbol{g}^k, \boldsymbol{v}\right\rangle \right\}, \tag{9.33}$$

where U is an orthonormal basis matrix for $\mathcal{U}$, V is a basis matrix for $\mathcal{V}$, and m is the dimension of $\mathcal{U}$. Let $\nabla^2 L_{\mathcal{U}}\left(\mathbf{0}; V^T \boldsymbol{g}^k\right)$ be the Hessian of the function defined in (9.33) at $\mathbf{0}$. As we do not assume the existence of the Hessian at $\boldsymbol{u}_k$, the condition

corresponding to (9.32) becomes

$$\left[U_k Q_k U_k^T - U \nabla^2 L_{\mathcal{U}}\left(\mathbf{0}; V^T \boldsymbol{g}^k\right)^{-1} U^T\right] \boldsymbol{g}^k = o\left(\|U^T \boldsymbol{g}^k\|\right). \tag{9.34}$$

Note also that (9.34) requires the Hessian $\nabla^2 L_{\mathcal{U}}\left(\mathbf{0}; V^T \boldsymbol{g}^k\right)$ to be positive definite for all k sufficiently large. For Example 9.3 we have $V^T = [1\ 0]$ and $U^T = [0\ 1]$. For k sufficiently large, $V^T \boldsymbol{g}^k$ corresponds to some γ^k sufficiently close to 0 and $\nabla^2 L_{\mathcal{U}}\left(\mathbf{0}; V^T \boldsymbol{g}^k\right) = \left(1 - |\gamma^k| a\right)$ is positive definite.

In summary, the $\mathcal{U}$-step generates a direction $\boldsymbol{u}_k^*$ as in (9.31) and requires (9.30) and (9.34). In practice, the matrices Q_k are updated using quasi-Newton formulas, making sure that dimensions of U_k and Q_k are consistent (see Remark 9.5). A variety of methods can be used to calculate U_k, some of which are discussed in Sect. 9.7.

9.6 An Implementable Algorithm with Exact Prox-Calculation

The previous section primarily laid the background for implementing the original $\mathcal{V}$-step with a prox-step and the original $\mathcal{U}$-step with a quasi-Newton $\mathcal{U}$-step. In the machine learning literature there is no shortage of functions whose proximal points can be computed exactly. In addition, nowadays various techniques can be applied to the quasi-Newton methods to reduce their computational cost. The resulting *prox-quasi-Newton $\mathcal{VU}$-algorithm* is given below.

Algorithm 9.3: The prox-quasi-Newton $\mathcal{VU}$-algorithm

Data: starting point $\boldsymbol{x}^0$, a tolerance ϵ for termination, exact computation of the prox operator for f with rules of updating parameter μ, a sequence of $n \times n_k$ matrices U_k approximating the basis of the subspace $\mathcal{U}_k$ with dimension n_k, and a sequence of $n_k \times n_k$ symmetric positive semidefinite matrices Q_k.

repeat

  $\mathcal{V}$-Step. Given $\boldsymbol{x}^k$, compute $\boldsymbol{p}^k$ and $\boldsymbol{g}^k$.

  $\mathcal{U}$-Step. Make a quasi-Newton step in $\boldsymbol{p}^k + \mathcal{U}_k$ approximately: compute the direction

$$\boldsymbol{u}_k^* := -Q_k U_k^T \boldsymbol{g}^k .$$

  Update. Set $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{p}^k + U_k u_k^*$ and set $k \leftarrow k + 1$.

until $\|\boldsymbol{g}^k\| \le \epsilon$;

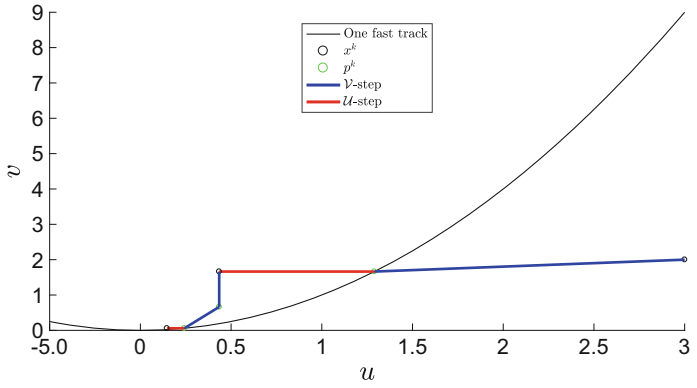

Fig. 9.6 Trajectory of the first 3 iterations of Algorithm 9.3 for function f in Example 9.3 with $a = 2$, $\mu = 1$ and $\boldsymbol{x}^0 = (2, 3)^T$

Figure 9.6 shows the trajectory of the first 3 iterations of Algorithm 9.3 for the function f in Example 9.3. Notice that the first and third $\mathcal{V}$-steps produce proximal points on the fast track. The second iteration skips $\mathcal{U}$-step because $f(\boldsymbol{x}^1) = f^1(x_1^1) > f^2(x_2^1)$ resulting a first null matrix, $Q_1 = 0$, while Q_0 and Q_2 are set to $\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{a} \end{bmatrix}$.

Remark 9.4 One iteration of Algorithm 9.3 can be shortened as

$$\boldsymbol{x}^{k+1} = \boldsymbol{p}^k + \mu^k U_k Q_k U_k^T \left(\boldsymbol{p}^k - \boldsymbol{x}^k \right), \tag{9.35}$$

a writing that puts in evidence the correction to the proximal step, incorporated to speed up the algorithm.

Under some conditions, Algorithm 9.3 has global convergence in the sense that every accumulation point the iteration sequence $\boldsymbol{x}^k$ is a minimizer of f. In addition, with the Dennis-Moré-type condition (9.34) the superlinear rate of convergence can be shown assuming convergence of the whole sequence of $\boldsymbol{x}^k$ to a minimizer of f. Here, we only give the main ideas of the convergence proof.

Consider the $n \times n$ matrix $\hat{U}_k$ whose first n_k columns are U_k and whose elements in other positions are all zero. The boundedness and convergence of U_k can be defined as that of $\hat{U}_k$ via the Frobenius norm. The boundedness and convergence of Q_k can be defined in a similar way. The convergence of Algorithm 9.3 requires the boundedness of U_k and Q_k. Additionally, the parameter sequence $\{\mu^k\}$ needs to be bounded (in agreement Assumption 9.1, condition (i), needed for R-linear convergence, given in Sect. 9.2).

Assumption 9.3 (Bounded Prox-Parameter) *The parameter μ^k in the prox-operator satisfies*

$$\mu_{\max} \geq \mu^k \geq \mu_{\min} > 0, \ \forall\, k.$$

We have the following convergence theorem for Algorithm 9.3.

Theorem 9.2 (Global Convergence of Prox-Quasi-Newton $\mathcal{VU}$-Scheme) *Let f be convex. Suppose the matrices U_k and Q_k used in Algorithm 9.3 are bounded. If Assumption 9.3 holds then every accumulation point of the sequence $\{\boldsymbol{x}^k\}$ generated by Algorithm 9.3 is a minimizer of f.*

To show this, take a convergent subsequence of $\boldsymbol{x}^k$. Due to the boundedness of the objects in (9.35), there exists a common subsequence such that the limit of (9.35) holds as $\boldsymbol{x}' = \boldsymbol{p}' + \mu' U' Q U'^T \left(\boldsymbol{p}' - \boldsymbol{x}'\right)$ where the convergence of $\boldsymbol{p}^k$ follows from [26, Theorem 2.26] due to the boundedness of μ_k. Then we can use the positive definiteness of the matrix $I + \mu' U' Q U'^T$ to derive that $\boldsymbol{p}' = \boldsymbol{x}'$, validating the theorem.

The superlinear rate of convergence can be shown under the two extra conditions discussed in Sect. 9.5, Eqs. (9.28) and (9.30).

Theorem 9.3 (Superlinear Convergence of Prox-Quasi-Newton $\mathcal{VU}$-Scheme) *Suppose that the sequence $\{\boldsymbol{x}^k\}$ generated by Algorithm 9.3 converges to $\overline{\boldsymbol{x}}$, $\mathbf{0} \in \operatorname{ri} \partial f\left(\overline{\boldsymbol{x}}\right)$ and Assumption If conditions (9.28), (9.30) and (9.34) are satisfied, then*

$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\|}{\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\|} = 0\,.$$

The proof can be divided into three parts. First, show that Proposition 9.2 can be applied on $\boldsymbol{p}^k$, $\boldsymbol{x}^k$ and $\boldsymbol{g}^k$ for k sufficiently large. This together with (9.28) yields

$$\lim_{k\to\infty} \frac{\|\left(\boldsymbol{p}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{V}}\|}{\|\left(\boldsymbol{p}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{U}}\|} = 0\,.$$

Second, define $\boldsymbol{u}_k = U^T \left(\boldsymbol{p}^k - \overline{\boldsymbol{x}}\right)$ and multiply (9.30) by U^T to obtain

$$U^T \boldsymbol{g}^k_{\mathcal{U}} = \nabla^2 L_{\mathcal{U}} \left(\mathbf{0}; \boldsymbol{g}^k_{\mathcal{V}}\right) \boldsymbol{u}_k + U^T o\left(\|U u_k\|\right). \tag{9.36}$$

Combine (9.36) and (9.34) to get

$$\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}} = \boldsymbol{x}^{k+1} - \boldsymbol{p}^k + \boldsymbol{p}^k - \overline{\boldsymbol{x}} = o\left(\|\boldsymbol{g}^k_{\mathcal{U}}\|\right) + o\left(\|U u_k\|\right). \tag{9.37}$$

Finally, use the properties of the proximal mapping that $\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\| \geq \|\boldsymbol{x}^k - \boldsymbol{p}^k\| \geq \|\left(\boldsymbol{x}^k - \boldsymbol{p}^k\right)_{\mathcal{U}}\|$ and $\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\| \geq \|\boldsymbol{p}^k - \overline{\boldsymbol{x}}\| \geq \|\left(\boldsymbol{p}^k - \overline{\boldsymbol{x}}\right)_{\mathcal{U}}\|$ and show together with (9.37) that the quotient $\frac{\|\boldsymbol{x}^{k+1} - \overline{\boldsymbol{x}}\|}{\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\|}$ goes to 0.

Remark 9.5 Algorithm 9.3 relies on two sequences of matrices: $\{U_k\}$ and $\{Q_k\}$. Although theoretically U_k is meant to approximate a basis of $\mathcal{U}_k$ and Q_k some second-order information of the Lagrangian, the convergence of Algorithm 9.3 only requires those objects to satisfy two conditions: first, they have to be bounded;

second, they have to satisfy the Dennis-Moré condition (9.34). Consequently, there is plenty of flexibility in their constructions. To have a $\mathcal{U}$-quasi-Newton method, one can choose Q_{k+1} in a way that it is close to Q_k and satisfies the secant equation (9.38)

$$Q_{k+1}U_{k+1}^T\left(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k\right)=U_{k+1}^T\left(\boldsymbol{g}^{k+1}-\boldsymbol{g}^k\right). \tag{9.38}$$

Note that the underlying function of the $\mathcal{U}$-quasi-Newton method is the $\mathcal{U}$-lagrangian, $L_{\mathcal{U}}$ (not f) and the associated point is $\boldsymbol{u}_k$ (not $\boldsymbol{x}^k$).

We note that in our context the Dennis-Moré condition (9.34) is different from its original form. First, Q_k and $\nabla^2 L_{\mathcal{U}}\left(\mathbf{0};\boldsymbol{g}_{\mathcal{V}}^k\right)^{-1}$ have to be accompanied by the associated basis matrices. Second, each Q_k not only corresponds to a point $\boldsymbol{u}_k$ (cf. (9.36)) but also the underlying $\mathcal{U}_k$-Lagrangian.

Finally, compared with the convergence proof in [20], here the existence of a primal-dual track is no longer needed. The readers can verify that the function f in Example 9.3 satisfy conditions (9.28) and (9.30).

9.7 Practical Information

In this section, we discuss some practical aspects of implementing $\mathcal{VU}$ algorithms including constructions of the matrix U_k and dealing with functions whose proximal points are difficult to compute, thus requiring the bundling mechanism to be incorporated in the process.

9.7.1 Calculating U_k

For some functions, a basis of $\mathcal{U}_k$ can be computed exactly, although U_k does not need to be a basis. Generally, a basis matrix for the subspace $\mathcal{U}_k$ can be constructed this way: find the maximum number of linearly independent vectors in $\partial f\left(\boldsymbol{x}^k\right)-\boldsymbol{s}^k$, for any $\boldsymbol{s}^k\in\partial f\left(\boldsymbol{x}^k\right)$ and take the columns of the matrix

$$V_k,\ \text{as a basis for } \mathcal{V}_k.$$

The columns of the matrix U_k form an orthonormal basis for the null-space of V_k. Such operations can be carried out if the full subdifferential is known, as it is the case for the *PDG structured* functions [18], a generalization of max-functions that includes maximum eigenvalue functions; see also [5].

Example 9.4 Consider a function of the form $h(\boldsymbol{x}) = q(\boldsymbol{x}) + f(\boldsymbol{x})$, where q is smooth and f is the L_1-norm. The definition of the $\mathcal{VU}$-decomposition reveals that $\mathcal{V}_h(\boldsymbol{x}) = \mathcal{V}(\boldsymbol{x})$ and $\mathcal{U}_h(\boldsymbol{x}) = \mathcal{U}(\boldsymbol{x})$, where $\mathcal{V}_h(\boldsymbol{x})$ and $\mathcal{U}_h(\boldsymbol{x})$ are the subspaces associated with the function h.

This means that the canonical basis $\left\{\boldsymbol{e}^j : x_j \neq 0\right\}$, where $\boldsymbol{e}^j$ is a vector in $\mathbb{R}^n$ whose j-th component is 1 and other components are 0, can be used for minimizing an arbitrary nonsmooth function in the form of h. We now explain how to construct the bases.

Consider the L_1-norm $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ in $\mathbb{R}^n$. It can be shown that $\partial f(\boldsymbol{x}) = \left\{\boldsymbol{s} \in \mathbb{R}^n : |s_1| \leq 1,\ \sum_{i=1}^n s_i x_i = \sum_{i=1}^n |x_i|\right\} = \text{conv} S(\boldsymbol{x})$, where

$$\begin{aligned} S(\boldsymbol{x}) :&= \left\{\boldsymbol{s} \in \mathbb{R}^n : s_i = \pm 1,\ \sum (s_i - \text{sign}(x_i))\, x_i = 0\right\} \\ &= \left\{\boldsymbol{s} \in \mathbb{R}^n : s_i = \pm 1,\ s_i = \text{sign}(x_i),\ \text{if } x_i \neq 0\right\}. \end{aligned}$$

Take $\boldsymbol{s}^x \in \mathbb{R}^n$ to be such that $s_i^x = 1$, if $x_i = 0$ and otherwise $s_i^x = \text{sign}(x_i)$. Then $\boldsymbol{s}^x \in \partial f(\boldsymbol{x})$ and $\mathcal{V}(\boldsymbol{x}) = \text{span}\,(S(\boldsymbol{x}) - \boldsymbol{s}^x)$. For each j such that $x_j = 0$, define

$$\boldsymbol{s}^j \in \mathbb{R}^n \text{ be such that } s_i^j = \begin{cases} -1, & \text{if } i = j, \\ 1, & \text{if } i \neq j \text{ and } x_i = 0, \\ \text{sign}(x_i), & \text{otherwise.} \end{cases}$$

We can verify that $\boldsymbol{s}^j \in S(\boldsymbol{x})$ and $\boldsymbol{s}^j - \boldsymbol{s}^x = -2\boldsymbol{e}^j$.

Define $B(\boldsymbol{x}) := \left\{\boldsymbol{s}^j - \boldsymbol{s}^x : x_j = 0\right\}$ then we have $B(\boldsymbol{x}) \subset S_\varepsilon(\boldsymbol{x}) - \boldsymbol{s}^x$. It follows that the vectors in $B(\boldsymbol{x})$ are linearly independent. Now we show that $B(\boldsymbol{x})$ is a basis matrix of $\mathcal{V}(\boldsymbol{x})$ by verifying that each vector in $S(\boldsymbol{x}) - \boldsymbol{s}^x$ is a linear combination of $B(\boldsymbol{x})$. Take arbitrary $\boldsymbol{s} \in S(\boldsymbol{x})$ then the definitions of $\boldsymbol{s}^x$ and $S(\boldsymbol{x})$ yield that $(\boldsymbol{s} - \boldsymbol{s}^x)_i = -2$, if $x_i = 0$ and $s_i = -1$; otherwise, 0. The vector $\boldsymbol{s} - \boldsymbol{s}^x$ can have more than one non-null element, being -2. From the definition of $\boldsymbol{s}^j$ we see that such vector can be a linear combination of vectors in $B(\boldsymbol{x})$. Consequently, respective bases for $\mathcal{V}(\boldsymbol{x})$ and $\mathcal{U}(\boldsymbol{x})$ are

$$-\frac{1}{2} B(\boldsymbol{x}) = \left\{\boldsymbol{e}^j : x_j = 0\right\} \quad \text{and} \quad \left\{\boldsymbol{e}^j : x_j \neq 0\right\}.$$

Some other functions of which the associated basis for $\mathcal{U}$ is easy to calculate include functions as in Example 9.1, that are partly smooth at a point $\overline{\boldsymbol{x}}$ relative to a manifold $\mathcal{M}$ which is equal to $\mathcal{U}$ itself. More precisely, by the second property in the definition of partial smoothness given in Remark 9.3, when $\mathcal{M}$ itself is a linear subspace, the basis of $\mathcal{U}$ can be computed by exploiting the structure of the

manifold. A list of functions that satisfy $\mathcal{M} = \mathcal{U}$ can be found in [28, Section 3.1]. For example, the L_∞-norm is partly smooth at any point $\bar{\boldsymbol{x}}$ relative to the subspace

$$\mathcal{M} := \{\boldsymbol{x} : \boldsymbol{x}_I = k\,\mathrm{sign}(\bar{\boldsymbol{x}}_I),\ k \in \mathbb{R}\}\,,\ \text{where } I = \{i : |\bar{x}_i| = \|\bar{\boldsymbol{x}}\|_\infty\}\,.$$

With this information, it is not difficult to deduce that a basis of $\mathcal{U}$ can just be $\left\{\boldsymbol{e}^j : |\bar{x}_j| < \|\bar{\boldsymbol{x}}\|_\infty\right\} \cup \bar{\boldsymbol{e}}$, where $\boldsymbol{e}^j$ has the same definition as in Example 9.4 and $\bar{\boldsymbol{e}}$ is a vector such that $\bar{\boldsymbol{e}}_i = \mathrm{sign}\,(\bar{x}_i)$ if $i \in I$ and 0 otherwise.

To conclude this section, we discuss how to incorporate a bundle routine in the prox-quasi-Newton $\mathcal{VU}$-algorithm 9.3, so that it can be employed without computing exactly the proximal point operator at each iteration.

9.7.2 Incorporating a Bundle Routine

Since computing exact proximal points is not possible except for specific functions, developing an inexact prox-step in Algorithm 9.3 is desirable. A mechanism suitable for this purpose can be drawn from the proximal bundle method sketched in Sect. 9.2. In Algorithm 9.1 therein, successive null steps ending in a serious step are interpreted as improving the inexact proximal point calculation until a sufficient level of accuracy is achieved, measured in terms of the descent step 5.

The authors of [20] introduced a primal-dual trajectory called *primal-dual track*. A primal track is a fast track such that its Jacobian is a basis matrix for the $\mathcal{U}$-subspace while a dual track is the collection of the minimum norm subgradient of f at points in the primal track. The idea of introducing a dual track stems from a strategy different from the one presented in this text. Rather than varying the choice of the parameter $\boldsymbol{g}_\mathcal{V}$ in $L_\mathcal{U}$ as in Sect. 9.5.1, the subgradient remains fixed at $\mathbf{0}$, so that the original $\mathcal{V}$-step produces a point $\boldsymbol{p}$ such that $\mathbf{0} \in P_\mathcal{V}\,(\partial f(\boldsymbol{p}))$. Then the $\mathcal{U}$-step would need a subgradient $\boldsymbol{g}' \in \partial f(\boldsymbol{p})$ with null $\mathcal{V}$-component: $\boldsymbol{g}'_\mathcal{V} = \boldsymbol{g}_\mathcal{V} = \mathbf{0}$. The natural candidate to approximate such subgradient is the minimum norm element of the proximal point subdifferential, that is, an element of the dual track.

The algorithm developed in [20] utilizes a proximal bundle routine composed by two QPs. A similar feature is present in the second-order bundle method to minimize the maximum eigenvalue functions introduced in [23].

Solving two successive quadratic programming problems per iteration is the price to pay to achieve superlinear convergence. As in the proximal bundle methods, the first QP problem computes the proximal point of a cutting-plane model. Specifically, given a model φ, and a prox-center $\boldsymbol{x}$, the associated proximal problem is

$$\min_{\boldsymbol{p}\in\mathbb{R}^n} \varphi(\boldsymbol{p}) + \frac{\mu}{2}\|\boldsymbol{p} - \boldsymbol{x}\|^2. \tag{9.39}$$

The model φ can be expressed by

$$\begin{aligned}\varphi(\boldsymbol{p}) &= \max_{i\in I}\left\{f(\boldsymbol{y}_i)+\langle \boldsymbol{s}_i,\, \boldsymbol{p}-\boldsymbol{y}_i\rangle\right\}\\ &= f(\boldsymbol{x})+\max_{i\in I}\left\{-e_i+\langle \boldsymbol{s}_i,\, \boldsymbol{p}-\boldsymbol{x}\rangle\right\},\end{aligned}$$

where I is some index set containing an index j such that $\boldsymbol{y}_j = \boldsymbol{x}$ and the linearization error is defined as $e_i := e\left(\boldsymbol{x}, \boldsymbol{y}_i\right) := f(\boldsymbol{x}) - f(\boldsymbol{y}_i) - \langle \boldsymbol{s}_i, \boldsymbol{x}-\boldsymbol{y}_i\rangle$ for $i \in I$. The problem (9.39) can be solved equivalently through the following quadratic programming subproblem

$$\min\left\{r+\frac{\mu}{2}\|\boldsymbol{p}-\boldsymbol{x}\|^2 :\ (r,\boldsymbol{p})\in\mathbb{R}^{n+1},\ r\geq f(\boldsymbol{x})-e_i+\langle \boldsymbol{s}_i,\, \boldsymbol{p}-\boldsymbol{x}\rangle,\ i\in I\right\}.$$

Its (unique) optimal solution $\left(\hat{r}, \hat{\boldsymbol{p}}\right)$ satisfies $\hat{r} = \varphi(\hat{\boldsymbol{p}})$. Here $\hat{\boldsymbol{p}} = \boldsymbol{p}_\mu\varphi(\boldsymbol{x})$ approximates $\boldsymbol{p}_\mu f(\boldsymbol{x})$. Afterwards, at $\hat{\boldsymbol{p}}$ a new cutting-plane is generated and indexed by i_+, that is, $\boldsymbol{y}_{i_+} := \hat{\boldsymbol{p}}$ with $\boldsymbol{s}_{i_+} \in \partial f(\hat{\boldsymbol{p}})$.

The second QP problem has the following form

$$\min\left\{r+\frac{1}{2}\|\boldsymbol{p}-\boldsymbol{x}\|^2 :\ (r,\boldsymbol{p})\in\mathbb{R}^{n+1},\ r\geq\langle \boldsymbol{s}_i,\, \boldsymbol{p}-\boldsymbol{x}\rangle,\ i\in\hat{I}\right\}, \tag{9.40}$$

where $\hat{I} := \left\{i\in I : \hat{r} = f(\boldsymbol{x}) - e_i + \left\langle \boldsymbol{s}_i, \hat{\boldsymbol{p}}-\boldsymbol{x}\right\rangle\right\} \cup \{i_+\}$. Its dual problem is

$$\min\left\{\frac{1}{2}\|\sum_{i\in\hat{I}}\alpha_i\boldsymbol{s}_i\|^2 : \alpha\geq 0,\ \sum_{i\in\hat{I}}\alpha_i = 1\right\}, \tag{9.41}$$

hence showing that the second QP solution provides an element with minimum norm among all the available subgradients.

The respective solutions of (9.40) and (9.41) are $(\overline{r}, \overline{\boldsymbol{p}})$ and $\overline{\alpha}$, satisfying

$$\overline{\boldsymbol{p}} - \boldsymbol{x} = -\hat{\boldsymbol{s}},\ \text{where } \hat{\boldsymbol{s}} := \sum_{i\in\hat{I}}\overline{\alpha_i}\boldsymbol{s}_i. \tag{9.42}$$

It can be shown [20, Lemma 5] that $\{\boldsymbol{s}_i\}_{i\in\hat{I}} \subset \partial_{\hat{\varepsilon}} f(\hat{\boldsymbol{p}})$, where

$$\hat{\varepsilon} = f(\hat{\boldsymbol{p}}) - \hat{r}.$$

The second QP produces $\hat{\boldsymbol{s}}$, the minimum norm element of $\operatorname{conv}\{\boldsymbol{s}_i\}_{i\in\hat{I}}$. As $\hat{\boldsymbol{p}}$ approximates $\boldsymbol{p}_\mu f(\boldsymbol{x})$, $\hat{\boldsymbol{s}}$ approximates the minimum norm element in $\partial f\left(\boldsymbol{p}_\mu f(\boldsymbol{x})\right)$. If there exists a primal-dual track, then $\boldsymbol{p}_\mu f(\boldsymbol{x})$ is on the primal track and hence, $\hat{\boldsymbol{s}}$ approximates a point in the dual track.

A very important by-product of this bundle routine is a basis matrix $\left[\hat{U}\ \hat{V}\right]$ for $\mathbb{R}^n$ such that $\hat{U}$ has orthonormal columns and $\hat{V}^T\hat{s} = \mathbf{0}$. (Recall that in [20], the parameter $\boldsymbol{g}_{\mathcal{V}}$ in the $\mathcal{U}$-Lagrangian is fixed to be $\mathbf{0}$.) The construction is done by defining the active index set of (9.41) at its optimal solution, that is, $\overline{I} := \left\{i \in \hat{I} : \bar{r} = \langle \boldsymbol{s}_i, \overline{\boldsymbol{p}} - \boldsymbol{x}\rangle\right\}$. Then it follows from (9.42) that $\bar{r} = -\langle \boldsymbol{s}_i, \hat{\boldsymbol{s}}\rangle$ for all $i \in \overline{I}$ and thus

$$\langle \boldsymbol{s}_i - \boldsymbol{s}_l, \hat{\boldsymbol{s}}\rangle = 0 \tag{9.43}$$

for all such i and for a fixed $l \in \overline{I}$. Collect the largest number of vectors $\boldsymbol{s}_i - \hat{\boldsymbol{s}}$ satisfying (9.43) such that they are linearly independent and let them form the columns of the matrix $\hat{V}$. Then $\hat{U}$ can be defined to be a matrix whose columns form an orthonormal basis for the null-space of $\hat{V}^T$ with $\hat{U}$ being the identity matrix if $\hat{V}$ is vacuous.

For the function in Example 9.2 and in a semilogarithmic scale, Fig. 9.7 shows the functional error at serious iterates computed with the inexact prox-quasi-Newton $\mathcal{VU}$-algorithm in [20] in red. The blue line corresponds to the output of N1CV2 an implementation of the proximal bundle method [12].

In addition to displaying graphically the superlinear rate of convergence of the $\mathcal{VU}$-algorithm, the figure also corroborates the degree of accuracy that can be achieved using a Newtonian method. For the example, the $\mathcal{VU}$-algorithm gives 10 digits, against 5 obtained by N1CV2. If high precision is a requirement for the nonsmooth problem, the extra computational work of solving two QPs instead of one can eventually pay off.

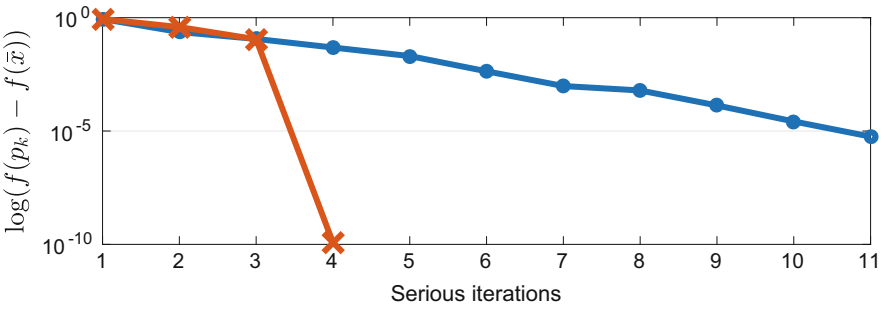

Fig. 9.7 Superlinear (red crosses) and linear (blue dots) convergence rate of the $\mathcal{VU}$ and bundle methods for the function in Example 9.2

9.8 Further Notes

For all the algorithms presented so far, the $\mathcal{U}$-step requires a matrix U_k approximating a basis of $\mathcal{U}_k$, a subspace associated with the iteration point $\boldsymbol{x}^k$. One may ask how precise this approximation has to be. Notice that the superlinear convergence needs the Dennis-Moré condition (9.34), which the sequence U_k has to satisfy. Actually, a proper construction of U_k helps the convergence and, in view of (9.34), it is desirable that

$$\lim_{k\to\infty} U_k Q_k U_k = U\nabla^2 L_{\mathcal{U}}\left(\mathbf{0}; V^T \boldsymbol{g}^k\right)^{-1} U^T,$$

where U is an orthonormal basis matrix for $\mathcal{U}$. Although this condition does not necessarily require that U_k converges to U (when considered as equivalent $n \times n$ matrices), it will be interesting to find some functions that can satisfy such condition. Essentially this boils down to the question of *continuity* of the underlying subspaces associated with U_k, as approximation of $\mathcal{U}_k$. The $\mathcal{VU}$-subspace definition is based on the subdifferential, which is not continuous as a set-valued function. By contrast, the well-known epsilon-subdifferential in convex analysis, enlarging the subdifferential, is continuous. This remark poses the question of studying the continuity of subspaces defined by enlargements. This point is considered in [15], a work where certain subspaces $\mathcal{V}_\varepsilon$ and $\mathcal{U}_\varepsilon$, among which those generated using $\partial_\varepsilon f(\bar{\boldsymbol{x}})$, are studied. The corresponding approximations of $\mathcal{V}$ and $\mathcal{U}$ are shown to be continuous set-valued functions for several types of functions, including finite-max functions.

So far the $\mathcal{VU}$-algorithms are designed for convex functions only. Generalizations to the nonconvex case can be made for functions with sufficient structure, for example the strongly transversal PDG-structure in [19]; see also [18]. All of the $\mathcal{VU}$-objects in the convex context can be constructed for these types of nonconvex functions. Additionally, a primal-dual track exists for these functions and proximal points are indeed on the primal track provided that the functions are prox-regular. A suitable bundling mechanism, like the one in [7], could then be employed to approximate the proximal point and, therefore, make implementable the $\mathcal{V}$-step; see also [8].

Acknowledgements The work of Shuai Liu was financially supported by FAPESP grant 2017/15936-2 and the work of Claudia Sagastizábal was partially supported by CNPq Grant 303905/2015-8 and FAPERJ Grant 203.052/2016.

References

1. Auslender, A.: Numerical methods for nondifferentiable convex optimization Math. Program. Study **30**, 102–126 (1987)
2. Bonnans, J.F., Gilbert, J.C., Lemaréchal, C., Sagastizábal, C.A.: A family of variable metric proximal methods Math. Program. **68**(1–3), 15–47 (1995)

3. Chen, X., Fukushima, M.: Proximal quasi-Newton methods for nondifferentiable convex optimization Math. Program. **85**(2, Ser. A), 313–334 (1999)
4. Correa, R., Lemaréchal, C.: Convergence of some algorithms for convex minimization Math. Program. **62**(1–3), 261–275 (1993)
5. Daniilidis, A., Sagastizábal, C., Solodov, M.: Identifying structure of nonsmooth convex functions by the bundle technique SIAM J. Optim. **20**(2), 820–840 (2009). https://doi.org/10.1137/080729864
6. Fukushima, M.: A descent algorithm for nonsmooth convex optimization Math. Program. **30**, 163–175 (1984)
7. Hare, W., Sagastizábal, C.: Computing proximal points of nonconvex functions Math. Program. **116**(1), 221–258 (2009). https://doi.org/10.1007/s10107-007-0124-6
8. Hare, W., Sagastizábal, C.: A redistributed proximal bundle method for nonconvex optimization SIAM J. Optim. **20**(5), 2442–2473 (2010)
9. Lemaréchal, C., Mifflin, R.: Global and superlinear convergence of an algorithm for one-dimensional minimization of convex functions Math. Program. Study **24**, 241–256 (1982)
10. Lemaréchal, C., Sagastizábal, C.: An approach to variable metric bundle methods. In: Henry, J., Yvon, J.P. (eds.) System Modelling and Optimization. Lecture Notes in Control and Information Sciences, vol. 197, pp. 144–162. Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/BFb0035464
11. Lemaréchal, C., Sagastizábal, C.: Practical aspects of the Moreau-Yosida regularization: theoretical preliminaries. SIAM J. Optim. **7**(2), 367–385 (1997). https://doi.org/10.1137/S1052623494267127
12. Lemaréchal, C., Sagastizábal, C.: Variable metric bundle methods: from conceptual to implementable forms. Math. Program. **76**(3), 393–410 (1997). https://doi.org/10.1007/BF02614390
13. Lemaréchal, C., Oustry, F., Sagastizábal, C.: The U-Lagrangian of a convex function. Trans. Am. Math. Soc. **352**(2), 711–729 (2000)
14. Lewis, A.S.: Active sets, nonsmoothness, and sensitivity. SIAM J. Optim. **13**(3), 702–725 (2002). https://doi.org/10.1137/s1052623401387623
15. Liu, S., Sagastizábal, C., Solodov, M.: Subdifferential Enlargements and Continuity Properties of the VU-decomposition in Convex Optimization. International Series of Numerical Mathematics, vol. 170. Birkhäuser/Springer, Cham (2018)
16. Mifflin, R.: A quasi-second-order proximal bundle algorithm. Math. Program. **73**(1), 51–72 (1996)
17. Mifflin, R., Sagastizábal, C.: Proximal points are on the fast track J. Convex Anal. **9**(2), 563–580 (2002)
18. Mifflin, R., Sagastizábal, C.: Primal-dual gradient structured functions: second-order results; links to epi-derivatives and partly smooth functions. SIAM J. Optim. **13**(4), 1174–1194 (2003)
19. Mifflin, R., Sagastizábal, C.: UV-smoothness and proximal point results for some nonconvex functions. Optim. Method Softw. **19**(5), 463–478 (2004). https://doi.org/10.1080/10556780410001704902
20. Mifflin, R., Sagastizábal, C.: A VU-algorithm for convex minimization Math. Program. **104**(2–3), 583–608 (2005)
21. Mifflin, R., Sagastizábal, C.: Optimization stories. In: Grötschel, M. (ed.) Extra Volume ISMP 2012, pp. 460 (2012)
22. Mifflin, R., Sun, D., Qi, L.: Quasi-Newton bundle-type methods for nondifferentiable convex optimization. SIAM J. Optim. **8**(2), 583–603 (1998). https://doi.org/10.1137/S1052623496303329
23. Oustry, F.: A second-order bundle method to minimize the maximum eigenvalue function Math. Program. **89**(1, Ser. B), 1–33 (2001). https://doi.org/10.1007/s101070000166. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0002928322&partnerID=40&md5=4ff600667e238f59008a3320f8e82dae
24. Rauf, A., Fukushima, M.: Globally convergent BFGS method for nonsmooth convex optimization. J. Optim. Theory Appl. **104**(3), 539–558 (1997)

25. Robinson, S.M.: Linear convergence of epsilon-subgradient descent methods for a class of convex functions. Math. Program. **86**(1), 41–50 (1999). https://doi.org/10.1007/s101070050078
26. Rockafellar, R.T., Wets, R.J.B.: Variational Analysis: Grundlehren Der Mathematischen Wissenschaften, vol. 317. Springer, Berlin (1998)
27. Sagastizábal, C.: A $\mathcal{VU}$-point of view of nonsmooth optimization. In: Proceedings of the International Congress of Mathematicians 2018 - Invited Lectures, vol. 3, pp. 3785–3806 (2018)
28. Vaiter, S., Deledalle, C., Fadili, J., Peyré, G., Dossal, C.: The degrees of freedom of partly smooth regularizers Ann. Inst. Stat. Math. **69**(4), 791–832 (2017). https://doi.org/10.1007/s10463-016-0563-z
29. Zhang, R., Treiman, J.: Upper-lipschitz multifunctions and inverse subdifferentials. Nonlinear Anal. Theory Methods Appl. **24**(2), 273–286 (1995). https://doi.org/10.1016/0362-546X(94)E0025-C. http://www.sciencedirect.com/science/article/pii/0362546X94E0025C

Chapter 10
Beyond the Oracle: Opportunities of Piecewise Differentiation

Andreas Griewank and Andrea Walther

Abstract For more than 30 years much of the research and development in nonsmooth optimization has been predicated on the assumption that the user provides an oracle that evaluates at any given $\boldsymbol{x} \in \mathbb{R}^n$ the objective function value $\varphi(\boldsymbol{x})$ and a generalized gradient $\boldsymbol{g} \in \partial\varphi(\boldsymbol{x})$ in the sense of Clarke. We will argue here that, if there is a realistic possibility of computing a vector $\boldsymbol{g}$ that is guaranteed to be a generalized gradient, then one must know so much about the way $\varphi : \mathbb{R}^n \to \mathbb{R}$ is calculated that more information about the behavior of φ in a neighborhood of the evaluation point can be extracted. Moreover, the latter can be achieved with reasonable effort and in a stable manner so that the *derivative information* provided varies Lipschitz continuously with respect to $\boldsymbol{x}$. In particular we describe the calculation of directionally active generalized gradients, generalized ε-gradients and the checking of first and second order optimality conditions. All this is based on the abs-linearization of a piecewise smooth objective in abs-normal form.

10.1 Motivation and Introduction

It is well understood that the convex set $\partial\varphi(\boldsymbol{x})$ of generalized gradients is highly volatile with respect to variations in $\boldsymbol{x}$, never mind that it is by definition outer semicontinuous as a set-valued mapping $\partial\varphi : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$. Moreover, due to Rademacher's theorem (see Theorem 1.5) for Lipschitzian functions on a Euclidean space, we can expect that almost everywhere we get a singleton $\partial\varphi(\boldsymbol{x}) = \{\partial\varphi(\boldsymbol{x})\}$ so that the users effort to somehow code up a guaranteed generalized gradient will rarely ever pay off during an optimization run. In fact in the paper [30], it is simply assumed that this exceptional event will never happen, so that at the iterates the actually generated $\boldsymbol{g}$ will always be a classical Fréchet gradient. However, it must

A. Griewank · A. Walther (✉)
Department of Mathematics, Humboldt University zu Berlin, Berlin, Germany
e-mail: griewank@math.hu-berlin.de; andrea.walther@math.hu-berlin.de

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_10

be noted that even when this optimistic assumption holds, nonsmooth functions may be poorly approximated by their tangent plane, because there can be a kink nearby about which the local gradient knows nothing. Fortunately, one can generate a local piecewise linear model that reflects such close-by derivative discontinuities, so that whatever algorithm one uses has a chance to deal with the nonsmoothness appropriately. In other words, we suggest to handle (possibly multiple and deflected) kinks at the level of the piecewise linearization.

This chapter is organized as follows. In the second section, we introduce the basic notation and discuss the relation between the oracle paradigm and piecewise differentiability. In Sect. 10.3, we discuss the framework of objective functions in abs-normal form and in Sect. 10.4 our approach to generate a local piecewise linear model. In Sect. 10.5, we show how one can get information about the gradients that are active in a neighborhood, in particular the gradients and ε-gradients. While these only allow the checking of stationarity and ε-stationarity, we discuss in Sect. 10.6 the issue of testing for criticality. In the same section, we introduce briefly an algorithm that actually allows to reach a stationary point. The various concepts discussed in this chapter are illustrated in Sect. 10.7 by means of the Crescent example. Section 10.8 discusses several ways of generating abs-linear approximations of functions including the Euclidean norm and compare their efficiency on a paradigmatic example. Finally, we give a summary and a conclusion in Sect. 10.9. Throughout we will consider only the unconstrained case, but most arguments and results carry over to constrained optimization.

10.2 The Oracle and Piecewise Differentiation

Throughout this chapter we assume that the objective $\varphi : \mathcal{D} \mapsto \mathbb{R}$ is locally Lipschitz on an open domain $\mathcal{D} \subset \mathbb{R}^n$. Moreover, we will use the notation and terminology

$$\text{Fréchet gradient:} \qquad \nabla\varphi(\boldsymbol{x}) \equiv \tfrac{\partial\varphi(\boldsymbol{x})}{\partial \boldsymbol{x}} : \mathcal{D} \mapsto \mathbb{R}^n \cup \emptyset;$$

$$\text{Limiting differential:} \quad \partial^L\varphi(\mathring{\boldsymbol{x}}) \equiv \overline{\lim}_{\boldsymbol{x}\to\mathring{\boldsymbol{x}}} \nabla\varphi(\boldsymbol{x}) : \mathcal{D} \rightrightarrows \mathbb{R}^n;$$

$$\text{Clarke differential:} \qquad \partial\varphi(\boldsymbol{x}) \equiv \operatorname{conv}(\partial^L\varphi(\boldsymbol{x})) : \mathcal{D} \rightrightarrows \mathbb{R}^n.$$

where conv denotes the convex hull. The individual elements of the limiting and the Clarke differential will be called limiting gradients and generalized gradients, respectively. The limiting differential is the outer semicontinuous limit of the Fréchet gradient in the Kuratowski-Painlevé sense [35, Section 4.B] so that we have more precisely

$$\partial^L\varphi(\boldsymbol{x}) \;\equiv\; \left\{ \lim_{i\to\infty} \nabla\varphi(\boldsymbol{x}_i) : \boldsymbol{x}_i \to \boldsymbol{x}, \mathcal{D} \ni \boldsymbol{x}_i \notin \mathcal{S} \right\}.$$

Here $\mathcal{S}$ denotes the set of exceptional points, where $\varphi(\boldsymbol{x})$ is not Fréchet differentiable. In the literature the limiting differential is often called the Bouligand differential or derivative.

Definition 10.1 (Oracle Paradigm) The locally Lipschitz continuous (LLC) function $\varphi : \mathbb{R}^n \mapsto \mathbb{R}$ is said to satisfy the oracle paradigm if at any $\boldsymbol{x} \in \mathbb{R}^n$ not only the function value $\varphi(\boldsymbol{x})$ but also at least one generalized gradient $\boldsymbol{g} \in \partial\varphi(\boldsymbol{x})$ can be made available to the optimization algorithm.

At first the task required by the oracle paradigm does not appear that hard in the piecewise differentiable case.

Definition 10.2 (Piecewise Differentiability) The LLC function $\varphi : \mathcal{D} \subset \mathbb{R}^n \mapsto \mathbb{R}$ is said to be $d > 0$ times piecewise differentiable if at any $\boldsymbol{x} \in \mathcal{D}$ there exists a selection function $\varphi_\sigma(\boldsymbol{x}) \in C^d(\mathcal{D})$ such that $\varphi(\boldsymbol{x}) = \varphi_\sigma(\boldsymbol{x})$. Here the signature σ belongs to some finite index set $\mathcal{E}$ labeling the selection functions φ_σ.

In the literature the elements of $\mathcal{E}$ are usually chosen as natural numbers, but we will give ourselves a little more freedom and later define them as tuples of a certain kind.

Definition 10.3 (Active Selections) The selection function φ_σ is said to be *active* at $\mathring{\boldsymbol{x}} \in \mathcal{D}$ if the *coincidence set* $M_\sigma = \{\boldsymbol{x} \in \mathcal{D} : \varphi(\boldsymbol{x}) = \varphi_\sigma(\boldsymbol{x})\}$ contains $\mathring{\boldsymbol{x}}$. Moreover φ_σ is called *essentially active* if $\mathring{\boldsymbol{x}}$ belongs to the closure of the interior of M_σ. Finally, it is called *conically active* if the tangent cone of M_σ at $\mathring{\boldsymbol{x}}$ has a nonempty interior, The index sets of the correspondingly active selection function indexes will be denoted by the chain $\mathcal{E} \supset \mathcal{E}_a(\mathring{\boldsymbol{x}}) \supset \mathcal{E}_e(\mathring{\boldsymbol{x}}) \supset \mathcal{E}_c(\mathring{\boldsymbol{x}})$.

The inclusion relations at the end of the last definition are easily verified. They become intuitively clear if one looks at the drawings in Fig. 10.1.

Lemma 10.1 (Scholtes [36]) *In the piecewise smooth case the limiting differential is the span of the essentially active gradients, i.e.,*

$$\partial^L \varphi(\mathring{\boldsymbol{x}}) = \bigcup_{\sigma \in \mathcal{E}_e(\boldsymbol{x})} \{\nabla \varphi_\sigma(\mathring{\boldsymbol{x}})\}$$

whose convex hull is of course the Clarke differential.

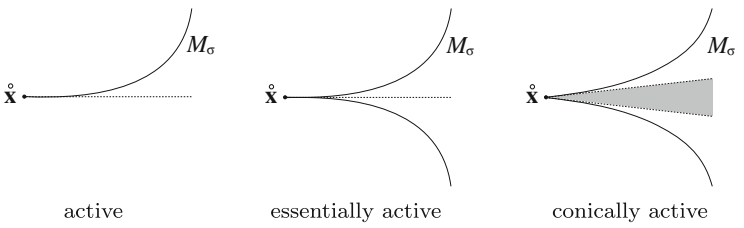

Fig. 10.1 Different coincidence sets with tangential cones

To realize the oracle here one would have to find a signature that is not only active but essentially active, which does not seem quite so simple. For our class it turns out to be easier to get the under- and overestimations

$$\emptyset \neq \partial^K \varphi(\mathring{\boldsymbol{x}}) \equiv \bigcup_{\sigma \in \mathcal{E}_c(\boldsymbol{x})} \left\{ \nabla \varphi_\sigma(\mathring{\boldsymbol{x}}) \right\} \subset \partial^L \varphi(\mathring{\boldsymbol{x}}) \subset \bigcup_{\sigma \in \mathcal{E}_a(\boldsymbol{x})} \left\{ \nabla \varphi_\sigma(\mathring{\boldsymbol{x}}) \right\}.$$

The first set on the left will be called the conic differential, as it contains only gradients $\nabla \varphi_\sigma(\mathring{\boldsymbol{x}})$ of selection functions that are conically active at $\mathring{\boldsymbol{x}}$. As we will see $\partial^K \varphi(\mathring{\boldsymbol{x}})$ is never empty and we will be able to compute one or even all of its finitely many elements for objectives in abs-normal form. It might be reasonably claimed that only the conically active gradients are relevant for optimizing φ in the vicinity of $\mathring{\boldsymbol{x}}$.

The last set on the right can be a gross-overestimation which might come about if one applies the generalized differentiation rules forward in a naive way. To avoid this overestimation one has to detect all selection functions that are active but not essentially active, a rather daunting task for a set of nonlinear $\varphi_\sigma(\boldsymbol{x})$ as the coincidence sets M_σ may be very complicated even if all $\varphi_\sigma(\boldsymbol{x})$ are assumed to be polynomial. Then realizing the oracle paradigm must be considered rather difficult.

A challenging question is how we can evaluate the multifunctions

$$\mathcal{E}_a : \boldsymbol{x} \in \mathcal{D} \rightrightarrows \mathcal{E}, \quad \mathcal{E}_e : \boldsymbol{x} \in \mathcal{D} \rightrightarrows \mathcal{E}, \quad \mathcal{E}_c : \boldsymbol{x} \in \mathcal{D} \rightrightarrows \mathcal{E}.$$

The first one appears easy except that testing equality in floating point arithmetic is always a little dicey. A popular format used in some software packages is that the coincidence sets are defined by $|\mathcal{E}|$ different systems of linear (or nonlinear) inequalities like

$$\sigma \in \mathcal{E}_a(\boldsymbol{x}) \iff A_\sigma \boldsymbol{x} \leq \boldsymbol{b}_\sigma \quad \text{for} \quad A_\sigma \in \mathbb{R}^{m_\sigma \times n}, \ \boldsymbol{b}_\sigma \in \mathbb{R}^{m_\sigma}.$$

The difficulty with this representation is that it suffers from three related drawbacks:

- likely exponential size of data structure;
- redundancy because pieces must fit together;
- numerical perturbations or typos destroy consistency.

Of course, everything is much worse in the nonlinear case. We view the representation by pieces as one of the main reasons why piecewise linear or smooth functions have not been used very much in scientific computing. Let us look at a small example.

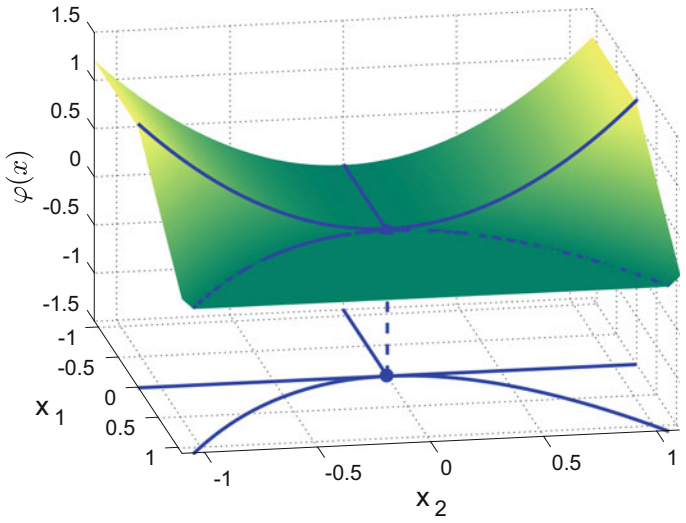

Fig. 10.2 Half pipe function with a priori five smooth pieces

The Half Pipe Example Firstly we consider the definition by pieces

$$\varphi : \mathbb{R}^2 \mapsto \mathbb{R}, \qquad \varphi(x_1, x_2) = \begin{cases} \varphi_{-1,1}(x_1, x_2) = x_2^2, & \text{if } x_1 \leq 0, \\ \varphi_{1,-1}(x_1, x_2) = x_2^2 - x_1, & \text{if } 0 \leq x_1 \leq x_2^2, \\ \varphi_{1,1}(x_1, x_2) = 0, & \text{if } 0 \leq x_2^2 \leq x_1. \end{cases}$$

The graph of this function is given by Fig. 10.2.
The double indexing of the functions will become clearer later. The corresponding coincidence sets $\mathcal{S}_{-1,1}$, $\mathcal{S}_{1,-1}$, $\mathcal{S}_{1,1}$ are all essentially active at the origin, but the double cusp shaped one in the middle is not conically active. The corresponding gradients are

$$\nabla\varphi_{-1,1}(0,0) = (0,0) = \nabla\varphi_{1,1}(0,0) \quad \text{and} \quad \nabla\varphi_{1,-1}(0,0) = (-1,0).$$

Hence we get the differentials

$$\{\nabla\varphi(0,0)\} \;=\; \{(0,0)\} \;=\; \partial^K\varphi(0,0) \subsetneq \partial^L\varphi(0,0) \;=\; \{(0,0),(-1,0)\}.$$

As we see the function is actually differentiable at the origin, which is reflected in the conic differential being a singleton containing only the Fréchet gradient $(0,0)$. In contrast the limiting differential picks up the gradient $(-1,0)$ from the double cusp $\mathcal{S}_{1,-1}$, which is not conically active. Of course the Clarke differential as the convex hull of the limiting differential is simply the line segment $\partial\varphi(0,0) \;=\; \{(\alpha,0) : \alpha \in [-1,0]\}$. To highlight the role of nonessentially active selection functions let us

introduce the function $\varphi_{0,1}(x_1, x_2) = x_2^2 + x_1$ with the coincidence set $\mathcal{S}_{0,1} = \{0\} \times \mathbb{R}$. This selection function is active at the origin but its gradient $\nabla\varphi_{0,1}(0, 0) = (0, 1)$ does and may not belong to the Clarke differential $[-1, 0] \times 0$. Hence it would be a failure for the oracle to return the gradient $(0, 1)$ of an active selection function as a generalized let alone limiting gradient.

Generally this difficulty arises because generalized differentiation rules are mostly only inclusions. Only the operations that are guaranteed to maintain convexity with respect to $\boldsymbol{x}$, namely conic combinations and pointwise maximization, i.e.,

$$v(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i u_i(\boldsymbol{x}) \quad \text{and} \quad v(\boldsymbol{x}) = \max_{i=1,\ldots,m} \{\alpha_i u_i(\boldsymbol{x})\} \quad \text{with}$$
$$\alpha_i \geq 0 \quad \text{for } i = 1, \ldots, m,$$

also propagate the corresponding generalized gradients as identities, see [7], such that

$$\partial v(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i \partial u_i(\boldsymbol{x}) \quad \text{and} \quad \partial v(\boldsymbol{x}) = \operatorname{conv}\{\alpha_i \partial u_i(\boldsymbol{x}) : u_i(\boldsymbol{x}) = v(\boldsymbol{x})\},$$

respectively. All other elementary operations, in particular subtraction and multiplication propagate generalized derivatives only as inclusions, i.e., we have

$$\partial(u - w) \subset \operatorname{conv}\{\partial u - \partial w\} \qquad \text{and} \qquad \partial(u \cdot w) \subset \operatorname{conv}\{w\,\partial u + u\,\partial w\},$$

where we have left off the argument $\boldsymbol{x}$ for notational simplicity. The same is true for the absolute value function $v = \operatorname{abs}(\boldsymbol{x})$ so that one can never be sure to obtain true generalized gradients when propagating such vectors forward through a chain of operations.

At a higher level, compositions of vector functions propagate generalized derivatives by the chain rule also as an inclusion, with identity holding only when one of the factors involved is smooth or at least subdifferentially regular [3, Definition 3.5]. The verification of this frequently assumed property was shown to be co-NP complete in [39] on the class of piecewise smooth functions $C^d_{\text{abs}}(\overline{\mathcal{D}})$ defined below. It actually amounts to local convexity of the piecewise linearization or equivalently the directional derivative $\varphi'(\boldsymbol{x}; \cdot)$ and is thus a rather strong and complex assumption. Finally, we note that for approximating generalized gradients by divided differences, see [3, Chapter 6], one has to rely on this convex set being finitely generated and thus polyhedral, which comes pretty close to assuming absnormality as defined in the next section. We will see, that assumption allows in fact the exact calculation of the conical differential $\partial^K\varphi(\boldsymbol{x})$ which is an always nonempty subset of the limiting differential $\partial^L\varphi(\boldsymbol{x})$.

10.3 Abs-Normal Objectives

For the half pipe example, one may consider several formulations. Firstly, what one might consider the "original" formulation in terms of max

$$\varphi(x_1, x_2) \ = \max\{x_2^2 - \max\{x_1, 0\}, 0\}.$$

Here the Lipschitz continuity is immediately apparent. Alternatively, rewriting max in terms of abs we get after some mechanical manipulations

$$\varphi(x_1, x_2) = \tfrac{1}{2}\left(x_2^2 - \tfrac{1}{2}(x_1 + |x_1|) + \left|x_2^2 - \tfrac{1}{2}(x_1 + |x_1|)\right|\right).$$

Now we have a formulation where all nonsmoothness is cast in terms of the absolute value function, which occurs at the two arguments x_1 and $x_2^2 - \frac{1}{2}(x_1 + |x_1|)$. Wherever these quantities change their sign we will have a kink in the function value. Therefore we name them switching variables and define them as

$$(z_1, z_2) = F(x_1, x_2, z_1) = \left(x_1, x_2^2 - \tfrac{1}{2}(x_1 + |z_1|)\right). \tag{10.1}$$

Substituting them into the original expression we get

$$\varphi(x_1, x_2) = f(\boldsymbol{x}, |\boldsymbol{z}|) = \tfrac{1}{2}\left(x_2^2 - \tfrac{1}{2}(x_1 + |z_1|) + |z_2|\right). \tag{10.2}$$

Now we have two Eqs. (10.1) and (10.2) that define the half pipe function in a nonredundant and stable way. Any of the coefficients, which are mostly 1 can be perturbed with the resulting function still being well defined and Lipschitz continuous. Moreover, we can label the various smooth function pieces by the vector $\boldsymbol{\sigma} = (\text{sign}(z_1), \text{sign}(z_2)) \in \{-1, 0, 1\}^2$, which is consistent with the labeling we used in the previous section.

More generally, we will consider the class of objective functions that are defined as compositions of smooth elemental functions and the absolute value function $\text{abs}(x) = |x|$. Hence they may also include $\max\{x, y\}$, $\min\{x, y\}$, and the positive part function $\text{pos}(x) \equiv \max\{x, 0\}$, which can all be easily cast in terms of an absolute value. By successively numbering all arguments of absolute value evaluations as *switching variables* z_i for $i = 1 \ldots s$, we obtain a piecewise smooth representation of $y = \varphi(\boldsymbol{x})$ in the abs-normal form

$$\boldsymbol{z} \ = \ F(\boldsymbol{x}, |\boldsymbol{z}|), \tag{10.3}$$

$$y \ = \ f(\boldsymbol{x}, |\boldsymbol{z}|), \tag{10.4}$$

where for $\mathcal{D} \subset \mathbb{R}^n$ open, $F : \overline{\mathcal{D}} \times \overline{\mathbb{R}^s_+} \mapsto \mathbb{R}^s$ and $f : \overline{\mathcal{D}} \times \overline{\mathbb{R}^s_+} \mapsto \mathbb{R}$ with $\overline{\mathcal{D}} \times \overline{\mathbb{R}^s_+} \subset \mathbb{R}^{n+s}$. Here, z_j can only influence z_i if $j < i$ so that when interpreting

F as a function of $|z|$, its Jacobian with respect to $|z|$ is strictly lower triangular. Consequently, we can evaluate for any $\boldsymbol{x}$ the unique, piecewise smooth value $z(\boldsymbol{x})$. In other words, we state the calculation of all switching variables as equality constraints and handle the vector of the absolute values of the switching variables as extra argument of the then smooth target function f. Sometimes, we write

$$\varphi(\boldsymbol{x}) \;\equiv\; f(\boldsymbol{x}, |z(\boldsymbol{x})|)$$

to denote the objective directly in terms of the argument vector $\boldsymbol{x}$ only. In this chapter, we are mostly interested in the case where the nonlinear elementals are all once or twice continuously differentiable. The resulting function class was first considered in [12] and is specified as follows:

Definition 10.4 For any $d \in \mathbb{N}$ and $\mathcal{D} \subset \mathbb{R}^n$, the set of functions $\varphi : \overline{\mathcal{D}} \mapsto \mathbb{R}$ defined by an abs-normal form (10.3)–(10.4) with $f, F \in C^d(\overline{\mathcal{D}} \times \overline{\mathbb{R}^s_+})$ is denoted by $C^d_{\text{abs}}(\overline{\mathcal{D}})$.

Recall that $C^d(\overline{\Omega})$ is the set of functions that possess continuous d-th derivatives in the open set Ω that can be continuously extended to the boundary $\partial\Omega = \overline{\Omega}\backslash\Omega$. In the usual case, where F and f are themselves compositions of smooth elemental functions φ_i these are assumed to be $C^d(\overline{\mathcal{D}}_i)$ functions on their respective domains $\mathcal{D}_i$ reachable from $\boldsymbol{x} \in \mathcal{D}$. The combinatorial structure of the nonsmooth function φ can be described by the signature vector and matrix

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{x}) \equiv \text{sign}(z(\boldsymbol{x})) \in \{-1, 0, +1\}^s \;\text{ and}$$
$$\Sigma \equiv \Sigma(\boldsymbol{x}) = \text{diag}(\boldsymbol{\sigma}(\boldsymbol{x})) \in \mathbb{R}^{s\times s}.$$

For fixed $\boldsymbol{\sigma}$ and the corresponding Σ we can locally solve Eq. (10.3) using $|z(\boldsymbol{x})| = \Sigma z(\boldsymbol{x})$ for $z(\boldsymbol{x})$ and thus have the implicitly defined selection function

$$\varphi_\sigma(\boldsymbol{x}) \;\equiv\; f(\boldsymbol{x}, \Sigma z(\boldsymbol{x})) \quad \text{such that} \quad z(\boldsymbol{x}) = F(\boldsymbol{x}, \Sigma z(\boldsymbol{x})). \tag{10.5}$$

Again due to the assumed triangularity, the system of equations is locally solvable for $z(\boldsymbol{x})$ by the implicit function theorem. Hence, the functions in $C^d_{\text{abs}}(\overline{\mathcal{D}})$ are certainly piecewise smooth as defined in Definition 10.2. In our scenario $\mathcal{E}$ is a subset of $\{-1, 0, 1\}^s$ by definition of $\boldsymbol{\sigma}$. Generally in the literature, it remains a little mysterious how a suitable index $\boldsymbol{\sigma}$ is chosen as a function of $\boldsymbol{x}$ such that the resulting function is continuous. In our function model the determination of the $\boldsymbol{\sigma}(\boldsymbol{x})$ is intertwined with the computation of the numerical values.

Related Piecewise Linear Functions The class $C^d_{\text{abs}}(\overline{\mathcal{D}})$ covers many piecewise smooth functions but certainly not all. For example, on a given triangulation of the plane or space, somebody may have spliced together different local models such that they fit continuously across the triangle edges or tetrahedron faces. In this situation it seems impossible to deduce from the properties of the function within one triangle

or tetrahedron anything about what is happening in the neighboring triangles or tetrahedrons, let alone further afield.

In contrast, in some sense the functions belonging to $C^d_{\rm abs}(\overline{\mathcal{D}})$ allow extrapolation from one polyhedron to its neighbors. We even harbor the hope that there might be reasonably efficient methods to globally optimize piecewise linear functions using their abs-linear form. That representation always exists but may be not so easy to construct.

On the other hand, we must admit that in the spliced situation the oracle paradigm appears quite natural with each limiting differential being just the gradient of the adjacent patches, i.e., one of the selection functions. However, except on triangular or quadrilateral grids it may again not be easy to decide which selection function is essentially active as defined in Definition 10.3. Of course any statement of stationarity or optimality will only apply to the patch and nothing can be said about the behavior of the piecewise smooth function in an open neighborhood, no matter how small.

Another class of possibly even linear piecewise smooth functions that does not fit within the $C^d_{\rm abs}(\overline{\mathcal{D}})$ framework are solution operators like

$$\varphi(\boldsymbol{x}) \;=\; \max\{\tfrac{1}{2}\boldsymbol{y}^T Q\,\boldsymbol{y} + \boldsymbol{c}^T\boldsymbol{y} : A\,\boldsymbol{y} \le B\boldsymbol{x} + \boldsymbol{c}\} \quad \text{with} \quad Q = Q^T \succ 0.$$

Here we may use a finite solver to compute the mathematically well defined piecewise linear function $\varphi(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ but the number of steps may vary depending on the argument $\boldsymbol{x}$. Moreover, there may be degeneracies whose numerical resolution requires if-statements and other program branches. As shown in [4], for implicitly defined functions like $G(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) = \boldsymbol{0}$ with G in abs-normal form, one can compute its abs-linear approximation $\Delta\boldsymbol{y}(\mathring{\boldsymbol{x}}, \Delta\boldsymbol{x})$ by a generalized version of the implicit function theorem. Albeit with some nontrivial effort, this could be integrated into the extended algorithmic differentiation (AD) software, which has not been done.

In the penultimate section of the paper we will consider the extension of $C^d_{\rm abs}(\overline{\mathcal{D}})$ to its superset $C^d_{\rm euc}(\overline{\mathcal{D}})$, which consists of all functions that can be evaluated as compositions of C^d elementals and the Euclidean norm $\|\cdot\| = \|\cdot\|_2$. These Lipschitz continuous functions are no longer piecewise smooth, but one can still construct abs-linear approximations that appear to be useful for optimization purposes.

It has recently been observed [21] that minimizing a function in abs-normal form is equivalent to solving the equality constrained MPEC

$$\begin{cases} \text{minimize} & \varphi = f(\boldsymbol{x}, \boldsymbol{u} + \boldsymbol{v}) \\ \text{subject to} & \boldsymbol{0} \le \boldsymbol{u} \perp \boldsymbol{v} \ge \boldsymbol{0}, \\ & \boldsymbol{u} - \boldsymbol{v} = F(\boldsymbol{x}, \boldsymbol{u} + \boldsymbol{v}). \end{cases}$$

Here, MPEC means mathematical programming with equilibrium constraints [31]. Notice that in general, i.e., without the triangularity of F with respect to $|z| = \boldsymbol{u} + \boldsymbol{v}$,

the MPEC may be quite hard to solve, if only because one cannot easily compute a feasible $z = u - v$ for any given x. Nevertheless, it was shown in [21] that in the triangular case the constraint qualification MPEC-LICQ is equivalent to the linear independent kink qualification (LIKQ) introduced in [15] for abs-normal objectives.

10.4 The Abs-Linear Approximation

All abs-normal objectives are strongly semismooth as defined in [8, Chapter 7] and thus their generalized gradients satisfy for fixed $\mathring{x}$ the backward approximation property [22]

$$\varphi(x) - \varphi(\mathring{x}) - g^T(x - \mathring{x}) \;=\; \mathcal{O}(\|x - \mathring{x}\|^2) \quad \text{for all} \quad g \in \partial\varphi(x).$$

While the vector version of this relation forms the basis of the semismooth Newton method it is not clear how it can be exploited for the purposes of unconstrained local optimization. Instead we aim for a generalization of the classical first order Taylor expansion, which is in some sense forward, from the reference point $\mathring{x}$ with the corresponding $\mathring{z} = z(\mathring{x})$ and $\mathring{y} = y(\mathring{x})$ to a trail point $x \approx \mathring{x}$ and the corresponding $z = z(x)$ and $y = y(x)$. From Eqs. (10.3) and (10.4), one obtains the smooth Taylor expansion

$$\begin{bmatrix} z - \mathring{z} \\ y - \mathring{y} \end{bmatrix} = \begin{bmatrix} Z & L \\ a^T & b^T \end{bmatrix} \begin{bmatrix} x - \mathring{x} \\ |z| - |\mathring{z}| \end{bmatrix} + \mathcal{O}\left(\begin{bmatrix} \|x - \mathring{x}\|^2 \\ \|z - \mathring{z}\|^2 \end{bmatrix} \right)$$

as abs-linear approximation of $y = \varphi(x)$. Here the matrices

$$L \equiv \frac{\partial}{\partial |z|} F(x, |z|) \in \mathbb{R}^{s \times s}, \quad Z \equiv \frac{\partial}{\partial x} F(x, |z|) \in \mathbb{R}^{s \times n} \tag{10.6}$$

and the vectors

$$a = \frac{\partial}{\partial x} f(x, |z|) \in \mathbb{R}^n, \quad b = \frac{\partial}{\partial |z|} f(x, |z|) \in \mathbb{R}^s \tag{10.7}$$

are evaluated at the reference point $(\mathring{x}, \mathring{z})$. Due to the triangularity of F and thus L one can easily check by induction that $\|z - \mathring{z}\| = \mathcal{O}(\|x - \mathring{x}\|)$. Hence we obtain with $\Delta x \equiv x - \mathring{x}$ and

$$\tilde{z} \equiv c + Z\Delta x + L|\tilde{z}| \equiv (\mathring{z} - L|\mathring{z}|) + Z\Delta x + L|\tilde{z}|, \tag{10.8}$$

the incremental approximation

$$\Delta\varphi(\mathring{\boldsymbol{x}};\,\Delta\boldsymbol{x}) \equiv \boldsymbol{a}^T\Delta\boldsymbol{x} + \boldsymbol{b}^T(|\tilde{\boldsymbol{z}}| - |\mathring{\boldsymbol{z}}|) \;=\; y - \mathring{y} + \mathcal{O}(\|\Delta\boldsymbol{x}\|^2), \tag{10.9}$$

or equivalently

$$\varphi(\mathring{\boldsymbol{x}} + \Delta\boldsymbol{x}) - \varphi(\mathring{\boldsymbol{x}}) \;=\; \Delta\varphi(\mathring{\boldsymbol{x}};\,\Delta\boldsymbol{x}) + \mathcal{O}(\|\Delta\boldsymbol{x}\|^2). \tag{10.10}$$

In other words, we have a generalized Taylor approximation with uniform error term $\mathcal{O}(\|\Delta\boldsymbol{x}\|^2)$, which is in contrast to directional differentiation completely independent of the direction $\Delta\boldsymbol{x}/\|\Delta\boldsymbol{x}\|$. This is possible because $\Delta\varphi(\mathring{\boldsymbol{x}};\,\Delta\boldsymbol{x})$ is with respect to $\Delta\boldsymbol{x}$ piecewise linear but in contrast to the directional derivative $\varphi'(\mathring{\boldsymbol{x}};\,\Delta\boldsymbol{x})$ not homogeneous. That means, it "knows" about nearby kinks, which is exactly the kind of information that any kind of strictly local generalized differentiation can not pick up. In previous papers [13, 18] we have derived the relation (10.10) by induction on the intermediate quantities occurring in the evaluation procedure of the overall $\varphi(\boldsymbol{x})$. This approach then directly yields the partial elemental derivatives, from which the "global" matrices and vectors $[L, Z, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}]$ can be accumulated by suitable variants of the chain rule.

For example let us consider the half pipe example in the abs-normal form (10.1) and (10.2). Then, we get by differentiation at any $\mathring{\boldsymbol{x}}$

$$L = \begin{bmatrix} 0 & 0 \\ -\frac{1}{2} & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} -0.25 \\ 0.5 \end{bmatrix}$$

and in dependence on the reference point $\mathring{\boldsymbol{x}} = (\mathring{x}_1, \mathring{x}_2)$ with $\mathring{\boldsymbol{z}} = \boldsymbol{z}(\mathring{\boldsymbol{x}})$

$$Z = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & 2\,\mathring{x}_2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{a} \;=\; \begin{bmatrix} -0.25 \\ \mathring{x}_2 \end{bmatrix}.$$

The original function and the resulting abs-linearization at $\mathring{\boldsymbol{x}} = (-1, 1)$ are illustrated in Fig. 10.3. At $\mathring{\boldsymbol{x}}$ we have $\mathring{\boldsymbol{z}} = (-1, 1)$ and thus $\mathring{\boldsymbol{\sigma}} = (-1, 1)$ which means that the function is locally completely smooth at $\mathring{\boldsymbol{x}}$ but the abs-linearization still has an idea where there are kinks. Notice that Z has the determinant $2x_2$ which means that at the origin where both switching variables are active the linear independence kink qualification, as introduced in [15] is not satisfied.

Evaluating $\Delta\varphi(\mathring{\boldsymbol{x}};\,\Delta\boldsymbol{x})$ via Eqs. (10.8) and (10.9) is quite cheap, provided the matrices and vectors $(Z, L, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, which constitute the abs-linear approximation are known. They are all first derivatives of smooth, composite functions, so they can be obtained by algorithmic or automatic differentiation. In fact the well known AD tools ADOL-C [38], CPP-AD [5] and Tapenade [20] have been extended to generate abs-linear approximations. Of course, just like in the smooth case, where the evaluation of the full Jacobian can be avoided through iterative methods that are based exclusively on tangents in the sense of Jacobian×vector products and co-

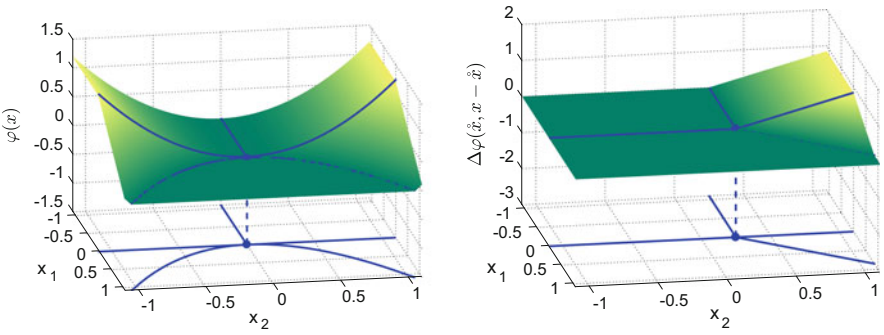

Fig. 10.3 The half pipe function and its abs-linearisation at $\mathring{\boldsymbol{x}} = (-1, 1)$

tangents or adjoints in the sense of row-vector×Jacobian products, such a matrix free approach can also be pursued for the abs-linear approximation. However, for notational simplicity we will assume in this chapter that the matrices Z and L are completely accumulated. As defined via Eqs. (10.8) and (10.9) for fixed $\mathring{\boldsymbol{x}}$ the function $\Delta\varphi(\mathring{\boldsymbol{x}}; \Delta\boldsymbol{x})$ is just an abs-normal function, where all operations other than the absolute value are linear or more precisely affine.

10.5 Checking Gradient Activity

Now the question arises what information about $\varphi(\boldsymbol{x})$ near the reference point $\mathring{\boldsymbol{x}}$ we can gain from the analysis of its abs-linear approximation $\Delta\varphi(\mathring{\boldsymbol{x}}; \Delta\boldsymbol{x})$ near $\Delta\boldsymbol{x} = \boldsymbol{0}$. For notational simplicity we set $\mathring{\boldsymbol{x}} = \boldsymbol{0}$, replace $\Delta\boldsymbol{x}$ by $\boldsymbol{x}$ and $\tilde{\boldsymbol{z}}$ by $\boldsymbol{z}$ as well as ignoring constant shifts in the objective function. Then we have simply the abs-linear minimization problem

$$\begin{cases} \text{minimize} & \Delta y(\boldsymbol{x}) \equiv \boldsymbol{a}^T\boldsymbol{x} + \boldsymbol{b}^T|\boldsymbol{z}| \\ \text{subject to} & \boldsymbol{z} \equiv \boldsymbol{c} + Z\boldsymbol{x} + L|\boldsymbol{z}|. \end{cases} \tag{10.11}$$

Due to the strict lower triangularity of L there is a unique piecewise linear $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$, which is a special case of the piecewise smooth $\boldsymbol{z}(\boldsymbol{x})$ considered before for the abs-normal φ itself. However, on the abs-linear level we have a much better chance of dealing with the nonsmoothness represented by the kinks explicitly. Then the full domain $\mathbb{R}^n$ is decomposed into polyhedra, which can be identified by the signature vector and matrix

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{x}) \equiv \operatorname{sign}(\boldsymbol{z}(\boldsymbol{x})) \in \{-1, 0, +1\}^s \quad \text{and}$$
$$\Sigma \equiv \Sigma(\boldsymbol{x}) = \operatorname{diag}(\boldsymbol{\sigma}(\boldsymbol{x})) \in \mathbb{R}^{s\times s}$$

now as a function of the piecewise linear $z(x)$. The inverse images

$$P_\sigma \equiv \{x \in \mathbb{R}^n : \sigma(x) = \sigma\} \tag{10.12}$$

are pairwise disjoint, relatively open polyhedra. Using the partial order of the signatures given by

$$\tilde{\sigma} \prec \sigma \quad \Longleftrightarrow \quad \tilde{\sigma}_i \sigma_i \leq \sigma_i^2 \quad \text{for} \quad i = 1 \ldots s,$$

we can define the essential closures

$$\bar{P}_\sigma \equiv \{x \in \mathbb{R}^n : \sigma(x) \prec \sigma\},$$

which are no longer disjoint and whose inclusion ordering corresponds exactly to the partial ordering $\prec$ of the signatures such that

$$\bar{P}_\sigma \subset \bar{P}_{\tilde{\sigma}} \iff \sigma \prec \tilde{\sigma}.$$

Hence, we see that $\mathring{x} = 0$ with $\mathring{\sigma} = \sigma(\mathring{x})$ belongs exactly to the essential closures $\bar{P}_\sigma$ for which $\sigma \succ \mathring{\sigma}$. Consequently, we find for some open ball $B(\mathring{x}; \rho)$

$$B(\mathring{x}; \rho) = \left\{ \bigcup_{\sigma \succ \mathring{\sigma}} P_\sigma \right\} \cap B(\mathring{x}; \rho).$$

Here, the σ on the right hand side can be restricted to be definite, i.e., only have nonzero components $\sigma_i = \pm 1$, which will be denoted by $\sigma \not\geq 0$. Within each P_σ, we have $|z| = \Sigma z$ so that one can solve the equality constraint on the right hand side of Eq. (10.11) for z to obtain the affine function

$$z(x) = (I - L\Sigma)^{-1}(c + Zx) \quad \text{for} \quad x \in \bar{P}_\sigma. \tag{10.13}$$

Note that due to the strict lower triangularity of L the unit lower triangular matrix $(I - L\Sigma)^{-1}$ is for any σ well defined and its elements are polynomial in the entries of L. For definite signatures $\sigma \not\geq 0$ the elements $x \in \bar{P}_\sigma$ are exactly characterized as solutions of the system of inequalities

$$\Sigma(I - L\Sigma)^{-1}(c + Zx) = (\Sigma - L)^{-1}(c + Zx) \geq 0.$$

If there is an $x \in \bar{P}_\sigma$ with definite signature $\sigma(x) \not\geq 0$ then the polyhedron P_σ has a nonempty interior. The converse needs not be true in the presence of degeneracy. From duality theory it is known that either, $\bar{P}_\sigma$ has a nonempty interior, in which case we call it full-dimensional, or the rows of $(\Sigma - L)^{-1} Z$ have a vanishing convex combination such that

$$\lambda^T (\Sigma - L)^{-1} Z = 0 \quad \text{with} \quad 0 \leq \lambda \neq 0.$$

Obviously this can be checked by standard linear optimization techniques. One can also check whether $\dim(\bar{P}_{\boldsymbol{\sigma}}) = n$ in which case we have the gradient

$$\boldsymbol{g}_{\boldsymbol{\sigma}} \;=\; \boldsymbol{a}^T + \boldsymbol{b}^T \Sigma (I - L\Sigma)^{-1} Z \;=\; \boldsymbol{a}^T + \boldsymbol{b}^T (\Sigma - L)^{-1} Z, \tag{10.14}$$

where the last equality relies on definiteness, i.e., $\boldsymbol{\sigma} \not\ni \mathbf{0}$, so that $\det(\Sigma) = \pm 1$.

Hence we obtain for the abs-linear approximation the nonempty set of limiting gradients

$$\partial^L_{\Delta \boldsymbol{x}} \Delta\varphi(\mathring{\boldsymbol{x}}; \Delta \boldsymbol{x})\Big|_{\Delta \boldsymbol{x} = \mathbf{0}} \;=\; \bigcup_{\mathbf{0} \not\ni \boldsymbol{\sigma} \succ \mathring{\boldsymbol{\sigma}}} \{\boldsymbol{a}^T + \boldsymbol{b}^T \Sigma (I - L\Sigma)^{-1} Z\}. \tag{10.15}$$

Conic Activity Now, let φ be again a general nonlinear C^d_{abs} function. It was shown in [13] and [27] that

$$\emptyset \;\neq\; \partial^K_{\boldsymbol{x}} \varphi(\boldsymbol{x})\Big|_{\boldsymbol{x} = \mathring{\boldsymbol{x}}} \equiv \partial^L_{\Delta \boldsymbol{x}} \Delta\varphi(\mathring{\boldsymbol{x}}, \Delta \boldsymbol{x})\Big|_{\Delta \boldsymbol{x} = \mathbf{0}} \;\subset\; \partial^L_{\boldsymbol{x}} \varphi(\boldsymbol{x})\Big|_{\boldsymbol{x} = \mathring{\boldsymbol{x}}},$$

which immediately implies the corresponding inclusion for the Clarke differential as convex hull of the limiting differential. The limiting gradients $\boldsymbol{g}_{\boldsymbol{\sigma}} \in \partial^K \varphi(\mathring{\boldsymbol{x}})$ are *conic* gradients of φ as defined in Definition 10.3. Then we have in fact $\boldsymbol{g}_{\boldsymbol{\sigma}} = \partial\varphi_{\boldsymbol{\sigma}}(\mathring{\boldsymbol{x}})$ for some $\boldsymbol{\sigma} \in \mathcal{E}_c(\mathring{\boldsymbol{x}})$. The limiting gradient $\partial^L \varphi(\mathring{\boldsymbol{x}})$ may contain other gradients $\partial\varphi_{\tilde{\boldsymbol{\sigma}}}(\mathring{\boldsymbol{x}})$ of selection functions $\varphi_{\tilde{\boldsymbol{\sigma}}}$ that are essentially active so that $\tilde{\boldsymbol{\sigma}} \in \mathcal{E}_e(\mathring{\boldsymbol{x}}) \setminus \mathcal{E}_c(\mathring{\boldsymbol{x}})$. If φ happens to be differentiable, but not necessarily strictly differentiable at $\mathring{\boldsymbol{x}}$ we have simply $\partial^K \varphi(\boldsymbol{x}) = \{\partial\varphi(\boldsymbol{x})\}$, which must be the case at almost all points in $\mathbb{R}^n$ by Rademacher's theorem. This applies also to functions like $\varphi(x) = |1 - \sin^2(x) - \cos^2(x)|$ where a conventional chain rule oriented "Are we differentiable at this point?" test, whose use is for example suggested in [6] would naturally always respond "no".

Back to the Oracle Obviously, the observations above also hold for generalized Jacobians of vector-valued functions, so abs-linearization also provides a practical procedure for implementing the semismooth Newton method [23, 28]. Then, only one element of $\partial^L \varphi(\mathring{\boldsymbol{x}}) \subset \partial\varphi(\mathring{\boldsymbol{x}})$ is required. The same holds true for the subgradient as well as the bundle methods. If checking the openness of the interiors of the candidate $\bar{P}_{\boldsymbol{\sigma}}$ appears too laborious, one can employ the technique of *polynomial escape* in a given preferred direction $\boldsymbol{d}_1 \in \mathbb{R}^n$. After complementing it with $(n-1)$ directions $\boldsymbol{d}_i$ for $i = 2 \ldots n$ such that $\det(\boldsymbol{d}_1 \ldots \boldsymbol{d}_n) \neq 0$ one knows that for some $\boldsymbol{\sigma}$ and all small $0 < t \approx 0$

$$\boldsymbol{x}(t) \;=\; \mathring{\boldsymbol{x}} + \sum_{i=1}^{n} t^i \boldsymbol{d}_i \;\in\; P_{\boldsymbol{\sigma}}$$

with $P_{\boldsymbol{\sigma}}$ being open. This corresponding $\boldsymbol{\sigma}$ and the corresponding gradient $\boldsymbol{g}_{\boldsymbol{\sigma}}$ can be calculated independently of the parameter t using the function `firstsign`

as described in [13]. It is a version of lexicographic differentiation introduced by Nesterov [34] and generalized to the case of composite functions by Khan and Barton [26]. The resulting $\boldsymbol{g}_{\sigma}$ is active in the direction $\boldsymbol{d}_1$ in that for some $\bar{t} > 0$

$$\boldsymbol{g}_{\sigma}^T \boldsymbol{d}_1 t \;=\; \Delta\varphi(\mathring{\boldsymbol{x}}; t\,\boldsymbol{d}_1) \;=\; \varphi(\mathring{\boldsymbol{x}} + t\boldsymbol{d}_1) - \varphi(\mathring{\boldsymbol{x}}) + \mathcal{O}(t^2) \quad \text{for} \quad 0 \le t \le \bar{t}.$$

The procedure for computing directionally active generalized gradients is actually matrix free and can be implemented efficiently using the so-called reverse mode of AD [14] except in the most degenerate cases. In those bad situations the complexity might equal that of the forward mode of AD, namely n times the complexity of evaluating the function $\varphi(\boldsymbol{x})$ itself, see also [25].
In the scenario $\varphi \in C^d_{\text{abs}}(\overline{\mathcal{D}})$ the directionally active gradient represents a little bit more then just any generalized gradient. The so-called *critical multiplier* gives important information about the nonsmoothness of φ from $\mathring{\boldsymbol{x}}$ in the direction $\boldsymbol{d}_1$. Moreover, if $\bar{t} < \infty$ we can also provide a gradient $\boldsymbol{g}^+(\mathring{\boldsymbol{x}}, \boldsymbol{d}_1)$ that is active in the direction $\boldsymbol{d}_1$ on the abs-linearization $\Delta\varphi(\mathring{\boldsymbol{x}}, \Delta\boldsymbol{x})$ at the point $\bar{t}\boldsymbol{d}_1$. The difference $\boldsymbol{g}^+(\mathring{\boldsymbol{x}}, \boldsymbol{d}_1) - \boldsymbol{g}(\mathring{\boldsymbol{x}}, \boldsymbol{d}_1)$ will then be a normal of the hyperplane separating the polyhedron before and beyond the kink location $\bar{t}\boldsymbol{d}_1$. This situation is depicted in Fig. 10.4, where the dashed lines represent the kinks. Of course we can expect that on the underlying nonlinear function the situation is similar up to perturbations of $\mathcal{O}(\|\bar{t}\boldsymbol{d}_1\|^2)$. This should provide a lot of useful information for any kind of method based on generalized gradients.

Provided the complementing of $\boldsymbol{d}_1$ by $(n-1)$ linearly independent directions is continuous, the mapping

$$(\mathring{\boldsymbol{x}}, \boldsymbol{d}_1) \in \mathbb{R}^{n+n} \;\mapsto\; (\boldsymbol{g}, \bar{t}) \in \mathbb{R}^n \times (0, \infty)$$

has the following property: The multiplier $\bar{t}$ is continuous in the sense of extended real valued functions. The gradient $\boldsymbol{g}$ itself may have jumps and reduces to the

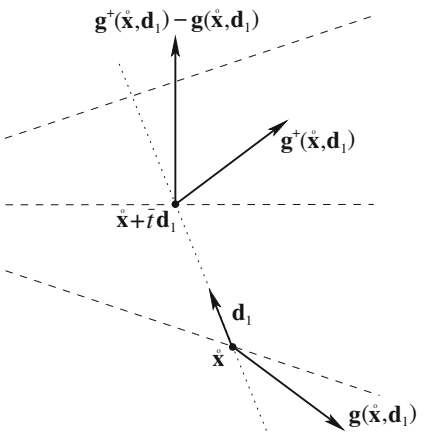

Fig. 10.4 Directional active gradients

Fréchet gradient, wherever that exists. The second gradient $\boldsymbol{g}^+(\boldsymbol{x};\boldsymbol{d}_1)$ will be in many cases an ε-gradient at $\mathring{\boldsymbol{x}}$, but obviously one of them will generally not be enough to model φ locally even in the convex case. Of course, bundle methods could collect several of them, picking up several hyperplanes at a time.

$\varepsilon-$**Activity** Since the limiting and Clarke generalized differentials are not inner semicontinuous [35], the minimal norm of their elements gives no indication of the distance to any stationary point, in particular stopping criteria can not be based on it. Therefore, nonsmooth analysis has partly abandoned the strictly local point of view and introduced $\varepsilon-$differentials, which take note of the objective function behavior nearby. In our notation the definition of the Goldstein ε-differential (see Definition 1.10) reads

$$\partial_\varepsilon^G \varphi(\mathring{\boldsymbol{x}}) \;\equiv\; \mathrm{conv}\left\{\bigcup \partial^L \varphi(\boldsymbol{x}) : \boldsymbol{x} \in \bar{B}(\mathring{\boldsymbol{x}};\varepsilon)\right\}.$$

By definition the Goldstein ε-differential is inner semicontinuous. Obviously, we have $\partial\varphi(\mathring{\boldsymbol{x}}) = \partial_0^G\varphi(\mathring{\boldsymbol{x}})$. The natural question is how the Goldstein ε-differential $\partial_\varepsilon^G\varphi(\mathring{\boldsymbol{x}})$ can actually be computed. Of course, looking at all points $\boldsymbol{x}$ in a spherical neighborhood of the reference point $\mathring{\boldsymbol{x}}$ and computing the convex hull of the union of the limiting subdifferentials $\partial^L\varphi(\boldsymbol{x})$ appears practically impossible.

In the case of limiting gradients as in Eq. (10.15) we only looked at signatures $\boldsymbol{\sigma}$ and their gradients $\boldsymbol{g}_{\boldsymbol{\sigma}}$ of which we were certain that they are active for the abs-linearization and thus the function itself at $\mathring{\boldsymbol{x}}$. For all these neighboring signatures $\boldsymbol{\sigma}$ we know that $\boldsymbol{\sigma} \succ \mathring{\boldsymbol{\sigma}}$ which is equivalent to $\Sigma\mathring{\boldsymbol{z}} \geq \mathbf{0}$. That means when $\mathring{z}_i \neq 0$ the σ_i must have the same sign as $\mathring{\sigma}_i$ and where $\mathring{z}_i = 0$ we may choose freely $\sigma_i \in \{-1, 1\}$. In order to get a larger set of gradients we may relax the condition on $\boldsymbol{\sigma}$ and only require that $\Sigma\mathring{\boldsymbol{z}} > -\varepsilon\,\boldsymbol{e}$ for the given $\varepsilon > 0$ and $\boldsymbol{e} = (1, 1, \ldots, 1)$. Then we define the corresponding limiting ε-differential

$$\partial_\varepsilon^L \varphi(\mathring{\boldsymbol{x}}) \;=\; \left\{\boldsymbol{a} + \boldsymbol{b}^T(\Sigma - L)^{-1}Z : \Sigma\mathring{\boldsymbol{z}} > -\varepsilon\,\boldsymbol{e}\right\} \;\supset\; \partial_0^L\varphi(\mathring{\boldsymbol{x}}) \;\supset\; \partial^L\varphi(\mathring{\boldsymbol{x}}), \tag{10.16}$$

and correspondingly "our" $\varepsilon-$differential simply as

$$\partial_\varepsilon\varphi(\mathring{\boldsymbol{x}}) = \mathrm{conv}(\partial_\varepsilon^L\varphi(\mathring{\boldsymbol{x}})).$$

Now we establish the desirable inner semicontinuity of both.

Lemma 10.2 *For fixed $\varepsilon > 0$ the multi-function $\boldsymbol{x} \rightrightarrows \partial_\varepsilon^L\varphi(\boldsymbol{x}) \subset \mathbb{R}^n$ and its convex hull $\boldsymbol{x} \rightrightarrows \partial_\varepsilon\varphi(\boldsymbol{x}) = \mathrm{conv}\{\partial_\varepsilon^L\varphi(\boldsymbol{x})\} \subset \mathbb{R}^n$ are inner semicontinuous.*

Proof First let us consider any $\mathring{\boldsymbol{g}} \in \partial_\varepsilon^L\varphi(\mathring{\boldsymbol{x}})$ and its corresponding Σ satisfying

$$\mathring{\boldsymbol{g}} \;=\; \mathring{\boldsymbol{a}}^T + \mathring{\boldsymbol{b}}^T(\Sigma - \mathring{L})^{-1}\mathring{Z} \quad \text{and} \quad \Sigma\mathring{\boldsymbol{z}} > -\varepsilon\,\boldsymbol{e}$$

with $\boldsymbol{\sigma}$ definite and thus $|\det(\Sigma)| = 1$ without loss of generality. Moreover consider any sequence $\boldsymbol{x}_k \to \mathring{\boldsymbol{x}}$ and the corresponding $\boldsymbol{z}_k \to \mathring{\boldsymbol{z}}$. Then we must have by continuity also $\Sigma \boldsymbol{z}_k \to \Sigma \mathring{\boldsymbol{z}} > -\varepsilon\, \boldsymbol{e}$ so that already $\Sigma \boldsymbol{z}_k > -\varepsilon\, \boldsymbol{e}$ for all large k. Thus the corresponding $\boldsymbol{g}_k = \boldsymbol{a}_k + \boldsymbol{b}_k^T (\Sigma - L_k)^{-1} Z_k$ belong to $\partial_\varepsilon^L \varphi(\boldsymbol{x}_k)$ and of course their limit is $\mathring{\boldsymbol{g}}$. Thus every element of $\partial_\varepsilon^L \varphi(\mathring{\boldsymbol{x}})$ is the limit of limiting ε-gradients at any sequence converging to $\mathring{\boldsymbol{x}}$. Any $\mathring{\boldsymbol{g}} \in \partial_\varepsilon \varphi(\mathring{\boldsymbol{x}})$ is a convex combination of at most $n+1$ elements of $\partial_\varepsilon^L \varphi(\mathring{\boldsymbol{x}})$. As we have shown above each one of them is the limit of elements of $\partial_\varepsilon^L \varphi(\boldsymbol{x}_k)$ for any given sequence $\boldsymbol{x}_k \to \mathring{\boldsymbol{x}}$. Their convex combinations with the same coefficients belong to $\partial_\varepsilon \varphi(\boldsymbol{x}_k)$, which completes the proof. □

The lemma implies in particular that if any sequence $\boldsymbol{x}_k$ converges to a point $\boldsymbol{x}$ that is ε-stationary, the smallest elements $\operatorname{short}(\partial_\varepsilon \varphi(\boldsymbol{x}_k))$ of $\partial_\varepsilon \varphi(\boldsymbol{x}_k)$ with respect to the Euclidean norm must converge to $\mathbf{0}$. Hence any stopping criterion $\|\operatorname{short}(\partial_\varepsilon^G \varphi(\boldsymbol{x}_k))\| < \delta$ for some positive δ must eventually be satisfied. Let us look at the situation in case of the half pipe function as defined in Eq. (10.1) at origin $\mathring{\boldsymbol{x}} = \mathbf{0}$, where we have

$$\begin{aligned}\partial^K \varphi(0,0) = \{(0,0)\} \subset \partial^L \varphi(0,0) = \{(-1,0),(0,0)\} \\ \subset \liminf_{\varepsilon \to 0} \partial_\varepsilon^G \varphi(0,0) = \liminf_{\varepsilon \to 0} \operatorname{conv}\{(-1,0),(0,2\,x_2) : |x_2| < \varepsilon\}\,.\end{aligned}$$

Here, the Goldstein ε-subdifferential can be computed exactly but that is a very special situation. From now on we only consider *our* limiting ε-differential at a particular convergent sequence.

If we had a sequence $\boldsymbol{x}_k$ that converges to $\mathbf{0} \in \mathbb{R}^2$ all the time staying in the quadratic crescent $\mathcal{S}_{1,1}$ where $0 < x_{k,1}$ and $x_{k,2}^2 > x_{k,1}$ then we have for all k the singleton

$$\partial_\varepsilon^L \varphi(\boldsymbol{x}_k) = \{(-1, 2\,x_{k,2})\} = \partial_\varepsilon^G \varphi(\boldsymbol{x}_k) = \operatorname{short}(\partial_\varepsilon^G \varphi(\boldsymbol{x}_k)).$$

Then the length of $\operatorname{short}(\partial_\varepsilon^G \varphi(\boldsymbol{x}_k))$ would stay constantly greater than 1 despite the convergence to the Clarke stationary and even critical point $\mathring{\boldsymbol{x}} = \mathbf{0}$. On the other hand since $\boldsymbol{z}_k \to \mathbf{0}$, the condition $\Sigma \boldsymbol{z}_k > -\varepsilon\, \boldsymbol{e}$ is eventually satisfied for all $\boldsymbol{\sigma} \in \{-1,1\}^2$ so that the late $\partial_\varepsilon^L \varphi(\boldsymbol{x}_k)$ are given by

$$\begin{aligned}\partial_\varepsilon^L \varphi(\boldsymbol{x}_k) &= \left\{ \boldsymbol{a}_k + \boldsymbol{b}^T \begin{bmatrix} \sigma_1 & 0 \\ \frac{1}{2} & \sigma_2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & 2x_{k,2} \end{bmatrix} \middle| \boldsymbol{\sigma} \in \{-1,1\}^2 \right\} \\ &= \left\{ \boldsymbol{a}_k + \boldsymbol{b}^T \begin{bmatrix} \sigma_1 & 0 \\ -\frac{1}{2}\sigma_1\sigma_2 & \sigma_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & 2x_{k,2} \end{bmatrix} \middle| \boldsymbol{\sigma} \in \{-1,1\}^2 \right\} \\ &= \left\{ \boldsymbol{a}_k + \boldsymbol{b}^T \begin{bmatrix} \sigma_1 & 0 \\ -\frac{1}{2}\sigma_2(\sigma_1+1) & 2\,x_{k,2}\sigma_2 \end{bmatrix} \middle| \boldsymbol{\sigma} \in \{-1,1\}^2 \right\}.\end{aligned}$$

These are actually four different generalized gradients. However as we let $x_{k,2}$ tend to zero we get

$$\lim_{k\to\infty} \partial_\varepsilon^L \varphi(\boldsymbol{x}_k) =$$
$$(-0.25\ 0) + \left\{ (-0.25\ 0.5) \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}, (-0.25,\ 0.5) \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, (-0.25,\ 0.5) \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \right\},$$

so that we get only the two generalized gradients

$$(-0.25,\ 0) + (-0.75,\ 0) = (-1,\ 0) \quad \text{and} \quad (-0.25,\ 0) + (0.25,\ 0) = (0,\ 0).$$

Hence, in this case we have that the limiting differential is indeed contained and thus equal to the inner limit of our limiting ε-differential as a sequence of points converging to the limit $\mathring{\boldsymbol{x}}$. The same relation applies to their convex hulls, which we simply call ε-differentials.

Beyond Mere Gradient Activity The property $\mathbf{0} \in \operatorname{conv}\{\partial^K \varphi(\mathring{\boldsymbol{x}})\}$ of conic stationarity is considerably more restrictive than that of Clarke stationarity, i.e., $\mathbf{0} \in \operatorname{conv}\{\partial^L \varphi(\mathring{\boldsymbol{x}})\}$, which in turn is more restrictive than $\mathbf{0} \in \partial_\varepsilon^G \varphi(\mathring{\boldsymbol{x}})$. However, all are merely depending on which gradients are active at some arbitrary points near the reference point $\mathring{\boldsymbol{x}}$. The relative positions do not matter, which is why $|x|$ and $-|x|$ have the same limiting gradient $\{-1, +1\}$ and generalized gradient as well as ε-differential, namely $[-1, 1]$, respectively. In this case, there is no difference between the conic, Clarke and Goldstein ε-differential. Obviously that is not very useful in the context of optimization, where one wants to distinguish between minimizers and maximizers. To do that one must look at a proper local model function.

10.6 Checking Criticality and Second Order Optimality

It is immediately clear from Eq. (10.10) that $\boldsymbol{x}_* = \mathring{\boldsymbol{x}}$ can only be a local minimizer of φ if it is a local minimizer of $\Delta\varphi(\mathring{\boldsymbol{x}}; \Delta\boldsymbol{x})$ with respect to $\Delta\boldsymbol{x} \approx \mathbf{0}$. We call that first order minimality (FOM). It is not difficult to see that on the function class $C^d_{\text{abs}}(\overline{\mathcal{D}})$ this property is equivalent to criticality as defined in [1] and [2], where $\mathbf{0} \in \mathbb{R}^n$ must be a Fréchet subgradient. The term "criticality" insinuates that critical points of φ should also be critical points of $-\varphi$, which is decidedly not the case. By the sign change first order minimal points turn into first order maximal points, which unfortunately yields the same acronym FOM. So the proper terminology remains to be decided upon.

In general, i.e., for functions outside $C^d_{\text{abs}}(\overline{\mathcal{D}})$ we know of no practical procedure to check a candidate point $\mathring{\boldsymbol{x}}$ for local minimality. Inside $C^d_{\text{abs}}(\overline{\mathcal{D}})$ that can be done by a simple analysis of the abs-linear approximation data $Z, L, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ with $\boldsymbol{c} = \mathring{\boldsymbol{y}}$ obtained from an extended AD tool in the following way. For simplicity, we assume

that for a given point $\boldsymbol{x}_*$ one has $z_i(\boldsymbol{x}_*) = 0$ for $i = 1 \ldots s$. This condition can be relaxed which requires technical reformulations [15]. For this reason we just concentrate on the simple case of full activity also called the localized case. Then one could first check, whether Z has full rank yielding LIKQ. As proven in the same paper, then first order optimality requires that for such a given point $\boldsymbol{x}_*$, there exists a Lagrange multiplier vector $\boldsymbol{\lambda}_* \in \mathbb{R}^s$ such that

$$\boldsymbol{a}^T(\boldsymbol{x}_*, \boldsymbol{0}) + \boldsymbol{\lambda}_*^T Z(\boldsymbol{x}_*, \boldsymbol{0}) = 0 \qquad \text{Tangential Stationarity (TS);} \tag{10.17}$$

$$F(\boldsymbol{x}_*, \boldsymbol{0}) = 0 \qquad \text{Full kink activity; and} \tag{10.18}$$

$$\boldsymbol{b}^T(\boldsymbol{x}_*, \boldsymbol{0}) + \boldsymbol{\lambda}_*^T L(\boldsymbol{x}_*, \boldsymbol{0}) \geq |\boldsymbol{\lambda}_*|^T \qquad \text{Normal Growth (NG)} \tag{10.19}$$

holds. Similar results apply if $\boldsymbol{x}_*$ is not localized in that some of the z_i are nonzero. It is important to note that these optimality conditions can be verified in polynomial time. If they do not hold, it is possible to construct a descent direction from the available derivative information $Z, L, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ as described in [15]. If all component inequalities hold strictly we say that Eq. (10.19) represents strict normal growth.

First order minimality can be ensured for cluster points of the so-called proximal iteration

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \underset{\Delta \boldsymbol{x}}{\operatorname{argmin}} \left\{ \varphi(\boldsymbol{x}_k + \Delta \boldsymbol{x}) + \frac{q}{2} \|\Delta \boldsymbol{x}\|^2 \right\}. \tag{10.20}$$

Here, q can be any positive constant or vary within some interval. The practical challenge for the proximal point concept is that the inner problem of minimizing the right hand side seems almost as hard as the direct minimization of φ.

Before we develop an approximate version where $\varphi(\boldsymbol{x}_k + \Delta \boldsymbol{x})$ is replaced by the more tractable $\Delta\varphi(\boldsymbol{x}_k; \Delta \boldsymbol{x})$ let us briefly look at second order necessary and sufficiency conditions.

Second Order Piecewise Differentiation and Conditions Here, we assume that $\varphi \in C^d_{\text{abs}}(\overline{\mathcal{D}})$ with $d \geq 2$ so that all second order derivatives of F and f are continuous on the respective domains. These derivatives are conventional except that they are only valid on certain subspaces in certain polyhedral domains. We therefore talk of second order piecewise differentiation. Abs-linearization is a form of first order piecewise differentiation but it is more powerful in that it works out the polyhedral decomposition at the same time, which is then relevant for higher order piecewise differentiation. As of now we believe that differentiation on nonpolyhedral domains is impractical. The equalities in our first order condition represent $n + s$ equations in the unknowns $(\boldsymbol{x}_*, \boldsymbol{\lambda}_*)$ whose Jacobian is given by the saddle point matrix

$$\begin{bmatrix} H & Z^T \\ Z & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(n+s)\times(n+s)} \tag{10.21}$$

with

$$H = H(\boldsymbol{x}_*, \boldsymbol{\lambda}_*) \equiv f(\boldsymbol{x}_*, \boldsymbol{0})_{\boldsymbol{xx}} + \Big(\boldsymbol{\lambda}^T F(\boldsymbol{x}_*, \boldsymbol{0})\Big)_{\boldsymbol{xx}} \in \mathbb{R}^{n \times n}.$$

Obviously the Hessian H is the second derivative of the Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{0}, \boldsymbol{\lambda}) = f(\boldsymbol{x}, \boldsymbol{0}) + \boldsymbol{\lambda}^T F(\boldsymbol{x}, \boldsymbol{0}) \tag{10.22}$$

with respect to $\boldsymbol{x}$. The saddle point Jacobian (10.21) is nonsingular provided we have second order sufficiency in that $U^T H U \succ 0$, where the columns of $U \in \mathbb{R}^{n \times (n-s)}$ span the null space of Z. Then we have a sufficient optimality condition in combination with tangential stationarity and strict normal growth. If $\det(U^T H U) = 0$, but the projected Hessian is still positive semidefinite we have a second order necessary condition of optimality. Wherever LIKQ holds the function φ will be smooth within $P_{\boldsymbol{\sigma}}$ but may have kinks (upward or downward) along certain normal directions. As was proven in [15] this geometry corresponds to the $\mathcal{VU}$-decomposition of Mifflin and Sagastizábal [33] and Lewis [29], where the kinks are restricted to point upward. Here we can define at a point $\boldsymbol{x}$ the pair of orthogonal subspaces

$$\mathcal{U}(\boldsymbol{x}) \equiv \text{range}\,(U(\boldsymbol{x})) \quad \text{and} \quad \mathcal{V}(\boldsymbol{x}) \equiv \mathcal{U}(\boldsymbol{x})^T.$$

It should be noted that the $\mathcal{VU}$-decomposition exists for some functions outside $C^d_{\text{abs}}(\overline{\mathcal{D}})$, for example again the Euclidean norm in two variables or more.

Reaching Criticality Based on the abs-linearisation described at the end of Sect. 10.3, the following iterative optimization algorithm was proposed in [13]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \underset{\Delta \boldsymbol{x}}{\text{argmin}} \left\{ \varphi(\boldsymbol{x}_k) + \Delta\varphi(\boldsymbol{x}_k; \Delta\boldsymbol{x}) + \tfrac{q}{2} \|\Delta \boldsymbol{x}\|^2 \right\}. \tag{10.23}$$

We call this approach SALMIN for successive abs-linear minimization. The penalty factor q of the quadratic term is an estimated bound on the discrepancy between φ and its local abs-linear model given by

$$\varphi(\boldsymbol{x}_k) + \Delta\varphi(\boldsymbol{x}_k; \Delta\boldsymbol{x}).$$

This method was shown in [13] to generate a sequence of iterates $\{\boldsymbol{x}_k\}_{k \in \mathbb{N}} \subset \mathbb{R}^n$ whose cluster points are first order minimal. If the inner problem of minimizing the regularized piecewise linear model is not solved exactly, but increments $\Delta\boldsymbol{x}$ that are merely Clarke stationary for $\Delta\varphi$ are accepted also, then the cluster points are guaranteed to be also Clarke stationary as shown in [9].

The SALMIN algorithm as stated in Eq. (10.23) can be interpreted also as a quadratic overestimation method, where the error between the model and the real function is bounded by a power of the distance, see, e.g., [11, 17]. This approach is in

some sense related to a proximal point method as stated in Eq. (10.20). However, in Eq. (10.23) the local abs-linear model of the function to be minimized at the current iterate $\boldsymbol{x}_k$ is used instead of the original function. This makes the solution of the inner optimization problem considerably easier in comparison to the proximal point method. Moreover, without looking at generalized gradients or $\varepsilon-$subdifferentials SALMIN has a very simple stopping criterion. The outer iteration terminates as soon as the objective function reduction promised by the solution of the inner problem falls below a user supplied tolerance.

One possible strategy to solve the inner problem, i.e., determine the minimizer of

$$\underset{\Delta \boldsymbol{x}}{\operatorname{argmin}} \left\{ \varphi(\boldsymbol{x}_k) + \Delta\varphi(\boldsymbol{x}_k; \Delta \boldsymbol{x}) + \tfrac{q}{2} \|\Delta \boldsymbol{x}\|^2 \right\},$$

exploits the polyhedral domain decomposition defined by Eq. (10.12). Starting with an arbitrary initial point and the corresponding polyhedron, one can derive an adapted QOP solver by exploiting the local first order optimality condition from [15] as stated for the localized case (10.18) in Eqs. (10.17) and (10.19). This strategy is based on the computation of stationary points by successively activating and dropping kinks appropriately as described in detail in [16].

10.7 Demonstration on Crescent

The various quantities that we promised as benefits of piecewise differentiation and the convergence behavior of SALMIN are illustrated on the two dimensional Crescent example [3, Nr. 21 in Section 9.1], namely

$$y = f(x_1, x_2) \;=\; \max\{x_1^2 + (x_2 - 1)^2 + x_2 - 1, -x_1^2 - (x_2 - 1)^2 + x_2 + 1\}$$

with the starting point $(-1.5, 2)$. In abs-normal form we can write

$$z_1 = F(x_1, x_2) = x_1^2 + (x_2 - 1)^2 - 1$$

and

$$y = f(x_1, x_2, |z_1|) = x_2 + |z_1|.$$

The new form was achieved by replacing $\max\{u, w\}$ with the equivalent value $\frac{1}{2}[u + w + \operatorname{abs}(w - u)]$ and then canceling various terms, which can of course be done by computer algebra or an AD package. Here one sees immediately that the set of kink locations is formed by the shifted unit circle $x_1^2 + (x_2 - 1)^2 = 1$.

Abs-Linear Approximation With respect to the abs-linear form we note that since there is only one switching variable we must have the trivial strictly lower triangular

matrix $L = 0 \in \mathbb{R}^{1\times 1}$. The remaining parts of the abs-linear form at a point $\mathring{\boldsymbol{x}}$ are given by

$$Z = \partial_{\boldsymbol{x}} z_1 = 2(\mathring{x}_1,\ \mathring{x}_2 - 1),\ \boldsymbol{a} = (0,\ 1)^T,\ b = 1 \ \text{ and}$$
$$c = \mathring{z} = \mathring{x}_1^2 + (\mathring{x}_2 - 1)^2 - 1.$$

The matrix Z has full rank except at the center $(0, 1)$ of the circle. Hence, LIKQ is satisfied everywhere on the kink circle.

Looking for Optimal Points To test for optimality we first look at tangential stationarity, which requires that

$$\mathbf{0} = (0, 1) + \lambda\, 2(x_1, x_2 - 1) \quad \text{and} \quad z_1 = 0.$$

This system of equations has the two solutions solution $(x_1, x_2) = (0, 0)$ with $\lambda = \frac{1}{2}$ and $(x_1, x_2) = (0, 2)$ with $\lambda = -\frac{1}{2}$. The normal growth requires that $b = \partial y/\partial |z_1| \equiv 1 \geq |\lambda| = \frac{1}{2}$ which is satisfied as strict inequality at both points. Thus we have at both points first order optimality, which is also known as criticality. Finally, at $\boldsymbol{x} = (0, 2)$ the null-space of $Z = (0, 2)$ is spanned by $U = (1, 0)^T$ so that the Hessian of the Lagrangian at the first order optimal point $\boldsymbol{x} = (0, 2)$ with $\lambda = -\frac{1}{2}$

$$-\tfrac{1}{2}(1,\ 0)\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = -1 < 0.$$

Here we have used that f is linear and hence its second derivatives vanish completely. Thus, the first order optimal point $(0, 2)$ does not satisfy the second order necessary condition and cannot be a minimizer. At the only other point satisfying tangential stationarity, namely the origin, we have $Z = (0, –2)$ so that with $U = (1, 0)^T$ and the positive Lagrange multiplier one obtains

$$\tfrac{1}{2}(1,\ 0)\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1 > 0.$$

This projected Hessian is positive definite and the origin is therefore a strict local minimizer and thus in fact the one and only global minimizer. Notice that the $\mathcal{VU}$-decomposition is well defined all around the kink circle with $\mathcal{V}$ being the radial direction, i.e., the normal of $z_1 = 0$ and $\mathcal{U}$ the tangential direction. Everywhere the kink is pointed upwards, although that need not be valid in general.

Let us go back and calculate the other goodies at some general point $\mathring{\boldsymbol{x}}$, say the usual starting point $\mathring{\boldsymbol{x}} = (\mathring{\boldsymbol{x}}_1, \mathring{\boldsymbol{x}}_2) = (-1.5, 2)$. There we have $\mathring{z} = \frac{9}{4}$ and thus $\mathring{\boldsymbol{\sigma}} = 1 = \Sigma$. Moreover, $\mathring{Z} = (-3,\ 2)$ so that independently of any preferred

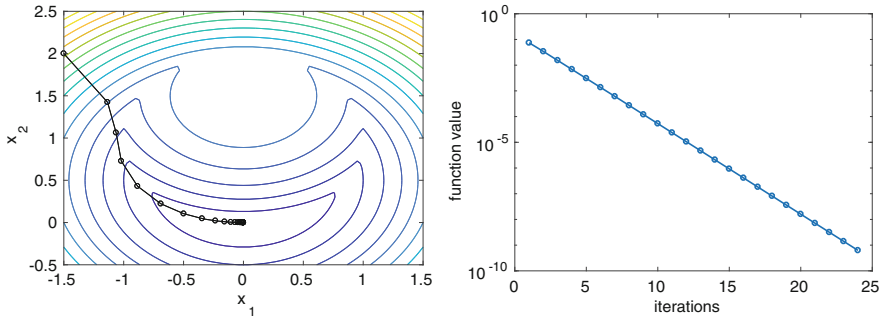

Fig. 10.5 Iterates and function values of SALMIN on Crescent example

direction differentiation yields the gradient

$$\mathring{\boldsymbol{g}} = \boldsymbol{a}^T + \boldsymbol{b}(\Sigma - L)^{-1}Z = (0,1) + 1 \cdot (-3,2) = (-3,3) \quad \Longrightarrow \quad -\mathring{\boldsymbol{g}} = (3,-3).$$

With respect to the limiting ε-subdifferential, according to Eq. (10.16) for $\varepsilon > 0$ we are admitting $\sigma \in \{-1, 1\}$ that satisfy $\sigma\frac{9}{4} > -\varepsilon$. Hence, $\sigma = -1$ and the corresponding

$$\boldsymbol{g}_\sigma = (0,\ 1) + (1)(-1)(-3,\ 2) = (3,\ -1)$$

will only be an ε-gradient when $\varepsilon > \frac{9}{4}$. Obviously that is a strong condition but then the reference point $(-1.5, 2)$ is quite some distance away from the next kink.

Finally, let us consider the performance of our SALMIN approach from the standard starting point with $q = 3$ constant. As one can see on the right hand side of Fig. 10.5 the convergence rate is clearly linear. It can not be better because no effort is made by SALMIN to approximate the curvature term that defines the circular valley.

We have applied earlier versions of SALMIN to most of the academic problems listed in [3]. The results in [9] are quite competitive with a generalization of BFGS [30] and the bundle method [24]. In fact the number of outer iterations is usually significantly smaller and a thorough comparison of the runtime cost of solving the inner problem remains to be done. In any case the inner loop of SALMIN is still undergoing rapid development, especially in view of larger dimensional applications. Another generalization that is under way is the extension to problems with constraints, which may be of complementarity type.

10.8 Covering the Euclidean Norm

In the context of geometric modeling, see, e.g., [32], one may easily think of optimization problems or systems of constraints that involve for $\boldsymbol{u} \in \mathbb{R}^k$ the Euclidean norm

$$||\boldsymbol{u}|| \;=\; \left(\sum_{i=1}^{k} u_i^2\right)^{\frac{1}{2}} \;=\; \|(u_1, \|(u_2, u_3, \ldots, u_k)\|)\|. \tag{10.24}$$

The identity on the right shows that the Euclidean norm in $k > 2$ variables can be expressed recursively in terms of the binary Euclidean norm $\|(u_1, u_2)\| = \sqrt{u_1^2 + u_2^2}$. This elemental function is of course a generalization of our beloved unary absolute value $\|u\| = |u|$ for $u \in \mathbb{R}$. Already the binary Euclidean norm is no longer piecewise differentiable, because at the origin one would need more than finitely many C^1 selection functions to represent it. However it is Lipschitz continuous with constant 1 and almost everywhere differentiable, which one can see directly without referring to Rademacher. As we already foreshadowed at the end of Sect. 10.3 we now consider the extension of $C^d_{\text{abs}}(\overline{\mathcal{D}})$ obtained by allowing not only the univariate abs$(\cdot)$ but its multivariate generalization $\|\cdot\|$.

Problems in $C^d_{\text{euc}}(\overline{\mathcal{D}})$ As main example we consider the simplest so-called location problem [19], which goes back to Fermat in the planar case. Given m distinct *client* points $\boldsymbol{y}_j \in \mathbb{R}^k$ for $j = 1, \ldots, m$ we are looking for a *supply* point $\boldsymbol{x}$ that minimizes the sum of the Euclidean distances to the clients.

$$\min \varphi_m(\boldsymbol{x}) \;\equiv\; \sum_{j=1}^{m} \|\boldsymbol{x} - \boldsymbol{y}_j\| \;\in C^{\infty}_{\text{euc}}(\mathbb{R}^n, \mathbb{R}).$$

The problem is convex, coercive and Lipschitz continuous so that it must have a nonempty compact and convex solution set. Moreover $\varphi_m(\boldsymbol{x})$ is also differentiable except where $\boldsymbol{x} = \boldsymbol{y}_j$ for some $1 \le j \le m$.

When $m = 3$ and thus w.l.o.g $n = 2$ we have the classical Fermat problem, whose solution was be constructed geometrically with a pair of compasses and ruler by Toricelli. Now suppose one has solved the problem in the horizontal plane, i.e., for three points and their geometric median

$$\boldsymbol{y}_1 = (y_{1,1}, y_{1,2}, 0),\;\; \boldsymbol{y}_2 = (y_{2,1}, y_{2,2}, 0),\;\; \boldsymbol{y}_3 = (y_{3,1}, y_{3,2}, 0),\;\; \boldsymbol{w} = (w_1, w_2, 0).$$

The minimizer $\boldsymbol{w}$ is the only stationary point of $\varphi_3(.)$ so that one has

$$\nabla\varphi_3(\boldsymbol{w}) = \boldsymbol{0} \quad \text{and} \quad \varphi_3(\boldsymbol{x}) = \varphi_3(\boldsymbol{w}) + \mathcal{O}(\|\boldsymbol{x} - \boldsymbol{w}\|^2).$$

Now let us add a forth data point $\boldsymbol{y}_4 = (y_{4,1}, y_{4,2}, y_{4,3})$ that is reasonable close to $\boldsymbol{w}$. Then we will have for the new $\varphi_4(.)$ that

$$\varphi_4(\boldsymbol{x}) \;=\; \varphi_3(\boldsymbol{x}) + \|\boldsymbol{x} - \boldsymbol{y}_4\| \;=\; \varphi_3(\boldsymbol{w}) + \|\boldsymbol{x} - \boldsymbol{y}_4\| + \mathcal{O}(\|\boldsymbol{x} - \boldsymbol{w}\|^2)$$

and in particular when $\boldsymbol{y}_4 = \boldsymbol{w}$

$$\varphi_4(\boldsymbol{x}) \;=\; \varphi_3(\boldsymbol{w}) + \|\boldsymbol{x} - \boldsymbol{w}\| + \mathcal{O}(\|\boldsymbol{x} - \boldsymbol{w}\|^2)$$

with $\boldsymbol{x} = \boldsymbol{w}$ as the nonsmooth (global) minimizer of $\varphi_4(.)$. Moreover, since for $\boldsymbol{x} \neq \boldsymbol{y}_4$

$$\nabla\varphi_4(\boldsymbol{x}) \;=\; \nabla\varphi_3(\boldsymbol{x}) + (\boldsymbol{x} - \boldsymbol{y}_4)/\|\boldsymbol{x} - \boldsymbol{y}_4\|$$

the same will be true for all $\boldsymbol{y}_4$ with $\|\nabla\varphi_3(\boldsymbol{y}_4)\| < 1$ and even $\|\nabla\varphi_3(\boldsymbol{y}_4)\| = 1$ since $\varphi_4(.)$ is convex. For each of these $\boldsymbol{y}_4$ we have a convex test problem with the global minimizer $\boldsymbol{x} = \boldsymbol{y}_4$, at which $\varphi_4(\boldsymbol{x})$ is dominated by the Euclidean norm and thus not differentiable.

A very similar situation arises in *compressive sensing* [10] where the distance $\|\boldsymbol{x} - \boldsymbol{y}_4\|$ of the variable vector $\boldsymbol{x}$ to a base point (often $\boldsymbol{y}_4 = \boldsymbol{0}$) is added to a smooth residual, here $\varphi_3(\boldsymbol{x})$. Then the base point is the *sparse* global minimizer as long as the smooth part is comparatively stationary. Only when the smooth part is as steep as the flanks of the norm term the minimizer can be pulled away from the reference points. Now the question arises how this type of problem can be minimized algorithmically. The simplest possible test problem in two variables would be

$$\varphi(x_1, x_2) \;=\; \sqrt{x_1^2 + x_2^2} + (\lambda, 0)^T\boldsymbol{x} \quad \text{with} \qquad |\lambda| \leq 1.$$

We have expressed the Euclidean norm implicitly and assume at first that it will not be recognizable to the optimization algorithm. Lewis and Overton [30] have called this problem the *tilted norm* function, which happens to be the only situation for which they can prove and not only observe the convergence of their BFGS method with a special line search. From a starting point with $x_2 = 0$, steepest descent with any kind of line search will behave exactly as on the univariate problem $|x| + \lambda x$. In our experience steepest descent with a Armijo type line-search stalls completely in the vicinity of the optimizer. Lewis and Overton have shown theoretically that in combination with their specially for nonsmooth problems adapted Armijo line-search, steepest descent exhibits a sublinear convergence rate in terms of the number of function evaluations.

In the one dimensional case with $\mathrm{abs}(x) = |\cdot|$ identified as such, our SALMIN approach would of course yield convergence from any initial point in one step. On the two dimensional problem without any hint of nonsmooth elementals it would behave like steepest descent with the coefficient q being incremented several times in each line search. Theoretically the fact of convergence can be deduced

by contradiction as follows. If there was a ball about $\boldsymbol{y}_4$ which was not reached by anyone of the iterates one could modify the convex function $\varphi_4(.)$ inside such that the cone singularity is smoothened out. Then our standard convergence theory would ensure convergence into the ball yielding a contradiction. Note that the other three points of nondifferentiability have a much higher function value and can therefore not be approached if the iteration is started below. Also, notice that we have assumed throughout like in [30] that the single point of nondifferentiability, i.e., the global minimizer itself is never reached exactly by any iterate. Of course a small fixed stepsize as is popular in machine learning will ultimately lead to oscillations back and forth across the base point. One might argue that the solution error may then be quite small during this chattering, but the whole purpose of these terms is to drive them exactly to zero and thus to achieve data sparsity.

So, en passant, we reach the tentative conclusion that on machine learning problems similar to Lasso [37] steepest descent converges sublinearly with linesearch and does not converge at all for a fixed stepsize. Obviously, some thing needs to be done to overcome this impasse.

Clipped Root Linearization We have seen in the previous section that approximating the Euclidean norm by its tangent plane (and equivalently the square root by its tangent line) does not yield good results on the kind of optimization problems in $C^d_{\rm euc}(\overline{\mathcal{D}})$. As it turns out the two approximation tasks are intimately related and by simply making a small modification to the root linearization we obtain a desired effect for the Euclidean root. Therefor we will go backward and start with the root, whose normal incremental linearization is given by

$$v = \sqrt{u} \quad \Longrightarrow \quad v + \Delta v = v + 0.5\Delta u/v \quad \Longleftrightarrow \quad \Delta v = 0.5\,\Delta u/v \tag{10.25}$$

with the tacit assumption that u and thus v are not exactly equal to zero. This propagation happens automatically under the rug when piecewise linearization is applied to a function evaluation procedure $y = \varphi(\boldsymbol{x})$. The value u and the increment Δu of the right hand side are computed from $\boldsymbol{x}$ and $\Delta \boldsymbol{x}$ via the preceding intermediate operations and Δv is the resulting increment of the left hand side. In other words the root is treated like any other differentiable univariate function, namely replaced by its tangent line. In contrast to the root itself, which is undefined for negative values, the linearization reaches arbitrarily large negative values. Therefore, one might argue that the user should be alerted in some way to the qualitative change for negative increments Δu much bigger than $u > 0$. The simple idea of taking the absolute value of the linear prediction leads to:

Definition 10.5 (Clipped Linearization) The *clipped linearization* of the root is given by

$$\begin{aligned} v = \mathrm{abs}(\sqrt{u}) \quad &\Longrightarrow \quad v + \Delta v = |v + 0.5\Delta u/v| \\ &\Longleftrightarrow \quad \Delta v = |v + 0.5\Delta u/v| - v. \end{aligned} \tag{10.26}$$

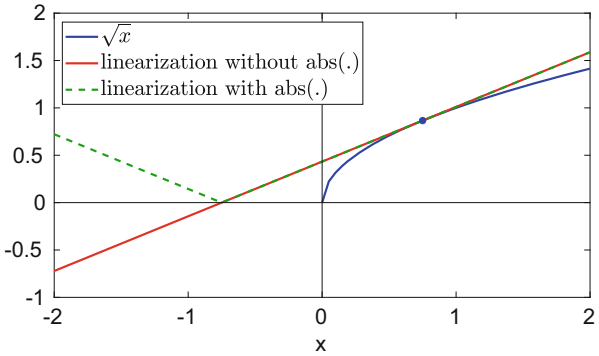

Fig. 10.6 Two different linearizations for the square root at $\mathring{\boldsymbol{x}} = 0.75$

Furthermore, we will call this technique of maintaining non-negativity or other bounds of the original elemental by its piecewise linearization as *clipping*.

Of course, while maintaining characteristic properties of the original elemental we have introduced an extra kink and thus made the piecewise linear model a bit more complicated. However, we will assume that there are lots of kinks anyhow so that a few more do not make a significant difference. The linear and clipped approximation of the square root are depicted in Fig. 10.6.
The straight tangent line has been replace by a V-shaped line touching the horizontal axis at $\Delta u = -2v$. The nice thing here is that one does not have to change anything in the evaluation procedure except extending all $\sqrt{u}$ to $\text{abs}(\sqrt{u})$, which is of course equivalent as far as the values themselves (but not the increments) are concerned. The piecewise linearization process by ADOL-C or some other abs-extended AD tool can then proceed as usual. Now the question is what that mechanism does to the Euclidean norm. The usual differentiation of the Euclidean norm of $\boldsymbol{u} \in \mathbb{R}^k$ in the composite form (10.24) gives the linear approximation

$$v = ||\boldsymbol{u}|| \Longrightarrow v + \Delta v = (\boldsymbol{u} + \Delta \boldsymbol{u})^T \boldsymbol{u}/\|\boldsymbol{u}\| \quad \Longleftrightarrow \quad \Delta v = \Delta \boldsymbol{u}^T \boldsymbol{u}/\|\boldsymbol{u}\|$$

again tacitly assuming that $\boldsymbol{u} \neq \boldsymbol{0}$ and equivalently $v \neq 0$. Now if again we extend $\|\boldsymbol{u}\|$ to $\text{abs}(\|\boldsymbol{u}\|)$ we get after some manipulations

$$\begin{aligned} v = \text{abs}(\|\boldsymbol{u}\|) \quad &\Longrightarrow \quad v + \Delta v = |(\boldsymbol{u} + \Delta \boldsymbol{u})^T \boldsymbol{u}|/\|\boldsymbol{u}\| \\ &\Longleftrightarrow \quad \Delta v = |(\boldsymbol{u} + \Delta \boldsymbol{u})^T \boldsymbol{u}|/\|\boldsymbol{u}\| - v. \end{aligned} \tag{10.27}$$

This approximation of the Euclidean norm is a V-shaped valley whose bottom line is orthogonal to the reference point $\boldsymbol{u}$ as illustrated in Fig. 10.7 Again there is no need for any substantial recoding but simply one has to extend all $v = ||\boldsymbol{u}||$ to $v = \text{abs}(\|\boldsymbol{u}\|)$ or even simpler use the expression (10.24) and extend $\sqrt{u}$ to $|\sqrt{u}|$ as suggested above.

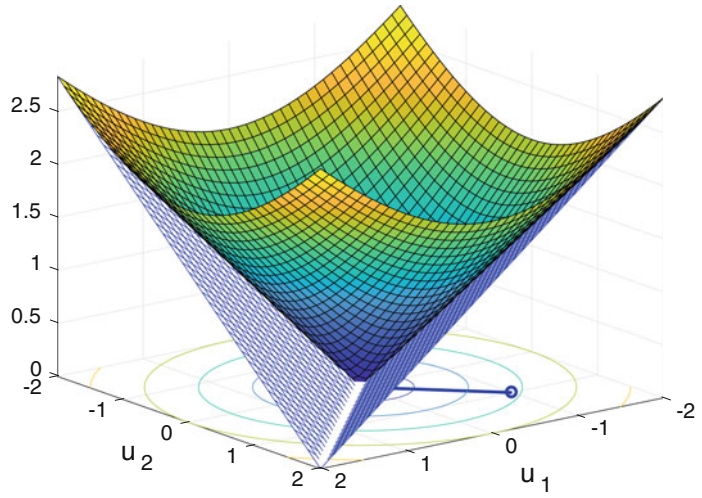

Fig. 10.7 The Euclidean norm $\|\boldsymbol{u}\|$ and its clipped linearization at $\mathring{\boldsymbol{u}} = (-1, 1)$

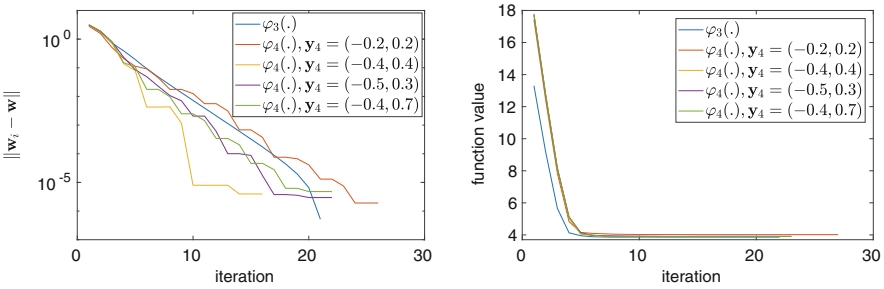

Fig. 10.8 Results of SALMIN with clipped square root for the Weber problem

We glossed a little bit about possible overflows when the scalar u or the vector $\boldsymbol{u}$ are small. Moreover, there is one important aspect that we have not mentioned. Namely the valley linearization of the Euclidean norm is not of second order and hence the generalized Taylor property (10.10) does no longer hold for the over-all abs-linearization. Of course, one might hope that does not stop whatever algorithm one is using from converging, albeit at a possibly reduced rate. Specifically applying the current version of SALMIN without any modifications to the location problem also called Weber problem [40] as described above we get the linear convergence behavior displayed in Fig. 10.8. The minimizer of $\varphi_3(.)$ with $\boldsymbol{y}_1 = (1, 1)$, $\boldsymbol{y}_2 = (-1, -1)$ and $\boldsymbol{y}_3 = (-1, 1)$ is $\boldsymbol{w} = (-0.577, 0.577, 0)$. Notice that the iteration appears to alternate between a step that barely reduces the distance to the solution, presumably moving along the bottom of the valley approximation and one that reduced the distance by a about a quarter. These numerical results appear quite satisfactory in view of the observation that normal descent methods are almost

certain to have a sublinear rate, which in the presence of rounding errors means stalling not all that close to a solution. To the best of our knowledge the clipped versions (10.26) and (10.27) of the piecewise linearization of root and Euclidean norm have not yet appeared in the literature.

10.9 Summary and Conclusion

As indicated by the title we tried to sow some doubts regarding the plausibility of the popular "oracle" scenario, i.e., the availability of the function value and one generalized gradient. The key claim is that, if there is a way to compute a vector that is guaranteed to be a generalized gradient then one can apply piecewise differentiation and obtains lots of other goodies, like directionally active gradients, critical multipliers, approximating separating planes, conically active generalized gradients, active $\varepsilon-$gradients and the whole local abs-linear approximation in the form of two matrices and three vectors. That full local model naturally leads to the SALMIN method for which there is now an extensive theory [9] and [16]. The $\varepsilon-$active gradients defined by (10.16) were firstly introduced in this paper and their relation to the classical $\varepsilon-$differential of Goldstein deserves further exploration. They certainly have the advantage of practical computability with polynomial effort. Also the clipped linearisation as defined in Definition 10.5 for the root is proposed here for the first time.

It remains to be seen, which class of problems are efficiently treatable by piecewise differentiation or not. In the penultimate section we looked at the extension of $C^d_{\text{abs}}(\overline{\mathcal{D}})$ to $C^d_{\text{euc}}(\overline{\mathcal{D}})$ by generalization of the absolute value to the Euclidean norm in two and thus arbitrary many variables. It is found that the piecewise linearization of the norm by a V-shaped valley rather than just its tangent plane appears very useful for avoiding the sublinear convergence of classical descent methods. The concept of abs-linearization is also extendable to reflexive Banach spaces and thus the optimization under PDE constraints. Finally let us remark that the abs-linear approximation can also be exploited for other fundamental numerical tasks like the solution of nonlinear systems and the integration of Lipschitz continuous dynamical systems.

References

1. Absil, P.A., Mahony, R., Andrews, B.: Convergence of the iterates of descent methods for analytic cost functions. SIAM J. Optim. **16**(2), 531–547 (2005)
2. Attouch, H., Bolte, J.: On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. Math. Program. **116**(1–2, Ser. B), 5–16 (2009)
3. Bagirov, A., Karmitsa, N., Mäkelä, M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, Berlin (2014)

4. Barton, P., Khan, K., Stechlinski, P., Watson, H.: Computationally relevant generalized derivatives: theory, evaluation and applications. Optim. Methods Softw. **33**(4–6), 1030–1072 (2018)
5. Bell, B.: CppAD. https://www.coin-or.org/CppAD/
6. Burke, J., Henrion, D., Lewis, A., Overton, M.: Stabilization via nonsmooth, nonconvex optimization. IEEE Trans. Autom. Control **51**(11), 1760–1769 (2006)
7. Clarke, F.: Optimization and Nonsmooth Analysis. SIAM, Philadelphia (1990)
8. Facchinei, F., Pang, J.S.: Finite-Dimensional Variational Inequalities and Complementarity Problems, vol. II. Springer, Berlin (2003)
9. Fiege, S., Walther, A., Griewank, A.: An algorithm for nonsmooth optimization by successive piecewise linearization. Math. Program. Ser. A (2018). https://doi.org/10.1007/s10107-018-1273-5
10. Foucart, S., Rauhut, H.: A Mathematical Introduction to Compressive Sensing. Birkhäuser/Springer, Basel (2013)
11. Griewank, A.: The modification of Newton's method for unconstrained optimization by bounding cubic terms. Technical Report NA/12, University of Cambridge (1981)
12. Griewank, A.: Automatic directional differentiation of nonsmooth composite functions. In: Recent Developments in Optimization. 7th French-German Conference on Optimization, Dijon, June 27–July 2, 1994, pp. 155–169. Springer, Berlin (1995)
13. Griewank, A.: On stable piecewise linearization and generalized algorithmic differentiation. Optim. Methods Softw. **28**(6), 1139–1178 (2013)
14. Griewank, A., Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia (2008)
15. Griewank, A., Walther, A.: First and second order optimality conditions for piecewise smooth objective functions. Optim. Methods Softw. **31**(5), 904–930 (2016)
16. Griewank, A., Walther, A.: Finite convergence of an active signature method to local minima of piecewise linear functions. Optim. Methods Softw. **34**, 1035–1055 (2018). https://doi.org/10.1080/10556788.2018.1546856
17. Griewank, A., Fischer, J., Bosse, T.: Cubic overestimation and secant updating for unconstrained optimization of $C^{2,1}$ functions. Optim. Methods Softw. **29**(5), 1075–1089 (2014)
18. Griewank, A., Streubel, T., Lehmann, L., Radons, M., Hasenfelder, R.: Piecewise linear secant approximation via algorithmic piecewise differentiation. Optim. Methods Softw. **33**(4–6), 1108–1126 (2018)
19. Hamacher, H., Nickel, S.: Classification of location models. Locat. Sci. **6**(1–4), 229–242 (1998)
20. Hascoët, L., Pascual, V.: The Tapenade automatic differentiation tool: principles, model, and specification. ACM Trans. Math. Softw. **39**(3), 20:1–20:43 (2013)
21. Hegerhorst-Schultchen, L., Kirches, C., Steinbach, M.: On the relation between mpecs and optimization problems in abs-normal form. Technical Report, Universität Hannover (2018). Available at optimization-online
22. Hintermüller, M.: Semismooth newton methods and applications. Technical Report, Department of Mathematics, Humboldt-University Berlin (2010)
23. Hintermüller, M., Stadler, G.: A semi-smooth Newton method for constrained linear-quadratic control problems. ZAMM. Z. Angew. Math. Mech. **83**(4), 219–237 (2003)
24. Karmitsa, N., Mäkelä, M.: Limited memory bundle method for large bound constrained nonsmooth optimization: convergence analysis. Optim. Methods Softw. **25**(6), 895–916 (2010)
25. Khan, K.: Branch-locking ad techniques for nonsmooth composite functions and nonsmooth implicit functions. Optim. Methods Softw. (2017). https://doi.org/10.1080/10556788.2017.1341506
26. Khan, K., Barton, P.: Evaluating an element of the Clarke generalized Jacobian of a composite piecewise differentiable function. ACM Trans. Math. Softw. **39**(4), 28 (2013)
27. Khan, K., Barton, P.: A vector forward mode of automatic differentiation for generalized derivative evaluation. Optim. Methods Softw. **30**(6), 1185–1212 (2015)
28. Klatte, D., Kummer, B.: Nonsmooth equations in optimization. In: Regularity, Calculus, Methods and Applications. Kluwer Academic Publishers, Amsterdam (2002)

29. Lewis, A.: Active sets, nonsmoothness, and sensitivity. SIAM J. Optim. **13**(3), 702–725 (2002)
30. Lewis, A., Overton, M.: Nonsmooth optimization via quasi-Newton methods. Math. Program. **141**(1–2, Ser. A), 135–163 (2013)
31. Leyffer, S.: Mathematical programs with complementarity constraints. SIAG/OPTViews-and-News **14**(1), 15–18 (2003)
32. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. SIAM Rev. **56**(1), 3–69 (2014)
33. Mifflin, R., Sagastizábal, C.: A science fiction story in nonsmooth optimization originating at IIASA. Doc. Math. **Extra Vol**, 291–300 (2012)
34. Nesterov, Y.: Lexicographic differentiation of nonsmooth functions. Math. Program. **104**(2–3, Ser. A), 669–700 (2005)
35. Rockafellar, R., Wets, R.B.: Variational Analysis. Springer, Berlin (1998)
36. Scholtes, S.: Introduction to Piecewise Differentiable Functions. Springer, Berlin (2012)
37. Tibshirani, R.: Regression shrinkage and selection via the LASSO. J. R. Stat. Soc. Ser. B **58**(1), 267–288 (1996)
38. Walther, A., Griewank, A.: Combinatorial Scientific Computing, Chapter Getting Started with ADOL-C, pp. 181–202. Chapman-Hall CRC Computational Science, Boca Raton (2012)
39. Walther, A., Griewank, A.: Characterizing and testing subdifferential regularity for piecewise smooth objective functions. SIAM J. Optim. **29**(2), 1473–1501 (2019)
40. Weber, A.: Über den Standort der Industrien. J.C.B. Mohr (Paul Siebeck), Tübingen (1922)

Chapter 11
Numerical Solution of Generalized Minimax Problems

Ladislav Lukšan, Ctirad Matonoha, and Jan Vlček

Abstract This contribution contains the description and investigation of three numerical methods for solving generalized minimax problems. These problems consists in the minimization of nonsmooth functions which are compositions of special smooth convex functions with maxima of smooth functions. The most important functions of this type are the sums of maxima of smooth functions. Section 11.2 is devoted to primal interior point methods which use solutions of nonlinear equations for obtaining minimax vectors. Section 11.3 contains investigation of smoothing methods, based on using exponential smoothing terms. Section 11.4 contains short description of primal-dual interior point methods based on transformation of generalized minimax problems to general nonlinear programming problems. Finally the last section contains results of numerical experiments.

11.1 Generalized Minimax Problems

In many practical problems we need to minimize functions that contain absolute values or pointwise maxima of smooth functions. Such functions are nonsmooth but they often have a special structure enabling the use of special methods that are more efficient than methods for minimization of general nonsmooth functions. The classical minimax problem, where $F(\boldsymbol{x}) = \max_{1 \le k \le m} f_k(\boldsymbol{x})$, or problems where the function to be minimized is a nonsmooth norm, e.g. $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_\infty$, $F(\boldsymbol{x}) = \|f_+(\boldsymbol{x})\|_\infty$, $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_1$, $F(\boldsymbol{x}) = \|f_+(\boldsymbol{x})\|_1$ with $f(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]^T$ and $f_+(\boldsymbol{x}) = [\max\{f_1(\boldsymbol{x}), 0\}, \ldots, \max\{f_m(\boldsymbol{x}), 0\}]^T$, are typical examples. Such functions can be considered as special cases of more general functions, so it is possible to formulate more general theories and construct more general numerical methods. One possibility for generalization of the classical

L. Lukšan (✉) · C. Matonoha · J. Vlček
Institute of Computer Science, The Czech Academy of Sciences, Prague, Czech Republic
e-mail: luksan@cs.cas.cz; matonoha@cs.cas.cz; vlcek@cs.cas.cz

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_11

minimax problem consists in the use of the function

$$F(\boldsymbol{x}) = \max_{1 \le k \le \overline{k}} \boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}), \tag{11.1}$$

where $\boldsymbol{p}_k \in \mathbb{R}^m$, $1 \le k \le \overline{k}$, and $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is a smooth mapping. This function is a special case of composite nonsmooth functions of the form $F(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \max_{1 \le k \le \overline{k}}(\boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}) + b_k)$, where $f_0 : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function [8, Section 14.1].

Remark 11.1 We can express all above mentioned minimax problems and nonsmooth norms in form (11.1).

(a) Setting $\boldsymbol{p}_k = \boldsymbol{e}_k$, where $\boldsymbol{e}_k$ is the k-th column of an identity matrix and $\overline{k} = m$, we obtain $F(\boldsymbol{x}) = \max_{1 \le k \le m} f_k(\boldsymbol{x})$ (the classical minimax).
(b) Setting $\boldsymbol{p}_k = \boldsymbol{e}_k$, $\boldsymbol{p}_{m+k} = -\boldsymbol{e}_k$ and $\overline{k} = 2m$, we obtain $F(\boldsymbol{x}) = \max_{1 \le k \le m} \max\{f_k(\boldsymbol{x}), -f_k(\boldsymbol{x})\} = \|f(\boldsymbol{x})\|_\infty$.
(c) Setting $\boldsymbol{p}_k = \boldsymbol{e}_k$, $\boldsymbol{p}_{m+1} = \boldsymbol{0}$ and $\overline{k} = m + 1$, we obtain $F(\boldsymbol{x}) = \max\{\max_{1 \le k \le m} f_k(\boldsymbol{x}), 0\} = \|f(\boldsymbol{x})_+\|_\infty$.
(d) If $\overline{k} = 2^m$ and $\boldsymbol{p}_k$, $1 \le k \le 2^m$, are mutually different vectors whose elements are either 1 or -1, we obtain $F(\boldsymbol{x}) = \sum_{k=1}^m \max\{f_k(\boldsymbol{x}), -f_k(\boldsymbol{x})\} = \|f(\boldsymbol{x})\|_1$.
(e) If $\overline{k} = 2^m$ and $\boldsymbol{p}_k$, $1 \le k \le 2^m$, are mutually different vectors whose elements are either 1 or 0, we obtain $F(\boldsymbol{x}) = \sum_{k=1}^m \max\{f_k(\boldsymbol{x}), 0\} = \|f_+(\boldsymbol{x})\|_1$.

Remark 11.2 Since the mapping $\boldsymbol{f}(\boldsymbol{x})$ is continuously differentiable, the function (11.1) is Lipschitz. Thus, if the point $\boldsymbol{x} \in \mathbb{R}^n$ is a local minimum of $F(\boldsymbol{x})$, then $\boldsymbol{0} \in \partial F(\boldsymbol{x})$ [25, Theorem 3.2.5] holds. According to [25, Theorem 3.2.13], one has

$$\partial F(\boldsymbol{x}) = (\nabla \boldsymbol{f}(\boldsymbol{x}))^T \operatorname{conv}\left\{\boldsymbol{p}_k : k \in \bar{\mathcal{I}}(\boldsymbol{x})\right\},$$

where $\bar{\mathcal{I}}(\boldsymbol{x}) = \{k \in \{1, \ldots, \overline{k}\} : \boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}) = F(\boldsymbol{x})\}$. Thus, if the point $\boldsymbol{x} \in \mathbb{R}^n$ is a local minimum of $F(\boldsymbol{x})$, then multipliers $\lambda_k \ge 0$, $1 \le k \le \overline{k}$, exist, such that $\lambda_k(\boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}) - F(\boldsymbol{x})) = 0$, $1 \le k \le \overline{k}$,

$$\sum_{k=1}^{\overline{k}} \lambda_k = 1 \quad \text{and} \quad \sum_{k=1}^{\overline{k}} \lambda_k J^T(\boldsymbol{x}) \boldsymbol{p}_k = \boldsymbol{0},$$

where $J(\boldsymbol{x})$ is a Jacobian matrix of the mapping $\boldsymbol{f}(\boldsymbol{x})$.

Remark 11.3 It is clear that a minimum of function (11.1) is a solution of a nonlinear programming problem consisting in minimization of a function $\tilde{F} : \mathbb{R}^{n+1} \to \mathbb{R}$, where $\tilde{F}(\boldsymbol{x}, z) = z$, on the set

$$\mathcal{C} = \{(\boldsymbol{x}, z) \in \mathbb{R}^{n+1} : \boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}) \le z,\ 1 \le k \le \overline{k}\}.$$

Denoting $c_k(\boldsymbol{x}, z) = \boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}) - z$ and $\boldsymbol{a}_k = \nabla c_k(\boldsymbol{x}, z)$, $1 \le k \le \overline{k}$, we obtain $\boldsymbol{a}_k = [\boldsymbol{p}_k^T J(\boldsymbol{x}), -1]^T$ and $\boldsymbol{g} = \nabla \tilde{F}(\boldsymbol{x}, z) = [\boldsymbol{0}^T, 1]^T$, so the necessary KKT conditions can be written in the form

$$\begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix} + \sum_{k=1}^{\overline{k}} \begin{bmatrix} J^T(\boldsymbol{x}) \boldsymbol{p}_k \\ -1 \end{bmatrix} \lambda_k = \boldsymbol{0},$$

$\lambda_k(\boldsymbol{p}_k^T \boldsymbol{f}(\boldsymbol{x}) - z) = 0$, where $\lambda_k \ge 0$, $1 \le k \le \overline{k}$, are the Lagrange multipliers and $z = F(\boldsymbol{x})$. Thus, we obtain the same necessary conditions for an extremum as in Remark 11.2.

From the examples given in Remark 11.1 it follows that composite nondifferentiable functions are not suitable for representation of the functions $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_1$ and $F(\boldsymbol{x}) = \|f_+(\boldsymbol{x})\|_1$ because in this case the expression on the right-hand side of (11.1) contains 2^m elements with vectors $\boldsymbol{p}_k$, $1 \le k \le 2^m$. In the subsequent considerations, we will choose a somewhat different approach. We will consider generalized minimax functions established in [5] and [23].

Definition 11.1 We say that $F : \mathbb{R}^n \to \mathbb{R}$ is a generalized minimax function if

$$F(\boldsymbol{x}) = h(F_1(\boldsymbol{x}), \ldots, F_m(\boldsymbol{x})), \quad F_k(\boldsymbol{x}) = \max_{1 \le l \le m_k} f_{kl}(\boldsymbol{x}), \quad 1 \le k \le m, \tag{11.2}$$

where $h : \mathbb{R}^m \to \mathbb{R}$ and $f_{kl} : \mathbb{R}^n \to \mathbb{R}$, $1 \le k \le m$, $1 \le l \le m_k$, are smooth functions satisfying the following assumptions.

Assumption 11.1 *Functions f_{kl}, $1 \le k \le m$, $1 \le l \le m_k$, are bounded from below on $\mathbb{R}^n$, so that there exists a constant $\underline{F} \in \mathbb{R}$ such that $f_{kl}(\boldsymbol{x}) \ge \underline{F}$, $1 \le k \le m$, $1 \le l \le m_k$, for all $\boldsymbol{x} \in \mathbb{R}^n$.*

Assumption 11.2 *Functions F_k, $1 \le k \le m$, are bounded from below on $\mathbb{R}^n$, so that there exist constants $\underline{F}_k \in \mathbb{R}$ such that $F_k(\boldsymbol{x}) \ge \underline{F}_k$, $1 \le k \le m$, for all $\boldsymbol{x} \in \mathbb{R}^n$.*

Assumption 11.3 *The function h is twice continuously differentiable and convex satisfying*

$$0 < \underline{h}_k \le \frac{\partial}{\partial z_k} h(\boldsymbol{z}) \le \overline{h}_k, \quad 1 \le k \le m, \tag{11.3}$$

for every $\boldsymbol{z} \in \mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^m : z_k \ge \underline{F}_k,\ 1 \le k \le m\}$ (vector $\boldsymbol{z} \in \mathbb{R}^m$ is called the minimax vector).

Assumption 11.4 *Functions $f_{kl}(\boldsymbol{x})$, $1 \le k \le m$, $1 \le l \le m_k$, are twice continuously differentiable on the convex hull of the level set*

$$\mathcal{D}_F(\overline{F}) = \{\boldsymbol{x} \in \mathbb{R}^n : F_k(\boldsymbol{x}) \le \overline{F},\ 1 \le k \le m\}$$

for a sufficiently large upper bound $\overline{F}$ and subsequently, constants $\overline{g}$ and $\overline{G}$ exist such that $\|\boldsymbol{g}_{kl}(\boldsymbol{x})\| \le \overline{g}$ and $\|G_{kl}(\boldsymbol{x})\| \le \overline{G}$ for all $1 \le k \le m$, $1 \le l \le m_k$, and $\boldsymbol{x} \in \operatorname{conv} \mathcal{D}_F(\overline{F})$, where $\boldsymbol{g}_{kl}(\boldsymbol{x}) = \nabla f_{kl}(\boldsymbol{x})$ and $G_{kl}(\boldsymbol{x}) = \nabla^2 f_{kl}(\boldsymbol{x})$.

Remark 11.4 The conditions imposed on the function $h(\boldsymbol{z})$ are relatively strong but many important nonsmooth functions satisfy them.

(1) Let $h : \mathbb{R} \to \mathbb{R}$ be an identity mapping, so $h(z) = z$ and $h'(z) = 1 > 0$. Then setting $\overline{k} = 1, m_1 = \overline{l}$ and

$$F(\boldsymbol{x}) = h(F_1(\boldsymbol{x})) = F_1(\boldsymbol{x}) = \max_{1 \le l \le \overline{l}} \boldsymbol{p}_l^T \boldsymbol{f}(\boldsymbol{x})$$

(since $f_{1l} = \boldsymbol{p}_l^T \boldsymbol{f}(\boldsymbol{x})$), we obtain the composite nonsmooth function (11.1) and therefore the functions $F(\boldsymbol{x}) = \max_{1 \le k \le m} f_k(\boldsymbol{x})$, $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_\infty$, $F(\boldsymbol{x}) = \|f_+(\boldsymbol{x})\|_\infty$.

(2) Let $h : \mathbb{R}^m \to \mathbb{R}$, where $h(\boldsymbol{z}) = z_1 + \cdots + z_m$, so $\frac{\partial}{\partial z_k} h(\boldsymbol{z}) = 1 > 0, 1 \le k \le m$. Then function (11.2) has the form

$$F(\boldsymbol{x}) = \sum_{k=1}^{m} F_k(\boldsymbol{x}) = \sum_{k=1}^{m} \max_{1 \le l \le m_k} f_{kl}(\boldsymbol{x}) \tag{11.4}$$

(the sum of maxima). If $m_k = 2$ and $F_k(\boldsymbol{x}) = \max\{f_k(\boldsymbol{x}), -f_k(\boldsymbol{x})\}$, we obtain the function $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_1$. If $m_k = 2$ and $F_k(\boldsymbol{x}) = \max\{f_k(\boldsymbol{x}), 0\}$, we obtain the function $F(\boldsymbol{x}) = \|f_+(\boldsymbol{x})\|_1$. It follows that the expression of functions $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_1$ and $F(\boldsymbol{x}) = \|f_+(\boldsymbol{x})\|_1$ by (11.2) contains only m summands and each summand is a maximum of two function values. Thus, this approach is much more economic than the use of formulas stated in Remark 11.1(d)–(e).

Remark 11.5 Since the functions $F_k(\boldsymbol{x})$, $1 \le k \le m$, are regular [25, Theorem 3.2.13], the function $h(\boldsymbol{z})$ is continuously differentiable, and $h_k = \frac{\partial}{\partial z_k} h(\boldsymbol{z}) > 0$, one can write [25, Theorem 3.2.9]

$$\partial F(\boldsymbol{x}) = \operatorname{conv} \sum_{k=1}^{m} h_k \partial F_k(\boldsymbol{x}) = \sum_{k=1}^{m} h_k \partial F_k(\boldsymbol{x})$$

$$= \sum_{k=1}^{m} h_k \operatorname{conv}\{\boldsymbol{g}_{kl}(\boldsymbol{x}) : l \in \bar{\mathcal{I}}_k(\boldsymbol{x})\},$$

where $\bar{\mathcal{I}}_k(\boldsymbol{x}) = \{l : 1 \le l \le m_k,\ f_{kl}(\boldsymbol{x}) = F_k(\boldsymbol{x})\}$. Thus, one has

$$\partial F(\boldsymbol{x}) = \sum_{k=1}^{m} h_k \sum_{l=1}^{m_k} \lambda_{kl} \boldsymbol{g}_{kl}(\boldsymbol{x}),$$

where for $1 \le k \le m$ it holds $\lambda_{kl} \ge 0$, $\lambda_{kl}(F_k(\boldsymbol{x}) - f_{kl}(\boldsymbol{x})) = 0$, $1 \le l \le m_k$, and $\sum_{l=1}^{m_k} \lambda_{kl} = 1$. Setting $u_{kl} = h_k \lambda_{kl}$, $1 \le k \le m$, $1 \le l \le m_k$, we can write

$$\partial F(\boldsymbol{x}) = \sum_{k=1}^{m} \sum_{l=1}^{m_k} u_{kl} \boldsymbol{g}_{kl}(\boldsymbol{x}),$$

where for $1 \le k \le m$ it holds $u_{kl} \ge 0$, $u_{kl}(F_k(\boldsymbol{x}) - f_{kl}(\boldsymbol{x})) = 0$, $1 \le l \le m_k$, and $\sum_{l=1}^{m_k} u_{kl} = h_k$. If a point $\boldsymbol{x} \in \mathbb{R}^n$ is a minimum of a function $F(\boldsymbol{x})$, then $\boldsymbol{0} \in \partial F(\boldsymbol{x})$, so there exist multipliers u_{kl}, $1 \le k \le m$, $1 \le l \le m_k$, such that

$$\sum_{k=1}^{m} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl} = \boldsymbol{0}, \quad \sum_{l=1}^{m_k} u_{kl} = h_k, \quad h_k = \frac{\partial}{\partial z_k} h(z), \tag{11.5}$$

$$u_{kl} \ge 0, \quad F_k(\boldsymbol{x}) - f_{kl}(\boldsymbol{x}) \ge 0, \quad u_{kl}(F_k(\boldsymbol{x}) - f_{kl}(\boldsymbol{x})) = 0. \tag{11.6}$$

Remark 11.6 Unconstrained minimization of function (11.2) is equivalent to the nonlinear programming problem

$$\begin{cases} \text{minimize} & \tilde{F}(\boldsymbol{x}, z) = h(z) \\ \text{subject to} & f_{kl}(\boldsymbol{x}) \le z_k, \quad 1 \le k \le m, \quad 1 \le l \le m_k. \end{cases} \tag{11.7}$$

The condition (11.3) is sufficient for satisfying equalities $z_k = F_k(\boldsymbol{x})$, $1 \le k \le m$, at the minimum point. Denoting $c_{kl}(\boldsymbol{x}, z) = f_{kl}(\boldsymbol{x}) - z_k$ and $\boldsymbol{a}_{kl}(\boldsymbol{x}, z) = \nabla c_{kl}(\boldsymbol{x}, z)$, $1 \le k \le m$, $1 \le l \le m_k$, we obtain $\boldsymbol{a}_{kl}(\boldsymbol{x}, z) = [\boldsymbol{g}_{kl}^T(\boldsymbol{x}), -\boldsymbol{e}_k^T]^T$, where $\boldsymbol{g}_{kl}(\boldsymbol{x})$ is a gradient of $f_{kl}(\boldsymbol{x})$ in $\boldsymbol{x}$ and $\boldsymbol{e}_k$ is the k-th column of the unit matrix of order m. Thus, the necessary first-order (KKT) conditions have the form

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = \sum_{k=1}^{m} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl} = \boldsymbol{0}, \quad \sum_{l=1}^{m_k} u_{kl} = h_k, \quad h_k = \frac{\partial}{\partial z_k} h(z), \tag{11.8}$$

$$u_{kl} \ge 0, \quad z_k - f_{kl}(\boldsymbol{x}) \ge 0, \quad u_{kl}(z_k - f_{kl}(\boldsymbol{x})) = 0, \tag{11.9}$$

where u_{kl}, $1 \le k \le m$, $1 \le l \le m_k$, are Lagrange multipliers and $z_k = F_k(\boldsymbol{x})$. So we obtain the same necessary conditions for an extremum as in Remark 11.5.

Remark 11.7 A classical minimax problem

$$F(\boldsymbol{x}) = \max_{1 \le k \le m} f_k(\boldsymbol{x}) \tag{11.10}$$

can be replaced with an equivalent nonlinear programming problem

$$\begin{cases} \text{minimize} & \tilde{F}(\boldsymbol{x}, z) = z \\ \text{subject to} & f_k(\boldsymbol{x}) \le z, \quad 1 \le k \le m, \end{cases} \tag{11.11}$$

and the necessary KKT conditions have the form

$$\sum_{k=1}^{m} \boldsymbol{g}_k(\boldsymbol{x}) u_k = \boldsymbol{0}, \quad \sum_{k=1}^{m} u_k = 1, \tag{11.12}$$

$$u_k \ge 0, \quad z - f_k(\boldsymbol{x}) \ge 0, \quad u_k(z - f_k(\boldsymbol{x})) = 0, \quad 1 \le k \le m. \tag{11.13}$$

Remark 11.8 Minimization of the sum of absolute values

$$F(\boldsymbol{x}) = \sum_{k=1}^{m} |f_k(\boldsymbol{x})| = \sum_{k=1}^{m} \max\{f_k^+(\boldsymbol{x}), f_k^-(\boldsymbol{x})\}, \tag{11.14}$$

where

$$f_k^+(\boldsymbol{x}) = f_k(\boldsymbol{x}), \quad f_k^-(\boldsymbol{x}) = -f_k(\boldsymbol{x})$$

can be replaced with an equivalent nonlinear programming problem

$$\begin{cases} \text{minimize} & \tilde{F}(\boldsymbol{x}, \boldsymbol{z}) = \sum\limits_{k=1}^{m} z_k \\ \text{subject to} & -z_k \le f_k(\boldsymbol{x}) \le z_k, \end{cases} \tag{11.15}$$

(there are two constraints $c_k^-(\boldsymbol{x}) = z_k - f_k(\boldsymbol{x}) \ge 0$ and $c_k^+(\boldsymbol{x}) = z_k + f_k(\boldsymbol{x}) \ge 0$ for each index $1 \le k \le m$) and the necessary KKT conditions have the form

$$\sum_{k=1}^{m} \boldsymbol{g}_k(\boldsymbol{x})(u_k^+ - u_k^-) = \boldsymbol{0}, \quad u_k^+ + u_k^- = 1, \tag{11.16}$$

$$u_k^+ \ge 0, \quad z_k - f_k(\boldsymbol{x}) \ge 0, \quad u_k^+(z_k - f_k(\boldsymbol{x})) = 0, \tag{11.17}$$

$$u_k^- \ge 0, \quad z_k + f_k(\boldsymbol{x}) \ge 0, \quad u_k^-(z_k + f_k(\boldsymbol{x})) = 0, \tag{11.18}$$

where $1 \le k \le m$. If we set $u_k = u_k^+ - u_k^-$ and use the equality $u_k^+ + u_k^- = 1$, we obtain $u_k^+ = (1 + u_k)/2$, $u_k^- = (1 - u_k)/2$. From conditions $u_k^+ \ge 0$, $u_k^- \ge 0$ the inequalities $-1 \le u_k \le 1$, or $|u_k| \le 1$, follow. The condition $u_k^+ + u_k^- = 1$ implies that the numbers u_k^+, u_k^- cannot be simultaneously zero, so either $z_k = f_k(\boldsymbol{x})$ or $z_k = -f_k(\boldsymbol{x})$, that is $z_k = |f_k(\boldsymbol{x})|$. If $f_k(\boldsymbol{x}) \ne 0$, it cannot simultaneously hold $z_k = f_k(\boldsymbol{x})$ and $z_k = -f_k(\boldsymbol{x})$, so the numbers u_k^+, u_k^- cannot be simultaneously nonzero. Then either $u_k = u_k^+ = 1$ and $z_k = f_k(\boldsymbol{x})$ or $u_k = -u_k^- = -1$ and $z_k = -f_k(\boldsymbol{x})$, that is $u_k = f_k(\boldsymbol{x})/|f_k(\boldsymbol{x})|$. Thus, the necessary KKT conditions have the form

$$\sum_{k=1}^{m} \boldsymbol{g}_k(\boldsymbol{x}) u_k = \boldsymbol{0}, \quad z_k = |f_k(\boldsymbol{x})|,$$

$$|u_k| \le 1, \quad u_k = \frac{f_k(\boldsymbol{x})}{|f_k(\boldsymbol{x})|}, \quad \text{if} \quad |f_k(\boldsymbol{x})| > 0. \tag{11.19}$$

Remark 11.9 Minimization of the sum of absolute values can also be reformulated so that more slack variables are used. We obtain the problem

$$\begin{cases} \text{minimize} & \tilde{F}(\boldsymbol{x}, \boldsymbol{z}) = \sum\limits_{k=1}^{m} (z_k^+ + z_k^-) \\ \text{subject to} & f_k(\boldsymbol{x}) = z_k^+ - z_k^-, \quad z_k^+ \ge 0, \quad z_k^- \ge 0, \end{cases} \tag{11.20}$$

where $1 \le k \le m$. This problem contains m general equality constraints and $2m$ simple bounds for $2m$ slack variables.

In the subsequent considerations, we will restrict ourselves to functions of the form (11.4), the sums of maxima that include most cases important for applications. In this case, it holds

$$h(\boldsymbol{z}) = \sum_{k=1}^{m} z_k, \quad \nabla h(\boldsymbol{z}) = \tilde{\boldsymbol{e}}, \quad \nabla^2 h(\boldsymbol{z}) = 0, \tag{11.21}$$

where $\tilde{\boldsymbol{e}} \in \mathbb{R}^m$ is a vector with unit elements. The case when $h(\boldsymbol{z})$ is a general function satisfying Assumption 11.3 is studied in [23].

11.2 Primal Interior Point Methods

11.2.1 Barriers and Barrier Functions

Primal interior point methods for equality constraint minimization problems are based on adding a barrier term containing constraint functions to the minimized function. A resulting barrier function, depending on a barrier parameter $0 < \mu \le$

$\overline{\mu} < \infty$, is successively minimized on $\mathbb{R}^n$ (without any constraints), where $\mu \downarrow 0$. Applying this approach on the problem (11.7), we obtain a barrier function

$$B_\mu(\boldsymbol{x}, \boldsymbol{z}) = h(\boldsymbol{z}) + \mu \sum_{k=1}^{m} \sum_{l=1}^{m_k} \varphi(z_k - f_{kl}(\boldsymbol{x})), \quad 0 < \mu \leq \overline{\mu}, \tag{11.22}$$

where $\varphi : (0, \infty) \to \mathbb{R}$ is a barrier which satisfies the following assumption.

Assumption 11.5 *Function $\varphi(t)$, $t \in (0, \infty)$, is twice continuously differentiable, decreasing, and strictly convex, with $\lim_{t \downarrow 0} \varphi(t) = \infty$. Function $\varphi'(t)$ is increasing and strictly concave such that $\lim_{t \uparrow \infty} \varphi'(t) = 0$. For $t \in (0, \infty)$ it holds $-t\varphi'(t) \leq 1$, $t^2\varphi''(t) \leq 1$. There exist numbers $\tau > 0$ and $\underline{c} > 0$ such that for $t < \tau$ it holds*

$$-t\varphi'(t) \geq \underline{c} \tag{11.23}$$

and

$$\varphi'(t)\varphi'''(t) - \varphi''(t)^2 > 0. \tag{11.24}$$

Remark 11.10 A logarithmic barrier function

$$\varphi(t) = \log t^{-1} = -\log t, \tag{11.25}$$

is most frequently used. It satisfies Assumption 11.5 with $\underline{c} = 1$ and $\tau = \infty$ but it is not bounded from below since $\log t \uparrow \infty$ for $t \uparrow \infty$. For that reason, barriers bounded from below are sometimes used, e.g. a function

$$\varphi(t) = \log(t^{-1} + \tau^{-1}) = -\log \frac{t\tau}{t + \tau}, \tag{11.26}$$

which is bounded from below by number $\underline{\varphi} = -\log \tau$, or a function

$$\varphi(t) = -\log t, \quad 0 < t \leq \tau, \qquad \varphi(t) = at^{-2} + bt^{-1} + c, \quad t \geq \tau, \tag{11.27}$$

which is bounded from below by number $\underline{\varphi} = c = -\log \tau - 3/2$, or a function

$$\varphi(t) = -\log t, \quad 0 < t \leq \tau, \qquad \varphi(t) = at^{-1} + bt^{-1/2} + c, \quad t \geq \tau, \tag{11.28}$$

which is bounded from below by number $\underline{\varphi} = c = -\log \tau - 3$. Coefficients a, b, c are chosen so that function $\varphi(t)$ as well as its first and second derivatives are continuous in $t = \tau$. All these barriers satisfy Assumption 11.5 [23] (the proof of this statement is trivial for logarithmic barrier (11.25)).

Even if bounded from below barriers (11.26)–(11.28) have more advantageous theoretical properties (Assumption 11.1 can be replaced with a weaker Assump-

tion 11.2 and the proof of Lemma 11.2 below is much simpler, see [23]), algorithms using logarithmic barrier (11.26) are usually more efficient. Therefore, we will only deal with methods using the logarithmic barrier $\varphi(t) = -\log t$ in the subsequent considerations.

11.2.2 Iterative Determination of a Minimax Vector

Suppose the function $h(z)$ is of form (11.21). Using the logarithmic barrier $\varphi(t) = -\log t$, function (11.22) can be written as

$$B_\mu(\boldsymbol{x}, z) = \sum_{k=1}^{m} z_k - \mu \sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(z_k - f_{kl}(\boldsymbol{x})), \quad 0 < \mu \le \overline{\mu}. \tag{11.29}$$

Further, we will denote $\boldsymbol{g}_{kl}(\boldsymbol{x})$ and $G_{kl}(\boldsymbol{x})$ gradients and Hessian matrices of functions $f_{kl}(\boldsymbol{x})$, $1 \le k \le m$, $1 \le l \le m_k$, and set

$$\begin{aligned} u_{kl}(\boldsymbol{x}, z) &= \frac{\mu}{z_k - f_{kl}(\boldsymbol{x})} \ge 0, \\ v_{kl}(\boldsymbol{x}, z) &= \frac{\mu}{(z_k - f_{kl}(\boldsymbol{x}))^2} = \frac{1}{\mu} u_{kl}^2(\boldsymbol{x}, z) \ge 0, \end{aligned} \tag{11.30}$$

and

$$\boldsymbol{u}_k(\boldsymbol{x}, z) = \begin{bmatrix} u_{k1}(\boldsymbol{x}, z) \\ \vdots \\ u_{km_k}(\boldsymbol{x}, z) \end{bmatrix}, \quad \boldsymbol{v}_k(\boldsymbol{x}, z) = \begin{bmatrix} v_{k1}(\boldsymbol{x}, z) \\ \vdots \\ v_{km_k}(\boldsymbol{x}, z) \end{bmatrix}, \quad \tilde{\boldsymbol{e}}_k = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Denoting by $\boldsymbol{g}(\boldsymbol{x}, z)$ the gradient of the function $B_\mu(\boldsymbol{x}, z)$ and $\gamma_k(\boldsymbol{x}, z) = \frac{\partial}{\partial z_k} B_\mu(\boldsymbol{x}, z)$, the necessary conditions for an extremum of the barrier function (11.22) can be written in the form

$$\boldsymbol{g}(\boldsymbol{x}, z) = \sum_{k=1}^{m} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x}, z) = \sum_{k=1}^{m} A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}, z) = \boldsymbol{0}, \tag{11.31}$$

$$\gamma_k(\boldsymbol{x}, z) = 1 - \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}, z) = 1 - \tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}, z) = 0, \quad 1 \le k \le m, \tag{11.32}$$

where $A_k(\boldsymbol{x}) = [\boldsymbol{g}_{k1}(\boldsymbol{x}), \ldots, \boldsymbol{g}_{km_k}(\boldsymbol{x})]$, which is a system of $n + m$ nonlinear equations for unknown vectors $\boldsymbol{x}$ and z. These equations can be solved by the Newton method. In this case, the second derivatives of the Lagrange function (which

are the first derivatives of expressions (11.31) and (11.32)) are computed. Denoting

$$G(\boldsymbol{x},z)=\sum_{k=1}^{m}\sum_{l=1}^{m_k}G_{kl}(\boldsymbol{x})u_{kl}(\boldsymbol{x},z), \tag{11.33}$$

the Hessian matrix of the Lagrange function and setting

$$U_k(\boldsymbol{x},z)=\mathrm{diag}[u_{k1}(\boldsymbol{x},z),\ldots,u_{km_k}(\boldsymbol{x},z)],$$
$$V_k(\boldsymbol{x},z)=\mathrm{diag}[v_{k1}(\boldsymbol{x},z),\ldots,v_{km_k}(\boldsymbol{x},z)]=\frac{1}{\mu}U_k^2(\boldsymbol{x},z),$$

we can write

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x},z)&=\sum_{k=1}^{m}\sum_{l=1}^{m_k}G_{kl}(\boldsymbol{x})u_{kl}(\boldsymbol{x},z)+\sum_{k=1}^{m}\sum_{l=1}^{m_k}\boldsymbol{g}_{kl}(\boldsymbol{x})v_{kl}(\boldsymbol{x},z)\boldsymbol{g}_{kl}^T(\boldsymbol{x})\\&=G(\boldsymbol{x},z)+\sum_{k=1}^{m}A_k(\boldsymbol{x})V_k(\boldsymbol{x},z)A_k^T(\boldsymbol{x}),\end{aligned} \tag{11.34}$$

$$\frac{\partial}{\partial z_k}\boldsymbol{g}(\boldsymbol{x},z)=-\sum_{l=1}^{m_k}\boldsymbol{g}_{kl}(\boldsymbol{x})v_{kl}(\boldsymbol{x},z)=-A_k(\boldsymbol{x})\boldsymbol{v}_k(\boldsymbol{x},z), \tag{11.35}$$

$$\frac{\partial}{\partial\boldsymbol{x}}\gamma_k(\boldsymbol{x},z)=-\sum_{l=1}^{m_k}v_{kl}(\boldsymbol{x},z)\boldsymbol{g}_{kl}^T(\boldsymbol{x})=-\boldsymbol{v}_k^T(\boldsymbol{x},z)A_k^T(\boldsymbol{x}), \tag{11.36}$$

$$\frac{\partial}{\partial z_k}\gamma_k(\boldsymbol{x},z)=\sum_{l=1}^{m_k}v_{kl}(\boldsymbol{x},z)=\tilde{\boldsymbol{e}}_k^T\boldsymbol{v}_k(\boldsymbol{x},z). \tag{11.37}$$

Using these formulas we obtain a system of linear equations describing a step of the Newton method

$$\begin{bmatrix}W(\boldsymbol{x},z) & -A_1(\boldsymbol{x})\boldsymbol{v}_1(\boldsymbol{x},z) & \cdots & -A_m(\boldsymbol{x})\boldsymbol{v}_m(\boldsymbol{x},z)\\ -\boldsymbol{v}_1^T(\boldsymbol{x},z)A_1^T(\boldsymbol{x}) & \tilde{\boldsymbol{e}}_1^T\boldsymbol{v}_1(\boldsymbol{x},z) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ -\boldsymbol{v}_m^T(\boldsymbol{x},z)A_m^T(\boldsymbol{x}) & 0 & \cdots & \tilde{\boldsymbol{e}}_m^T\boldsymbol{v}_m(\boldsymbol{x},z)\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta z_1\\ \vdots\\ \Delta z_m\end{bmatrix} \tag{11.38}$$
$$=-\begin{bmatrix}\boldsymbol{g}(\boldsymbol{x},z)\\ \gamma_1(\boldsymbol{x},z)\\ \vdots\\ \gamma_m(\boldsymbol{x},z)\end{bmatrix},$$

where

$$W(\boldsymbol{x},z)=G(\boldsymbol{x},z)+\sum_{k=1}^{m}A_k(\boldsymbol{x})V_k(\boldsymbol{x},z)A_k^T(\boldsymbol{x}). \tag{11.39}$$

Setting

$$C(\boldsymbol{x},z)=[A_1(\boldsymbol{x})\boldsymbol{v}_1(\boldsymbol{x},z),\ldots,A_m(\boldsymbol{x})\boldsymbol{v}_m(\boldsymbol{x},z)],$$
$$D(\boldsymbol{x},z)=\mathrm{diag}[\tilde{\boldsymbol{e}}_1^T\boldsymbol{v}_1(\boldsymbol{x},z),\ldots,\tilde{\boldsymbol{e}}_m^T\boldsymbol{v}_m(\boldsymbol{x},z)]$$

and $\boldsymbol{\gamma}(\boldsymbol{x},z)=[\gamma_1(\boldsymbol{x},z),\ldots,\gamma_m(\boldsymbol{x},z)]^T$, a step of the Newton method can be written in the form

$$\begin{bmatrix} W(\boldsymbol{x},z) & -C(\boldsymbol{x},z)\\ -C^T(\boldsymbol{x},z) & D(\boldsymbol{x},z)\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta z\end{bmatrix}=-\begin{bmatrix}\boldsymbol{g}(\boldsymbol{x},z)\\ \boldsymbol{\gamma}(\boldsymbol{x},z)\end{bmatrix}. \tag{11.40}$$

The diagonal matrix $D(\boldsymbol{x},z)$ is positive definite since it has positive diagonal elements.

During iterative determination of a minimax vector we know a value of the parameter μ and vectors $\boldsymbol{x}\in\mathbb{R}^n$, $z\in\mathbb{R}^m$ such that $z_k>F_k(\boldsymbol{x})$, $1\le k\le m$. Using formula (11.40) we determine direction vectors $\Delta\boldsymbol{x}$, Δz. Then, we choose a step length α so that

$$B_\mu(\boldsymbol{x}+\alpha\Delta\boldsymbol{x},z+\alpha\Delta z)<B_\mu(\boldsymbol{x},z) \tag{11.41}$$

and $z_k+\alpha\Delta z_k>F_k(\boldsymbol{x}+\alpha\Delta\boldsymbol{x})$, $1\le k\le m$. Finally, we set $\boldsymbol{x}_+=\boldsymbol{x}+\alpha\Delta\boldsymbol{x}$, $z_+=z+\alpha\Delta z$ and determine a new value $\mu_+<\mu$. If the matrix of system of equations (11.40) is positive definite, inequality (11.41) is satisfied for a sufficiently small value of the step length α.

Theorem 11.1 *Let the matrix $G(\boldsymbol{x},z)$ given by (11.33) be positive definite. Then the matrix of system of equations (11.40) is positive definite.*

Proof The matrix of system of equations (11.40) is positive definite if and only if the matrix D and its Schur complement $W-CD^{-1}C^T$ are positive definite [7, Theorem 2.5.6]. The matrix D is positive definite since it has positive diagonal elements. Further, it holds

$$W-CD^{-1}C^T=G+\sum_{k=1}^{m}\left(A_kV_kA_k^T-A_kV_k\tilde{\boldsymbol{e}}_k(\tilde{\boldsymbol{e}}_k^TV_k\tilde{\boldsymbol{e}}_k)^{-1}(A_kV_k\tilde{\boldsymbol{e}}_k)^T\right),$$

matrices $A_kV_kA_k^T-A_kV_k\tilde{\boldsymbol{e}}_k(\tilde{\boldsymbol{e}}_k^TV_k\tilde{\boldsymbol{e}}_k)^{-1}(A_kV_k\tilde{\boldsymbol{e}}_k)^T$, $1\le k\le m$, are positive semidefinite due to the Schwarz inequality and the matrix G is positive definite by the assumption. □

11.2.3 Direct Determination of a Minimax Vector

Now we will show how to solve system of equations (11.31)–(11.32) by direct determination of a minimax vector using two-level optimization

$$z(\boldsymbol{x};\mu) = \operatorname*{argmin}_{z\in\mathbb{R}^m} B_\mu(\boldsymbol{x},z), \tag{11.42}$$

and

$$\boldsymbol{x}^* = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} \hat{B}(\boldsymbol{x};\mu), \quad \hat{B}(\boldsymbol{x};\mu) \stackrel{\Delta}{=} B_\mu(\boldsymbol{x},z(\boldsymbol{x};\mu)). \tag{11.43}$$

The problem (11.42) serves for determination of an optimal vector $z(\boldsymbol{x};\mu)\in\mathbb{R}^m$. Let $\tilde{B}_\mu(z) = B_\mu(\boldsymbol{x},z)$ for a fixed chosen vector $\boldsymbol{x}\in\mathbb{R}^n$. The function $\tilde{B}_\mu(z)$ is strictly convex (as a function of a vector z), since it is a sum of convex function (11.21) and strictly convex functions $-\mu\log(z_k - f_{kl}(\boldsymbol{x}))$, $1\le k\le m$, $1\le l\le m_k$. A minimum of the function $\tilde{B}_\mu(z)$ is its stationary point, so it is a solution of system of equations (11.32) with Lagrange multipliers (11.30). The following theorem shows that this solution exists and is unique.

Theorem 11.2 *The function $\tilde{B}_\mu(z) : (F(\boldsymbol{x}),\infty)\to\mathbb{R}$ has a unique stationary point which is its global minimum. This stationary point is characterized by a system of equations $\boldsymbol{\gamma}(\boldsymbol{x},z)=\boldsymbol{0}$, or*

$$1 - \tilde{\boldsymbol{e}}_k^T\boldsymbol{u}_k = 1 - \sum_{l=1}^{m_k}\frac{\mu}{z_k - f_{kl}(\boldsymbol{x})} = 0, \quad 1\le k\le m, \tag{11.44}$$

which has a unique solution $z(\boldsymbol{x};\mu)\in\mathbb{Z}\subset\mathbb{R}^m$ such that

$$F_k(\boldsymbol{x}) < F_k(\boldsymbol{x}) + \mu < z_k(\boldsymbol{x};\mu) < F_k(\boldsymbol{x}) + m_k\mu \tag{11.45}$$

for $1\le k\le m$.

Proof Definition 11.1 implies $f_{kl}(\boldsymbol{x})\le F_k(\boldsymbol{x})$, $1\le k\le m$, $1\le l\le m_k$, where the equality occurs for at least one index l.

(a) If (11.44) holds, then we can write

$$1 = \sum_{l=1}^{m_k}\frac{\mu}{z_k - f_{kl}(\boldsymbol{x})} > \frac{\mu}{z_k - F_k(\boldsymbol{x})} \;\Leftrightarrow\; z_k - F_k(\boldsymbol{x}) > \mu,$$

$$1 = \sum_{l=1}^{m_k}\frac{\mu}{z_k - f_{kl}(\boldsymbol{x})} < \frac{m_k\mu}{z_k - F_k(\boldsymbol{x})} \;\Leftrightarrow\; z_k - F_k(\boldsymbol{x}) < m_k\mu,$$

which proves inequalities (11.45).

(b) Since

$$\gamma_k(\boldsymbol{x}, F+\mu) = 1 - \sum_{l=1}^{m_k} \frac{\mu}{\mu + F_k(\boldsymbol{x}) - f_{kl}(\boldsymbol{x})} < 1 - \frac{\mu}{\mu} = 0,$$

$$\gamma_k(\boldsymbol{x}, F+m_k\mu) = 1 - \sum_{l=1}^{m_k} \frac{\mu}{m_k\mu + F_k(\boldsymbol{x}) - f_{kl}(\boldsymbol{x})} > 1 - \frac{m_k\mu}{m_k\mu} = 0,$$

and the function $\gamma_k(\boldsymbol{x}, z_k)$ is continuous and decreasing in $F_k(\boldsymbol{x}) + \mu < z_k(\boldsymbol{x}; \mu) < F_k(\boldsymbol{x}) + m_k$ by (11.37), the equation $\gamma_k(\boldsymbol{x}, z_k) = 0$ has a unique solution in this interval. Since the function $\tilde{B}_\mu(\boldsymbol{z})$ is convex this solution corresponds to its global minimum.

□

System (11.44) is a system of m scalar equations with localization inequalities (11.45). These scalar equations can be efficiently solved by robust methods described e.g. in [14] and [15] (details are stated in [22]). Suppose that $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x}; \mu)$ and denote

$$\hat{B}(\boldsymbol{x}; \mu) = \sum_{k=1}^{m} z_k(\boldsymbol{x}; \mu) - \mu \sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x})). \tag{11.46}$$

To find a minimum of $B_\mu(\boldsymbol{x}, \boldsymbol{z})$ in $\mathbb{R}^{n+m}$, it suffices to minimize $\hat{B}(\boldsymbol{x}; \mu)$ in $\mathbb{R}^n$.

Theorem 11.3 *Consider the barrier function (11.46). Then*

$$\nabla \hat{B}(\boldsymbol{x}; \mu) = \sum_{k=1}^{m} A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}; \mu), \tag{11.47}$$

$$\begin{aligned} \nabla^2 \hat{B}(\boldsymbol{x}; \mu) &= W(\boldsymbol{x}; \mu) - C(\boldsymbol{x}; \mu) D^{-1}(\boldsymbol{x}; \mu) C^T(\boldsymbol{x}; \mu) \\ &= G(\boldsymbol{x}; \mu) + \sum_{k=1}^{m} A_k(\boldsymbol{x}) V_k(\boldsymbol{x}; \mu) A_k^T(\boldsymbol{x}) \\ &\quad - \sum_{k=1}^{m} \frac{A_k(\boldsymbol{x}) V_k(\boldsymbol{x}; \mu) \tilde{\boldsymbol{e}}_k \tilde{\boldsymbol{e}}_k^T V_k(\boldsymbol{x}; \mu) A_k^T(\boldsymbol{x})}{\tilde{\boldsymbol{e}}_k^T V_k(\boldsymbol{x}; \mu) \tilde{\boldsymbol{e}}_k}, \end{aligned} \tag{11.48}$$

where $G(\boldsymbol{x}; \mu) = G(\boldsymbol{x}, \boldsymbol{z}(\boldsymbol{x}; \mu))$ *and* $W(\boldsymbol{x}; \mu)$, $C(\boldsymbol{x}; \mu)$, $D(\boldsymbol{x}; \mu)$, $U_k(\boldsymbol{x}; \mu)$, $V_k(\boldsymbol{x}; \mu) = U_k^2(\boldsymbol{x}; \mu)/\mu$, $1 \le k \le m$, *are obtained by the same substitution. A solution of equation*

$$\nabla^2 \hat{B}(\boldsymbol{x}; \mu) \Delta \boldsymbol{x} = -\nabla \hat{B}(\boldsymbol{x}; \mu) \tag{11.49}$$

is identical with $\Delta \boldsymbol{x}$ given by (11.40), where $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x}; \mu)$ (so $\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{z}(\boldsymbol{x}; \mu)) = \boldsymbol{0}$).

Proof Differentiating the barrier function (11.46) and using (11.32) we obtain

$$\begin{aligned}\nabla \hat{B}(\boldsymbol{x}; \mu) &= \sum_{k=1}^{m} \frac{\partial}{\partial \boldsymbol{x}} z_k(\boldsymbol{x}; \mu) - \sum_{k=1}^{m} \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}; \mu) \left(\frac{\partial}{\partial \boldsymbol{x}} z_k(\boldsymbol{x}; \mu) - \frac{\partial}{\partial \boldsymbol{x}} f_{kl}(\boldsymbol{x}) \right) \\ &= \sum_{k=1}^{m} \frac{\partial}{\partial \boldsymbol{x}} z_k(\boldsymbol{x}; \mu) \left(1 - \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}; \mu) \right) + \sum_{k=1}^{m} \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}; \mu) \frac{\partial}{\partial \boldsymbol{x}} f_{kl}(\boldsymbol{x}) \\ &= \sum_{k=1}^{m} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x}; \mu) = \sum_{k=1}^{m} A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}; \mu),\end{aligned}$$

where

$$u_{kl}(\boldsymbol{x}; \mu) = \frac{\mu}{z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x})}, \quad 1 \le k \le m, \quad 1 \le l \le m_k. \tag{11.50}$$

Formula (11.48) can be obtained by additional differentiation of relations (11.32) and (11.47) using (11.50). A simpler way is based on using (11.40). Since (11.32) implies $\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{z}(\boldsymbol{x}; \mu)) = \boldsymbol{0}$, we can substitute $\boldsymbol{\gamma} = \boldsymbol{0}$ into (11.40), which yields the equation

$$\left(W(\boldsymbol{x}, \boldsymbol{z}) - C(\boldsymbol{x}, \boldsymbol{z}) D^{-1}(\boldsymbol{x}, \boldsymbol{z}) C^T(\boldsymbol{x}, \boldsymbol{z}) \right) \Delta \boldsymbol{x} = -\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{z}),$$

where $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{x}; \mu)$, that confirms validity of formulas (11.48) and (11.49) (details can be found in [22]). □

Remark 11.11 To determine an inverse of the Hessian matrix, one can use a Woodbury formula [7, Theorem 12.1.4] which gives

$$\begin{aligned}(\nabla^2 \hat{B}(\boldsymbol{x}; \mu))^{-1} = {} & W^{-1}(\boldsymbol{x}; \mu) - W^{-1}(\boldsymbol{x}; \mu) C(\boldsymbol{x}; \mu) \\ & \left(C^T(\boldsymbol{x}; \mu) W^{-1}(\boldsymbol{x}; \mu) C(\boldsymbol{x}; \mu) - D(\boldsymbol{x}; \mu) \right)^{-1} \\ & C^T(\boldsymbol{x}; \mu) W^{-1}(\boldsymbol{x}; \mu).\end{aligned} \tag{11.51}$$

If the matrix $\nabla^2 \hat{B}(\boldsymbol{x}; \mu)$ is not positive definite, it can be replaced by a matrix $LL^T = \nabla^2 \hat{B}(\boldsymbol{x}; \mu) + E$, obtained by the Gill–Murray decomposition [10]. Note that it is more advantageous to use system of linear equations (11.40) instead of (11.49) for determination of a direction vector $\Delta \boldsymbol{x}$ because the system of nonlinear equations (11.44) is solved with prescribed finite precision, and thus a vector $\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{z})$, defined by (11.32), need not be zero.

From

$$V_k(\boldsymbol{x};\mu)=\frac{1}{\mu}U_k^2(\boldsymbol{x};\mu),\quad \boldsymbol{u}_k(\boldsymbol{x};\mu)\geq 0,\quad \tilde{\boldsymbol{e}}_k^T\boldsymbol{u}_k(\boldsymbol{x};\mu)=1,\quad 1\leq k\leq m,$$

it follows that $\|V_k(\boldsymbol{x};\mu)\|\uparrow\infty$ if $\mu\downarrow 0$, so Hessian matrix (11.48) may be ill-conditioned if the value μ is very small. From this reason, we use a lower bound $\underline{\mu}>0$ for μ.

Theorem 11.4 *Let Assumption 11.4 be satisfied and $\mu\geq\underline{\mu}>0$. If $G(\boldsymbol{x};\mu)$ is uniformly positive definite (if a constant $\underline{G}$ exists such that $\boldsymbol{v}^TG(\boldsymbol{x};\mu)\boldsymbol{v}\geq\underline{G}\|\boldsymbol{v}\|^2$), then there is a number $\overline{\kappa}\geq 1$ such that $\kappa(\nabla^2\hat{B}(\boldsymbol{x};\mu))\leq\overline{\kappa}$.*

Proof

(a) Using (11.30), (11.48), and Assumption 11.4, we obtain

$$\begin{aligned}\|\nabla^2\hat{B}(\boldsymbol{x};\mu)\|&\leq\left\|G(\boldsymbol{x};\mu)+\sum_{k=1}^{m}A_k(\boldsymbol{x})V_k(\boldsymbol{x};\mu)A_k^T(\boldsymbol{x})\right\|\\&\leq\sum_{k=1}^{m}\sum_{l=1}^{m_k}\left(|G_{kl}(\boldsymbol{x})u_{kl}(\boldsymbol{x},\mu)|+\frac{1}{\mu}\left|u_{kl}^2(\boldsymbol{x};\mu)\boldsymbol{g}_{kl}(\boldsymbol{x})\boldsymbol{g}_{kl}^T(\boldsymbol{x})\right|\right)\\&\leq\frac{\overline{m}}{\mu}\left(\overline{\mu}\,\overline{G}+\overline{g}^2\right)\triangleq\frac{\overline{c}}{\mu}\leq\frac{\overline{c}}{\underline{\mu}},\end{aligned}\tag{11.52}$$

because $0\leq u_{kl}(\boldsymbol{x};\mu)\leq\tilde{\boldsymbol{e}}_k^T\boldsymbol{u}_k(\boldsymbol{x};\mu)=1$, $1\leq k\leq m$, $1\leq l\leq m_k$, by (11.44).

(b) From the proof of Theorem 11.1 it follows that the matrix $\nabla^2\hat{B}(\boldsymbol{x};\mu)-G(\boldsymbol{x};\mu)$ is positive semidefinite. Therefore,

$$\underline{\lambda}(\nabla^2\hat{B}(\boldsymbol{x};\mu))\geq\underline{\lambda}(G(\boldsymbol{x};\mu))\geq\underline{G}.$$

(c) Since (a) implies $\overline{\lambda}(\nabla^2\hat{B}(\boldsymbol{x};\mu))=\|\nabla^2\hat{B}(\boldsymbol{x};\mu)\|\leq\overline{c}/\underline{\mu}$, using (b) we can write

$$\kappa(\nabla^2\hat{B}(\boldsymbol{x};\mu))=\frac{\overline{\lambda}(\nabla^2\hat{B}(\boldsymbol{x};\mu))}{\underline{\lambda}(\nabla^2\hat{B}(\boldsymbol{x};\mu))}\leq\frac{\overline{c}}{\underline{\mu}\,\underline{G}}\triangleq\overline{\kappa}.\tag{11.53}$$

□

Remark 11.12 If there exists a number $\overline{\kappa}>0$ such that $\kappa(\nabla^2\hat{B}(\boldsymbol{x}_i;\mu_i))\leq\overline{\kappa}$, $i\in\mathbb{N}$, the direction vector $\Delta\boldsymbol{x}_i$, given by solving a system of equations $\nabla^2\hat{B}(\boldsymbol{x}_i;\mu_i)\Delta\boldsymbol{x}_i=-\nabla\hat{B}(\boldsymbol{x}_i;\mu_i)$, satisfies the condition

$$(\Delta\boldsymbol{x}_i)^T\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\leq-\varepsilon_0\|\Delta\boldsymbol{x}_i\|\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|,\quad i\in\mathbb{N},\tag{11.54}$$

where $\varepsilon_0=1/\sqrt{\overline{\kappa}}$ and $\boldsymbol{g}(\boldsymbol{x};\mu)=\nabla\hat{B}(\boldsymbol{x};\mu)$. Then, for arbitrary numbers $0<\varepsilon_1\leq\varepsilon_2<1$ one can find a step length parameter $\alpha_i>0$ such that for $\boldsymbol{x}_{i+1}=\boldsymbol{x}_i+\alpha_i\Delta\boldsymbol{x}_i$

it holds

$$\varepsilon_1 \leq \frac{\hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \hat{B}(\boldsymbol{x}_i;\mu_i)}{\alpha_i(\Delta\boldsymbol{x}_i)^T\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)} \leq \varepsilon_2, \tag{11.55}$$

so there exists a number $c > 0$ such that (see [26, Section 3.2])

$$\hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \hat{B}(\boldsymbol{x}_i;\mu_i) \leq -c\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|^2, \quad i \in \mathbb{N}. \tag{11.56}$$

If Assumption 11.4 is not satisfied, then only $(\Delta\boldsymbol{x}_i)^T\boldsymbol{g}(\boldsymbol{x}_i;\mu_i) < 0$ holds (because the matrix $\nabla^2\hat{B}(\boldsymbol{x};\mu)$ is positive definite by Theorem 11.1) and

$$\hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \hat{B}(\boldsymbol{x}_i;\mu_i) \leq 0, \quad i \in \mathbb{N}. \tag{11.57}$$

11.2.4 Implementation

Remark 11.13 In (11.39), it is assumed that $G(\boldsymbol{x},\boldsymbol{z})$ is the Hessian matrix of the Lagrange function. Direct computation of the matrix $G(\boldsymbol{x};\mu) = G(\boldsymbol{x},\boldsymbol{z}(\boldsymbol{x};\mu))$ is usually difficult (one can use automatic differentiation as described in [13]). Thus, various approximations $G \approx G(\boldsymbol{x};\mu)$ are mostly used.

- The matrix $G \approx G(\boldsymbol{x};\mu)$ can be determined using differences

$$G\boldsymbol{w}_j = \frac{1}{\delta}\left(\sum_{k=1}^{m} A_k(\boldsymbol{x}+\delta\boldsymbol{w}_j)\boldsymbol{u}_k(\boldsymbol{x};\mu) - \sum_{k=1}^{m} A(\boldsymbol{x})\boldsymbol{u}_k(\boldsymbol{x};\mu)\right).$$

  The vectors $\boldsymbol{w}_j$, $1 \leq j \leq \overline{k}$, are chosen so that the number of them is as small as possible [4, 27].
- The matrix $G \approx G(\boldsymbol{x};\mu)$ can be determined using the variable metric methods [17]. The vectors

$$\boldsymbol{d} = \boldsymbol{x}_+ - \boldsymbol{x}, \quad \boldsymbol{y} = \sum_{k=1}^{m} A_k(\boldsymbol{x}_+)\boldsymbol{u}_k(\boldsymbol{x}_+;\mu) - \sum_{k=1}^{m} A_k(\boldsymbol{x})\boldsymbol{u}_k(\boldsymbol{x}_+;\mu)$$

  are used for an update of G.
- If the problem is separable (i.e. $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, are functions of a small number $n_{kl} = O(1)$ of variables), one can set as in [12]

$$G = \sum_{k=1}^{m}\sum_{l=1}^{m_k} Z_{kl}\hat{G}_{kl}Z_{kl}^T u_{kl}(\boldsymbol{x},\boldsymbol{z}),$$

where the reduced Hessian matrices $\hat{G}_{kl}$ are updated using the reduced vectors $\hat{\boldsymbol{d}}_{kl} = Z_{kl}^T(\boldsymbol{x}_+ - \boldsymbol{x})$ and $\hat{\boldsymbol{y}}_{kl} = Z_{kl}(\boldsymbol{g}_{kl}(\boldsymbol{x}_+) - \boldsymbol{g}_{kl}(\boldsymbol{x}))$.

Remark 11.14 The matrix $G \approx G(\boldsymbol{x}; \mu)$ obtained by the approach stated in Remark 11.13 can be ill-conditioned so condition (11.54) (with a chosen value $\varepsilon_0 > 0$) may not be satisfied. In this case it is possible to restart the iteration process and set $G = I$. Then $\overline{G} = 1$ and $\underline{G} = 1$ in (11.52) and (11.53), so it is a higher probability of fulfilment of condition (11.54). If the choice $G = I$ does not satisfy (11.54), we set $\Delta\boldsymbol{x} = -\boldsymbol{g}(\boldsymbol{x}; \mu)$ (a steepest descent direction).

An update of μ is an important part of interior point methods. Above all, $\mu \downarrow 0$ must hold, which is a main property of interior point methods. Moreover, rounding errors may cause that $z_k(\boldsymbol{x}; \mu) = F_k(\boldsymbol{x})$ when the value μ is small (because $F_k(\boldsymbol{x}) < z_k(\boldsymbol{x}; \mu) \leq F_k(\boldsymbol{x}) + m_k\mu$ and $F_k(\boldsymbol{x}) + m_k\mu \to F_k(\boldsymbol{x})$ if $\mu \downarrow 0$), which leads to a breakdown (division by $z_k(\boldsymbol{x}; \mu) - F_k(\boldsymbol{x}) = 0$) when computing $1/(z_k(\boldsymbol{x}; \mu) - F_k(\boldsymbol{x}))$. Therefore, we need to use a lower bound $\underline{\mu}$ for a barrier parameter (e.g. $\underline{\mu} = 10^{-8}$ when computing in double precision).

The efficiency of interior point methods also depends on the way of decreasing the value of a barrier parameter. The following heuristic procedures proved successful in practice, where $\underline{g}$ is a suitable constant.

Procedure A

Phase 1. If $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \geq \underline{g}$, then $\mu_{i+1} = \mu_i$ (the value of a barrier parameter is unchanged).

Phase 2. If $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| < \underline{g}$, then

$$\mu_{i+1} = \max\left\{\tilde{\mu}_{i+1},\ \underline{\mu},\ 10\,\varepsilon_M|F(\boldsymbol{x}_{i+1})|\right\}, \tag{11.58}$$

where $F(\boldsymbol{x}_{i+1}) = F_1(\boldsymbol{x}_{i+1}) + \cdots + F_m(\boldsymbol{x}_{i+1})$, ε_M is a machine precision, and

$$\tilde{\mu}_{i+1} = \min\left\{\max\{\lambda\mu_i,\ \mu_i/(\sigma\mu_i + 1)\}, \max\{\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2,\ 10^{-2k}\}\right\}. \tag{11.59}$$

The values $\underline{\mu} = 10^{-8}$, $\lambda = 0.85$, and $\sigma = 100$ are usually used.

Procedure B

Phase 1. If $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2 \geq \vartheta\mu_i$, then $\mu_{i+1} = \mu_i$ (the value of a barrier parameter is unchanged).

Phase 2. If $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2 < \vartheta\mu_i$, then

$$\mu_{i+1} = \max\{\underline{\mu}, \|\boldsymbol{g}_i(\boldsymbol{x}_i; \mu_i)\|^2\}. \tag{11.60}$$

The values $\underline{\mu} = 10^{-8}$ and $\vartheta = 0.1$ are usually used.

The choice of $\underline{g}$ in Procedure A is not critical. We can set $\underline{g} = \infty$ but a lower value is sometimes more advantageous. Formula (11.59) requires several comments. The first argument of the minimum controls the decreasing speed of the value of a barrier parameter which is linear (a geometric sequence) for small i (the term $\lambda\mu_i$) and sublinear (a harmonic sequence) for large i (the term $\mu_i/(\sigma\mu_i + 1)$). Thus, the second argument ensuring that the value μ is small in a neighborhood of a desired solution is mainly important for large i. This situation may appear if the gradient norm $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|$ is small even if $\boldsymbol{x}_i$ is far from a solution. The idea of Procedure B proceeds from the fact that a barrier function $\hat{B}(\boldsymbol{x}; \mu)$ should be minimized with a sufficient precision for a given value of a parameter μ.

The considerations up to now are summarized in Algorithm 11.1 introduced in the Appendix. This algorithm supposes that the matrix $A(\boldsymbol{x})$ is sparse. If it is dense, the algorithm is simplified because there is no symbolic decomposition.

11.2.5 Global Convergence

Now we prove the global convergence of the method realized by Algorithm 11.1.

Lemma 11.1 *Let numbers $z_k(\boldsymbol{x}; \mu)$, $1 \le k \le m$, be solutions of Eq. (11.44). Then*

$$\frac{\partial}{\partial \mu} z_k(\boldsymbol{x}; \mu) > 0, \ \ 1 \le k \le m, \ \ \frac{\partial}{\partial \mu} \hat{B}(\boldsymbol{x}; \mu) = -\sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x})).$$

Proof Differentiating (11.44) with respect to μ, one can write for $1 \le k \le m$

$$-\sum_{l=1}^{m_k} \frac{1}{z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x})} + \sum_{l=1}^{m_k} \frac{\mu}{(z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x}))^2} \frac{\partial}{\partial \mu} z_k(\boldsymbol{x}; \mu) = 0,$$

which after multiplication of μ together with (11.30) and (11.44) gives

$$\frac{\partial}{\partial \mu} z_k(\boldsymbol{x}; \mu) = \left(\sum_{l=1}^{m_k} \frac{\mu^2}{(z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x}))^2} \right)^{-1} = \left(\sum_{l=1}^{m_k} u_{kl}^2(\boldsymbol{x}; \mu) \right)^{-1} > 0.$$

Differentiating the function

$$\hat{B}(\boldsymbol{x}; \mu) = \sum_{k=1}^{m} z_k(\boldsymbol{x}; \mu) - \mu \sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(z_k(\boldsymbol{x}; \mu) - f_{kl}(\boldsymbol{x})) \tag{11.61}$$

and using (11.44) we obtain

$$\begin{aligned}\frac{\partial}{\partial\mu}\hat{B}(\boldsymbol{x};\mu) &= \sum_{k=1}^{m}\frac{\partial}{\partial\mu}z_k(\boldsymbol{x};\mu) - \sum_{k=1}^{m}\sum_{l=1}^{m_k}\log(z_k(\boldsymbol{x};\mu) - f_{kl}(\boldsymbol{x})) \\ &\quad - \sum_{k=1}^{m}\sum_{l=1}^{m_k}\frac{\mu}{z_k(\boldsymbol{x};\mu) - f_{kl}(\boldsymbol{x})}\frac{\partial}{\partial\mu}z_k(\boldsymbol{x};\mu) \\ &= \frac{\partial}{\partial\mu}z_k(\boldsymbol{x};\mu)\sum_{k=1}^{m}\left(1 - \sum_{l=1}^{m_k}\frac{\mu}{z_k(\boldsymbol{x};\mu) - f_{kl}(\boldsymbol{x})}\right) \\ &\quad - \sum_{k=1}^{m}\sum_{l=1}^{m_k}\log(z_k(\boldsymbol{x};\mu) - f_{kl}(\boldsymbol{x})) \\ &= -\sum_{k=1}^{m}\sum_{l=1}^{m_k}\log(z_k(\boldsymbol{x};\mu) - f_{kl}(\boldsymbol{x})).\end{aligned}$$

□

Lemma 11.2 *Let Assumption 11.1 be satisfied. Let* $\{\boldsymbol{x}_i\}$ *and* $\{\mu_i\}$, $i \in \mathbb{N}$, *be the sequences generated by Algorithm 11.1. Then the sequences* $\{\hat{B}(\boldsymbol{x}_i;\mu_i)\}$, $\{z(\boldsymbol{x}_i;\mu_i)\}$, *and* $\{F(\boldsymbol{x}_i)\}$, $i \in \mathbb{N}$, *are bounded. Moreover, there exists a constant* $L \geq 0$ *such that for* $i \in \mathbb{N}$ *it holds*

$$\hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) \leq \hat{B}(\boldsymbol{x}_{i+1};\mu_i) + L(\mu_i - \mu_{i+1}). \tag{11.62}$$

Proof

(a) We first prove boundedness from below. Using (11.61) and Assumption 11.1, one can write

$$\begin{aligned}\hat{B}(\boldsymbol{x};\mu) - \underline{F} &= \sum_{k=1}^{m}z_k(\boldsymbol{x};\mu) - \underline{F} - \mu\sum_{k=1}^{m}\sum_{l=1}^{m_k}\log(z_k(\boldsymbol{x};\mu) - f_{kl}(\boldsymbol{x})) \\ &\geq \sum_{k=1}^{m}\left(z_k(\boldsymbol{x};\mu) - \underline{F} - m_k\mu\log(z_k(\boldsymbol{x};\mu) - \underline{F})\right).\end{aligned}$$

A convex function $\psi(t) = t - m\mu\log(t)$ has a unique minimum at the point $t = m\mu$ because $\psi'(m\mu) = 1 - m\mu/m\mu = 0$. Thus, it holds

$$\hat{B}(\boldsymbol{x};\mu) \geq \underline{F} + \sum_{k=1}^{m}(m_k\mu - m_k\mu\log(m_k\mu))$$

$$\geq \underline{F} + \sum_{k=1}^{m} \min\{0,\ m_k \overline{\mu}(1 - \log(m_k \overline{\mu})\}$$

$$\geq \underline{F} + \sum_{k=1}^{m} \min\{0,\ m_k \overline{\mu}(1 - \log(2m_k \overline{\mu})\} \stackrel{\Delta}{=} \underline{B}.$$

Boundedness from below of sequences $\{z(\boldsymbol{x}_i; \mu_i)\}$ and $\{F(\boldsymbol{x}_i)\}$, $i \in \mathbb{N}$, follows from inequalities (11.45) and Assumption 11.1.

(b) Now we prove boundedness from above. Similarly as in (a) we can write

$$\hat{B}(\boldsymbol{x}; \mu) - \underline{F} \geq \sum_{k=1}^{m} \frac{z_k(\boldsymbol{x}; \mu) - \underline{F}}{2} + \sum_{k=1}^{m} \left(\frac{z_k(\boldsymbol{x}; \mu) - \underline{F}}{2} - m_k \mu \log(z_k(\boldsymbol{x}; \mu) - \underline{F}) \right).$$

A convex function $t/2 - m\mu \log(t)$ has a unique minimum at the point $t = 2m\mu$. Thus, it holds

$$\hat{B}(\boldsymbol{x}; \mu) \geq \sum_{k=1}^{m} \frac{z_k(\boldsymbol{x}; \mu) - \underline{F}}{2} + \underline{F} + \sum_{k=1}^{m} \min\{0,\ m\overline{\mu}(1 - \log(2m_k \overline{\mu}))\}$$

$$= \sum_{k=1}^{m} \frac{z_k(\boldsymbol{x}; \mu) - \underline{F}}{2} + \underline{B}$$

or

$$\sum_{k=1}^{m} (z_k(\boldsymbol{x}; \mu) - \underline{F}) \leq 2(\hat{B}(\boldsymbol{x}; \mu) - \underline{B}). \tag{11.63}$$

Using the mean value theorem and Lemma 11.1, we obtain

$$\hat{B}(\boldsymbol{x}_{i+1}; \mu_{i+1}) - \hat{B}(\boldsymbol{x}_{i+1}; \mu_i)$$

$$= \sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(z_k(\boldsymbol{x}_{i+1}; \tilde{\mu}_i) - f_{kl}(\boldsymbol{x}_{i+1}))(\mu_i - \mu_{i+1})$$

$$\leq \sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(z_k(\boldsymbol{x}_{i+1}; \mu_i) - f_{kl}(\boldsymbol{x}_{i+1}))(\mu_i - \mu_{i+1})$$

$$\leq \sum_{k=1}^{m} m_k \log(z_k(\boldsymbol{x}_{i+1}; \mu_i) - \underline{F})(\mu_i - \mu_{i+1}), \tag{11.64}$$

where $0 < \mu_{i+1} \leq \tilde{\mu}_i \leq \mu_i$. Since $\log(t) \leq t/e$ (where $e = \exp(1)$) for $t > 0$, we can write using inequalities (11.63), (11.64), and (11.45)

$$\begin{aligned}
\hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) - \underline{B} &\leq \hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \underline{B} \\
&\quad + \sum_{k=1}^{m} m_k \log(z_k(\boldsymbol{x}_{i+1};\mu_i) - \underline{F})(\mu_i - \mu_{i+1}) \\
&\leq \hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \underline{B} \\
&\quad + e^{-1} \sum_{k=1}^{m} m_k (z_k(\boldsymbol{x}_{i+1};\mu_i) - \underline{F})(\mu_i - \mu_{i+1}) \\
&\leq \hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \underline{B} \\
&\quad + 2e^{-1}\overline{m}(\hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \underline{B})(\mu_i - \mu_{i+1}) \\
&= (1 + \lambda\delta_i)(\hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \underline{B}) \\
&\leq (1 + \lambda\delta_i)(\hat{B}(\boldsymbol{x}_i;\mu_i) - \underline{B}),
\end{aligned}$$

where $\lambda = 2\overline{m}/e$ and $\delta_i = \mu_i - \mu_{i+1}$. Therefore,

$$\begin{aligned}
\hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) - \underline{B} &\leq \prod_{j=1}^{i}(1 + \lambda\delta_j)(\hat{B}(\boldsymbol{x}_1;\mu_1) - \underline{B}) \\
&\leq \prod_{i=1}^{\infty}(1 + \lambda\delta_i)(\hat{B}(\boldsymbol{x}_1;\mu_1) - \underline{B}) \qquad (11.65)
\end{aligned}$$

and since

$$\sum_{i=1}^{\infty} \lambda\delta_i = \lambda \sum_{i=1}^{\infty}(\mu_i - \mu_{i+1}) = \lambda(\overline{\mu} - \lim_{i\uparrow\infty} \mu_i) \leq \lambda\overline{\mu}$$

the expression on the right-hand side of (11.65) is finite. Thus, the sequence $\{\hat{B}(\boldsymbol{x}_i;\mu_i)\}$, $i \in \mathbb{N}$, is bounded from above and the sequences $\{z(\boldsymbol{x}_i;\mu_i)\}$ and $\{F(\boldsymbol{x}_i)\}$, $i \in \mathbb{N}$, are bounded from above as well by (11.63) and (11.45).

(c) Finally, we prove formula (11.62). Using (11.64) and (11.45) we obtain

$$\begin{aligned}
&\hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) - \hat{B}(\boldsymbol{x}_{i+1};\mu_i) \\
&\qquad \leq \sum_{k=1}^{m} m_k \log(z_k(\boldsymbol{x}_{i+1};\mu_i) - \underline{F})(\mu_i - \mu_{i+1})
\end{aligned}$$

$$\leq \sum_{k=1}^{m} m_k \log(F_k(\boldsymbol{x}_{i+1}) + m_k\mu_i - \underline{F})(\mu_i - \mu_{i+1})$$

$$\leq \sum_{k=1}^{m} m_k \log(\overline{F} + m_k\overline{\mu} - \underline{F})(\mu_i - \mu_{i+1})$$

$$\stackrel{\Delta}{=} L(\mu_i - \mu_{i+1})$$

(the existence of a constant $\overline{F}$ follows from boundedness of a sequence $\{F(\boldsymbol{x}_i)\}$, $i \in \mathbb{N}$), which together with (11.57) gives $\hat{B}(\boldsymbol{x}_{i+1}; \mu_{i+1}) \leq \hat{B}(\boldsymbol{x}_i; \mu_i) + L(\mu_i - \mu_{i+1})$, $i \in \mathbb{N}$. Thus, it holds

$$\hat{B}(\boldsymbol{x}_i; \mu_i) \leq \hat{B}(\boldsymbol{x}_1; \mu_1) + L(\mu_1 - \mu_i) \leq \hat{B}(\boldsymbol{x}_1; \mu_1) + L\overline{\mu} \stackrel{\Delta}{=} \overline{B}, \quad i \in N. \tag{11.66}$$

□

The upper bounds $\overline{g}$ and $\overline{G}$ are not used in Lemma 11.2, so Assumption 11.4 may not be satisfied. Thus, there exists an upper bound $\overline{F}$ (independent of $\overline{g}$ and $\overline{G}$) such that $F(\boldsymbol{x}_i) \leq \overline{F}$ for all $i \in \mathbb{N}$. This upper bound can be used in definition of a set $\mathcal{D}_F(\overline{F})$ in Assumption 11.4.

Lemma 11.3 *Let Assumption 11.4 and the assumptions of Lemma 11.2 be satisfied. Then, if we use Procedure A or Procedure B for an update of parameter μ, the values $\{\mu_i\}$, $i \in \mathbb{N}$, form a non-decreasing sequence such that $\mu_i \downarrow 0$.*

Proof The value of parameter μ is unchanged in the first phase of Procedure A or Procedure B. Since a function $\hat{B}(\boldsymbol{x}; \mu)$ is continuous, bounded from below by Lemma 11.2, and since inequality (11.56) is satisfied (with $\mu_i = \mu$), it holds $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu)\| \downarrow 0$ if phase 1 contains an infinite number of subsequent iterative steps [26, Section 3.2]. Thus, there exists a step (with index i) belonging to the first phase such that either $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu)\| < \underline{g}$ in Procedure A or $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu)\|^2 < \vartheta\mu$ in Procedure B. However, this is in contradiction with the definition of the first phase. Thus, there exists an infinite number of steps belonging to the second phase, where the value of parameter μ is decreased so that $\mu_i \downarrow 0$. □

Theorem 11.5 *Let assumptions of Lemma 11.3 be satisfied. Consider a sequence $\{\boldsymbol{x}_i\}$, $i \in \mathbb{N}$, generated by Algorithm 11.1, where $\underline{\delta} = \underline{\varepsilon} = \underline{\mu} = 0$. Then*

$$\lim_{i\uparrow\infty} \sum_{k=1}^{m} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}_i) u_{kl}(\boldsymbol{x}_i; \mu_i) = \boldsymbol{0}, \quad \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}_i; \mu_i) = 1,$$

$$z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i) \geq 0, \quad u_{kl}(\boldsymbol{x}_i; \mu_i) \geq 0,$$

$$\lim_{i\uparrow\infty} u_{kl}(\boldsymbol{x}_i; \mu_i)(z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i)) = 0$$

for $1 \leq k \leq m$ and $1 \leq l \leq m_k$.

Proof

(a) Equalities $\tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}_i;\mu_i) = 1$, $1 \le k \le m$, are satisfied by (11.44) because $\underline{\delta} = 0$. Inequalities $z_k(\boldsymbol{x}_i;\mu_i) - f_{kl}(\boldsymbol{x}_i) \ge 0$ and $u_{kl}(\boldsymbol{x}_i;\mu_i) \ge 0$ follow from formulas (11.45) and statement (11.50).

(b) Relations (11.56) and (11.62) yield

$$\begin{aligned}\hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) - \hat{B}(\boldsymbol{x}_i;\mu_i) &= (\hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) - \hat{B}(\boldsymbol{x}_{i+1};\mu_i)) \\ &\quad + (\hat{B}(\boldsymbol{x}_{i+1};\mu_i) - \hat{B}(\boldsymbol{x}_i;\mu_i)) \\ &\le L\,(\mu_i - \mu_{i+1}) - c\,\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|^2\end{aligned}$$

and since $\lim_{i\uparrow\infty} \mu_i = 0$ (Lemma 11.3), we can write by (11.66) that

$$\begin{aligned}\underline{B} &\le \lim_{i\uparrow\infty} \hat{B}(\boldsymbol{x}_{i+1};\mu_{i+1}) \\ &\le \hat{B}(\boldsymbol{x}_1;\mu_1) + L\sum_{i=1}^{\infty}(\mu_i - \mu_{i+1}) - c\sum_{i=1}^{\infty}\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|^2 \\ &\le \hat{B}(\boldsymbol{x}_1;\mu_1) + L\overline{\mu} - c\sum_{i=1}^{\infty}\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|^2 = \overline{B} - c\sum_{i=1}^{\infty}\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|^2.\end{aligned}$$

Thus, it holds

$$\sum_{i=1}^{\infty}\|\boldsymbol{g}(\boldsymbol{x}_i;\mu_i)\|^2 \le \frac{1}{c}(\overline{B} - \underline{B}) < \infty,$$

which gives $\boldsymbol{g}(\boldsymbol{x}_i;\mu_i) = \sum_{k=1}^{m}\sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}_i)u_{kl}(\boldsymbol{x}_i;\mu_i) \downarrow \boldsymbol{0}$.

(c) Let indices $1 \le k \le m$ and $1 \le l \le m_k$ are chosen arbitrarily. Using (11.50) and Lemma 11.3 we obtain

$$u_{kl}(\boldsymbol{x}_i;\mu_i)(z_k(\boldsymbol{x}_i;\mu_i) - f_{kl}(\boldsymbol{x}_i)) = \frac{\mu_i(z_k(\boldsymbol{x}_i;\mu_i) - f_{kl}(\boldsymbol{x}_i))}{z_k(\boldsymbol{x}_i;\mu_i) - f_{kl}(\boldsymbol{x}_i)} = \mu_i \downarrow 0.$$

□

Corollary 11.1 *Let the assumptions of Theorem 11.5 be satisfied. Then, every cluster point $\boldsymbol{x} \in \mathbb{R}^n$ of a sequence $\{\boldsymbol{x}_i\}$, $i \in \mathbb{N}$, satisfies necessary KKT conditions (11.8)–(11.9) where $\boldsymbol{z}$ and $\boldsymbol{u}$ (with elements z_k and u_{kl}, $1 \le k \le m$, $1 \le l \le m_k$) are cluster points of sequences $\{\boldsymbol{z}(\boldsymbol{x}_i;\mu_i)\}$ and $\{\boldsymbol{u}(\boldsymbol{x}_i;\mu_i)\}$, $i \in \mathbb{N}$.*

Now we will suppose that the values $\underline{\delta}$, $\underline{\varepsilon}$, and $\underline{\mu}$ are nonzero and show how a precise solution of the system of KKT equations will be after termination of computation.

Theorem 11.6 *Let the assumptions of Lemma 11.3 be satisfied. Consider a sequence $\{\boldsymbol{x}_i\}$, $i \in \mathbb{N}$, generated by Algorithm 11.1. Then, if the values $\underline{\delta} > 0$,*

$\underline{\varepsilon} > 0$, and $\underline{\mu} > 0$ are chosen arbitrarily, there exists an index $i \geq 1$ such that

$$\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \leq \underline{\varepsilon}, \quad \left|1 - \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}_i; \mu_i)\right| \leq \underline{\delta},$$
$$z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i) \geq 0, \quad u_{kl}(\boldsymbol{x}_i; \mu_i) \geq 0,$$
$$u_{kl}(\boldsymbol{x}_i; \mu_i)(z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i)) \leq \underline{\mu},$$

for $1 \leq k \leq m$ and $1 \leq l \leq m_k$.

Proof Inequality $|1 - \tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}_i; \mu_i)| \leq \underline{\delta}$ follows immediately from the fact that the equation $\tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}_i; \mu_i) = 1$, $1 \leq k \leq m$, is solved with precision $\underline{\delta}$. Inequalities $z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i) \geq 0$, $u_{kl}(\boldsymbol{x}_i; \mu_i) \geq 0$ follow from formulas (11.45) and statement (11.50) as in the proof of Theorem 11.5. Since $\mu_i \downarrow 0$ and $\boldsymbol{g}(\boldsymbol{x}_i; \mu_i) \downarrow \boldsymbol{0}$ by Lemma 11.3 and Theorem 11.5, there exists an index $i \geq 1$ such that $\mu_i \leq \underline{\mu}$ and $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \leq \underline{\varepsilon}$. Using (11.50) we obtain

$$u_{kl}(\boldsymbol{x}_i; \mu_i)(z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i)) = \frac{\mu_i (z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i))}{z_k(\boldsymbol{x}_i; \mu_i) - f_{kl}(\boldsymbol{x}_i)} = \mu_i \leq \underline{\mu}.$$

□

Theorem 11.5 is a standard global convergence result. If the stopping parameters $\underline{\delta}, \underline{\varepsilon}, \underline{\mu}$ are zero, the sequence of generated points converges to the point satisfying the KKT conditions for the equivalent nonlinear programming problem. Theorem 11.6 determines a precision of the obtained solution if the stopping parameters are nonzero.

11.2.6 Special Cases

Both the simplest and most widely considered generalized minimax problem is the classical minimax problem (11.10), when $m = 1$ in (11.4) (in this case we write m, z, $\boldsymbol{u}$, $\boldsymbol{v}$, U, V, A instead of m_1, z_1, $\boldsymbol{u}_1$, $\boldsymbol{v}_1$, U_1, V_1, A_1). For solving a classical minimax problem one can use Algorithm 11.1, where a major part of computation is very simplified. System of equations (11.38) is of order $n + 1$ and has the form

$$\begin{bmatrix} G(\boldsymbol{x}, z) + A(\boldsymbol{x})V(\boldsymbol{x}, z)A^T(\boldsymbol{x}) & -A(\boldsymbol{x})V(\boldsymbol{x}, z)\tilde{\boldsymbol{e}} \\ -\tilde{\boldsymbol{e}}^T V(\boldsymbol{x}, z)A^T(\boldsymbol{x}) & \tilde{\boldsymbol{e}}^T V(\boldsymbol{x}, z)\tilde{\boldsymbol{e}} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta z \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}, z) \\ \gamma(\boldsymbol{x}, z) \end{bmatrix}, \tag{11.67}$$

where $\boldsymbol{g}(\boldsymbol{x}, z) = A(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}, z)$, $\gamma(\boldsymbol{x}, z) = 1 - \tilde{\boldsymbol{e}}^T \boldsymbol{u}(\boldsymbol{x}, z)$, $V(\boldsymbol{x}, z) = U^2(\boldsymbol{x}, z)/\mu = \mathrm{diag}[u_1^2(\boldsymbol{x}, z), \ldots, u_m^2(\boldsymbol{x}, z)]/\mu$, and $u_k(\boldsymbol{x}, z) = \mu/(z - f_k(\boldsymbol{x}))$, $1 \leq k \leq m$. System

of equations (11.44) is reduced to one nonlinear equation

$$1 - \tilde{\boldsymbol{e}}^T \boldsymbol{u}(\boldsymbol{x}, z) = 1 - \sum_{k=1}^{m} \frac{\mu}{z - f_k(\boldsymbol{x})} = 0, \tag{11.68}$$

whose solution $z(\boldsymbol{x}; \mu)$ lies in the interval $F(\boldsymbol{x}) + \mu \le z(\boldsymbol{x}; \mu) \le F(\boldsymbol{x}) + m\mu$. To find this solution by robust methods from [14, 15] is not difficult. A barrier function has the form

$$\hat{B}(\boldsymbol{x}; \mu) = z(\boldsymbol{x}; \mu) - \mu \sum_{k=1}^{m} \log(z(\boldsymbol{x}; \mu) - f_k(\boldsymbol{x})) \tag{11.69}$$

with $\nabla \hat{B}(\boldsymbol{x}; \mu) = A(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}; \mu)$ and

$$\nabla^2 \hat{B}(\boldsymbol{x}; \mu) = G(\boldsymbol{x}; \mu) + A(\boldsymbol{x})V(\boldsymbol{x}; \mu)A^T(\boldsymbol{x}) - \frac{A(\boldsymbol{x})V(\boldsymbol{x}; \mu)\tilde{\boldsymbol{e}}\tilde{\boldsymbol{e}}^T V(\boldsymbol{x}; \mu)A^T(\boldsymbol{x})}{\tilde{\boldsymbol{e}}^T V(\boldsymbol{x}; \mu)\tilde{\boldsymbol{e}}}.$$

If we write system (11.67) in the form

$$\begin{bmatrix} W(\boldsymbol{x}, z) & -\boldsymbol{c}(\boldsymbol{x}, z) \\ -\boldsymbol{c}^T(\boldsymbol{x}, z) & \delta(\boldsymbol{x}, z) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta z \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}, z) \\ \gamma(\boldsymbol{x}, z) \end{bmatrix},$$

where $W(\boldsymbol{x}, z) = G(\boldsymbol{x}, z) + A(\boldsymbol{x})V(\boldsymbol{x}, z)A^T(\boldsymbol{x})$, $\boldsymbol{c}(\boldsymbol{x}, z) = A(\boldsymbol{x})V(\boldsymbol{x}, z)\tilde{\boldsymbol{e}}$ and $\delta(\boldsymbol{x}, z) = \tilde{\boldsymbol{e}}^T V(\boldsymbol{x}, z)\tilde{\boldsymbol{e}}$, then

$$\nabla^2 \hat{B}(\boldsymbol{x}; \mu) = W(\boldsymbol{x}; \mu) - \frac{\boldsymbol{c}(\boldsymbol{x}; \mu)\boldsymbol{c}^T(\boldsymbol{x}; \mu)}{\delta(\boldsymbol{x}; \mu)}.$$

Since

$$\begin{bmatrix} W & -\boldsymbol{c} \\ -\boldsymbol{c}^T & \delta \end{bmatrix}^{-1} = \begin{bmatrix} W^{-1} - W^{-1}\boldsymbol{c}\,\omega^{-1}\boldsymbol{c}^T H^{-1} & -W^{-1}\boldsymbol{c}\,\omega^{-1} \\ -\omega^{-1}\boldsymbol{c}^T W^{-1} & -\omega^{-1} \end{bmatrix},$$

where $\omega = \boldsymbol{c}^T W^{-1}\boldsymbol{c} - \delta$, we can write

$$\begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta z \end{bmatrix} = - \begin{bmatrix} W & -\boldsymbol{c} \\ -\boldsymbol{c}^T & \delta \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g} \\ \gamma \end{bmatrix} = \begin{bmatrix} W^{-1}(\boldsymbol{c}\,\Delta z - \boldsymbol{g}) \\ \Delta z \end{bmatrix},$$

where

$$\Delta z = \omega^{-1}(\boldsymbol{c}^T W^{-1}\boldsymbol{g} + \gamma).$$

The matrix W is sparse if the matrix $A(\boldsymbol{x})$ has sparse columns. If the matrix W is not positive definite, we can use the Gill–Murray decomposition

$$W + E = LL^T, \tag{11.70}$$

where E is a positive semidefinite diagonal matrix. Then we solve equations

$$LL^T \boldsymbol{p} = \boldsymbol{g}, \quad LL^T \boldsymbol{q} = \boldsymbol{c} \tag{11.71}$$

and set

$$\Delta z = \frac{\boldsymbol{c}^T \boldsymbol{p} + \gamma}{\boldsymbol{c}^T \boldsymbol{q} - \delta}, \quad \Delta \boldsymbol{x} = \boldsymbol{q}\, \Delta z - \boldsymbol{p}. \tag{11.72}$$

If we solve the classical minimax problem, Algorithm 11.1 must be somewhat modified. In Step 2, we solve only Eq. (11.68) instead of the system of equations (11.44). In Step 4, we determine a vector $\Delta \boldsymbol{x}$ by solving Eq. (11.71) and using relations (11.72). In Step 4, we use the barrier function (11.69) (the nonlinear equation (11.68) must be solved at the point $\boldsymbol{x} + \alpha \Delta \boldsymbol{x}$).

Minimization of a sum of absolute values, i.e., minimization of the function (11.14) is another important generalized minimax problem. In this case, a barrier function has the form

$$\begin{aligned} B_\mu(\boldsymbol{x}, z) &= \sum_{k=1}^m z_k - \mu \sum_{k=1}^m \log(z_k - f_k(\boldsymbol{x})) - \mu \sum_{k=1}^m \log(z_k + f_k(\boldsymbol{x})) \\ &= \sum_{k=1}^m z_k - \mu \sum_{k=1}^m \log(z_k^2 - f_k^2(\boldsymbol{x})), \end{aligned} \tag{11.73}$$

where $z_k > |f_k(\boldsymbol{x})|$, $1 \le k \le m$. Differentiating $B_\mu(\boldsymbol{x}, z)$ with respect to $\boldsymbol{x}$ and z we obtain the necessary conditions for an extremum

$$\begin{aligned} &\sum_{k=1}^m \frac{2\mu f_k(\boldsymbol{x})}{z_k^2 - f_k^2(\boldsymbol{x})}\, \boldsymbol{g}_k(\boldsymbol{x}) = \sum_{k=1}^m u_k(\boldsymbol{x}, z_k)\, \boldsymbol{g}_k(\boldsymbol{x}) = \boldsymbol{0}, \\ &u_k(\boldsymbol{x}, z_k) = \frac{2\mu f_k(\boldsymbol{x})}{z_k^2 - f_k^2(\boldsymbol{x})} \end{aligned} \tag{11.74}$$

and

$$1 - \frac{2\mu z_k}{z_k^2 - f_k^2(\boldsymbol{x})} = 1 - u_k(\boldsymbol{x}, z_k) \frac{z_k}{f_k(\boldsymbol{x})} = 0 \quad \Rightarrow \quad u_k(\boldsymbol{x}, z_k) = \frac{f_k(\boldsymbol{x})}{z_k}, \tag{11.75}$$

where $\boldsymbol{g}_k(\boldsymbol{x}) = \nabla f_k(\boldsymbol{x})$, $1 \le k \le m$, which corresponds to (11.31)–(11.32). Equations in (11.44) are quadratic of the form

$$\frac{2\mu z_k(\boldsymbol{x};\mu)}{z_k^2(\boldsymbol{x};\mu) - f_k^2(\boldsymbol{x})} = 1 \quad \Leftrightarrow \quad z_k^2(\boldsymbol{x};\mu) - f_k^2(\boldsymbol{x}) = 2\mu z_k(\boldsymbol{x};\mu), \tag{11.76}$$

where $1 \le k \le m$, and their solutions is given by

$$z_k(\boldsymbol{x};\mu) = \mu + \sqrt{\mu^2 + f_k^2(\boldsymbol{x})}, \quad 1 \le k \le m, \tag{11.77}$$

(the second solutions of quadratic equations (11.76) do not satisfy the condition $z_k > |f_k(\boldsymbol{x})|$, so the obtained vector $\boldsymbol{z}$ does not belong to a domain of $\tilde{B}_\mu(z)$). Using (11.75) and (11.77) we obtain

$$u_k(\boldsymbol{x};\mu) = u_k(\boldsymbol{x}, z_k(\boldsymbol{x};\mu)) = \frac{f_k(\boldsymbol{x})}{z_k(\boldsymbol{x};\mu)} = \frac{f_k(\boldsymbol{x})}{\mu + \sqrt{\mu^2 + f_k^2(\boldsymbol{x})}} \tag{11.78}$$

for $1 \le k \le m$ and

$$\begin{aligned}\hat{B}(\boldsymbol{x};\mu) &= B(\boldsymbol{x}, \boldsymbol{z}(\boldsymbol{x};\mu)) = \sum_{k=1}^{m} z_k(\boldsymbol{x};\mu) - \mu \sum_{k=1}^{m} \log(z_k^2(\boldsymbol{x};\mu) - f_k^2(\boldsymbol{x})) \\ &= \sum_{k=1}^{m} z_k(\boldsymbol{x};\mu) - \mu \sum_{k=1}^{m} \log(2\mu z_k(\boldsymbol{x};\mu)) \\ &= \sum_{k=1}^{m} \big[z_k(\boldsymbol{x};\mu) - \mu \log(z_k(\boldsymbol{x};\mu))\big] - \mu m \log(2\mu). \end{aligned} \tag{11.79}$$

Using these expressions, we can write formulas (11.47) and (11.48) in the form

$$\nabla \hat{B}(\boldsymbol{x};\mu) = \sum_{k=1}^{m} \boldsymbol{g}_k(\boldsymbol{x}) u_k(\boldsymbol{x};\mu) \tag{11.80}$$

and

$$\nabla^2 \hat{B}(\boldsymbol{x};\mu) = W(\boldsymbol{x};\mu) = \sum_{k=1}^{m} G_k(\boldsymbol{x}) u_k(\boldsymbol{x};\mu) + \sum_{k=1}^{m} \boldsymbol{g}_k(\boldsymbol{x}) v_k(\boldsymbol{x};\mu) \boldsymbol{g}_k^T(\boldsymbol{x}), \tag{11.81}$$

where

$$G_k(\boldsymbol{x}) = \nabla^2 f_k(\boldsymbol{x}), \quad v_k(\boldsymbol{x};\mu) = \frac{2\mu}{z_k^2(\boldsymbol{x};\mu) + f_k^2(\boldsymbol{x})}, \quad 1 \le k \le m. \tag{11.82}$$

A vector $\Delta\boldsymbol{x} \in \mathbb{R}^n$ is determined by solving the equation

$$\nabla^2 \hat{B}(\boldsymbol{x};\mu)\Delta\boldsymbol{x} = -\boldsymbol{g}(\boldsymbol{x};\mu), \tag{11.83}$$

where $\boldsymbol{g}(\boldsymbol{x};\mu) = \nabla\hat{B}(\boldsymbol{x};\mu) \neq 0$. From (11.83) and (11.81) it follows

$$(\Delta\boldsymbol{x})^T \boldsymbol{g}(\boldsymbol{x};\mu) = -(\Delta\boldsymbol{x})^T \nabla^2 \hat{B}(\boldsymbol{x};\mu)\Delta\boldsymbol{x} \le -(\Delta\boldsymbol{x})^T G(\boldsymbol{x};\mu)\Delta\boldsymbol{x},$$

so if a matrix $G(\boldsymbol{x};\mu)$ is positive definite, a matrix $\nabla\hat{B}(\boldsymbol{x};\mu)$ is positive definite as well (since a diagonal matrix $V(\boldsymbol{x};\mu)$ is positive definite by (11.82)) and $(\Delta\boldsymbol{x})^T \boldsymbol{g}(\boldsymbol{x};\mu) < 0$ holds (a direction vector $\Delta\boldsymbol{x}$ is descent for a function $\hat{B}(\boldsymbol{x};\mu)$).

If we minimize a sum of absolute values, Algorithm 11.1 needs to be somewhat modified. In Step 2, we solve quadratic equations (11.76) whose solutions are given by (11.77). In Step 4, we determine a vector $\Delta\boldsymbol{x}$ by solving Eq. (11.83), where matrix $\nabla^2\hat{B}(\boldsymbol{x};\mu)$ is given by (11.83). In Step 4, we use the barrier function (11.79).

11.3 Smoothing Methods

11.3.1 Basic Properties

Similarly as in Sect. 11.2.1 we will restrict ourselves to sums of maxima, where a mapping $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ is a sum of its arguments, so (11.4) holds. Smoothing methods for minimization of sums of maxima replace function (11.4) by a smoothing function

$$S(\boldsymbol{x};\mu) = \sum_{k=1}^{m} S_k(\boldsymbol{x};\mu), \tag{11.84}$$

where

$$\begin{aligned} S_k(\boldsymbol{x};\mu) &= \mu \log \sum_{l=1}^{m_k} \exp\left(\frac{f_{kl}(\boldsymbol{x})}{\mu}\right) \\ &= F_k(\boldsymbol{x}) + \mu \log \sum_{l=1}^{m_k} \exp\left(\frac{f_{kl}(\boldsymbol{x}) - F_k(\boldsymbol{x})}{\mu}\right), \end{aligned} \tag{11.85}$$

depending on a smoothing parameter $0 < \mu \leq \overline{\mu}$, which is successively minimized on $\mathbb{R}^n$ with $\mu \downarrow 0$. Since $f_{kl}(\boldsymbol{x}) \leq F_k(\boldsymbol{x})$, $1 \leq l \leq m_k$, and the equality arises for at least one index, at least one exponential function on the right-hand side of (11.85) has the value 1, so the logarithm is positive. Thus $F_k(\boldsymbol{x}) \leq S_k(\boldsymbol{x}; \mu) \leq F_k(\boldsymbol{x}) + \mu \log m_k$, $1 \leq k \leq m$, hold. Therefore

$$F(\boldsymbol{x}) \leq S(\boldsymbol{x}; \mu) \leq F(\boldsymbol{x}) + \mu \sum_{k=1}^{m} \log m_k, \tag{11.86}$$

so $S(\boldsymbol{x}; \mu) \to F(\boldsymbol{x})$ if $\mu \downarrow 0$.

Remark 11.15 Similarly as in Sect. 11.2.2 we will denote $\boldsymbol{g}_{kl}(\boldsymbol{x})$ and $G_{kl}(x)$ the gradients and Hessian matrices of functions $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, and

$$\boldsymbol{u}_k(\boldsymbol{x}; \mu) = \begin{bmatrix} u_{k1}(\boldsymbol{x}; \mu) \\ \vdots \\ u_{km_k}(\boldsymbol{x}; \mu) \end{bmatrix}, \quad \tilde{\boldsymbol{e}}_k = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix},$$

where

$$u_{kl}(\boldsymbol{x}; \mu) = \frac{\exp(f_{kl}(\boldsymbol{x})/\mu)}{\sum_{l=1}^{m_k} \exp(f_{kl}(\boldsymbol{x})/\mu)} = \frac{\exp((f_{kl}(\boldsymbol{x}) - F_k(\boldsymbol{x}))/\mu)}{\sum_{l=1}^{m_k} \exp((f_{kl}(\boldsymbol{x}) - F_k(\boldsymbol{x}))/\mu)}. \tag{11.87}$$

Thus, it holds $u_{kl}(\boldsymbol{x}; \mu) \geq 0$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, and

$$\tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}; \mu) = \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}; \mu) = 1. \tag{11.88}$$

Further, we denote $A_k(\boldsymbol{x}) = J_k^T(\boldsymbol{x}) = [\boldsymbol{g}_{k1}(\boldsymbol{x}), \ldots, \boldsymbol{g}_{km_k}(\boldsymbol{x})]$ and $U_k(\boldsymbol{x}; \mu) = \operatorname{diag}[u_{k1}(\boldsymbol{x}; \mu), \ldots, u_{km_k}(\boldsymbol{x}; \mu)]$ for $1 \leq k \leq m$.

Theorem 11.7 *Consider the smoothing function (11.84). Then*

$$\nabla S(\boldsymbol{x}; \mu) = \boldsymbol{g}(\boldsymbol{x}; \mu) \tag{11.89}$$

and

$$\begin{aligned} \nabla^2 S(\boldsymbol{x}; \mu) &= G(\boldsymbol{x}; \mu) + \frac{1}{\mu} \sum_{k=1}^{m} A_k(\boldsymbol{x}) U_k(\boldsymbol{x}; \mu) A_k^T(\boldsymbol{x}) \\ &\quad - \frac{1}{\mu} \sum_{k=1}^{m} A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}; \mu) (A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}; \mu))^T \\ &= G(\boldsymbol{x}; \mu) + \frac{1}{\mu} A(\boldsymbol{x}) U(\boldsymbol{x}; \mu) A^T(\boldsymbol{x}) - \frac{1}{\mu} C(\boldsymbol{x}; \mu) C(\boldsymbol{x}; \mu)^T \end{aligned} \tag{11.90}$$

where $\boldsymbol{g}(\boldsymbol{x};\mu) = \sum_{k=1}^{m} A_k(\boldsymbol{x})\boldsymbol{u}_k(\boldsymbol{x};\mu) = A(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x})$ *and*

$$G(\boldsymbol{x};\mu) = \sum_{k=1}^{m} G_k(\boldsymbol{x})\boldsymbol{u}_k(\boldsymbol{x};\mu), \quad A(\boldsymbol{x}) = [A_1(\boldsymbol{x}), \ldots, A_m(\boldsymbol{x})],$$
$$U(\boldsymbol{x};\mu) = \mathrm{diag}[U_1(\boldsymbol{x};\mu), \ldots, U_m(\boldsymbol{x};\mu)],$$
$$C(\boldsymbol{x};\mu) = [A_1(\boldsymbol{x})\boldsymbol{u}_1(\boldsymbol{x};\mu), \ldots, A_m(\boldsymbol{x})\boldsymbol{u}_m(\boldsymbol{x};\mu)].$$

Proof Obviously,

$$\nabla S(\boldsymbol{x};\mu) = \sum_{k=1}^{m} \nabla S_k(\boldsymbol{x};\mu), \quad \nabla^2 S(\boldsymbol{x};\mu) = \sum_{k=1}^{m} \nabla^2 S_k(\boldsymbol{x};\mu).$$

Differentiating functions (11.85) and using (11.87) we obtain

$$\begin{aligned}\nabla S_k(\boldsymbol{x};\mu) &= \frac{\mu}{\sum_{l=1}^{m_k} \exp(f_{kl}(\boldsymbol{x})/\mu)} \sum_{l=1}^{m_k} \frac{1}{\mu} \exp(f_{kl}(\boldsymbol{x})/\mu)\boldsymbol{g}_{kl}(\boldsymbol{x}) \\ &= \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x};\mu) = A_k(\boldsymbol{x})\boldsymbol{u}_k(\boldsymbol{x};\mu). \end{aligned} \tag{11.91}$$

Adding up these expressions yields (11.89). Further, it holds

$$\begin{aligned}\nabla u_{kl}(\boldsymbol{x};\mu) &= \frac{1}{\mu} \frac{\exp(f_{kl}(\boldsymbol{x})/\mu)}{\sum_{l=1}^{m_k} \exp(f_{kl}(\boldsymbol{x})/\mu)} \boldsymbol{g}_{kl}(\boldsymbol{x}) \\ &\quad - \frac{\exp(f_{kl}(\boldsymbol{x})/\mu)}{\left(\sum_{l=1}^{m_k} \exp(f_{kl}(\boldsymbol{x})/\mu)\right)^2} \sum_{l=1}^{m_k} \frac{1}{\mu} \exp(f_{kl}(\boldsymbol{x})/\mu)\boldsymbol{g}_{kl}(\boldsymbol{x}) \\ &= \frac{1}{\mu} u_{kl}(\boldsymbol{x};\mu)\boldsymbol{g}_{kl}(\boldsymbol{x}) - \frac{1}{\mu} u_{kl}(\boldsymbol{x};\mu) \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x};\mu)\boldsymbol{g}_{kl}(\boldsymbol{x}). \end{aligned} \tag{11.92}$$

Differentiating (11.91) and using (11.92) we obtain

$$\begin{aligned}\nabla^2 S_k(\boldsymbol{x};\mu) &= \sum_{l=1}^{m_k} G_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x};\mu) + \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) \nabla u_{kl}(\boldsymbol{x};\mu) \\ &= G_k(\boldsymbol{x};\mu) + \frac{1}{\mu} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x};\mu) \boldsymbol{g}_{kl}^T(\boldsymbol{x}) \end{aligned}$$

$$
\begin{aligned}
&\quad - \frac{1}{\mu} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x}; \mu) \left(\sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x}; \mu) \right)^T \\
&= G_k(\boldsymbol{x}; \mu) + \frac{1}{\mu} A_k(\boldsymbol{x}) U_k(\boldsymbol{x}; \mu) A_k^T(\boldsymbol{x}) \\
&\quad - \frac{1}{\mu} A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}; \mu) (A_k(\boldsymbol{x}) \boldsymbol{u}_k(\boldsymbol{x}; \mu))^T,
\end{aligned}
$$

where $G_k(\boldsymbol{x}; \mu) = \sum_{l=1}^{m_k} G_{kl}(\boldsymbol{x}) u_{kl}(\boldsymbol{x}; \mu)$. Adding up these expressions yields (11.90). □

Remark 11.16 Note that using (11.90) and the Schwarz inequality we obtain

$$
\begin{aligned}
\boldsymbol{v}^T \nabla^2 S(\boldsymbol{x}; \mu) \boldsymbol{v} &= \boldsymbol{v}^T G(\boldsymbol{x}; \mu) \boldsymbol{v} \\
&\quad + \frac{1}{\mu} \sum_{k=1}^{m} \left(\boldsymbol{v}^T A_k(\boldsymbol{x}) U_k(\boldsymbol{x}; \mu) A_k^T(\boldsymbol{x}) \boldsymbol{v} - \frac{(\boldsymbol{v}^T A_k(\boldsymbol{x}) U_k(\boldsymbol{x}; \mu) \tilde{\boldsymbol{e}}_k)^2}{\tilde{\boldsymbol{e}}_k^T U_k(\boldsymbol{x}; \mu) \tilde{\boldsymbol{e}}_k} \right) \\
&\geq \boldsymbol{v}^T G(\boldsymbol{x}; \mu) \boldsymbol{v},
\end{aligned}
$$

because $\tilde{\boldsymbol{e}}_k^T U_k(\boldsymbol{x}; \mu) \tilde{\boldsymbol{e}}_k = \tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}; \mu) = 1$, so the Hessian matrix $\nabla^2 S(\boldsymbol{x}; \mu)$ is positive definite if the matrix $G(\boldsymbol{x}; \mu)$ is positive definite.

Using Theorem 11.7, a step of the Newton method can be written in the form $\boldsymbol{x}_+ = \boldsymbol{x} + \alpha \Delta \boldsymbol{x}$ where

$$
\nabla^2 S(\boldsymbol{x}; \mu) \Delta \boldsymbol{x} = -\nabla S(\boldsymbol{x}; \mu),
$$

or

$$
\left(W(\boldsymbol{x}; \mu) - \frac{1}{\mu} C(\boldsymbol{x}; \mu) C^T(\boldsymbol{x}; \mu) \right) \Delta \boldsymbol{x} = -\boldsymbol{g}(\boldsymbol{x}; \mu), \tag{11.93}
$$

where

$$
W(\boldsymbol{x}; \mu) = G(\boldsymbol{x}; \mu) + \frac{1}{\mu} A(\boldsymbol{x}) U(\boldsymbol{x}; \mu) A^T(\boldsymbol{x}), \quad \boldsymbol{g}(\boldsymbol{x}; \mu) = A(\boldsymbol{x}) \boldsymbol{u}(\boldsymbol{x}; \mu). \tag{11.94}
$$

A matrix W in (11.94) has the same structure as a matrix W in (11.48) and, by Theorem 11.7, smoothing function (11.84) has similar properties as the barrier function (11.46). Thus, one can use an algorithm that is analogous to Algorithm 11.1 and considerations stated in Remark 11.12, where $S(\boldsymbol{x}; \mu)$ and $\nabla^2 S(\boldsymbol{x}; \mu)$ are used instead of $\hat{B}(\boldsymbol{x}; \mu)$ and $\nabla^2 \hat{B}(\boldsymbol{x}; \mu)$. It means that

$$
S(\boldsymbol{x}_{i+1}; \mu_i) - S(\boldsymbol{x}_i; \mu_i) \leq -c \|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2 \quad \text{for all } i \in \mathbb{N}, \tag{11.95}
$$

if Assumption 11.4 is satisfied and

$$S(\boldsymbol{x}_{i+1};\mu_i) - S(\boldsymbol{x}_i;\mu_i) \leq 0 \quad \text{for all } i \in \mathbb{N} \tag{11.96}$$

in remaining cases.

The considerations up to now are summarized in Algorithm 11.2 introduced in the Appendix. This algorithm differs from Algorithm 11.1 in that a nonlinear equation $\tilde{\boldsymbol{e}}^T \boldsymbol{u}(\boldsymbol{x};\mu) = 1$ need not be solved in Step 2 (because (11.88) follows from (11.87)), Eq. (11.93)–(11.94) instead of (11.71)–(11.72) are used in Step 4, and a barrier function $\hat{B}(\boldsymbol{x};\mu)$ is replaced with a smoothing function $S(\boldsymbol{x};\mu)$ in Step 6. Note that the parameter μ in (11.84) has different meaning than the same parameter in (11.46), so we could use another procedure for its update in Step 7. However, it is becoming apparent that using Procedure A or Procedure B is very efficient. On the other hand, it must be noted that using exponential functions in Algorithm 11.2 has certain disadvantages. Computation of the values of exponential functions is more time consuming than performing standard arithmetic operations and underflow may also happen (i.e. replacing nonzero values by zero values) if the value of a parameter μ is very small.

11.3.2 Global Convergence

Now we prove the global convergence of the smoothing method realized by Algorithm 11.2.

Lemma 11.4 *Choose a fixed vector $\boldsymbol{x} \in \mathbb{R}^n$. Then $S_k(\boldsymbol{x};\mu) : (0,\infty) \to \mathbb{R}$, $1 \leq k \leq m$, are nondecreasing convex functions of $\mu > 0$ and*

$$0 \leq \log \underline{m}_k \leq \frac{\partial}{\partial \mu} S_k(\boldsymbol{x};\mu) \leq \log m_k, \tag{11.97}$$

where $\underline{m}_k$ is a number of active functions (for which $f_{kl}(\boldsymbol{x}) = F_k(\boldsymbol{x})$) and

$$\frac{\partial}{\partial \mu} S_k(\boldsymbol{x};\mu) = \log \sum_{l=1}^{m_k} \exp\left(\frac{f_{kl}(\boldsymbol{x}) - F_k(\boldsymbol{x})}{\mu}\right) - \sum_{l=1}^{m_k} \left(\frac{f_{kl}(\boldsymbol{x}) - F_k(\boldsymbol{x})}{\mu}\right) u_{kl}(\boldsymbol{x};\mu). \tag{11.98}$$

Proof Denoting $\varphi_{kl}(\boldsymbol{x};\mu) = (f_{kl}(\boldsymbol{x}) - F_k(\boldsymbol{x}))/\mu \leq 0$, $1 \leq k \leq m$, so

$$\varphi'_{kl}(\boldsymbol{x};\mu) \stackrel{\Delta}{=} \frac{\partial}{\partial \mu} \varphi_{kl}(\boldsymbol{x};\mu) = -\frac{\varphi_{kl}(\boldsymbol{x};\mu)}{\mu} \geq 0,$$

we can write by (11.85) that

$$S_k(\boldsymbol{x};\mu) = F_k(\boldsymbol{x}) + \mu \log \sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu)$$

and

$$\begin{aligned}\frac{\partial}{\partial \mu} S_k(\boldsymbol{x};\mu) &= \log \sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu) + \mu \frac{\sum_{l=1}^{m_k} \varphi'_{kl}(\boldsymbol{x};\mu) \exp \varphi_{kl}(\boldsymbol{x};\mu)}{\sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu)} \\ &= \log \sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu) - \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) u_{kl}(\boldsymbol{x};\mu) \geq 0, \quad (11.99)\end{aligned}$$

because $\varphi_{kl}(\boldsymbol{x};\mu) \leq 0$, $u_{kl}(\boldsymbol{x};\mu) \geq 0$, $1 \leq k \leq m$, and $\varphi_{kl}(\boldsymbol{x};\mu) = 0$ holds for at least one index. Thus, functions $S_k(\boldsymbol{x};\mu)$, $1 \leq k \leq m$, are nondecreasing. Differentiating (11.87) with respect to μ we obtain

$$\begin{aligned}\frac{\partial}{\partial \mu} u_{kl}(\boldsymbol{x};\mu) &= -\frac{1}{\mu} \frac{\varphi_{kl}(\boldsymbol{x};\mu) \exp \varphi_{kl}(\boldsymbol{x};\mu)}{\sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu)} \\ &\quad + \frac{1}{\mu} \frac{\exp \varphi_{kl}(\boldsymbol{x};\mu)}{\sum_{l=1}^{k} m \exp \varphi_{kl}(\boldsymbol{x};\mu)} \frac{\sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) \exp \varphi_{kl}(\boldsymbol{x};\mu)}{\sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu)} \\ &= \frac{1}{\mu} u_{kl}(\boldsymbol{x};\mu) \left(-\varphi_{kl}(\boldsymbol{x};\mu) + \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) u_{kl}(\boldsymbol{x};\mu) \right). \quad (11.100)\end{aligned}$$

Differentiating (11.99) with respect to μ and using Eqs. (11.88) and (11.100) we can write

$$\begin{aligned}\frac{\partial^2}{\partial \mu^2} S_k(\boldsymbol{x};\mu) &= -\frac{1}{\mu} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) u_{kl}(\boldsymbol{x};\mu) \\ &\quad + \frac{1}{\mu} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) u_{kl}(\boldsymbol{x};\mu) - \frac{1}{\mu} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) \frac{\partial}{\partial \mu} u_{kl}(\boldsymbol{x};\mu) \\ &= -\frac{1}{\mu} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) \frac{\partial}{\partial \mu} u_{kl}(\boldsymbol{x};\mu) \\ &= \frac{1}{\mu^2} \left(\sum_{l=1}^{m_k} \varphi_{kl}^2(\boldsymbol{x};\mu) u_{kl}(\boldsymbol{x};\mu) \right) \left(\sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x};\mu) \right) \\ &\quad - \frac{1}{\mu^2} \left(\sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) u_{kl}(\boldsymbol{x};\mu) \right)^2 \geq 0,\end{aligned}$$

because

$$\left(\sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu)u_{kl}(\boldsymbol{x};\mu)\right)^2 = \left(\sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu)\sqrt{u_{kl}(\boldsymbol{x};\mu)}\sqrt{u_{kl}(\boldsymbol{x};\mu)}\right)^2$$
$$\leq \sum_{l=1}^{m_k} \varphi_{kl}^2(\boldsymbol{x};\mu)u_{kl}(\boldsymbol{x};\mu) \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x};\mu)$$

holds by the Schwarz inequality. Thus, functions $S_k(\boldsymbol{x};\mu)$, $1 \leq k \leq m$, are convex, so their derivatives $\frac{\partial}{\partial\mu}S_k(\boldsymbol{x};\mu)$ are nondecreasing. Obviously, it holds

$$\lim_{\mu\downarrow 0} \frac{\partial}{\partial\mu} S_k(\boldsymbol{x};\mu) = \lim_{\mu\downarrow 0} \log \sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu) - \lim_{\mu\downarrow 0} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu)u_{kl}(\boldsymbol{x};\mu)$$
$$= \log \underline{m}_k - \frac{1}{\underline{m}_k} \lim_{\mu\downarrow 0} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu) \exp \varphi_{kl}(\boldsymbol{x};\mu) = \log \underline{m}_k,$$

because $\varphi_{kl}(\boldsymbol{x};\mu) = 0$ if $f_{kl}(\boldsymbol{x}) = F_k(\boldsymbol{x})$ and $\lim_{\mu\downarrow 0} \varphi_{kl}(\boldsymbol{x};\mu) = -\infty$, $\lim_{\mu\downarrow 0} \varphi_{kl}(\boldsymbol{x};\mu) \exp \varphi_{kl}(\boldsymbol{x};\mu) = 0$ if $f_{kl}(\boldsymbol{x}) < F_k(\boldsymbol{x})$. Similarly, it holds

$$\lim_{\mu\uparrow\infty} \frac{\partial}{\partial\mu} S_k(\boldsymbol{x};\mu) = \lim_{\mu\uparrow\infty} \log \sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x};\mu) - \lim_{\mu\uparrow\infty} \sum_{l=1}^{m_k} \varphi_{kl}(\boldsymbol{x};\mu)u_{kl}(\boldsymbol{x};\mu)$$
$$= \log m,$$

because $\lim_{\mu\uparrow\infty} \varphi_{kl}(\boldsymbol{x};\mu) = 0$ and $\lim_{\mu\uparrow\infty} |u_{kl}(\boldsymbol{x};\mu)| \leq 1$ for $1 \leq k \leq m$. □

Lemma 11.5 *Let Assumptions 11.2 and 11.4 be satisfied. Then the values μ_i, $i \in \mathbb{N}$, generated by Algorithm 11.2, create a nonincreasing sequence such that $\mu_i \downarrow 0$.*

Proof Lemma 11.5 is a direct consequence of Lemma 11.3 because the same procedures for an update of a parameter μ are used and (11.95) holds. □

Theorem 11.8 *Let the assumptions of Lemma 11.5 be satisfied. Consider a sequence $\{\boldsymbol{x}_i\}$ $i \in \mathbb{N}$, generated by Algorithm 11.2, where $\underline{\varepsilon} = \underline{\mu} = 0$. Then*

$$\lim_{i\uparrow\infty} \sum_{k=1}^{m} \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}_i;\mu_i)\boldsymbol{g}_{kl}(\boldsymbol{x}_i) = \boldsymbol{0}, \quad \sum_{l=1}^{m_k} u_{kl}(\boldsymbol{x}_i;\mu_i) = 1$$

and

$$F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i) \geq 0, \quad u_{kl}(\boldsymbol{x}_i;\mu_i) \geq 0, \quad \lim_{i\uparrow\infty} u_{kl}(\boldsymbol{x}_i;\mu_i)(F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i)) = 0$$

for $1 \le k \le m$ *and* $1 \le l \le m_k$.

Proof

(a) Equations $\tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}_i; \mu_i) = 1$ for $1 \le k \le m$ follow from (11.88). Inequalities $F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i) \ge 0$ and $u_{kl}(\boldsymbol{x}_i; \mu_i) \ge 0$ for $1 \le k \le m$ and $1 \le l \le m_k$ follow from (11.4) and (11.87).

(b) Since $S_k(\boldsymbol{x}; \mu)$ are nondecreasing functions of the parameter μ by Lemma 11.4 and (11.95) holds, we can write

$$\underline{F} \le \sum_{k=1}^{m} F_k(\boldsymbol{x}_{i+1}) \le S(\boldsymbol{x}_{i+1}; \mu_{i+1}) \le S(\boldsymbol{x}_{i+1}; \mu_i)$$

$$\le S(\boldsymbol{x}_i; \mu_i) - c\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2 \le S(\boldsymbol{x}_1; \mu_1) - c \sum_{j=1}^{i} \|\boldsymbol{g}(\boldsymbol{x}_j; \mu_j)\|^2,$$

where $\underline{F} = \sum_{k=1}^{m} \underline{F}_k$ and $\underline{F}_k$, $1 \le k \le m$, are lower bounds from Assumption 11.2. Thus, it holds

$$\underline{F} \le \lim_{i \uparrow \infty} S(\boldsymbol{x}_{i+1}; \mu_{i+1}) \le S(\boldsymbol{x}_1; \mu_1) - c \sum_{i=1}^{\infty} \|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2,$$

or

$$\sum_{i=1}^{\infty} \|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\|^2 \le \frac{1}{c}(S(\boldsymbol{x}_1; \mu_1) - \underline{F}),$$

so $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \downarrow 0$, which together with inequalities $0 \le u_{kl}(\boldsymbol{x}_i; \mu_i) \le 1$, $1 \le k \le m$, $1 \le l \le m_k$, gives $\lim_{i \uparrow \infty} u_{kl}(\boldsymbol{x}_i; \mu_i) \boldsymbol{g}_{kl}(\boldsymbol{x}_i) = \boldsymbol{0}$.

(c) Let indices $1 \le k \le m$ and $1 \le l \le m_k$ be chosen arbitrarily. Using (11.87) we get

$$0 \le u_{kl}(\boldsymbol{x}_i; \mu_i)(F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i)) = -\mu_i \frac{\varphi_{kl}(\boldsymbol{x}_i; \mu_i) \exp \varphi_{kl}(\boldsymbol{x}_i; \mu_i)}{\sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x}_i; \mu_i)}$$

$$\le -\mu_i \varphi_{kl}(\boldsymbol{x}_i; \mu_i) \exp \varphi_{kl}(\boldsymbol{x}_i; \mu_i) \le \frac{\mu_i}{e},$$

where $\varphi_{kl}(\boldsymbol{x}_i; \mu_i)$, $1 \le k \le m$, $1 \le l \le m_k$, are functions used in the proof of Lemma 11.4, because

$$\sum_{l=1}^{m_k} \exp \varphi_{kl}(\boldsymbol{x}_i; \mu_i) \ge 1$$

and the function $t \exp t$ attains its minimal value $-1/e$ at the point $t = -1$. Since $\mu_i \downarrow 0$, we obtain $u_{kl}(\boldsymbol{x}_i; \mu_i)(F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i)) \downarrow 0$.

□

Corollary 11.2 *Let the assumptions of Theorem 11.8 be satisfied. Then every cluster point $\boldsymbol{x} \in \mathbb{R}^n$ of a sequence $\{\boldsymbol{x}_i\}$, $i \in \mathbb{N}$, satisfies the necessary KKT conditions (11.5) and (11.6), where $\boldsymbol{u}$ (with elements $\boldsymbol{u}_k$, $1 \leq k \leq m$) is a cluster point of a sequence $\{\boldsymbol{u}(\boldsymbol{x}_i; \mu_i)\}$, $i \in \mathbb{N}$.*

Now we will suppose that the values $\underline{\varepsilon}$ and $\underline{\mu}$ are nonzero and show how a precise solution of the system of KKT equations will be after termination of computation of Algorithm 11.2.

Theorem 11.9 *Let the assumptions of Theorem 11.5 be satisfied and let $\{\boldsymbol{x}_i\}$, $i \in \mathbb{N}$, be a sequence generated by Algorithm 11.2. Then, if the values $\underline{\varepsilon} > 0$ and $\underline{\mu} > 0$ are chosen arbitrarily, there exists an index $i \geq 1$ such that*

$$\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \leq \underline{\varepsilon}, \quad \tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}_i; \mu_i) = 1, \quad 1 \leq k \leq m,$$

and

$$F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i) \geq 0, \quad u_{kl}(\boldsymbol{x}_i; \mu_i) \geq 0, \quad u_{kl}(\boldsymbol{x}_i; \mu_i)(F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i)) \leq \frac{\underline{\mu}}{e}$$

for all $1 \leq k \leq m$ and $1 \leq l \leq m_k$.

Proof Equalities $\tilde{\boldsymbol{e}}_k^T \boldsymbol{u}_k(\boldsymbol{x}_i; \mu_i) = 1$, $1 \leq k \leq m$, follow from (11.88). Inequalities $F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i) \geq 0$ and $u_{kl}(\boldsymbol{x}_i; \mu_i) \geq 0$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, follow from (11.10) and (11.87). Since $\mu_i \downarrow 0$ holds by Lemma 11.5 and $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \downarrow 0$ holds by Theorem 11.8, there exists an index $i \geq 1$ such that $\mu_i \leq \underline{\mu}$ and $\|\boldsymbol{g}(\boldsymbol{x}_i; \mu_i)\| \leq \underline{\varepsilon}$. By (11.87), as in the proof of Theorem 11.8, one can write

$$u_{kl}(\boldsymbol{x}_i; \mu_i)(F_k(\boldsymbol{x}_i) - f_{kl}(\boldsymbol{x}_i)) \leq -\mu_i \varphi_{kl}(\boldsymbol{x}_i; \mu_i) \exp \varphi_{kl}(\boldsymbol{x}_i; \mu_i) \leq \frac{\mu_i}{e} \leq \frac{\underline{\mu}}{e}$$

for $1 \leq k \leq m$ and $1 \leq l \leq m_k$. □

Theorems 11.8 and 11.9 have the same meaning as Theorems 11.5 and 11.6 introduced in Sect. 11.2.5.

11.3.3 Special Cases

Both the simplest and most widely considered generalized minimax problem is the classical minimax problem (11.10), when $m = 1$ in (11.4) (in this case we write m and z instead of m_1 and z_1). For solving a classical minimax problem one can use Algorithm 11.2, where a major part of computation is very simplified. A step of the

Newton method can be written in the form $\boldsymbol{x}_+ = \boldsymbol{x} + \alpha\Delta\boldsymbol{x}$ where

$$\nabla^2 S(\boldsymbol{x};\mu)\Delta\boldsymbol{x} = -\nabla S(\boldsymbol{x};\mu),$$

or

$$\left(W(\boldsymbol{x};\mu) - \frac{1}{\mu}\boldsymbol{g}(\boldsymbol{x};\mu)\boldsymbol{g}^T(\boldsymbol{x};\mu)\right)\Delta\boldsymbol{x} = -\boldsymbol{g}(\boldsymbol{x};\mu), \tag{11.101}$$

where

$$W(\boldsymbol{x};\mu) = G(\boldsymbol{x};\mu) + \frac{1}{\mu}A(\boldsymbol{x})U(\boldsymbol{x};\mu)A^T(\boldsymbol{x}), \quad \boldsymbol{g}(\boldsymbol{x};\mu) = A(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x};\mu). \tag{11.102}$$

Since

$$\left(W - \frac{1}{\mu}\boldsymbol{g}\boldsymbol{g}^T\right)^{-1} = W^{-1} + \frac{W^{-1}\boldsymbol{g}\boldsymbol{g}^T W^{-1}}{\mu - \boldsymbol{g}^T W^{-1}\boldsymbol{g}}$$

holds by the Sherman–Morrison formula, the solution of system of equations (11.101) can be written in the form

$$\Delta\boldsymbol{x} = \frac{\mu}{\boldsymbol{g}^T W^{-1}\boldsymbol{g} - \mu}W^{-1}\boldsymbol{g}. \tag{11.103}$$

If a matrix W is not positive definite, it may be replaced with a matrix $LL^T = W+E$ obtained by the Gill–Murray decomposition described in [10]. Then, we solve an equation

$$LL^T\boldsymbol{p} = \boldsymbol{g}, \tag{11.104}$$

and set

$$\Delta\boldsymbol{x} = \frac{\mu}{\boldsymbol{g}^T\boldsymbol{p} - \mu}\boldsymbol{p}. \tag{11.105}$$

Minimization of a sum of absolute values, i.e., minimization of the function (11.14) is another important generalized minimax problem. In this case, a smoothing function has the form

$$\begin{aligned} S(\boldsymbol{x};\mu) &= F(\boldsymbol{x}) \\ &\quad + \mu\sum_{k=1}^{m}\log\left(\exp\left(-\frac{|f_k(\boldsymbol{x})| - f_k^+(\boldsymbol{x})}{\mu}\right) + \exp\left(-\frac{|f_k(\boldsymbol{x})| - f_k^-(\boldsymbol{x})}{\mu}\right)\right) \\ &= \sum_{k=1}^{m}|f_k(\boldsymbol{x})| + \mu\sum_{k=1}^{m}\log\left(1 + \exp\left(-\frac{2|f_k(\boldsymbol{x})|}{\mu}\right)\right), \end{aligned}$$

because $f_k^+(\boldsymbol{x}) = |f_k(\boldsymbol{x})|$ if $f_k(\boldsymbol{x}) \geq 0$ and $f_k^-(\boldsymbol{x}) = |f_k(\boldsymbol{x})|$ if $f_k(\boldsymbol{x}) \leq 0$, and by Theorem 11.7 we have

$$\nabla S(\boldsymbol{x};\mu) = \sum_{k=1}^m (\boldsymbol{g}_k^+ u_k^+ + \boldsymbol{g}_k^- u_k^-) = \sum_{k=1}^m \boldsymbol{g}_k (u_k^+ - u_k^-) = \sum_{k=1}^m \boldsymbol{g}_k u_k = \boldsymbol{g}(\boldsymbol{x};\mu),$$

$$\begin{aligned}\nabla^2 S(\boldsymbol{x};\mu) &= \sum_{k=1}^m G_k (u_k^+ - u_k^-) + \frac{1}{\mu} \sum_{k=1}^m \boldsymbol{g}_k \boldsymbol{g}_k^T (u_k^+ + u_k^-) \\ &\quad - \frac{1}{\mu} \sum_{k=1}^m \boldsymbol{g}_k \boldsymbol{g}_k^T (u_k^+ - u_k^-)^2 = G(\boldsymbol{x};\mu) + \frac{1}{\mu} \sum_{k=1}^m \boldsymbol{g}_k \boldsymbol{g}_k^T (1 - u_k^2),\end{aligned}$$

(because $u_k^+ + u_k^- = 1$), where $\boldsymbol{g}_k = \boldsymbol{g}_k(\boldsymbol{x})$,

$$\begin{aligned}u_k = u_k^+ - u_k^- &= \frac{\exp\left(-\frac{|f_k(\boldsymbol{x})| - f_k^+(\boldsymbol{x})}{\mu}\right) - \exp\left(-\frac{|f_k(\boldsymbol{x})| - f_k^-(\boldsymbol{x})}{\mu}\right)}{\exp\left(-\frac{|f_k(\boldsymbol{x})| - f_k^+(\boldsymbol{x})}{\mu}\right) + \exp\left(-\frac{|f_k(\boldsymbol{x})| - f_k^-(\boldsymbol{x})}{\mu}\right)} \\ &= \frac{1 - \exp\left(-\frac{2|f_k(\boldsymbol{x})|}{\mu}\right)}{1 + \exp\left(-\frac{2|f_k(\boldsymbol{x})|}{\mu}\right)} \operatorname{sign}(f_k(\boldsymbol{x})),\end{aligned}$$

and

$$1 - u_k^2 = \frac{4 \exp\left(-\frac{2|f_k(\boldsymbol{x})|}{\mu}\right)}{\left(1 + \exp\left(-\frac{2|f_k(\boldsymbol{x})|}{\mu}\right)\right)^2},$$

and where $\operatorname{sign}(f_k(\boldsymbol{x}))$ is a sign of a function $f_k(\boldsymbol{x})$.

11.4 Primal-Dual Interior Point Methods

11.4.1 Basic Properties

Primal interior point methods for solving nonlinear programming problems profit from the simplicity of obtaining and keeping a point in the interior of the feasible set (for generalized minimax problems, it suffices to set $z_k > F_k(\boldsymbol{x})$, $1 \leq k \leq m$). Minimization of a barrier function without constraints and a direct computation of multipliers u_{kl}, $1 \leq k \leq m$, $1 \leq l \leq m_k$, are basic features of these methods. Primal-dual interior point methods are intended for solving general nonlinear programming

problems, where it is usually impossible to assure validity of constraints. These methods guarantee feasibility of points by adding slack variables, which appear in a barrier term added to the objective function. Positivity of the slack variables is assured algorithmically (by a step length selection). Minimization of a barrier function with equality constraints and an iterative computation of the Lagrange multipliers (dual variables) are the main features of primal-dual interior point methods.

Consider function (11.4). As is mentioned in the introduction, minimization of this function is equivalent to the nonlinear programming problem

$$\begin{cases} \text{minimize} & \sum\limits_{k=1}^{m} z_k \\ \text{subject to} & f_{kl}(\boldsymbol{x}) \le z_k, \quad 1 \le k \le m, \quad 1 \le l \le m_k. \end{cases} \tag{11.106}$$

Using slack variables $s_{kl} > 0$, $1 \le k \le m$, $1 \le l \le m_k$, and a barrier function

$$B_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{s}) = \sum_{k=1}^{m} z_k - \mu \sum_{k=1}^{m} \sum_{l=1}^{m_k} \log(s_{kl}), \tag{11.107}$$

a solving of the problem (11.106) can be transformed to a successive solving of problems

$$\begin{cases} \text{minimize} & B_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{s}) \\ \text{subject to} & f_{kl}(\boldsymbol{x}) + s_{kl} - z_k = 0, \quad 1 \le k \le m, \quad 1 \le l \le m_k, \end{cases} \tag{11.108}$$

where $\mu \downarrow 0$. Necessary conditions for an extremum of the problem (11.108) have the form

$$\begin{aligned} & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = \sum_{k=1}^{m} \sum_{l=1}^{m_k} \boldsymbol{g}_{kl}(\boldsymbol{x}) u_{kl} = \boldsymbol{0}, \\ & 1 - \sum_{l=1}^{m_k} u_{kl} = 0, \quad 1 \le k \le m, \\ & u_{kl} s_{kl} - \mu = 0, \quad 1 \le k \le m, \quad 1 \le l \le m_k, \\ & f_{kl}(\boldsymbol{x}) + s_{kl} - z_k = 0, \quad 1 \le k \le m, \quad 1 \le l \le m_k, \end{aligned}$$

which is $n + m + 2\bar{m}$ equations for $n + m + 2\bar{m}$ unknowns (vectors $\boldsymbol{x}$, $\boldsymbol{z} = [z_k]$, $\boldsymbol{s} = [s_{kl}]$, $\boldsymbol{u} = [u_{kl}]$, $1 \le k \le m$, $1 \le l \le m_k$), where $\bar{m} = m_1 + \cdots + m_m$. Denote

$A(\boldsymbol{x}) = [A_1(\boldsymbol{x}), \ldots, A_m(\boldsymbol{x})]$, $\boldsymbol{f} = [f_{kl}]$, $S = \mathrm{diag}[s_{kl}]$, $U = \mathrm{diag}[u_{kl}]$, $1 \le k \le m$, $1 \le l \le m_k$, and

$$E = \begin{bmatrix} \tilde{\boldsymbol{e}}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{e}}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \tilde{\boldsymbol{e}}_m \end{bmatrix}, \quad \tilde{\boldsymbol{e}} = \begin{bmatrix} \tilde{\boldsymbol{e}}_1 \\ \tilde{\boldsymbol{e}}_2 \\ \vdots \\ \tilde{\boldsymbol{e}}_m \end{bmatrix}, \quad \boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

(matrices $A_k(\boldsymbol{x})$, vectors $\tilde{\boldsymbol{e}}_k$, and numbers z_k, $1 \le k \le m$, are defined in Sect. 11.2.2). Applying the Newton method to this system of nonlinear equations, we obtain a system of linear equations for increments (direction vectors) $\Delta\boldsymbol{x}$, $\Delta\boldsymbol{z}$, $\Delta\boldsymbol{s}$, $\Delta\boldsymbol{u}$. After arrangement and elimination

$$\Delta\boldsymbol{s} = -U^{-1}S(\boldsymbol{u} + \Delta\boldsymbol{u}) + \mu S^{-1}\tilde{\boldsymbol{e}}, \tag{11.109}$$

this system has the form

$$\begin{bmatrix} G(\boldsymbol{x},\boldsymbol{u}) & \boldsymbol{0} & A(\boldsymbol{x}) \\ \boldsymbol{0} & \boldsymbol{0} & -E^T \\ A^T(\boldsymbol{x}) & -E & -U^{-1}S \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{z} \\ \Delta\boldsymbol{u} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) \\ \tilde{\boldsymbol{e}} - E^T\boldsymbol{u} \\ \boldsymbol{f}(\boldsymbol{x}) - E\boldsymbol{z} + \mu U^{-1}\tilde{\boldsymbol{e}} \end{bmatrix}, \tag{11.110}$$

where $G(\boldsymbol{x},\boldsymbol{u}) = \sum_{k=1}^{m}\sum_{l=1}^{m_k} G_{kl}(\boldsymbol{x})u_{kl}$. Vector $\tilde{\boldsymbol{e}}$ in the equation $\tilde{\boldsymbol{e}} - E^T\boldsymbol{u} = \boldsymbol{0}$ has unit elements, but its dimension is different from the dimension of a vector $\tilde{\boldsymbol{e}}$ in (11.109).

For solving this linear system, we cannot advantageously use the structure of a generalized minimax problem (because substituting $z_k = F_k(\boldsymbol{x}) = \max_{1\le l\le m_k} f_{kl}(\boldsymbol{x})$ we would obtain a nonsmooth problem whose solution is much more difficult). Therefore, we need to deal with a general nonlinear programming problem. To simplify subsequent considerations, we use the notation $\tilde{\boldsymbol{x}} = [\boldsymbol{x}^T, z^T]^T$,

$$\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}},\boldsymbol{u}) = \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) \\ \tilde{\boldsymbol{e}} - E^T\boldsymbol{u} \end{bmatrix}, \quad \tilde{G}(\tilde{\boldsymbol{x}},\boldsymbol{u}) = \begin{bmatrix} G(\boldsymbol{x},\boldsymbol{u}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \quad \tilde{A}(\tilde{\boldsymbol{x}}) = \begin{bmatrix} A(\boldsymbol{x}) \\ -E^T \end{bmatrix}, \tag{11.111}$$

and write (11.110) in the form

$$\begin{bmatrix} \tilde{G}(\tilde{\boldsymbol{x}},\boldsymbol{u}) & \tilde{A}(\tilde{\boldsymbol{x}}) \\ \tilde{A}^T(\tilde{\boldsymbol{x}}) & -U^{-1}S \end{bmatrix} \begin{bmatrix} \Delta\tilde{\boldsymbol{x}} \\ \Delta\boldsymbol{u} \end{bmatrix} = - \begin{bmatrix} \tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}},\boldsymbol{u}) \\ \boldsymbol{c}(\tilde{\boldsymbol{x}}) + \mu U^{-1}\tilde{\boldsymbol{e}} \end{bmatrix}, \tag{11.112}$$

where $\boldsymbol{c}(\tilde{\boldsymbol{x}}) = \boldsymbol{f}(\boldsymbol{x}) - E\boldsymbol{z}$. This system of equations is more advantageous against systems (11.49) and (11.93) in that its matrix does not depend on the barrier parameter μ, so it is not necessary to use a lower bound $\underline{\mu}$. On the other hand, system (11.112) has a dimension $n + m + \bar{m}$, while systems (11.49) and (11.93)

have dimensions n. It would be possible to eliminate the vector $\Delta \boldsymbol{u}$, so the resulting system

$$(\tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) + \tilde{A}(\tilde{\boldsymbol{x}}) M^{-1} \tilde{A}^T(\tilde{\boldsymbol{x}})) \Delta \tilde{\boldsymbol{x}} = -\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) - \tilde{A}(\tilde{\boldsymbol{x}})(M^{-1} \boldsymbol{c}(\tilde{\boldsymbol{x}}) + \mu S^{-1} \tilde{\boldsymbol{e}}), \tag{11.113}$$

where $M = U^{-1} S$, would have dimension $n + m$ (i.e., $n + 1$ for classical minimax problems). Nevertheless, as follows from the equation $u_{kl} s_{kl} = \mu$, either $u_{kl} \downarrow 0$ or $s_{kl} \downarrow 0$ if $\mu \downarrow 0$, so some elements of a matrix M^{-1} may tend to infinity, which increases the condition number of system (11.113). Conversely, the solution of Eq. (11.112) is easier if the elements of a matrix M are small (if $M = 0$, we obtain the saddle point system, which can be solved by efficient iterative methods [1, 18]). Therefore, it is advantageous to split the constraints to active with $s_{kl} \le \tilde{\varepsilon} u_{kl}$ (we denote active quantities by $\hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}})$, $\hat{A}(\tilde{\boldsymbol{x}})$, $\hat{\boldsymbol{s}}$, $\Delta \hat{\boldsymbol{s}}$, $\hat{S}$, $\hat{\boldsymbol{u}}$, $\Delta \hat{\boldsymbol{u}}$, $\hat{U}$, $\hat{M} = \hat{U}^{-1} \hat{S}$) and inactive with $s_{kl} > \tilde{\varepsilon} u_{kl}$ (we denote inactive quantities by $\check{\boldsymbol{c}}(\tilde{\boldsymbol{x}})$, $\check{A}(\tilde{\boldsymbol{x}})$, $\check{\boldsymbol{s}}$, $\Delta \check{\boldsymbol{s}}$, $\check{S}$, $\check{\boldsymbol{u}}$, $\Delta \check{\boldsymbol{u}}$, $\check{U}$, $\check{M} = \check{U}^{-1} \check{S}$). Eliminating inactive equations from (11.112) we obtain

$$\Delta \check{\boldsymbol{u}} = \check{M}^{-1}(\check{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \check{A}(\tilde{\boldsymbol{x}})^T \Delta \tilde{\boldsymbol{x}}) + \mu \check{S}^{-1} \tilde{\boldsymbol{e}} \tag{11.114}$$

and

$$\begin{bmatrix} \hat{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) & \hat{A}(\tilde{\boldsymbol{x}}) \\ \hat{A}^T(\tilde{\boldsymbol{x}}) & -\hat{M} \end{bmatrix} \begin{bmatrix} \Delta \tilde{\boldsymbol{x}} \\ \Delta \hat{\boldsymbol{u}} \end{bmatrix} = - \begin{bmatrix} \hat{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) \\ \hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \mu \hat{U}^{-1} \tilde{\boldsymbol{e}} \end{bmatrix}, \tag{11.115}$$

where

$$\hat{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = G(\tilde{\boldsymbol{x}}, \boldsymbol{u}) + \check{A}(\tilde{\boldsymbol{x}}) \check{M}^{-1} \check{A}^T(\tilde{\boldsymbol{x}}),$$
$$\hat{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = \boldsymbol{g}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) + \check{A}(\tilde{\boldsymbol{x}})(\check{M}^{-1} \check{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \mu \check{S}^{-1} \tilde{\boldsymbol{e}}),$$

and $\hat{M} = \hat{U}^{-1} \hat{S}$ is a diagonal matrix of order $\hat{m}$, where $0 \le \hat{m} \le \bar{m}$ is the number of active constraints. Substituting (11.114) into (11.109) we can write

$$\Delta \hat{\boldsymbol{s}} = -\hat{M}(\hat{\boldsymbol{u}} + \Delta \hat{\boldsymbol{u}}) + \mu \hat{U}^{-1} \tilde{\boldsymbol{e}}, \quad \Delta \check{\boldsymbol{s}} = -(\check{\boldsymbol{c}} + \check{A}^T \Delta \tilde{\boldsymbol{x}} + \check{\boldsymbol{s}}). \tag{11.116}$$

The matrix of the linear system (11.115) is symmetric, but indefinite, so its Choleski decomposition cannot be determined. In this case, we use either dense [3] or sparse [6] Bunch–Parlett decomposition for solving this system. System (11.115) (especially if it is large and sparse) can be efficiently solved by iterative conjugate gradient method with indefinite preconditioner [20]. If the vectors $\Delta \tilde{\boldsymbol{x}}$ and $\Delta \hat{\boldsymbol{u}}$ are solutions of system (11.115), we determine vector $\Delta \check{\boldsymbol{u}}$ by (11.114) and vectors $\Delta \hat{\boldsymbol{s}}$, $\Delta \check{\boldsymbol{s}}$ by (11.116).

Having vectors $\Delta \tilde{\boldsymbol{x}}$, $\Delta \boldsymbol{s}$, $\Delta \boldsymbol{u}$, we need to determine a step length $\alpha > 0$ and set

$$\tilde{\boldsymbol{x}}_+ = \tilde{\boldsymbol{x}} + \alpha \Delta \tilde{\boldsymbol{x}}, \quad \boldsymbol{s}_+ = \boldsymbol{s}(\alpha), \quad \boldsymbol{u}_+ = \boldsymbol{u}(\alpha), \tag{11.117}$$

where $\boldsymbol{s}(\alpha)$ and $\boldsymbol{u}(\alpha)$ are vector functions such that $\boldsymbol{s}(\alpha) > 0$, $\boldsymbol{s}'(0) = \Delta\boldsymbol{s}$ and $\boldsymbol{u}(\alpha) > 0$, $\boldsymbol{u}'(0) = \Delta\boldsymbol{u}$. This step is not trivial, because we need to decrease both the value of the barrier function $\tilde{B}_\mu(\tilde{\boldsymbol{x}}, \boldsymbol{s}) = B_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{s})$ and the norm of constraints $\|\boldsymbol{c}(\tilde{\boldsymbol{x}})\|$, and also to assure positivity of vectors $\boldsymbol{s}$ and $\boldsymbol{u}$. We can do this in several different ways: using either the augmented Lagrange function [20, 21] or a bi-criterial filter [9, 29] or a special algorithm [11, 16]. In this section, we confine our attention to the augmented Lagrange function which has (for the problem (11.106)) the form

$$P(\alpha) = \tilde{B}_\mu(\tilde{\boldsymbol{x}} + \alpha\Delta\tilde{\boldsymbol{x}}, \boldsymbol{s}(\alpha)) + (\boldsymbol{u} + \Delta\boldsymbol{u})^T(\boldsymbol{c}(\tilde{\boldsymbol{x}} + \alpha\Delta\tilde{\boldsymbol{x}}) + \boldsymbol{s}(\alpha)) + \frac{\sigma}{2}\|\boldsymbol{c}(\tilde{\boldsymbol{x}} + \alpha\Delta\tilde{\boldsymbol{x}}) + \boldsymbol{s}(\alpha)\|^2, \tag{11.118}$$

where $\sigma \geq 0$ is a penalty parameter. The following theorem, whose proof is given in [20], holds.

Theorem 11.10 *Let $\boldsymbol{s} > 0$, $\boldsymbol{u} > 0$ and let vectors $\Delta\tilde{\boldsymbol{x}}$, $\Delta\hat{\boldsymbol{u}}$ be solutions of the linear system*

$$\begin{bmatrix} \hat{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) & \hat{A}(\tilde{\boldsymbol{x}}) \\ \hat{A}^T(\tilde{\boldsymbol{x}}) & -\hat{M} \end{bmatrix} \begin{bmatrix} \Delta\tilde{\boldsymbol{x}} \\ \Delta\hat{\boldsymbol{u}} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) \\ \hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \mu\hat{U}^{-1}\tilde{\boldsymbol{e}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r} \\ \hat{\boldsymbol{r}} \end{bmatrix}, \tag{11.119}$$

where $\boldsymbol{r}$ and $\hat{\boldsymbol{r}}$ are residual vectors, and let vectors $\Delta\check{\boldsymbol{u}}$ and $\Delta\boldsymbol{s}$ be determined by (11.114) and (11.116). Then

$$P'(0) = -(\Delta\tilde{\boldsymbol{x}})^T\tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u})\Delta\tilde{\boldsymbol{x}} - (\Delta\boldsymbol{s})^T M^{-1}\Delta\boldsymbol{s} - \sigma\|\boldsymbol{c}(\tilde{\boldsymbol{x}}) + \boldsymbol{s}\|^2 + (\Delta\tilde{\boldsymbol{x}})^T\boldsymbol{r} + \sigma(\hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \hat{\boldsymbol{s}})^T\hat{\boldsymbol{r}}. \tag{11.120}$$

If

$$\sigma > -\frac{(\Delta\tilde{\boldsymbol{x}})^T\tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u})\Delta\tilde{\boldsymbol{x}} + (\Delta\boldsymbol{s})^T M^{-1}\Delta\boldsymbol{s}}{\|\boldsymbol{c}(\tilde{\boldsymbol{x}}) + \boldsymbol{s}\|^2} \tag{11.121}$$

and if system (11.115) is solved in such a way that

$$(\Delta\tilde{\boldsymbol{x}})^T\boldsymbol{r} + \sigma(\hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \hat{\boldsymbol{s}})^T\hat{\boldsymbol{r}} < (\Delta\tilde{\boldsymbol{x}})^T\tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u})\Delta\tilde{\boldsymbol{x}} + (\Delta\boldsymbol{s})^T M^{-1}\Delta\boldsymbol{s} + \sigma(\|\boldsymbol{c}(\tilde{\boldsymbol{x}}) + \boldsymbol{s}\|^2), \tag{11.122}$$

then $P'(0) < 0$.

Inequality (11.122) is significant only if linear system (11.115) is solved iteratively and residual vectors $\boldsymbol{r}$ and $\hat{\boldsymbol{r}}$ are nonzero. If these vectors are zero, then (11.122) follows immediately from (11.121). Inequality (11.121) serves for determination of a penalty parameter, which should be as small as possible. If the matrix $\tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u})$ is positive semidefinite, then the right-hand side of (11.121) is negative and we can choose $\sigma = 0$.

11.4.2 Implementation

The algorithm of the primal-dual interior point method consists of four basic parts: determination of the matrix $G(\boldsymbol{x}, \boldsymbol{u})$ or its approximation, solving linear system (11.115), a step length selection, and an update of the barrier parameter μ. The matrix $G(\boldsymbol{x}, \boldsymbol{u})$ has form (11.33), so its approximation can be determined in the one of the ways introduced in Remark 11.13.

The linear system (11.115), obtained by determination and subsequent elimination of inactive constraints in the way described in the previous subsection, is solved either directly using the Bunch–Parlett decomposition or iteratively by the conjugate gradient method with the indefinite preconditioner

$$C = \begin{bmatrix} \hat{D} & \hat{A}(\tilde{\boldsymbol{x}}) \\ \hat{A}^T(\tilde{\boldsymbol{x}}) & -\hat{M} \end{bmatrix}, \tag{11.123}$$

where $\hat{D}$ is a positive definite diagonal matrix that approximates matrix $\hat{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u})$. An iterative process is terminated if residual vectors satisfy conditions (11.122) and

$$\|\boldsymbol{r}\| \le \omega \|\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u})\|, \quad \|\hat{\boldsymbol{r}}\| \le \omega \min\{\|\hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \mu \hat{U}^{-1}\tilde{\boldsymbol{e}}\|, \|\hat{\boldsymbol{c}}(\tilde{\boldsymbol{x}}) + \hat{\boldsymbol{s}}\|\},$$

where $0 < \omega < 1$ is a prescribed precision. The directional derivative $P'(0)$ given by (11.118) should be negative. There are two possibilities how this requirement can be achieved. We either determine the value $\sigma \ge 0$ satisfying inequality (11.121), which implies $P'(0) < 0$ if (11.122) holds (Theorem 11.10), or set $\sigma = 0$ and ignore inequality (11.122). If $P'(0) \ge 0$, we determine a diagonal matrix $\tilde{D}$ with elements

$$\begin{cases} \tilde{D}_{jj} = \underline{\Gamma}, & \text{if} \quad \frac{\|\tilde{\boldsymbol{g}}\|}{10}|\tilde{G}_{jj}| < \underline{\Gamma}, \\ \tilde{D}_{jj} = \frac{\|\tilde{\boldsymbol{g}}\|}{10}|\tilde{G}_{jj}|, & \text{if} \quad \underline{\Gamma} \le \frac{\|\tilde{\boldsymbol{g}}\|}{10}|\tilde{G}_{jj}| \le \overline{\Gamma}, \\ \tilde{D}_{jj} = \overline{\Gamma}, & \text{if} \quad \overline{\Gamma} < \frac{\|\tilde{\boldsymbol{g}}\|}{10}|\tilde{G}_{jj}|, \end{cases} \tag{11.124}$$

for $1 \le j \le n + m$, where $\tilde{\boldsymbol{g}} = \tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u})$ and $0 < \underline{\Gamma} < \overline{\Gamma}$, set $\tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = \tilde{D}$ and restart the iterative process by solving new linear system (11.115).

We use functions $\boldsymbol{s}(\alpha) = [s_{kl}(\alpha)]$, $\boldsymbol{u}(\alpha) = [u_{kl}(\alpha)]$, where $s_{kl}(\alpha) = s_{kl} + \alpha_{s_{kl}} \Delta s_{kl}$, $u_{kl}(\alpha) = u_{kl} + \alpha_{u_{kl}} \Delta u_{kl}$ and

$$\begin{cases} \alpha_{s_{kl}} = \alpha, & \text{if} \quad \Delta s_{kl} \ge 0, \\ \alpha_{skl} = \min\{\alpha, -\gamma \frac{s_{kl}}{\Delta s_{kl}}\}, & \text{if} \quad \Delta s_{kl} < 0, \\ \alpha_{u_{kl}} = \alpha, & \text{if} \quad \Delta u_{kl} \ge 0, \\ \alpha_{u_{kl}} = \min\{\alpha, -\gamma \frac{u_{kl}}{\Delta u_{kl}}\}, & \text{if} \quad \Delta u_{kl} < 0 \end{cases}$$

when choosing a step length using the augmented Lagrange function. A parameter $0 < \gamma < 1$ (usually $\gamma = 0.99$) assures the positivity of vectors $\boldsymbol{s}_+$ and $\boldsymbol{u}_+$ in (11.117). A parameter $\alpha > 0$ is chosen to satisfy the inequality $P(\alpha) - P(0) \le \varepsilon_1 \alpha P'(0)$, which is possible because $P'(0) < 0$ and a function $P(\alpha)$ is continuous.

After finishing the iterative step, a barrier parameter μ should be updated. There exist several heuristic procedures for this purpose. The following procedure proposed in [28] seems to be very efficient.

Procedure C

Compute the centrality measure

$$\varrho = \frac{\bar{m}\ \min_{kl}\{s_{kl}u_{kl}\}}{\boldsymbol{s}^T\boldsymbol{u}},$$

where $\bar{m} = m_1 + \cdots + m_m$ and $1 \le k \le m$, $1 \le l \le m_k$. Compute the value

$$\lambda = 0.1 \min\left\{0.05\frac{1-\varrho}{\varrho}, 2\right\}^3$$

and set $\mu = \lambda \boldsymbol{s}^T\boldsymbol{u}/\bar{m}$.

The considerations up to now are summarized in Algorithm 11.3 introduced in the Appendix.

11.5 Numerical Experiments

The methods studied in this contribution were tested by using two collections of test problems TEST14 and TEST15 described in [19], which are the parts of the UFO system [24] and can be downloaded from the web-page www.cs.cas.cz/luksan/test.html. Both these collections contain 22 problems with functions $f_k(\boldsymbol{x})$, $1 \le k \le m$, $\boldsymbol{x} \in \mathbb{R}^n$, where n is an input parameter and $m \ge n$ depends on n (we have used the values $n = 100$ and $n = 1000$ for numerical experiments). Functions $f_k(\boldsymbol{x})$, $1 \le k \le m$, have a sparse structure (the Jacobian matrix of a mapping $\boldsymbol{f}(\boldsymbol{x})$ is sparse), so sparse matrix decompositions can be used for solving linear equation systems.

The tested methods, whose results are reported in Tables 11.1, 11.2, 11.3, 11.4, and 11.5 introduced in the Appendix, are denoted by five letters. The first pair of letters gives the problem type: either a classical minimax MX (when a function $F(\boldsymbol{x})$ has form (11.10) or $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_\infty$ holds) or a sum of absolute values SA (when $F(\boldsymbol{x}) = \|f(\boldsymbol{x})\|_1$ holds). Further two letters specify the method used:

`PI` –the primal interior point method (Sect. 11.2),
`SM` –the smoothing method (Sect. 11.3),
`DI` –the primal-dual interior point method (Sect. 11.4).

The last letter denotes the procedure for updating a barrier parameter μ (procedures A and B are described in Sect. 11.2.4 and procedure C is described in Sect. 11.4.2).

Table 11.1 TEST14 (minimization of maxima) — 22 problems

	Newton methods: n=100						Variable metric methods: n=100					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
MXPI-A	2232	7265	11575	0.74	4	-	2849	5078	2821	0.32	2	-
MXPI-B	2184	5262	9570	0.60	1	-	1567	2899	1589	0.24	1	-
MXSM-A	3454	11682	21398	1.29	5	-	4444	12505	4465	1.03	-	-
MXSM-B	10241	36891	56399	4.15	3	-	8861	32056	8881	2.21	1	1
MXDI-C	1386	2847	14578	0.90	2	-	2627	5373	2627	0.96	3	-
	Newton methods: n=1000						Variable metric methods: n=1000					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
MXPI-A	1386	3735	7488	5.58	4	-	3237	12929	3258	5.91	6	-
MXPI-B	3153	6885	12989	9.03	4	-	1522	3287	1544	2.68	5	-
MXSM-A	10284	30783	82334	54.38	7	-	4221	9519	4242	8.00	8	-
MXSM-B	18279	61180	142767	87.76	6	-	13618	54655	13639	45.10	9	1
MXDI-C	3796	6677	48204	49.95	6	-	2371	5548	2371	18.89	3	-

Table 11.2 TEST14 (L_∞-approximation) — 22 problems

	Newton methods: n=100						Variable metric methods: n=100					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
MXPI-A	2194	5789	10553	0.67	3	-	2890	5049	2912	0.48	1	-
MXPI-B	6767	17901	39544	3.79	4	-	1764	3914	1786	0.37	2	-
MXSM-A	3500	9926	23568	1.79	7	-	8455	23644	8476	4.69	4	-
MXSM-B	15858	48313	92486	8.33	5	-	9546	34376	9566	2.59	9	1
MXDI-C	1371	2901	11580	1.12	8	-	2467	5130	2467	1.59	3	-
	Newton methods: n=1000						Variable metric methods: n=1000					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
MXPI-A	4110	14633	20299	18.89	4	-	1549	2636	1571	2.51	3	-
MXPI-B	6711	31618	29939	30.73	7	-	1992	6403	2013	4.96	4	-
MXSM-A	9994	24333	88481	67.45	11	-	6164	15545	6185	29.37	8	-
MXSM-B	23948	84127	182604	149.63	8	-	24027	92477	24048	132.08	8	1
MXDI-C	3528	9084	26206	49.78	12	-	1932	2845	1932	18.73	5	-

The columns of all tables correspond to two classes of methods. The Newton methods use approximations of the Hessian matrices of the Lagrange function obtained by gradient differences [4] and variable metric methods update approximations of the Hessian matrices of the partial functions by the methods belonging to the Broyden family [12] (Remark 11.13).

Table 11.3 TEST15 (L_∞-approximation) — 22 problems

	Newton methods: n=100						Variable metric methods: n=100					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
MXPI-A	15525	20272	55506	4.41	1	-	6497	8204	6518	1.37	3	-
MXPI-B	7483	17999	27934	3.27	5	-	1764	7598	2488	0.74	2	-
MXSM-A	17574	29780	105531	11.09	4	-	9879	15305	9900	5.95	-	-
MXSM-B	13446	29249	81938	6.80	9	1	9546	34376	9566	2.59	3	-
MXDI-C	980	1402	7356	0.79	1	-	1179	1837	1179	1.06	2	-
	Newton methods: n=1000						Variable metric methods: n=1000					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
MXPI-A	10325	15139	32422	39.30	6	-	6484	9904	6502	13.77	2	-
MXPI-B	14836	30724	46864	68.70	10	-	7388	15875	7409	19.98	3	-
MXSM-A	11722	24882	69643	61.65	10	-	6659	12824	6681	41.55	8	-
MXSM-B	13994	31404	86335	78.45	9	1	15125	25984	15147	61.62	10	-
MXDI-C	1408	2406	10121	15.63	6	-	2228	3505	2228	35.13	10	-

Table 11.4 TEST14 (L_1-approximation) — 22 problems

	Newton methods: n=100						Variable metric methods: n=100					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
SAPI-A	1647	5545	8795	0.63	5	-	12265	23579	12287	1.37	2	1
SAPI-B	1957	7779	10121	0.67	6	-	4695	6217	10608	0.67	3	-
SASM-A	1677	4505	16079	0.74	3	-	20025	27369	20047	2.83	4	-
SASM-B	2389	8085	23366	1.18	4	-	5656	11637	5678	1.02	2	-
SADI-C	4704	13012	33937	4.16	7	1	6547	7012	6547	9.18	8	-
	Newton methods: n=1000						Variable metric methods: n=1000					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
SAPI-A	7592	19621	46100	15.39	4	-	22277	36610	22298	19.09	7	1
SAPI-B	9067	35463	56292	19.14	6	-	16650	35262	16672	14.47	6	1
SASM-A	5696	13534	41347	15.28	4	-	20020	30736	20042	23.05	5	1
SASM-B	8517	30736	57878	23.60	6	-	18664	28886	18686	18.65	5	1
SADI-C	6758	11011	47960	94.78	11	1	13123	14610	13124	295.46	8	2

The tables contain total numbers of iterations NIT, function evaluations NFV, gradient evaluations NFG, and also the total computational time, the number of problems with the value $\overline{\Delta}$ decreased and the number of failures (the number of unsolved problems). The decrease of the maximum step length $\overline{\Delta}$ is used for three reasons. First, too large steps may lead to overflows if arguments of functions (roots,

Table 11.5 TEST15 (L_1-approximation) — 22 problems

	Newton methods: n=100						Variable metric methods: n=100					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
SAPI-A	15760	21846	58082	4.24	8	-	39469	58157	39486	6.28	4	1
SAPI-B	4592	17050	17778	1.46	5	-	5932	25035	5952	1.48	6	1
SASM-A	10098	14801	610511	3.54	5	-	9162	28421	9184	3.65	6	1
SASM-B	4528	14477	290379	2.94	8	-	3507	9036	3528	1.27	6	-
SADI-C	877	1373	6026	0.84	3	-	15528	15712	15529	14.49	5	1
	Newton methods: n=1000						Variable metric methods: n=1000					
Method	NIT	NFV	NFG	Time	Δ	Fail	NIT	NFV	NFG	Time	Δ	Fail
SAPI-A	18519	39319	70951	61.04	5	-	27308	44808	27327	36.64	4	1
SAPI-B	12405	57969	43189	55.06	7	-	12712	32179	12731	21.48	8	1
SASM-A	19317	32966	113121	62.65	8	-	22264	42908	22284	62.46	7	1
SASM-B	14331	33572	86739	57.56	6	-	12898	42479	12919	47.05	7	1
SADI-C	2093	3681	12616	20.01	3	1	23957	28000	23960	186.92	5	3

logarithms, exponentials) lie outside of their definition domain. Second, the change of $\overline{\Delta}$ can affect the finding of a suitable (usually global) minimum. Finally, it can prevent from achieving a domain in which the objective function has bad behavior leading to a loss of convergence. The number of times the step length has decreased is in some sense a symptom of robustness (a lower number corresponds to higher robustness).

Several conclusions can be done from the results stated in these tables.

- The smoothing methods are less efficient than the primal interior point methods. For testing the smoothing methods, we had to use the value $\underline{\mu} = 10^{-6}$, while the primal interior methods work well with the smaller value $\underline{\mu} = 10^{-8}$, which gives more precise results.
- The primal-dual interior point methods are slower than the primal interior point methods, especially for the reason that system of equations (11.115) is indefinite, so we cannot use the Choleski (or the Gill–Murray [10]) decomposition. If the matrix of linear system (11.115) is large and sparse, we can use the Bunch–Parlett decomposition [6]. In this case, a large fill-in of new nonzero elements (and thus to overflow of the operational memory or large extension of the computational time) may appear. In this case, we can also use the iterative conjugate gradient method with an indefinite preconditioner [18], however, the ill-conditioned systems can require a large number of iterations and thus a large computational time.
- It cannot be uniquely decided whether Procedure A is better than Procedure B. The Newton methods usually work better with Procedure A while the variable metric methods are more efficient with Procedure B.

- The variable metric methods are usually faster because it is not necessary to determine the elements of the Hessian matrix of the Lagrange function by gradient differences. The Newton methods seem to be more robust (especially in case of L_1-approximation).

Appendix

Algorithm 11.1: Primal interior point method

Data: A tolerance for the gradient norm of the Lagrange function $\underline{\varepsilon} > 0$. A precision for determination of a minimax vector $\underline{\delta} > 0$. Bounds for a barrier parameter $0 < \underline{\mu} < \overline{\mu}$. Coefficients for decrease of a barrier parameter $0 < \lambda < 1$, $\sigma > 1$ (or $0 < \vartheta < 1$). A tolerance for a uniform descent $\varepsilon_0 > 0$. A tolerance for a step length selection $\varepsilon_1 > 0$. A maximum step length $\overline{\Delta} > 0$.

Input. A sparsity pattern of the matrix $A(\boldsymbol{x}) = [A_1(\boldsymbol{x}), \ldots, A_m(\boldsymbol{x})]$. A starting point $\boldsymbol{x} \in \mathbb{R}^n$.

Step 1. *(Initiation)* Choose $\mu \leq \overline{\mu}$. Determine a sparse structure of the matrix $W = W(\boldsymbol{x}; \mu)$ from the sparse structure of the matrix $A(\boldsymbol{x})$ and perform a symbolic decomposition of the matrix W (described in [2, Section 1.7.4]). Compute values $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, values $F_k(\boldsymbol{x}) = \max_{1 \leq l \leq m_k} f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, and the value of objective function (11.4). Set $r = 0$ (restart indicator).

Step 2. *(Termination)* Solve nonlinear equations (11.44) with precision $\underline{\delta}$ to obtain a minimax vector $\boldsymbol{z}(\boldsymbol{x}; \mu)$ and a vector of Lagrange multipliers $\boldsymbol{u}(\boldsymbol{x}; \mu)$. Determine a matrix $A = A(\boldsymbol{x})$ and a vector $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x}; \mu) = A(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}; \mu)$. If $\mu \leq \underline{\mu}$ and $\|\boldsymbol{g}\| \leq \underline{\varepsilon}$, terminate the computation.

Step 3. *(Hessian matrix approximation)* Set $G = G(\boldsymbol{x}; \mu)$ or compute an approximation G of the Hessian matrix $G(\boldsymbol{x}; \mu)$ using gradient differences or using quasi-Newton updates (Remark 11.13).

Step 4. *(Direction determination)* Determine a matrix $\nabla^2 \hat{B}(\boldsymbol{x}; \mu)$ by (11.48) and a vector $\Delta\boldsymbol{x}$ by solving Eq. (11.49) with the right-hand side defined by (11.47).

Step 5. *(Restart)* If $r = 0$ and (11.54) does not hold, set $G = I$, $r = 1$ and go to Step 4. If $r = 1$ and (11.54) does not hold, set $\Delta\boldsymbol{x} = -\boldsymbol{g}$. Set $r = 0$.

Step 6. *(Step length selection)* Determine a step length $\alpha > 0$ satisfying inequalities (11.55) (for a barrier function $\hat{B}(\boldsymbol{x}; \mu)$ defined by (11.46)) and $\alpha \leq \overline{\Delta}/\|\Delta\boldsymbol{x}\|$. Note that nonlinear equations (11.44) are solved at the point $\boldsymbol{x} + \alpha\Delta\boldsymbol{x}$. Set $\boldsymbol{x} := \boldsymbol{x} + \alpha\Delta\boldsymbol{x}$. Compute values $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, values $F_k(\boldsymbol{x}) = \max_{1 \leq l \leq m_k} f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, and the value of objective function (11.4).

Step 7. *(Barrier parameter update)* Determine a new value of a barrier parameter $\mu \geq \underline{\mu}$ using Procedure A or Procedure B. Go to Step 2.

The values $\underline{\varepsilon} = 10^{-6}$, $\underline{\delta} = 10^{-6}$, $\underline{\mu} = 10^{-8}$, $\overline{\mu} = 1$, $\lambda = 0.85$, $\sigma = 100$, $\vartheta = 0.1$, $\varepsilon_0 = 10^{-8}$, $\varepsilon_1 = 10^{-4}$, and $\overline{\Delta} = 1000$ were used in our numerical experiments.

Algorithm 11.2: Smoothing method

Data: A tolerance for the gradient norm of the smoothing function $\underline{\varepsilon} > 0$. Bounds for a smoothing parameter $0 < \underline{\mu} < \overline{\mu}$. Coefficients for decrease of a smoothing parameter $0 < \lambda < 1$, $\sigma > 1$ (or $0 < \vartheta < 1$). A tolerance for a uniform descent $\varepsilon_0 > 0$. A tolerance for a step length selection $\varepsilon_1 > 0$. A maximum step length $\overline{\Delta} > 0$.

Input. A sparsity pattern of the matrix $A(\boldsymbol{x}) = [A_1(\boldsymbol{x}), \ldots, A_m(\boldsymbol{x})]$. A starting point $\boldsymbol{x} \in \mathbb{R}^n$.

Step 1. *(Initiation)* Choose $\mu \leq \overline{\mu}$. Determine a sparse structure of the matrix $W = W(\boldsymbol{x}; \mu)$ from the sparse structure of the matrix $A(\boldsymbol{x})$ and perform a symbolic decomposition of the matrix W (described in [2, Section 1.7.4]). Compute values $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, values $F_k(\boldsymbol{x}) = \max_{1 \leq l \leq m_k} f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, and the value of objective function (11.4). Set $r = 0$ (restart indicator).

Step 2. *(Termination)* Determine a vector of smoothing multipliers $\boldsymbol{u}(\boldsymbol{x}; \mu)$ by (11.87). Determine a matrix $A = A(\boldsymbol{x})$ and a vector $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x}; \mu) = A(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}; \mu)$. If $\mu \leq \underline{\mu}$ and $\|\boldsymbol{g}\| \leq \underline{\varepsilon}$, terminate the computation.

Step 3. *(Hessian matrix approximation)* Set $G = G(\boldsymbol{x}; \mu)$ or compute an approximation G of the Hessian matrix $G(\boldsymbol{x}; \mu)$ using gradient differences or using quasi-Newton updates (Remark 11.13).

Step 4. *(Direction determination)* Determine the matrix W by (11.94) and the vector $\Delta\boldsymbol{x}$ by (11.93) using the Gill–Murray decomposition of the matrix W.

Step 5. *(Restart)* If $r = 0$ and (11.54) does not hold, set $G = I$, $r = 1$ and go to Step 4. If $r = 1$ and (11.54) does not hold, set $\Delta\boldsymbol{x} = -\boldsymbol{g}$. Set $r = 0$.

Step 6. *(Step length selection)* Determine a step length $\alpha > 0$ satisfying inequalities (11.55) (for a smoothing function $S(\boldsymbol{x}; \mu)$) and $\alpha = \overline{\Delta}/\|\Delta\boldsymbol{x}\|$. Set $\boldsymbol{x} := \boldsymbol{x} + \alpha\Delta\boldsymbol{x}$. Compute values $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, values $F_k(\boldsymbol{x}) = \max_{1 \leq l \leq m_k} f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, and the value of the objective function (11.4).

Step 7. *(Smoothing parameter update)* Determine a new value of the smoothing parameter $\mu \geq \underline{\mu}$ using Procedure A or Procedure B. Go to Step 2.

The values $\underline{\varepsilon} = 10^{-6}$, $\underline{\mu} = 10^{-6}$, $\overline{\mu} = 1$, $\lambda = 0.85$, $\sigma = 100$, $\vartheta = 0.1$, $\varepsilon_0 = 10^{-8}$, $\varepsilon_1 = 10^{-4}$, and $\overline{\Delta} = 1000$ were used in our numerical experiments.

Algorithm 11.3: Primal-dual interior point method

Data: A tolerance for the gradient norm $\underline{\varepsilon} > 0$. A parameter for determination of active constraints $\tilde{\varepsilon} > 0$. A parameter for initiation of slack variables and Lagrange multipliers $\delta > 0$. An initial value of the barrier parameter $\overline{\mu} > 0$. A precision for the direction determination $0 \leq \omega < 1$. A parameter for the step length selection $0 < \gamma < 1$. A tolerance for the step length selection $\varepsilon_1 > 0$. Maximum step length $\overline{\Delta} > 0$.

Input. A sparsity pattern of the matrix $A(\boldsymbol{x}) = [A_1(\boldsymbol{x}), \ldots, A_m(\boldsymbol{x})]$. A starting point $\boldsymbol{x} \in \mathbb{R}^n$.

Step 1. *(Initialization)* Compute values $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, and set $F_k(\boldsymbol{x}) = \max_{1 \leq l \leq m_k} f_{kl}(\boldsymbol{x})$, $z_k = F_k(\boldsymbol{x}) + \delta$, $1 \leq k \leq m$. Compute values $c_{kl}(\tilde{\boldsymbol{x}}) = f_{kl}(\boldsymbol{x}) - z_k$, and set $s_{kl} = -c_{kl}(\tilde{\boldsymbol{x}})$, $u_{kl} = \delta$. Set $\mu = \overline{\mu}$ and compute the value of the barrier function $\tilde{B}_\mu(\tilde{\boldsymbol{x}}, \boldsymbol{s})$.

Step 2. *(Termination)* Determine a matrix $\tilde{A}(\tilde{\boldsymbol{x}})$ and a vector $\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = \tilde{A}(\tilde{\boldsymbol{x}})\boldsymbol{u}$ by (11.111). If the KKT conditions $\|\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}, \boldsymbol{u})\| \leq \underline{\varepsilon}$, $\|\boldsymbol{c}(\tilde{\boldsymbol{x}}) + \boldsymbol{s}\| \leq \underline{\varepsilon}$, and $\boldsymbol{s}^T\boldsymbol{u} \leq \underline{\varepsilon}$ are satisfied, terminate the computation.

Step 3. *(Hessian matrix approximation)* Set $G = G(\boldsymbol{x}, \boldsymbol{u})$ or compute an approximation G of the Hessian matrix $G(\boldsymbol{x}, \boldsymbol{u})$ using gradient differences or utilizing quasi-Newton updates (Remark 11.13). Determine a parameter $\sigma \geq 0$ by (11.121) or set $\sigma = 0$. Split the constraints into active if $\hat{s}_{kl} \leq \tilde{\varepsilon}\hat{u}_{kl}$ and inactive if $\check{s}_{kl} > \tilde{\varepsilon}\check{u}_{kl}$.

Step 4. *(Direction determination)* Determine the matrix $\tilde{G} = \tilde{G}(\tilde{\boldsymbol{x}}, \boldsymbol{u})$ by (11.111) (where the Hessian matrix $G(\boldsymbol{x}, \boldsymbol{u})$ is replaced with its approximation G). Determine vectors $\Delta\tilde{\boldsymbol{x}}$ and $\Delta\hat{\boldsymbol{u}}$ by solving linear system (11.115), a vector $\Delta\check{\boldsymbol{u}}$ by (11.114), and a vector $\Delta\boldsymbol{s}$ by (11.116). Linear system (11.115) is solved either directly using the Bunch–Parlett decomposition (we carry out both the symbolic and the numeric decompositions in this step) or iteratively by the conjugate gradient method with indefinite preconditioner (11.123). Compute the derivative of the augmented Lagrange function by formula (11.120).

Step 5. *(Restart)* If $P'(0) \geq 0$, determine a diagonal matrix $\tilde{D}$ by (11.124), set $\tilde{G} = \tilde{D}$, $\sigma = 0$, and go to Step 4.

Step 6. *(Step length selection)* Determine a step length parameter $\alpha > 0$ satisfying inequalities $P(\alpha) - P(0) \leq \varepsilon_1 \alpha P'(0)$ and $\alpha \leq \overline{\Delta}/\|\Delta\boldsymbol{x}\|$. Determine new vectors $\tilde{\boldsymbol{x}} := \tilde{\boldsymbol{x}} + \alpha\Delta\tilde{\boldsymbol{x}}$, $\boldsymbol{s} := \boldsymbol{s}(\alpha)$, $\boldsymbol{u} := \boldsymbol{u}(\alpha)$ by (11.117). Compute values $f_{kl}(\boldsymbol{x})$, $1 \leq k \leq m$, $1 \leq l \leq m_k$, and set $c_{kl}(\tilde{\boldsymbol{x}}) = f_{kl}(\boldsymbol{x}) - z_k$, $1 \leq k \leq m$, $1 \leq l \leq m_k$. Compute the value of the barrier function $\tilde{B}_\mu(\tilde{\boldsymbol{x}}, \boldsymbol{s})$.

Step 7. *(Barrier parameter update)* Determine a new value of the barrier parameter $\mu \geq \underline{\mu}$ using Procedure C. Go to Step 2.

The values $\underline{\varepsilon} = 10^{-6}$, $\tilde{\varepsilon} = 0.1$, $\delta = 0.1$, $\omega = 0.9$, $\gamma = 0.99$, $\overline{\mu} = 1$, $\varepsilon_1 = 10^{-4}$, and $\overline{\Delta} = 1000$ were used in our numerical experiments.

References

1. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. Acta Numer. **14**, 1–137 (2005)
2. Björck, A.: Numerical Methods in Matrix Computations. Springer, New York (2015)
3. Bunch, J.R., Parlett, B.N.: Direct methods for solving symmetric indefinite systems of linear equations. SIAM J. Numer. Anal. **8**, 639–655 (1971)
4. Coleman, T.F., Moré, J.J.: Estimation of sparse Hessian matrices and graph coloring problems. Math. Program. **28**, 243–270 (1984)
5. Di Pillo, G., Grippo, L., Lucidi, S.: Smooth Transformation of the generalized minimax problem. J. Optim. Theory Appl. **95**, 1–24 (1997)
6. Duff, I.S., Gould, N.I.M., Reid, J.K., Turner, K.: The factorization of sparse symmetric indefinite metrices. IMA J. Numer. Anal. **11**, 181–204 (1991)
7. Fiedler, M.: Special Matrices and Their Applications in Numerical Mathematics. Dover, New York (2008)
8. Fletcher, R.: Practical Methods of Optimization. Wiley, New York (1987)
9. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. **91**, 239–269 (2002)
10. Gill, P.E., Murray, W.: Newton type methods for unconstrained and linearly constrained optimization. Math. Program. **7**, 311–350 (1974)
11. Gould, N.I.M., Toint, P.L.: Nonlinear programming without a penalty function or a filter. Math. Program. **122**, 155–196 (2010)
12. Griewank, A., Toint, P.L.: Partitioned variable metric updates for large-scale structured optimization problems. Numer. Math. **39**, 119–137 (1982)
13. Griewank, A., Walther, A.: Evaluating Derivatives. SIAM, Philadelphia (2008)
14. Le, D.: Three new rapidly convergent algorithms for finding a zero of a function. SIAM J. Sci. Stat. Comput. **6**, 193–208 (1985)
15. Le, D.: An efficient derivative-free method for solving nonlinear equations. ACM Trans. Math. Softw. **11**, 250–262 (1985)
16. Liu, X., Yuan, Y.: A sequential quadratic programming method without a penalty function or a filter for nonlinear equality constrained optimization. SIAM J. Optim. **21**, 545–571 (2011)
17. Lukšan, L., Spedicato, E.: Variable metric methods for unconstrained optimization and nonlinear least squares. J. Comput. Appl. Math. **124**, 61–93 (2000)
18. Lukšan, L., Vlček, J.: Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems. Numer. Linear Algebra Appl. **5**, 219–247 (1998)
19. Lukšan, L., Vlček, J.: Sparse and partially separable test problems for unconstrained and equality constrained optimization. Technical Report V-767, Prague, ICS AS CR (1998)
20. Lukšan, L., Matonoha, C., Vlček, J.: Interior-point method for non-linear non-convex optimization. Numer. Linear Algebra Appl. **11**, 431–453 (2004)
21. Lukšan, L., Matonoha, C., Vlček, J.: Interior-point method for large-scale nonlinear programming. Optim. Methods Softw. **20**, 569–582 (2005)
22. Lukšan, L., Matonoha, C., Vlček, J.: Primal interior-point method for large sparse minimax optimization. Kybernetika **45**, 841–864 (2009)
23. Lukšan, L., Matonoha, C., Vlček, J.: Primal interior-point method for minimization of generalized minimax functions. Kybernetika **46**, 697–721 (2010)
24. Lukšan, L., Tůma, M., Matonoha, C., Vlček, J., Ramešová, N., Šiška, M., Hartman, J.: UFO 2017. Interactive System for Universal Functional Optimization. Technical Report V-1252, Prague, ICS AS CR (2017)
25. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization. World Scientific, London (1992)
26. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)

27. Tůma, M.: A note on direct methods for approximations of sparse Hessian matrices. Appl. Math. **33**, 171–176 (1988)
28. Vanderbei, J., Shanno, D.F.: An interior point algorithm for nonconvex nonlinear programming. Comput. Optim. Appl. **13**, 231–252 (1999)
29. Wächter, A., Biegler, L.: Line search filter methods for nonlinear programming. Motivation and global convergence. SIAM J. Comput. **16**, 1–31 (2005)

Part III
Methods for Special Problems

Chapter 12
Bundle Methods for Inexact Data

Welington de Oliveira and Mikhail Solodov

Abstract Many applications of optimization to real-life problems lead to nonsmooth objective and/or constraint functions that are assessed through "noisy" oracles. In particular, only some approximations to the function and/or subgradient values are available, while exact values are not. For example, this is the typical case in Lagrangian relaxation of large-scale (possibly mixed-integer) optimization problems, in stochastic programming, and in robust optimization, where the oracles perform some *numerical procedure* to evaluate functions and subgradients, such as solving one or more optimization subproblems, multidimensional integration, or simulation. As a consequence, one cannot expect such oracles to provide exact data on the function values and/or subgradients. We review algorithms based on the bundle methodology, mostly developed quite recently, that have the ability to handle inexact data. We adopt an approach which, although not exaustive, covers various classes of bundle methods and various types of inexact oracles, for unconstrained and convexly constrained problems (with both convex and nonconvex objective functions), as well as nonsmooth mixed-integer optimization.

12.1 Introduction

Nonsmooth optimization (NSO) appears often in connection with real-life problems that are too difficult to solve directly and need to be decomposed: instead of dealing directly with the difficult problem one may choose (or even have to choose) to solve a sequence of simpler problems (subproblems); see, e.g., [15, 66–68]. For example, this is a common strategy in stochastic programming [66], in Lagrangian relaxation

W. de Oliveira
CMA – Centre de Mathématiques Appliquées, MINES ParisTech, PSL – Research University, Sophia Antipolis, France
e-mail: welington.oliveira@mines-paristech.fr

M. Solodov (✉)
IMPA – Instituto de Matemática Pura e Aplicada, Rio de Janeiro, RJ, Brazil
e-mail: solodov@impa.br

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_12

[8, 50], and in Benders' decomposition [7, Chapter 11]. Lagrangian relaxation leads to nonsmooth convex problems. (Generalized) Benders' decomposition may give rise to constrained nonsmooth and nonconvex optimization problems [25, 61]. In any case, the resulting nonsmooth functions can only be evaluated by an oracle solving (inner) optimization (sub)problems. Solving those subproblems exactly, along many iterations, is at the very least impractical, and is usually not even possible anyway. Similar situations arise when simulation or other numerical procedures are required.

In the NSO setting, the *oracle information* comes in the form of a functional value and *one* subgradient (i.e., the full subdifferential is not available). As is well known, bundle methods, dating back to 1975 (in particular, [49] and [82]), are nowadays among the most efficient algorithms for NSO. Further developments have been in several directions: algorithms with limited memory [42], methods for nonlinearly constrained problems [44, 69], bilevel problems [12, 71], nonmonotone versions [3, 19], second-order methods [58], nonconvex objective functions [27, 33, 41, 57, 63, 76], DC programming [13, 19, 28, 40], convex multiobjetive optimization [60], algorithms for combinatorial and mixed-integer optimization [11, 81], semidefinite programming [24, 35], distributed optimization [39], among others. The original proximal variant of bundle methods has been generalized in [26]. Moreover, other (than proximal) stabilizations have been developed: the level bundle methods proposed in [51], the trust-region variant in [38], the proximal Chebychev center algorithm in [64], and the doubly stabilized bundle method in [18].

Since their invention, and for about 20 years, convergence theory of bundle methods could only handle *exact* oracles, i.e., the exact value of the function and a true subgradient were required, at every point of evaluation. Inexactness was first introduced in [45]; however, approximations of both the function and its subgradients were required to be *asymptotically exact*, i.e., the noise/perturbations had to vanish in the limit. In the context of Lagrangian relaxation, for example, this presumes that we can solve the optimization subproblems with an arbitrarily high precision. While this can be accepted as realistic in some cases, it is certainly not so in general. Next, inexact oracles were considered in [22] for level bundle methods, in [36] for proximal bundle, and in [59] in a special bundle method for the maximal-eigenvalue function. In [22] and [59], the oracle is still asymptotically exact, as in [45]. In [36] it is assumed that the exact value of the function is available, while the subgradient can be computed approximately. The attractive feature of the analysis in [36] is that, unlike in any previous work on bundle methods, the perturbation in subgradient evaluations need not vanish in the limit. On the other hand, the exact value of the function is still needed, and so this setting is not suitable for some important applications of bundle methods and, in particular, for the Lagrangian relaxation. This is because in that case, evaluating the function and its subgradient is the same task, which amounts to computing a solution of the subproblem (exactly). Nonvanishing perturbations in both the function and subgradient values were introduced in [70]. The next advance, including the idea of *noise attenuation* is [47]. On the side of level bundle variants, [22] was extended in [16, 54, 75].

The more recent developments in handling inexact data made the new variants of bundle methods even more suitable for various real-life applications; see, e.g., [15]. Works such as [16, 20, 34, 77, 81] provide different ways inexact data can be handled by bundle methods, allowing solution of difficult NSO problems (sometimes even exactly, or at least with the desired accuracy, depending on the oracle's assumptions).

In what follows, we discuss various bundle algorithms for variants of the optimization problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in G \ \text{ and } \ h(\boldsymbol{x}) \leq 0, \end{cases} \tag{12.1}$$

where $G \subset \mathbb{R}^n$ and the functions $f, h : \mathbb{R}^n \to \mathbb{R}$ are assessed through inexact oracles. Note that in the nonsmooth setting, there is no loss of generality in considering a scalar constraint function h, as it can be defined as the maximum of all the constraint functions. Usually, but not always, f, h and G would be assumed to be convex, and not necessarily all of them would be present in the problem. Specific assumptions would be stated as needed.

The rest of this chapter is organized as follows. We start in Sect. 12.2 with defining different types of inexact oracles and presenting some examples of how inexact data appears naturally in applications. By assuming convexity of the involved functions, Sect. 12.3 reviews a family of inexact level bundle methods for the problem (12.1) with either convex or nonconvex bounded set G. In Sect. 12.4 we consider inexact proximal bundle methods for unconstrained and linearly constrained convex problems. A recent algorithm combining level and proximal ideas is given in Sect. 12.5. An inexact proximal bundle method for nonconvex objective functions is discussed in Sect. 12.6. Finally, Sect. 12.7 contains some concluding remarks and research perspectives.

12.2 Inexact Oracles: The Main Assumptions and Examples

Let the functions f and h in (12.1) be convex, and assessed via inexact oracles. Specifically, for each given $\boldsymbol{x} \in \mathbb{R}^n$ an *upper* oracle delivers inexact information on f, namely

$$\begin{aligned} &\text{(i)} \quad f_{\boldsymbol{x}} = f(\boldsymbol{x}) - \eta_{\boldsymbol{x}}^v; \qquad \text{and} \\ &\text{(ii)} \quad \boldsymbol{\xi}_{\boldsymbol{x}} \in \mathbb{R}^n : f(\cdot) \geq f_{\boldsymbol{x}} + \langle \boldsymbol{\xi}_{\boldsymbol{x}}, \cdot - \boldsymbol{x} \rangle - \eta_{\boldsymbol{x}}^s, \ \eta_{\boldsymbol{x}}^v \leq \eta, \ \eta_{\boldsymbol{x}}^s \leq \eta \text{ for all } \boldsymbol{x} \in \mathbb{R}^n, \end{aligned} \tag{12.2}$$

and, for the function h, a *lower* oracle provides

$$\begin{aligned} &\text{(i)} \quad h_{\boldsymbol{x}} = h(\boldsymbol{x}) - \epsilon_{\boldsymbol{x}}; \qquad \text{and} \\ &\text{(ii)} \quad \boldsymbol{\zeta}_{\boldsymbol{x}} \in \mathbb{R}^n : h(\cdot) \geq h_{\boldsymbol{x}} + \langle \boldsymbol{\zeta}_{\boldsymbol{x}}, \cdot - \boldsymbol{x} \rangle, \ \epsilon_{\boldsymbol{x}} \leq \epsilon \text{ for all } \boldsymbol{x} \in \mathbb{R}^n. \end{aligned} \tag{12.3}$$

In the above, the oracles' output is (f_x, ξ_x) and (h_x, ζ_x), respectively. The subscripts v and s on the errors in (12.2) make the distinction between function *value* and *subgradient* errors. The bounds $\eta, \epsilon \geq 0$ on the unknown errors η_x^v, η_x^s and ϵ_x, are possibly unknown as well. However, there are also important situations in which these bounds can actually be chosen by the user and sent, together with x, to the oracles (see Example 12.2 below).

The exact oracles for f and h correspond to taking $\eta \equiv \epsilon \equiv 0$, and compute $f_x = f(x)$, $h_x = h(x)$, together with their true subgradients. The important subclass of lower oracles returns lower linearizations: $h(x) - \epsilon_x = h_x \leqslant h(x)$ and $h(\cdot) \geqslant h_x + \langle \zeta_x, \cdot - x \rangle$. *Upper oracles*, by contrast, can overestimate function values: in (12.2) η_x^v can be negative and η_x^s can be positive. We assume the h-oracle to be of lower type for simplicity: as shown in [75, Subsection 5.3], dealing with upper oracles for the constraint function is possible, but requires extra steps in the (level) bundle algorithm. We omit such extra steps to avoid technicalities that may obscure presentation of the main ideas. Interested readers are referred to [75, 78] for a comprehensive treatment of upper oracles for both objective and constraint functions.

We next show how lower and upper oracles appear naturally in practice.

12.2.1 Examples of Inexact Oracles

We start with an application which is perhaps the main motivation for investigating algorithms for nonsmooth optimization with inexact data.

Example 12.1 (Lagrangian Relaxation: Inexact Oracle) Consider the following problem:

$$\begin{cases} \text{maximize} & \varphi(u) \\ \text{subject to} & u \in U \ \text{ and } \ c(u) = 0 \in \mathbb{R}^n, \end{cases}$$

where $\varphi : \mathbb{R}^m \to \mathbb{R}$ and $c : \mathbb{R}^m \to \mathbb{R}^n$ are continuous functions, and $U \subset \mathbb{R}^m$ is a nonempty compact set. For a Lagrange multiplier $x \in \mathbb{R}^n$, the dual function of the above problem is given by $f(x) := \max_{u \in U} L(u, x)$ with $L(u, x) := \varphi(u) + \langle x, c(u) \rangle$. Notice that $L(u, \cdot)$ is convex for any $u \in U$ fixed.

Given x_0, the oracle (solver) computes a point $u_0 \in U$ which (approximately) maximizes the function $L(\cdot, x_0)$ over U, and returns $f_{x_0} := L(u_0, x_0)$ and $\xi_{x_0} = c(u_0) \in \partial_x L(u_0, x_0)$. Convexity of $L(u_0, \cdot)$ and the definition of $f(\cdot)$ yield that

$$f(x) \geq L(u_0, x) \geq L(u_0, x_0) + \langle \xi_{x_0}, x - x_0 \rangle = f_{x_0} + \langle \xi_{x_0}, x - x_0 \rangle .$$

Therefore, $f_{x_0} = L(u_0, x_0)$ and $\xi_{x_0} = c(u_0)$ satisfy the two first lines in (12.2) with $\eta^v_{x_0} = f(x_0) - L(u_0, x_0) \geq 0$ (unknown) and $\eta^s_{x_0} \equiv 0$, i.e., this is a lower oracle for f. The boundedness assumption $\eta_{x_0} \leq \eta$ is satisfied if the considered solver is ensured to compute an η-solution u_0 to the subproblem $\max_{u \in U} L(u, x_0)$, i.e., a point $u_0 \in U$ satisfying the condition $\max_{u \in U} L(u, x_0) \leq L(u_0, x_0) + \eta$.

More versatile oracles can be obtained if one has control over the solver employed to compute approximate solutions of $\max_{u \in U} L(u, x)$. This interesting setting is discussed in the next example.

Example 12.2 (Lagrangian Relaxation: On-Demand Accuracy) In some practical situations (e.g., when $\max_{u \in U} L(u, x_0)$ is a combinatorial problem), the error bound η can be chosen and sent to the solver, so that some η-solution u_0 is returned. In other words, the decision-maker can decide to request more or less accuracy from the oracle. Note that requesting high accuracy at unpromising candidate solutions can be a waste of time. More specifically, along the iterative process of minimizing f, the decision-maker may have access to the best estimation $f^{tar} = f_{x_j}$ of the optimal value. Then, if a new point $x_i \neq x_j$ is identified to be a poor solution candidate, the process of computing $f(x_i)$, i.e., of solving $\max_{u \in U} L(u, x_i)$, can be interrupted early (at some point $u_i \in U$). The oracle delivers the inexact data $f_{x_i} = L(u_i, x_i)$ and $\xi_i = c(u_i)$ without satisfying the requested tolerance η. This is the case if the inequality $L(u_i, x_i) > f^{tar}$ is verified when computing $f(x_i)$. It is worth mentioning that this kind of test is available in solvers such as Gurobi (www.gurobi.com) and CPLEX (www.cplex.com), for example.

An inexact oracle equipped with such procedures is called *oracle with on-demand accuracy*, introduced in [16] and further investigated in [23] and [77].

Another example of how inexact data can arise naturally comes from stochastic programming.

Example 12.3 (Two-Stage Stochastic Programs) Consider a stochastic program with decision variables organized in two levels, denoted by x and y for the first and second stages, respectively. Let $\Omega \subset \mathbb{R}^m$ be a set containing *finitely many* elements (scenarios). If $\omega \in \Omega$ represents uncertainty, for a convex function $\varphi : \mathbb{R}^n \to \mathbb{R}$, vectors $q(\omega)$ and matrices $T(\omega)$ and W, the corresponding two-stage program with fixed recourse is

$$\begin{cases} \text{minimize} & \varphi(x) + \mathbb{E}[\langle q(\omega), y \rangle] \\ \text{subject to} & T(\omega)x + Wy = b(\omega) \quad \text{for all } \omega \in \Omega\,, \\ & x \in G,\ y \geqslant 0\,, \end{cases}$$

where $\mathbb{E}[\cdot]$ stands for the expected value with respect to the probability measure on the set Ω. For fixed $\boldsymbol{x}$ and $\boldsymbol{\omega}$, the *recourse* function

$$Q(\boldsymbol{x};\boldsymbol{\omega}) := \begin{cases} \text{minimize} & \langle \boldsymbol{q}(\boldsymbol{\omega}), \boldsymbol{y}\rangle \\ \text{subject to} & \boldsymbol{y} \geqslant 0\,,\ W\boldsymbol{y} = \boldsymbol{b}(\boldsymbol{\omega}) - T(\boldsymbol{\omega}) \end{cases}$$

gives in (12.1) an objective of the form $f(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + \mathbb{E}[Q(\boldsymbol{x};\boldsymbol{\omega})]$, which is finite-valued when the recourse is relatively complete.

For each fixed $\boldsymbol{x}$ and a given realization $\boldsymbol{\omega}$, the evaluation of the recourse function can be done by solving the dual linear program

$$\begin{cases} \text{maximize} & \langle \boldsymbol{b}(\boldsymbol{\omega}) - T(\boldsymbol{\omega}), \boldsymbol{u}\rangle \\ \text{subject to} & W^T\boldsymbol{u} \leq \boldsymbol{q}(\boldsymbol{\omega})\,. \end{cases}$$

If, to speed up calculations, instead of performing the maximization for the considered $\boldsymbol{\omega}$ we just take a feasible point $\boldsymbol{u}_{\boldsymbol{x},\boldsymbol{\omega}}$ (satisfying $W^T\boldsymbol{u}_{\boldsymbol{x},\boldsymbol{\omega}} \leqslant \boldsymbol{q}(\boldsymbol{\omega})$), then an oracle giving $f_{\boldsymbol{x}} := \varphi(\boldsymbol{x}) + \mathbb{E}[\langle \boldsymbol{b}(\boldsymbol{\omega}) - T(\boldsymbol{\omega}), \boldsymbol{u}_{\boldsymbol{x},\boldsymbol{\omega}}\rangle]$, and $\boldsymbol{\xi}_{\boldsymbol{x}} := \nabla\varphi(\boldsymbol{x}) - \mathbb{E}[T(\boldsymbol{\omega})^T\boldsymbol{u}_{\boldsymbol{x},\boldsymbol{\omega}}]$ is of lower type and fits (12.2) with $\eta^s_{\boldsymbol{x}} \equiv 0$.

Alternatively, if we select a (much smaller) subset $\Omega_k \subset \Omega$ of scenarios to perform the subproblem optimization, an upper oracle can be obtained by setting $f_{\boldsymbol{x}}$ and $\boldsymbol{\xi}_{\boldsymbol{x}}$ as above but with $\mathbb{E}$ replaced by $\mathbb{E}_k$, the expected value with respect to the probability of (fewer) scenarios in Ω_k. See [17] for more details.

Nonlinearly constrained variants of stochastic programs arise naturally when one needs to handle risk measures.

Example 12.4 (CVaR Two-Stage Stochastic Programs) With the notation of the previous example, consider the following nonlinearly constrained two-stage stochastic program with parameters $\beta \in (0, 1]$ and $\rho \in \mathbb{R}$:

$$\begin{cases} \text{minimize} & \varphi(\boldsymbol{x}) + \mathbb{E}[Q(\boldsymbol{x};\boldsymbol{\omega})] \\ \text{subject to} & \boldsymbol{x} \in G,\ \mathrm{CVaR}_\beta[Q(\boldsymbol{x};\boldsymbol{\omega})] \leq \rho\,, \end{cases}$$

where CVaR_β is the *conditional value at risk*. Let $\mathbb{P}(\boldsymbol{\omega})$ be the probability associated to the scenario $\boldsymbol{\omega} \in \Omega$. It can be seen [23] that the convex constraint CVaR_β can be evaluated as the optimal value of the following LP (for $\boldsymbol{x}$ fixed):

$$\mathrm{CVaR}_\beta[Q(\boldsymbol{x};\boldsymbol{\omega})] := \begin{cases} \text{maximize} & \displaystyle\sum_{\boldsymbol{\omega}\in\Omega} \pi_{\boldsymbol{\omega}} Q(\boldsymbol{x};\boldsymbol{\omega}) \\ \text{subject to} & 0 \leq \pi_{\boldsymbol{\omega}} \leq \mathbb{P}(\boldsymbol{\omega}) \quad \text{for all } \boldsymbol{\omega} \in \Omega, \\ & \displaystyle\sum_{\boldsymbol{\omega}\in\Omega} \pi_{\boldsymbol{\omega}} = \beta\,, \end{cases}$$

whose solution can be easily computed by sorting the costs $Q(\boldsymbol{x};\boldsymbol{\omega})$, $\boldsymbol{\omega}\in\Omega$, and assigning as much as possible weights $\pi_{\boldsymbol{\omega}}$ to the highest costs.

By performing inexact optimization to compute the recourse functions $Q(\boldsymbol{x};\boldsymbol{\omega})$, one is inexactly evaluating the functions $f(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + \mathbb{E}[Q(\boldsymbol{x};\boldsymbol{\omega})]$ and $h(\boldsymbol{x}) := \mathrm{CVaR}_{\beta}[Q(\boldsymbol{x};\boldsymbol{\omega})] - \rho$. In [23] it is shown how to design efficient inexact oracles for these functions.

There are applications where the source of inexactness is not in solving optimization subproblems, but is associated to simulation and numerical integration.

Example 12.5 (Probability Maximization Problem) Let $(\boldsymbol{\omega}, \Omega, \mathbb{P})$ be a probability space and $c : \mathbb{R}^n \to \mathbb{R}^m$ be a concave mapping. A variant of the so-called *probability maximization problem* is

$$\begin{cases} \text{maximize} & \mathbb{P}[c(\boldsymbol{x}) \geq \boldsymbol{\omega}] \\ \text{subject to} & \boldsymbol{x} \in G\,. \end{cases}$$

Problems of this form can be seen as reformulation of chance-constrained optimization problems. They arise, for example, in cascaded-reservoir management [15, 79], financial risk management, and some other areas [9]. If the continuous probability distribution of $\boldsymbol{\omega}$ is log-concave, then the function $f(\boldsymbol{x}) := -\log(\mathbb{P}[c(\boldsymbol{x}) \geq \boldsymbol{\omega}])$ is convex (and can be nonsmooth, depending on $\mathbb{P}$ and the covariance matrix of $\boldsymbol{\omega}$). Since a multidimensional integral needs to be numerically computed to evaluate the function, an exact oracle for f is impossible in practice. Inexact values can also be computed by using Monte-Carlo simulation and variance reduction techniques [29]. Such processes induce, in general, upper oracles. Lower estimations of f are possible in some particular cases (for instance when $\boldsymbol{\omega}$ follows a multivariate Gaussian probability distribution) [2]. It should be noted though that an error bound $\eta > 0$ in (12.2) can be expected, but cannot be fully guaranteed, as there is always some nonzero probability of a large error showing up (see [78, Lemma 5.1] for details).

If the distribution of $\boldsymbol{\omega}$ is not log-concave, then f is not convex, and algorithms for nonconvex optimization need to be employed.

The above examples show how inexact oracles arise in connection with practical problems. In the remaining of this chapter, we review several bundle algorithms that are able to handle inexact data.

12.3 Inexact Level Bundle Methods

Throughout this section we make the assumption that f and h in (12.1) are convex functions, and that inexact oracles satisfying (12.2) and (12.3) are available. We also assume that G is a compact set, but not necessary a convex one.

A versatile family of bundle algorithms, called *level bundle* methods, was introduced in [45, 51]. As will be seen in what follows, simple level bundle variants can handle both exact and inexact data in the same fashion, without any special treatment or modifications in the case of inexact oracles. This is different from proximal bundle methods, which require modifications in the inexact setting.

12.3.1 Models, Localizer Sets, and Optimality Certificate

Observe that the two first lines in (12.2) and (12.3) yield

$$\begin{aligned} f(\cdot) &\geq f(\boldsymbol{x}) + \langle \boldsymbol{\xi}_{\boldsymbol{x}}, \cdot - \boldsymbol{x} \rangle - (\eta_{\boldsymbol{x}}^{v} + \eta_{\boldsymbol{x}}^{s}) \,, \\ h(\cdot) &\geq h(\boldsymbol{x}) + \langle \boldsymbol{\zeta}_{\boldsymbol{x}}, \cdot - \boldsymbol{x} \rangle - \epsilon_{\boldsymbol{x}} \,, \end{aligned} \tag{12.4}$$

from which, evaluating at $\boldsymbol{x}$ we deduce that $\eta_{\boldsymbol{x}}^{v} + \eta_{\boldsymbol{x}}^{s} \geq 0$ (regardless of the signs of the individual error terms) and $\epsilon_{\boldsymbol{x}} \geq 0$. As a result,

$$\boldsymbol{\xi}_{\boldsymbol{x}} \in \partial_{(\eta_{\boldsymbol{x}}^{v} + \eta_{\boldsymbol{x}}^{s})} f(\boldsymbol{x}) \quad \text{with} \quad \eta_{\boldsymbol{x}}^{v} + \eta_{\boldsymbol{x}}^{s} \geq 0 \quad \text{for all} \;\; \boldsymbol{x} \in \mathbb{R}^n \,, \tag{12.5}$$

and

$$\boldsymbol{\zeta}_{\boldsymbol{x}} \in \partial_{(\epsilon_{\boldsymbol{x}})} f(\boldsymbol{x}) \quad \text{with} \quad \epsilon_{\boldsymbol{x}} \geq 0 \quad \text{for all} \;\; \boldsymbol{x} \in \mathbb{R}^n \,. \tag{12.6}$$

Even if in (12.2) and (12.3) the values of the bounds for errors η and ϵ are unknown, the inequalities above imply that $\eta \geq \eta_{\boldsymbol{x}}^{v} \geq -\eta_{\boldsymbol{x}}^{s} \geq -\eta$ and $\epsilon \geq \epsilon_{\boldsymbol{x}} \geq 0$. Thus, oracle errors are bounded from below (by $-\eta$ and 0, respectively).

Let k be the index of the current iteration. Having called the oracle at previous iterates $\boldsymbol{x}_j$, bundle algorithms accumulate linearizations of the functions to construct their cutting-plane models:

$$\hat{f}_k(\cdot) := \max_{j \in \mathcal{J}_k} \{ f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \cdot - \boldsymbol{x}_j \rangle \} \leq f(\cdot) + \eta, \tag{12.7}$$

$$\hat{h}_k(\cdot) := \max_{j \in \mathcal{J}_k} \{ h_{\boldsymbol{x}_j} + \langle \boldsymbol{\zeta}_{\boldsymbol{x}_j}, \cdot - \boldsymbol{x}_j \rangle \} \leq h(\cdot), \tag{12.8}$$

where $\mathcal{J}_k$ is an index set (typically $\mathcal{J}_k \subset \{1, \ldots, k\}$), and the inequalities follow from the second lines in (12.2) and (12.3). With these models and an additional

parameter $f_k^{lev} \in \mathbb{R}$, we can define the following *localizer set*:

$$\mathcal{L}_k := \{\boldsymbol{x} \in G : \hat{f}_k(\boldsymbol{x}) \leq f_k^{lev},\ \hat{h}_k(\boldsymbol{x}) \leq 0\}\,. \tag{12.9}$$

When the constraint function h is not present, the localizer set becomes the level set of the cutting-model $\hat{f}_k$, justifying thus the name "*level bundle methods*". The following is a simple but useful result that will allow to determine approximate lower bounds for the optimal value f^* of the problem (12.1).

Lemma 12.1 *If $\mathcal{L}_k = \emptyset$, then $f_k^{lev} \leq f^* + \eta$.*

Proof It follows from (12.7) and (12.8) that $\mathcal{L}_k$ approximates the level set of f over the feasible set in the following sense:

$$\{\boldsymbol{x} \in G : f(\boldsymbol{x}) \leq f_k^{lev} - \eta,\ h(\boldsymbol{x}) \leqslant 0\} \subset \mathcal{L}_k\,.$$

Hence, if $\mathcal{L}_k$ is an empty set, then so is $\{\boldsymbol{x} \in G : f(\boldsymbol{x}) \leq f_k^{lev} - \eta,\ h(\boldsymbol{x}) \leqslant 0\}$. The feasible set $\{\boldsymbol{x} \in G : h(\boldsymbol{x}) \leqslant 0\}$ is nonempty, by the standing assumption. As a result, $f_k^{lev} - \eta$ must be a lower bound for f^*. □

Let f_0^{low} be a given lower bound for the optimal value f^*. A handy manner to update such a bound along the iterative process is to set $f_{k+1}^{low} := f_k^{lev}$ if $\mathcal{L}_k = \emptyset$, and $f_{k+1}^{low} := f_k^{low}$ otherwise. With this rule, Lemma 12.1 ensures that $f_k^{low} \leq f^* + \eta$ for all k. We can thus define the useful *improvement function*

$$F_j(f_k^{low}) := \max\{f_{\boldsymbol{x}_j} - f_k^{low},\ h_{\boldsymbol{x}_j}\} \quad \text{and} \quad \Delta_k := \min_{j \leq k} F_j(f_k^{low})\,. \tag{12.10}$$

Remark 12.1 (Approximate Optimality Test) Note that if $F^j(f_k^{low}) \leq \delta^{tol}$ for some index $j \leq k$ and the given tolerance $\delta^{tol} \geq 0$ (which is the case when $\Delta_k \leq \delta^{tol}$), then $h_{\boldsymbol{x}_j} \leq \delta^{tol}$ and $f_{\boldsymbol{x}_j} \leq f_k^{low} + \delta^{tol}$. By using the oracles' assumptions we get $h(\boldsymbol{x}_j) - \epsilon \leq h_{\boldsymbol{x}_j} \leq \delta^{tol}$ and $f(\boldsymbol{x}_j) - \eta \leq f_{\boldsymbol{x}_j} \leq f_k^{low} + \delta^{tol} \leq f^* + \eta + \delta^{tol}$, i.e.,

$$h(\boldsymbol{x}_j) \leq \epsilon + \delta^{tol} \quad \text{and} \quad f(\boldsymbol{x}_j) \leq f^* + 2\eta + \delta^{tol}\,.$$

In other words, $\boldsymbol{x}_j$ is a $(\max\{2\eta, \epsilon\} + \delta^{tol})$-solution to the problem (12.1). We can thus employ the value Δ_k as an (approximate) optimality certificate: if it holds that $\Delta_k \leq \delta^{tol}$, then $\boldsymbol{x}^{best} := \boldsymbol{x}_j$ for j yielding the minimum in (12.10) is an approximate solution in the given sense. □

12.3.2 An Algorithmic Pattern

We now present an inexact level bundle method for the class of problems in the form (12.1), having convex functions f and h, and a compact set G.

Algorithm 12.1: Inexact level bundle method: an algorithmic pattern

Data: Parameters $\gamma \in (0,1)$, $\delta^{tol} \geq 0$, and a lower bound f_0^{low} for (12.1).

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_0 \in G$ and call the oracles to compute $(f_{\boldsymbol{x}_0}, \boldsymbol{\xi}_{\boldsymbol{x}_0})$ and $(h_{\boldsymbol{x}_0}, \boldsymbol{\zeta}_{\boldsymbol{x}_0})$. Set $\mathcal{J}_0 := \{0\}$ and $k := 0$.

Step 1. (*Stopping test*) Compute Δ_k as in (12.10). If $\Delta_k \leq \delta^{tol}$, then **stop** and return $\boldsymbol{x}^{best} := \boldsymbol{x}_j$ for j yielding the minimum in (12.10).

Step 2. (*Next iterate*) Set $f_k^{lev} := f_k^{low} + \gamma \Delta_k$ and define the localizer set $\mathcal{L}_k$ as in (12.9). If $\mathcal{L}_k = \emptyset$, then define $f_{k+1}^{low} := f_k^{lev}$ and go to Step 4. Otherwise, set $f_{k+1}^{low} := f_k^{low}$ and choose $\boldsymbol{x}_{k+1} \in \mathcal{L}_k$.

Step 3. (*Oracle call*) Compute the data $(f_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}_{\boldsymbol{x}_{k+1}})$ and $(h_{\boldsymbol{x}_{k+1}}, \boldsymbol{\zeta}_{\boldsymbol{x}_{k+1}})$.

Step 4. (*Bundle management*) Set $\mathcal{J}_{k+1} := \mathcal{J}_k \cup \{k+1\}$ if $\boldsymbol{x}_{k+1}$ is available and $\mathcal{J}_{k+1} := \mathcal{J}_k$ otherwise. Set $k := k+1$ and go back to Step 1.

Algorithm 12.1 is a simplistic version of [75, Algorithm 1]: it does not provide either an implementable rule to define $x_{k+1} \in \mathcal{L}_k$ nor a scheme for keeping the size of the bundle $\mathcal{J}_k$ bounded (limited memory). We shall return to these more specific issues a bit later, after proving convergence of the method to $(\max\{2\eta, \epsilon\} + \delta^{tol})$-solution of the problem (12.1).

Note that the algorithm defines the next trial point in the localizer set $\mathcal{L}_k$. Hence, for all $j \in \mathcal{J}_k$, we have that:

(i) $f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x}_{k+1} - \boldsymbol{x}_j \rangle \leq f_k^{lev}$; and
(ii) $h_{\boldsymbol{x}_j} + \langle \boldsymbol{\zeta}_{\boldsymbol{x}_j}, \boldsymbol{x}_{k+1} - \boldsymbol{x}_j \rangle \leq 0$.

Using the Cauchy–Schwarz inequality in (i), we obtain that $f_{\boldsymbol{x}_j} - f_k^{lev} \leq \|\boldsymbol{\xi}_{\boldsymbol{x}_j}\| \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_j\| \leq \|\Lambda_f\| \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_j\|$, whereas inequality (ii) yields $h_{\boldsymbol{x}_j} \leq \|\boldsymbol{\zeta}_{\boldsymbol{x}_j}\| \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_j\| \leq \|\Lambda_h\| \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_j\|$. The existence of finite constants Λ_f and Λ_h above is ensured by [37, Proposition 6.2.2], because the oracle errors are bounded, (12.5) and (12.6) hold, and G is a bounded set. By taking $\Lambda := \max\{\Lambda_f, \Lambda_h\}$ we have thus shown that

$$\begin{aligned} \Delta_k \leq F_j(f_k^{low}) &= \max\{f_{\boldsymbol{x}_j} - f_k^{lev}, h_{\boldsymbol{x}_j}\} \\ &\leq \Lambda \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_j\| \quad \text{for all } j \in \mathcal{J}_k\,. \end{aligned} \tag{12.11}$$

Theorem 12.1 *For the problem* (12.1)*, assume that f and h are convex, $G \neq \emptyset$ is compact, and that the inexact oracles satisfy* (12.2) *and* (12.3)*. If $\delta^{tol} > 0$, then after finitely many steps Algorithm 12.1 stops with a* $(\max\{2\eta, \epsilon\} + \delta^{tol})$*-solution to the problem* (12.1)*.*

Proof Suppose that Algorithm 12.1 does not terminate. In particular, this implies that $\Delta_k > \delta^{tol} > 0$ for all k. Then, using also (12.11), we obtain that $0 < \delta^{tol} \leq \Lambda \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_j\|$ for all k and all $j \leq k$. This contradicts the fact that the bounded sequence $(\boldsymbol{x}_k) \subset G$ has a convergent subsequence. Therefore, the inequality $\Delta_k \leq \delta^{tol}$ must hold after finitely many steps. Remark 12.1 then concludes the proof. □

As long as $\boldsymbol{x}_{k+1} \in \mathcal{L}_k$ and $\mathcal{J}_k = \{1, 2, \ldots, k\}$, the above result ensures convergence (in the given sense) of Algorithm 12.1 for any nonempty compact set G, regardless of its structure. For instance, G can be a mixed-integer set, or even something more complicated. If one can either verify that $\mathcal{L}_k$ is empty or, otherwise, pick *any* point in $\mathcal{L}_k$ in some practical/implementable manner, Algorithm 12.1 is an appealing general strategy, due to its reliability and simplicity.

12.3.3 Exploiting the Domain

Let $\hat{\boldsymbol{x}}_k$ be the stability center, for instance: the last iterate $\boldsymbol{x}_{k-1}$, the best candidate $\boldsymbol{x}^{best}$, or another point chosen by some rule. Then, determining the next trial point $\boldsymbol{x}_{k+1}$ in $\mathcal{L}_k$ (given in (12.9)) can be done by minimizing the *distance* to the stability center $\hat{\boldsymbol{x}}_k$, i.e., $\boldsymbol{x}_{k+1}$ solves

$$\begin{cases} \text{minimize} & d(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{L}_k \,. \end{cases} \tag{12.12}$$

Appropriate choices for the distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ and for the stability center $\hat{\boldsymbol{x}}_k$ depend on the structure of G. In what follows, we discuss some possible variants of Algorithm 12.1 by exploiting this structure.

12.3.3.1 The Combinatorial Setting

Let G be composed of binary variables: $G = \{\boldsymbol{x} \in \{0, 1\}^n : A\boldsymbol{x} \leq b\}$, for some given matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$. Taking $d(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) := \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2$, we get that $d(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) = \frac{1}{2}\sum_{i=1}^n \boldsymbol{x}_i - \sum_{i=1}^n \boldsymbol{x}_i \hat{\boldsymbol{x}}_i + \frac{1}{2}\sum_{i=1}^n \hat{\boldsymbol{x}}_i$. Hence, the master problem (12.12) boils down to the binary LP:

$$\begin{cases} \text{minimize} & \langle \frac{1}{2}\mathbf{1} - \hat{\boldsymbol{x}}_k, \boldsymbol{x} \rangle \\ \text{subject to} & f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \leq f_k^{lev}, \quad j \in \mathcal{J}_k, \\ & h_{\boldsymbol{x}_j} + \langle \boldsymbol{\zeta}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \leq 0, \quad j \in \mathcal{J}_k, \\ & A\boldsymbol{x} \leq b, \ \boldsymbol{x} \in \{0, 1\}^n. \end{cases}$$

We point out that one does not need to solve this subproblem up to optimality at every iteration of Algorithm 12.1: merely a feasible point, or a certificate that the feasible set is empty, is enough to ensure convergence of the level bundle method. That said, solving up to optimality may decrease the number of oracle calls, as reported in [81].

12.3.3.2 The Mixed-Integer Setting

Suppose that G is the mixed-integer set $G = \{\boldsymbol{x} = (\boldsymbol{x}_c, \boldsymbol{x}_i) \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} : A\boldsymbol{x} \leq b\}$. The work [11] proposes to define the next trial point as (approximate) solution of the following mixed-integer master problem:

$$\begin{cases} \text{minimize} & \|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|_\diamond \\ \text{subject to} & f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \leq f_k^{lev}, \quad j \in \mathcal{J}_k, \\ & h_{\boldsymbol{x}_j} + \langle \boldsymbol{\zeta}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \leq 0, \quad j \in \mathcal{J}_k, \\ & A\boldsymbol{x} \leq b, \ \boldsymbol{x} = (\boldsymbol{x}_c, \boldsymbol{x}_i) \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}, \end{cases} \tag{12.13}$$

where $d(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|_\diamond$, and $\|\cdot\|_\diamond$ is a given norm, for instance the L_1, L_∞ or L_2-norms. For the two first choices, (12.13) becomes a MILP, whereas for $\|\cdot\|_\diamond = \|\cdot\|^2$ (12.13) is MIQP. For these choices good numerical results were reported in [11] for dealing with convex MINLP problems: when compared to classical methods for this type of problems, the given level bundle methods could reduce the number of oracle calls in around 22%, which can yield a significant CPU time reduction depending on the computational cost of the oracles.

12.3.3.3 The Convex Setting

Let now G be a closed convex set, the more friendly setting considered in most bundle methods in the literature [16, 22, 45, 51, 75]. We stick with the classical choice for the stability function $d(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) = \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2$, yielding the following QP master problem:

$$\begin{cases} \text{minimize} & \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \leq f_k^{lev}, \quad j \in \mathcal{J}_k, \\ & h_{\boldsymbol{x}_j} + \langle \boldsymbol{\zeta}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \leq 0, \quad j \in \mathcal{J}_k, \\ & \boldsymbol{x} \in G. \end{cases} \tag{12.14}$$

The next iterate is therefore the projection of the stability center onto the localizer set. The projection provides some useful inequalities that permit to analyze the complexity of Algorithm 12.1. To this end, let us consider special iterations called

"critical iterations", the ones which make significant progress in the quest of solving (12.1) or improving the current lower bound f_k^{low}. More specifically, critical iterations would be indexed by ℓ and are defined as follows: set $\Delta_{k(0)} = \Delta_0$, $\ell = 0$ and update

$$\ell := \ell + 1, \ k(\ell) := k \quad \text{whenever} \quad \Delta_k \leq (1-\gamma)\Delta_{k(\ell)} \ \text{ or } \ \mathcal{L}_k = \emptyset\,.$$

We can thus split the iterative process of Algorithm 12.1 into cycles: denote the ℓ-th cycle as $\mathcal{K}^\ell := \{k(\ell), k(\ell)+1, \ldots, k(\ell+1)-1\}$.

Proposition 12.1 *Consider Algorithm 12.1 and assume that G is a compact convex set with diameter $D < \infty$ and $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}} \in G$ is fixed for all $k \in \mathcal{K}^\ell$. Then, any iteration index $k \in \mathcal{K}^\ell$ with $\Delta_k > \delta^{tol} > 0$ may differ no more from $k(\ell)$ than the following bound:*

$$k - k(\ell) + 1 \leq \left(\frac{\Lambda D}{(1-\gamma)\Delta_k}\right)^2 .$$

Proof It follows from the definition of the cycle $\mathcal{K}^\ell$ that $f_k^{low} = f_{k(\ell)}^{low}$ and Δ_k is nonincreasing for all $k \in \mathcal{K}^\ell$. This shows that f_k^{lev} is nonincreasing as well. Hence, $\mathcal{L}_k \subset \mathcal{L}_{k-1}$ for all $k, k-1 \in \mathcal{K}^\ell$ (because the bundle $\mathcal{J}_k$ keeps all linearizations). Consider the iteration indexed by $k-1 \in \mathcal{K}^\ell$. The projection of $\hat{\boldsymbol{x}}$ onto $\mathcal{L}_{k-1}$ yields $\boldsymbol{x}_k$ and, moreover, the inequality

$$\langle \hat{\boldsymbol{x}} - \boldsymbol{x}_k, \boldsymbol{x}_k - \boldsymbol{x}\rangle \geq 0 \quad \text{for all } \ \boldsymbol{x} \in \mathcal{L}_{k-1}\,.$$

As $\boldsymbol{x}_{k+1} \in \mathcal{L}_k \subset \mathcal{L}_{k-1}$, the inequality $\langle \hat{\boldsymbol{x}} - \boldsymbol{x}_k, \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle \geq 0$ used in $\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}\|^2 = \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k + (\boldsymbol{x}_k - \hat{\boldsymbol{x}})\|^2$ gives

$$\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}\|^2 \geq \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\|^2 + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2\,.$$

By employing (12.11) with $j = k$ and recalling that $\Delta_k > (1-\gamma)\Delta_{k(\ell)}$ by the definition of $k(\ell)$, we obtain that

$$\begin{aligned} D^2 \geq \|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}\|^2 &\geq \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\|^2 + \left(\frac{\Delta_k}{\Lambda}\right)^2 \\ &> \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\|^2 + \left(\frac{(1-\gamma)\Delta_{k(\ell)}}{\Lambda}\right)^2 . \end{aligned}$$

The result then follows by some simple manipulations of the above relation; see [75, Lemma 4] for more details. □

We note that the hypothesis $\hat{\boldsymbol{x}} \in G$ is only used to ensure that the relation $D^2 \geq \|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}\|^2$ holds. In fact, we could use any (non-feasible) stability center $\hat{\boldsymbol{x}}$: since

G is compact, the distance $\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}\|^2$ would be bounded by another constant, say $\tilde{D}^2$, and the given complexity analysis would hold with D replaced with $\tilde{D}$. We next estimate the maximum number of oracle calls to obtain $\Delta_k \leq \delta^{tol}$.

Theorem 12.2 *Under the assumptions of Proposition 12.1, assume further that $\delta^{tol} > 0$ in Algorithm 12.1. Then, to reach an optimality measure Δ_k smaller than $\delta^{tol} > 0$ Algorithm 12.1 performs at most*

$$\left(1 + \frac{f^* + \eta - f_0^{low}}{\gamma \delta^{tol}}\right)\left(\frac{\Lambda D}{(1-\gamma)\delta^{tol}}\right)^2$$

iterations.

Proof Notice that every time $\mathcal{L}_k$ is empty the lower bound f_k^{low} of $f^* + \eta$ is increased by an amount of $\gamma \Delta_k$ ($> \gamma \delta^{tol}$). Since f_0^{low} is finite, the maximum number of cycles ℓ^{mx} times the stepsize $\gamma \delta^{tol}$ is less than $f^* + \eta - f_0^{low}$, i.e., $\ell^{mx} \leqslant (f^* + \eta - f_0^{low})/(\gamma \delta^{tol})$. The result then follows from Proposition 12.1 by noting that each cycle $\mathcal{K}^\ell$ has at most $(\Lambda D)^2/((1-\gamma)\delta^{tol})^2$ iterations. See [75, Theorem 2] for more details. □

Proposition 12.1 requires the bundle of information $\mathcal{J}_k$ to keep all the linearizations indexed by $k \in \mathcal{K}^\ell$. However, the bundle can be managed arbitrarily at critical iterations $k(\ell)$: for instance, we could simply empty $\mathcal{J}_k$ by keeping only the new linearization indexed by $k(\ell)$. We mention that for a limited-memory variant of Algorithm 12.1, one needs to include the so-called aggregate linearizations, as is standard in modern bundle methods; see, for instance, [75, Step 7 of Algorithm 1].

12.3.4 Handling Oracles with On-Demand Accuracy

We highlight that Algorithm 12.1 does not make any special treatment of inexact data: it works exactly the same way for both exact and inexact oracles. More efficient variants are obtained if we can control the oracles' inexactness. We now address this subject by focusing on oracles with *on-demand accuracy*.

Following [23], we say that $\boldsymbol{x}_k \in G$ is an *f-substantial iterate* if the inexact value of the function $f_{\boldsymbol{x}_k}$ meets the descent target $f_k^{tar} \in \mathbb{R}$. Similarly, $\boldsymbol{x}_k \in G$ is said to be an *h-substantial iterate* if the inexact constraint value $h_{\boldsymbol{x}_k}$ meets the feasibility target $h_k^{tar} \in \mathbb{R}$. An iterate $\boldsymbol{x}_k$ is said to be *substantial* if it is both f- and h-substantial. Moreover, we denote with $\mathcal{S}$ the index set gathering iterations that provided substantial iterates. We will refer to $\mathcal{S}$ as the *substantial set*. The aim of this section is to make a few changes in Algorithm 12.1, in order to deal with lower oracles with asymptotically vanishing errors for substantial iterates. Such oracles are referred to as lower oracles with on-demand accuracy [75]. They satisfy (12.2) and (12.3) with the following additional assumptions, where $0 \leq \eta_k \leq \eta$, $f_k^{tar} \in \mathbb{R}$

and $h_k^{tar} \in \mathbb{R}$ are given parameters:

$$\begin{cases} \eta^s_{\boldsymbol{x}_k} \equiv 0, & \text{(lower oracle)}, \\ \eta^v_{\boldsymbol{x}_k} \leq \eta_k, & \text{if } f_{\boldsymbol{x}_k} \leq f_k^{tar}, \\ \epsilon^v_{\boldsymbol{x}_k} \leq \eta_k, & \text{if } h_{\boldsymbol{x}_k} \leq h_k^{tar}. \end{cases} \tag{12.15}$$

This kind of oracles provide information with accuracy up to η_k for substantial iterates.

We emphasize that the assumptions (12.2) and (12.3) satisfying also (12.15) are still quite general and cover many situations of interest. For given $\boldsymbol{x}_k$ and $\eta_k = 0$, the oracle provides exact information if both $f_{\boldsymbol{x}_k}$ and $h_{\boldsymbol{x}_k}$ meet the targets. Moreover, if both targets f_k^{tar} and h_k^{tar} are set as $+\infty$ for all iterations and $\eta_k = 0$, then the oracles are exact. *Asymptotically exact* versions of these oracles are obtained if $f_k^{tar} = h_k^{tar} = +\infty$ for all k and $\eta_k \to 0$. Combinations between partially exact and asymptotically exact versions are also possible. For this setting, in order to get an exact solution to the problem (12.1) the algorithm must force $\eta_k \to 0$ (only) for substantial iterates.

Let f_k^{low} be a proven lower bound for f^*, the optimal value of (12.1). At any substantial iterate $j \in \mathcal{S}$, the oracle bound η_j is known and can be exploited in the definition of the improvement function. We therefore define $\Delta_k := \min_{j \leq k} F_j^{\mathcal{S}}(f_k^{low})$ with

$$F_j^{\mathcal{S}}(f_k^{low}) := \begin{cases} \max\{f_{\boldsymbol{x}_j} - f_k^{low},\, h_{\boldsymbol{x}_j}\} + \eta_j, & \text{if } j \in \mathcal{S}, \\ +\infty, & \text{otherwise.} \end{cases} \tag{12.16}$$

Due to the *on-demand accuracy* assumption, Lemma 9 in [75] ensures that $F_j^{\mathcal{S}}(f_k^{low}) \geq \max\{f(\boldsymbol{x}_j) - f_k^{low},\, h(\boldsymbol{x}_j)\} \geq 0$. Hence, if $F_j^{\mathcal{S}}(f_k^{low}) = 0$ then $\boldsymbol{x}_j$ is an optimal solution to the problem (12.1) and $f_k^{low} = f^*$. In this case $\eta_j = 0$, i.e., the oracle is exact at the substantial iterate $\boldsymbol{x}_j$.

In order to obtain $\Delta_k \to 0$, it is necessary to force η_k to zero for substantial iterates. This can be done by controlling the oracle error in Step 3 of Algorithm 12.1 as follows:

Step 3'. (*On-demand accuracy*)

Step 3.1. (*Controlling the error*) Update the oracle error by setting $\eta_{k+1} = \theta \Delta_{k(\ell)}$, for a given $\theta \in (0, (1-\gamma)^2)$ (here $k(\ell)$ the last critical iteration).

Step 3.2. (*Target updating*) Set $f_{k+1}^{tar} = f_k^{lev} + (1-\gamma)\Delta_k$ and $h_{k+1}^{tar} = (1-\gamma)\Delta_k$.

Step 3.3. (*Oracle call*) Compute data $(f_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}_{\boldsymbol{x}_{k+1}})$ and $(h_{\boldsymbol{x}_{k+1}}, \boldsymbol{\zeta}_{\boldsymbol{x}_{k+1}})$ satisfying (12.2), (12.3) and (12.15).

We have the following result, whose proof can be found in [75, Theorem 6].

Theorem 12.3 *Consider Algorithm 12.1 with $\delta^{tol} = 0$, Step 3 replaced by Step 3' above, and optimality certificate given in (12.16). Let $\boldsymbol{x}_k^{best}$ be the point defining the*

value of Δ_k. Then $\Delta_k \to 0$ and every accumulation point of the sequence $\{\boldsymbol{x}_k^{best}\}$ is an exact solution to the problem (12.1).

The interest of using oracles with on-demand accuracy is evident: the algorithm can compute an exact solution to the problem, even though most of its iterations are inexact. The main idea is that there is no gain in "wasting" time for computing exact data at non-promising points (whose function values are greater than the specified targets f_k^{tar} and h_k^{tar}). As shown by the numerical experience in [16, 23, 75], for a large battery of stochastic programs (two-stage, risk-free, risk-averse and chance-constrained ones) it is possible to save up to 79% of the total computational time (while still finding an exact solution) just by letting the bundle method control how accurate the oracle information should be at substantial iterates.

In the linearly-constrained setting, the complexity analysis of level bundle methods with on-demand accuracy is very similar to the one of exact bundle methods, which is known for being optimal or nearly-optimal [6, 12]. We refer interested readers to [16, Subsection 3.2.3] for a formal statement.

We finalize this section by mentioning that there exist specialized inexact bundle methods dealing with unbounded set G; see [16, Section 4] for oracles with on-demand accuracy and [54] for the more general setting. However, none of these works handle nonlinearly constrained problems. Nonlinearly constrained problems with unbounded set G are considered in [78], but in a proximal bundle framework.

12.4 Inexact Proximal Bundle Methods

For the sake of simplicity, in most part of this section we drop the nonlinear constraint h in (12.1) and focus on the simpler setting of

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in G, \end{cases} \tag{12.17}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $G \subset \mathbb{R}^n$ is a nonempty convex and closed set, typically polyhedral. The nonlinearly constrained setting will be discussed in Sect. 12.4.4.3.

To deal with the problem (12.17) we now consider proximal bundle methods [38, 49]. As in the level variant, proximal bundle algorithms define new iterates by making use of a cutting-plane model $\hat{f}_k$ for f, a stability center $\hat{\boldsymbol{x}}_k \in G$ and a stabilization function d. For the classical choice $d(\boldsymbol{x}, \hat{\boldsymbol{x}}_k) = \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2$, with $t_k > 0$ a prox-parameter, the new iterate $\boldsymbol{x}_{k+1}$ of a proximal bundle algorithm is the solution of the master problem

$$\begin{cases} \text{minimize} & \hat{f}_k(\boldsymbol{x}) + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & \boldsymbol{x} \in G\,. \end{cases} \tag{12.18}$$

As is well known, if G is polyhedral, then (12.18) is equivalent to solving the following QP:

$$\begin{cases} \text{minimize} & r + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \le r, \quad j \in \mathcal{J}_k, \\ & \boldsymbol{x} \in G,\ r \in \mathbb{R}. \end{cases}$$

It is worth to mention that for the same model $\hat{f}_k$ and the same stability center $\hat{\boldsymbol{x}}_k$, one can find the proximal and level parameters t_k and f_k^{lev} such that (12.12) (without the nonlinear constraint h) and (12.18) generate the same next iterate $\boldsymbol{x}_{k+1}$. In this formal theoretical sense, the level and proximal approaches can be considered equivalent. But the details of the implementation and practical performance can be quite different. In particular, the key parameters are updated by strategies which are specific to each of the methods, and thus the generated iterates are not the same in practice.

12.4.1 Descent Test and Optimality Certificate

The stability center in (inexact) proximal bundle methods is usually updated according to the following *descent rule*. Let

$$v_k := f_{\hat{\boldsymbol{x}}_k} - \hat{f}_k(\boldsymbol{x}_{k+1}), \tag{12.19}$$

and let $\kappa \in (0, 1)$ be a parameter.

$$\text{If } f_{\boldsymbol{x}_{k+1}} \le f_{\hat{\boldsymbol{x}}_k} - \kappa v_k, \text{ then } \hat{\boldsymbol{x}}_{k+1} := \boldsymbol{x}_{k+1}; \text{ otherwise } \hat{\boldsymbol{x}}_{k+1} := \hat{\boldsymbol{x}}_k. \tag{12.20}$$

Some other descent rules allowing more flexibility in defining stability centers (or providing strong convergence in general Hilbert spaces) can be found in [3, 26, 77] and [80], respectively.

Note first that in the setting of inexact oracles one can have $v_k < 0$ in (12.19), which renders the descent test (12.20) meaningless. In the convex case, this situation can only be caused by the oracle inexactness. The method then checks whether v_k is (or is not) sufficiently positive, to detect when inaccuracy is becoming excessive/cumbersome. If v_k is not (sufficiently) positive, the method increases the prox-parameter t_k. The motivation for this is as follows. If v_k is not positive, (12.19) suggests that $\boldsymbol{x}_{k+1}$ does not sufficiently minimize the model $\hat{f}_k(\cdot)$. Increasing t_k in (12.18) decreases the weight of the quadratic term therein, thus increasing the relative weight of $\hat{f}_k(\cdot)$ in the minimization problem. This has the effect of decreasing the value of the model at candidate points, eventually (once t_k is large enough) making v_k in (12.19) acceptably positive. Once the latter is achieved,

the iteration proceeds as in a standard bundle method. This procedure is called *noise attenuation* [47].

In what follows we provide some useful ingredients to check inexact optimality and to keep the bundle of information $\mathcal{J}_{k+1}$ limited. Assuming G is polyhedral, or that an appropriate constraint qualification [72] holds in (12.18), the optimality conditions for (12.18) yield that

$$\boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_k - t_k \hat{\boldsymbol{\xi}}_k, \quad \text{with} \quad \hat{\boldsymbol{\xi}}_k := \boldsymbol{p}_k^f + \boldsymbol{p}_k^G,$$

where $\boldsymbol{p}_k^f \in \partial \hat{f}_k(\boldsymbol{x}_{k+1})$ and $\boldsymbol{p}_k^G \in \partial i_G(\boldsymbol{x}_{k+1})$, i_G being the indicator function of the set G. Making use of Lagrange multipliers α_j^k associated to the constraints $f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x}_{k+1} - \boldsymbol{x}_j \rangle \leq r$, $j \in \mathcal{J}_k$, it further holds that

$$\boldsymbol{p}_k^f := \sum_{j \in \mathcal{J}_k} \alpha_j^k \boldsymbol{\xi}_{\boldsymbol{x}_j}, \quad \sum_{j \in \mathcal{J}_k} \alpha_j^k = 1, \ \alpha_j^k \geq 0, \ j \in \mathcal{J}_k \, .$$

Such multipliers can be used to save storage without impairing convergence. More precisely, "inactive" indices, corresponding to $\alpha_j^k = 0$, can be dropped for the next iteration: $\mathcal{J}_{k+1} \supset \{j \in \mathcal{J}_k : \alpha_j^k \neq 0\}$. An even more economical model can be constructed using the so-called *aggregate linearization*

$$\bar{f}_{k_a}(\boldsymbol{x}) := \hat{f}_k(\boldsymbol{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \boldsymbol{x} - \boldsymbol{x}_{k+1} \rangle,$$

which satisfies $\bar{f}_{k_a}(\boldsymbol{x}) \leq f_k(\boldsymbol{x}) + i_G(\boldsymbol{x}) \leq f(\boldsymbol{x}) + \eta + i_G(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$, by the oracle assumptions in (12.2) and the definition of $\hat{\boldsymbol{\xi}}_k$. As in the exact proximal bundle methods, this linearization plays an important role in the bundle management: it can be shown that if the new cutting-plane model is defined by the *minimal bundle* $\mathcal{J}_{k+1} := \{k+1, k_a\}$, this is still enough to ensure convergence.

The following result establishes a helpful connection between the predicted decrease $v_k := f_{\hat{\boldsymbol{x}}_k} - \hat{f}_k(\boldsymbol{x}_{k+1})$ and the *aggregate error* $\hat{e}_k := f_{\hat{\boldsymbol{x}}_k} - \bar{f}_{k_a}(\hat{\boldsymbol{x}}_k)$. In particular, it gives a certificate of (approximate) optimality to the problem (12.17).

Lemma 12.2 *It holds that* $\hat{e}_k \geqslant -2\eta$, $v_k = \hat{e}_k + t_k \|\hat{\boldsymbol{\xi}}_k\|^2$, *and*

$$\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k + 2\eta)} [f(\hat{\boldsymbol{x}}_k) + i_G(\hat{\boldsymbol{x}}_k)].$$

Proof It follows from the definitions of $\hat{e}_k$, v_k, $\boldsymbol{x}_{k+1}$ and $\bar{f}_{k_a}$ that

$$\hat{e}_k = f_{\hat{\boldsymbol{x}}_k} - \bar{f}_{k_a}(\hat{\boldsymbol{x}}_k) = f_{\hat{\boldsymbol{x}}_k} - [\hat{f}_k(\boldsymbol{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1} \rangle] = v_k - t_k \|\hat{\boldsymbol{\xi}}_k\|^2.$$

In addition, the oracle definition (12.2) and inequality $\bar{f}_{k_a}(\boldsymbol{x}) \leq f(\boldsymbol{x}) + \eta$ for all $\boldsymbol{x} \in G$ give: $\hat{e}_k = f_{\hat{\boldsymbol{x}}_k} - \bar{f}_{k_a}(\hat{\boldsymbol{x}}_k) \geq (f(\hat{\boldsymbol{x}}_k) - \eta) - (f(\hat{\boldsymbol{x}}_k) + \eta) = -2\eta$.

Let $\boldsymbol{x} \in \mathbb{R}^n$ be an arbitrary point. Proposition 12.5 and the oracle assumptions (12.2) yield

$$\begin{aligned}\eta + f(\boldsymbol{x}) + i_G(\boldsymbol{x}) &\geq \bar{f}_{k_a}(\boldsymbol{x}) = \hat{f}_k(\boldsymbol{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \boldsymbol{x} - \boldsymbol{x}_{k+1}\rangle \\ &= f_{\hat{\boldsymbol{x}}_k} - (f_{\hat{\boldsymbol{x}}_k} - \hat{f}_k(\boldsymbol{x}_{k+1})) + \langle \hat{\boldsymbol{\xi}}_k, \boldsymbol{x} - \hat{\boldsymbol{x}}_k\rangle + \langle \hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1}\rangle \\ &= f_{\hat{\boldsymbol{x}}_k} - (\hat{e}_k + t_k\|\hat{\boldsymbol{\xi}}_k\|^2) + \langle \hat{\boldsymbol{\xi}}_k, \boldsymbol{x} - \hat{\boldsymbol{x}}_k\rangle + \langle \hat{\boldsymbol{\xi}}_k, t_k\hat{\boldsymbol{\xi}}_k\rangle \\ &\geq f(\hat{\boldsymbol{x}}_k) - \eta - \hat{e}_k + \langle \hat{\boldsymbol{\xi}}_k, \boldsymbol{x} - \hat{\boldsymbol{x}}_k\rangle\,.\end{aligned}$$

Since $i_G(\hat{\boldsymbol{x}}_k) = 0$, we conclude that $\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k+2\eta)}[f(\hat{\boldsymbol{x}}_k) + i_G(\hat{\boldsymbol{x}}_k)]$. □

As a consequence of Lemma 12.2, if for some infinite index set $\mathcal{K} \subset \{0, 1, 2, \ldots\}$ we have

$$\limsup_{\mathcal{K}\ni k\to\infty} \hat{e}_k \leq 0 \quad \text{and} \quad \lim_{\mathcal{K}\ni k\to\infty} \|\hat{\boldsymbol{\xi}}_k\| = 0, \tag{12.21}$$

then every accumulation point $\bar{\boldsymbol{x}}$ of the sequence $\{\hat{\boldsymbol{x}}_k : k \in \mathcal{K}\}$ satisfies $\mathbf{0} \in \partial_{2\eta}[f(\bar{\boldsymbol{x}}) + i_G(\bar{\boldsymbol{x}})]$, i.e., $\bar{\boldsymbol{x}}$ is a 2η-solution to (12.17). This gives the needed (approximate) optimality measures.

12.4.2 Inexact Proximal Bundle Algorithm

We now present an inexact proximal bundle method for convex problems in the form (12.17). We refer to [47] for the original algorithm and to [20] for further enhancements.

In the exact setting, the aggregate error $\hat{e}_k$ is always nonnegative. This implies the inequality $v_k = \hat{e}_k + t_k\|\hat{\boldsymbol{\xi}}_k\|^2 \geq t_k\|\hat{\boldsymbol{\xi}}_k\|^2 = \frac{1}{t_k}\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_k\|^2$. The inaccuracy detection inequality $\hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|^2$ at Step 3 implies that the oracle error is too excessive: note that in this case

$$v_k = \hat{e}_k + t_k\|\hat{\boldsymbol{\xi}}_k\|^2 < (1-\tau)t_k\|\hat{\boldsymbol{\xi}}_k\|^2 = \frac{1-\tau}{t_k}\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_k\|^2,$$

i.e., the predicted decrease is not sufficiently positive. The role of increasing t_k in this situation had been already discussed above.

Algorithm 12.2: Inexact proximal bundle method

Data: Stopping tolerances $\delta_1^{tol}, \delta_2^{tol} \geq 0$, a descent parameter $\kappa \in (0,1)$, a stepsize $t_0 \geq \underline{t} > 0$, and $\tau \in [\frac{1}{2}, 1)$.

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_0 \in G$, set $\hat{\boldsymbol{x}}_0 := \boldsymbol{x}_0$, and call the oracle to compute $(f_{\boldsymbol{x}_0}, \boldsymbol{\xi}_{\boldsymbol{x}_0})$. Set $\mathcal{J}_0 := \{0\}$, $k := 0$ and $\mathtt{na} := 0$.

Step 1. (*Trial point computation*) Find $\boldsymbol{x}_{k+1}$ solving (12.18) and let α^k denote the corresponding optimal simplicial multiplier. Compute $\hat{\boldsymbol{\xi}}_k = (\hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1})/t_k$, $v_k = f_{\hat{\boldsymbol{x}}_k} - \hat{f}_k(\boldsymbol{x}_{k+1})$, and $\hat{e}_k = v_k - t_k\|\hat{\boldsymbol{\xi}}_k\|^2$.

Step 2. (*Stopping criterion*) If $\|\hat{\boldsymbol{\xi}}_k\| \leq \delta_1^{tol}$ and $\hat{e}_k \leq \delta_2^{tol}$, **stop**.

Step 3. (*Inaccuracy detection*) If $\hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|^2$, set $t_k := 10t_k$, $\mathtt{na} := k$ and loop back to step 1.

Step 4. (*Oracle call and descent test*) Call the inexact oracle (12.2) to compute $(f_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}_{\boldsymbol{x}_{k+1}})$. If the descent test (12.20) holds, then declare the iterate serious: set $\hat{\boldsymbol{x}}_{k+1} := \boldsymbol{x}_{k+1}$, $\mathtt{na} := 0$. Otherwise, declare the iterate null: set $\hat{x}_{k+1} := \hat{x}_k$.

Step 5. (*Bundle management*) Choose $\mathcal{J}_{k+1} \supseteq \{j \in \mathcal{J}^k : \alpha_j^k \neq 0\} \cup \{k+1\}$. (Another possibility is $\mathcal{J}_{k+1} := \{k+1, k_a\}$.)

Step 6. (*Stepsize updating and loop*) If the iterate was declared serious, select $t_{k+1} \geq t_k$. If the iterate was declared null, either set $t_{k+1} := t_k$, or choose $t_{k+1} \in [0.1t_k, t_k]$ such that $t_{k+1} \geq \underline{t}$ if $\mathtt{na} = 0$. Increase k by 1 and go to Step 1.

12.4.3 Convergence Analysis

Throughout this section, we consider Algorithm 12.2 with $\delta_1^{tol} = \delta_2^{tol} = 0$, applied to the problem (12.17), assumed to have a solution. The oracle satisfies (12.2).

We start the analysis considering the infinite noise attenuation loop.

Proposition 12.2 (Infinite Noise Attenuation Loop) *Suppose the algorithm loops indefinitely between Steps 1 and 3, after the last serious iterate $\bar{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{k_0}$ is generated. Then (12.21) holds with $\mathcal{K} = \{0, 1, 2, \ldots\}$ and $\bar{\boldsymbol{x}}$ is a 2η-solution to the problem (12.17).*

Proof Since the algorithm loops indefinitely between Steps 1 and 3, we have that $\hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|^2$ for all k large enough. Then, Lemma 12.2 ensures that $-2\eta \leq \hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|^2 < 0$ for k large enough. Letting $k \to \infty$ in this relation, and recalling that $t_k \to +\infty$ by Step 3 of the algorithm, we conclude that (12.21) holds. □

Next, we proceed as usual in proximal bundle methods, with two possibilities:

(i) the algorithm generates an infinite sequence of null steps after a last serious iterate;
(ii) an infinite sequence of serious steps is produced.

Note that $\hat{e}_k \geq -\tau t_k \|\hat{\boldsymbol{\xi}}_k\|^2$ for all k yielding either a null or a serious step (as otherwise the step is that of noise attenuation), and in particular $v_k \geq 0$ for such iterations.

The next result addresses item (i). It makes use of the crucial inequality

$$0 \geq \limsup_{k\to\infty} [f_{\boldsymbol{x}_k} - \hat{f}_{k-1}(\boldsymbol{x}_k)], \tag{12.22}$$

whose proof can be consulted in [47, Lemma 3.3, Equation (3.6)] or [20, Theorem 6.4 and Subsection 7.3].

Proposition 12.3 (Infinite Loop of Null Steps) *If the algorithm generates an infinite sequence of null steps after the last serious iterate $\bar{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{k_0}$ is obtained, then (12.21) holds with $\mathcal{K} = \{0, 1, 2, \ldots\}$ and $\bar{\boldsymbol{x}}$ is a 2η-solution to the problem (12.17).*

Proof In this case, the descent test (12.20) never holds for k large enough, i.e., $f_{\boldsymbol{x}_{k+1}} > f_{\hat{\boldsymbol{x}}_{k_0}} - \kappa v_k$. We then obtain that

$$\begin{aligned} f_{\boldsymbol{x}_{k+1}} - \hat{f}_k(\boldsymbol{x}_{k+1}) &= f_{\boldsymbol{x}_{k+1}} - f_{\hat{\boldsymbol{x}}_{k_0}} + (f_{\hat{\boldsymbol{x}}_{k_0}} - \hat{f}_k(\boldsymbol{x}_{k+1})) \\ &\geq -\kappa v_k + (f_{\hat{\boldsymbol{x}}_{k_0}} - \hat{f}_k(\boldsymbol{x}_{k+1})) = (1-\kappa) v_k\,, \end{aligned}$$

where $v_k \geq 0$ for the type of iterations in considertion. Using (12.22), we conclude that $v_k \to 0$. As $v_k = \hat{e}_k + t_k \|\hat{\boldsymbol{\xi}}_k\|^2$, $\hat{e}_k \geq -\tau t_k \|\hat{\boldsymbol{\xi}}_k\|^2$ in this case and $t_k \geq \underline{t} > 0$, the relations (12.21) follow. □

We now consider the sequence of serious steps.

Proposition 12.4 (Infinite Number of Serious Steps) *If the algorithm generates an infinite sequence of serious steps, then the relations in (12.21) hold for $\mathcal{K} = \{k \in \mathbb{N} : \text{the descent test (12.20) is satisfied for } k\}$, and every accumulation point of $\{\hat{\boldsymbol{x}}_k : k \in \mathcal{K}\}$ is a 2η-solution to the problem (12.17).*

Proof Let $f^* > -\infty$ be the optimal value of (12.17). Then the oracle ensures that $f_{\boldsymbol{x}_k} \geq f^* - \eta$ for all k. For $k \in \mathcal{K}$, it holds that $f_{\hat{\boldsymbol{x}}_{k+1}} \leq f_{\hat{\boldsymbol{x}}_k} - \kappa v_k$, where $v_k \geq 0$. Then for any $k \in \mathcal{K}$, we obtain that

$$\begin{aligned} \infty > f_{\hat{\boldsymbol{x}}_0} - (f^* - \eta) \geq f_{\hat{\boldsymbol{x}}_0} - f_{\hat{\boldsymbol{x}}_{k+1}} &= \sum_{m=0}^{k} [f_{\hat{\boldsymbol{x}}_m} - f_{\hat{\boldsymbol{x}}_{m+1}}] \\ &\geq \kappa \sum_{m=0}^{k} v_m\,, \end{aligned}$$

where $m \in \mathcal{K}$. Letting in the above $k \to \infty$ implies that $\sum_{k=0,\, k\in\mathcal{K}}^{\infty} v_k$ is finite, and hence, $v_k \to 0$ as $\mathcal{K} \ni k \to \infty$. The rest of the reasoning is the same as in the end of the proof of Proposition 12.3. □

Combining Propositions 12.2, 12.3 and 12.4, convergence analysis of Algorithm 12.2 is now summarized as follows.

Theorem 12.4 *Suppose that the inexact oracle satisfies* (12.2)*, and that in Algorithm 12.2 one sets* $\delta_1^{tol} = \delta_2^{tol} = 0$*. Then the optimality certificate* (12.21) *holds for some index set* $\mathcal{K}$*, and every accumulation point* $\bar{\boldsymbol{x}}$ *of the sequence* $\{\hat{\boldsymbol{x}}_k : k \in \mathcal{K}\}$ *is a* 2η*-solution to the problem* (12.17)*.*

We remark that $(\hat{\boldsymbol{x}}_k)$ is guaranteed to be bounded (and thus have accumulation points) if f has bounded level sets. Indeed, since v_k is nonnegative at serious steps, $f_{\hat{\boldsymbol{x}}_k} \leq f_{\hat{\boldsymbol{x}}_0} \leq f(\hat{\boldsymbol{x}}_0) + \eta$ for all k. This shows that $f(\hat{\boldsymbol{x}}_k) \leq f(\hat{\boldsymbol{x}}_0) + 2\eta$ for all k.

On the other hand, Theorem 4.5 in [20] does not even assume that the problem has a solution to prove that $\limsup_{\mathcal{K} \ni k \to \infty} f_{\hat{\boldsymbol{x}}_k} \leq \inf_{\boldsymbol{x} \in G} f(\boldsymbol{x}) + \eta$. For this, the stopping test of Algorithm 12.2 has to be changed to the following: stop when $\|\hat{\boldsymbol{\xi}}_k\|$ and $\max\{0, \hat{e}_k + \langle \hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{x}}_k \rangle\}$ are small enough. Employing the latter quantities is useful for proving convergence without imposing boundedness on the sequence of serious steps; see [20, Subsection 4.2] for details.

12.4.4 Inexact Proximal Bundle Method Variants

As for the level bundle method, the proximal bundle algorithm also has many variants as discussed, for example, in [20, Section 7]. Below we list some of them. But first, let us note that with exact oracles, Algorithm 12.2 becomes one of the usual exact proximal bundle methods. In particular, the noise attenuation procedure in Step 3 is never triggered (because the aggregate error is always nonnegative in the exact convex case). Then, Theorem 12.4 holds with $\eta = 0$.

12.4.4.1 Partially Inexact Proximal Bundle Method

Let the inexact oracle satisfy (12.2) and (12.15) with $f_k^{tar} := f_{\hat{x}_k} - \kappa v_k$. Then, Algorithm 12.2 becomes a partially inexact method, in which at serious iterates evaluations are by an oracle with on-demand accuracy, but null iterates are of the general inexact type. It can be shown that in this setting the algorithm computes an exact solution to the problem (12.17), see [20, Subsection 7.1].

12.4.4.2 Incremental Proximal Bundle Method

Suppose that the objective function in (12.17) has the additive structure, i.e., $f(\boldsymbol{x}) := \sum_{i=1}^m f^i(\boldsymbol{x})$, with $f^i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, being convex functions. For example, this is the case in Lagrangian relaxation of multistage stochastic integer programming, where every component function f^i is the optimal value of a mixed-integer suproblem

$$f^i(\boldsymbol{x}) := \begin{cases} \text{maximize} & \varphi^i(\boldsymbol{u}) + \langle \boldsymbol{x}, c^i(\boldsymbol{u}) \rangle \\ \text{subject to} & \boldsymbol{u} \in U^i \subset \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. \end{cases} \tag{12.23}$$

We assume that it is possible for any $\boldsymbol{x}$ and $\eta > 0$ to find η-solutions $\boldsymbol{u}^i_{\boldsymbol{x}}$ to the problem (12.23), that is,

$$\begin{aligned} &\boldsymbol{u}^i_{\boldsymbol{x}} \in U^i \ : \ \varphi^i(\boldsymbol{u}^i_{\boldsymbol{x}}) + \langle \boldsymbol{x}, c^i(\boldsymbol{u}^i_{\boldsymbol{x}})\rangle \geq f^i(\boldsymbol{x}) - \eta, \\ &\quad \text{yielding } \ f^i_{\boldsymbol{x}} = \varphi^i(\boldsymbol{u}^i_{\boldsymbol{x}}) \quad \text{and} \quad \boldsymbol{\xi}^i_{\boldsymbol{x}} := c^i(\boldsymbol{u}^i_{\boldsymbol{x}})\,. \end{aligned} \tag{12.24}$$

However, the computational effort required may be substantial and will depend on the choice of η. The number of subproblems m may be large, adding further difficulty to the oracle. It may be possible to simultaneously carry out more than one such calculation using parallel computing.

In the context in consideration, the only favorable properties of f are its convexity and the additive structure. *Incremental bundle methods* [14, 21, 77] take advantage of this structure using (possibly inexact) information about some of the functions f^i and their subgradients, but trying to avoid evaluating all the m functions (i.e., solving all the m subproblems (12.23)), at least at certain "unpromising" trial points.

At the k-th iteration of the algorithm, having generated trial points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ and the corresponding lower-oracle pairs $(f^i_{\boldsymbol{x}_j}, \boldsymbol{\xi}^i_{\boldsymbol{x}_j})$ as defined in (12.24), the *disaggregate* model for f is obtained as follows:

$$f^{\boldsymbol{m}}_k(\boldsymbol{x}) := \sum_{i=1}^{m} \hat{f}^i_k(\boldsymbol{x}) \quad \text{with} \quad \hat{f}^i_k(\boldsymbol{x}) := \max_{j \in \mathcal{J}^i_k} \{ f^i_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}^i_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \} \tag{12.25}$$

for some $\mathcal{J}^i_k \subseteq \{1, \ldots, k\}$. Since the oracle is of the lower type, it can be shown (see, e.g., [14, Lemma 2.3]) that if $\mathcal{J}^i_k \supset \mathcal{J}_k$ for all $i = 1, \ldots, m$, then $\hat{f}_k(\boldsymbol{x}) \leq f^{\boldsymbol{m}}_k(\boldsymbol{x}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$, i.e., the inexact disaggregate model $f^{\boldsymbol{m}}_k$ is a better approximation of f than $\hat{f}_k$.

Aside from offering greater accuracy, the disaggregate model has another advantage, which has already been exploited in [21]: it allows for "partial" oracle evaluations in which information about the value of f at a new trial point $\boldsymbol{x}_{k+1}$ may be obtained without calling all the m individual subproblem oracles (12.24). Specifically, for any $I_{k+1} \subseteq \{1, \ldots, m\}$, we have

$$f_{I_{k+1}} \ := \underbrace{\sum_{i \in I_{k+1}} f^i_{\boldsymbol{x}_{k+1}}}_{|I_{k+1}| \text{ oracle calls}} + \underbrace{\sum_{j \notin I_{k+1}} \hat{f}^j_k(\boldsymbol{x}_{k+1})}_{\substack{m - |I_{k+1}| \text{ model} \\ \text{approximations}}} \ \leq \ f(\boldsymbol{x}_{k+1})\,. \tag{12.26}$$

The inequality (12.26) holds because $f^i_{\boldsymbol{x}_{k+1}} \leq f^i(\boldsymbol{x}_{k+1})$ and $\hat{f}^i_k(\boldsymbol{x}_{k+1}) \leq f^i(\boldsymbol{x}_{k+1})$ for $i = 1, \ldots, m$. Note that the component functions f^j with $j \notin I_{k+1}$ are not accessed by the oracles.

When examining an iterate $\boldsymbol{x}_{k+1}$, the method will sometimes not require the exact value $f(\boldsymbol{x}_{k+1})$, and will be able to discard the point based only on the information that $f(\boldsymbol{x}_{k+1}) > f_k^{tar}$ for some "target" value. If we can check this condition using a bound of the form $f_{I_{k+1}}$ as defined in (12.26), we may be able to save on oracle calls by calling $|I_{k+1}| < m$ subproblem oracles at $\boldsymbol{x}_{k+1}$, instead of all the m oracles. The pseudo-code below outlines a class of procedures for attaining this goal by incrementally enlarging the set I_{k+1}, as needed.

Algorithm 12.3: Managing oracle calls

Data: A point $\boldsymbol{x}_{k+1} \in G$, a target value $f_k^{tar} \in \mathbb{R}$, bounds η_{k+1}^i, $i = 1, \ldots, m$ for the oracle errors.

Initialize $I_{k+1} \leftarrow \emptyset$ and $f_{I_{k+1}} := \hat{f}_k(\boldsymbol{x}_{k+1})$.

while $|I_{k+1}| < m$ *and* $f_{I_{k+1}} \leq f_k^{tar}$ **do**

  Select some nonempty $J \subseteq \{1, \ldots, m\} \backslash I_{k+1}$;

  for $i \in J$ **do**

    Call oracle (12.24) to compute $(f^i_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}^i_{\boldsymbol{x}_{k+1}})$ with accuracy $\eta_{k+1}^i \geq 0$;

  Set $I_{k+1} := I_{k+1} \cup J$ and compute $f_{I_{k+1}}$ from (12.26);

Set $f_{\boldsymbol{x}_{k+1}} := f_{I_{k+1}}$, return I_{k+1}, $f_{\boldsymbol{x}_{k+1}}$ and $\{(f^i_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}^i_{\boldsymbol{x}_{k+1}}),\ i \in I_{k+1}\}$.

Initially, the algorithm above approximates the value $f(\boldsymbol{x}_{k+1})$ by using the cutting-plane models $\hat{f}^i$, $i = 1, \ldots, m$. It then selects a set of subproblems $J \subseteq \{1, \ldots, m\}$ and calls their oracles. Next, it includes J into the set I_{k+1} and updates its estimate $f_{I_{k+1}}$ of $f(\boldsymbol{x}_{k+1})$. If this calculation is sufficient to establish that $f(\boldsymbol{x}_{k+1}) > f_k^{tar}$, or already all subproblem oracles have been called, it exits. Otherwise, it repeats this procedure with another subset J of oracles.

In serial computing, it would be most economical to fix $|J| = 1$ and thus evaluate a single subproblem oracle per iteration. In a parallel setting, it might be advantageous to choose $|J| > 1$, and perform the oracle calls concurrently.

Note that Algorithm 12.3 is not specific about how to select the set J of subproblem oracles to invoke at each iteration. Determining the best strategy for selecting J will likely require some experimental study and could well be application-dependent. One can imagine greedy strategies in which one attempts to first select oracles that seem likely to be quick to evaluate, or to yield a large increase in the objective estimate, or some combination of these criteria.

If U^i in (12.23) is a compact set, c^i is a continuous function, and errors $\eta^i \geq 0$ chosen in Algorithm 12.3 are bounded, then the overall error of the lower oracle scheme above is bounded regardless of the size of the index set I_{k+1}, [14, Lemma 2.5]. Thus, Algorithm 12.2 can possibly be employed with Algorithm 12.3 to explore additive structures in difficult optimization problems.

The recent work [77] proposes a more general family of incremental bundle methods for problems with additive structure. The methods in [77] require two additional assumptions, that are satisfied in many relevant applications. One is that the Lipschitz constant of the functions must be available. The other is that

oracles must also produce upper estimates on the function values. Exploiting upper estimates is an interesting feature that allows to approximate function values at points where the oracle had not been called. Furthermore, the methods in [77] can skip oracle calls entirely for some of the component functions, not only at null steps as in [14] (and Algorithm 12.2 above combined with Algorithm 12.3), but also at serious steps. These methods can work with oracles whose error bound is either known or unknown. Furthermore, such oracles may not necessarily be able to attain arbitrary precision requested by the user/algorithm, or even provide reliable upper bounds. Of course, when dealing with such general oracles, some properties of the incremental algorithm might be lost (e.g., exactness of the computed solution).

12.4.4.3 Nonlinearly Constrained Problems

We next comment briefly on inexact proximal bundle methods for nonlinearly constrained problems of the form (12.1) where, mostly for simplicity, we consider that $G = \mathbb{R}^n$.

If the Slater condition holds, i.e., $h(z) < 0$ for some $z \in \mathbb{R}^n$, then the *exact penalty* approach can be used. Recall that the Slater condition ensures that the set of Lagrange multipliers associated to any solution of (12.1) is nonempty and bounded [72]. Then, for any ρ greater than the largest Lagrange multiplier associated to some solution to (12.1), in principle, the problem (12.1) can be solved minimizing the exact penalty function

$$F(\boldsymbol{x}) := f(\boldsymbol{x}) + \rho \max\{h(\boldsymbol{x}), 0\},$$

i.e., the original problem is reduced to the unconstrained setting (12.17), with f replaced by F. Note, however, that Lagrange multipliers are obviously unknown (just like the solution itself), and thus a suitable value of the penalty parameter ρ is unkown as well. It thus needs to be adjusted along iterations, using appropriate strategies.

A more general approach is presented in [71], where a bilevel problem is considered (i.e., that of minimizing one function over the set of minimizers of another). The usual constrained optimization case is obtained taking the lower-level function as a penalization of infeasibility, e.g., $\max\{h(\boldsymbol{x}), 0\}$. The method in [71] does not require constraint qualifications, though "penalization" is not exact. In [71] exact oracles are assumed, but it should be possible to introduce inexactness along of nowadays well understood patterns.

Approximations of the constrained problem by bundle-like linearly constrained subproblems introduces an *exogenous* inaccuracy that can be easily handled together with the genuine errors in the f- and h-oracles. With a model of the form $\hat{F}_k(\boldsymbol{x}) := \hat{f}_k(\boldsymbol{x}) + \rho \max\{\hat{h}_k(\boldsymbol{x}), 0\}$, the property that $\hat{F}_k(\cdot) \leq F(\cdot) + \eta_F$ (for some overall error $\eta_F \geq 0$) follows from the f- and h-models and oracles. As already noted, the difficulty lies in estimating the penalty parameter. Also, in the

inexact case, it appears that a more accurate F-oracle is needed. A possible update is $\rho_k = \max\{\rho_{k-1}, \lambda_k + 1\}$, where $\lambda_k \geq 0$ is a Lagrange multiplier associated to the solution $\boldsymbol{x}_{k+1}$ of the QP

$$\begin{cases} \text{minimize} & \hat{f}_k(\boldsymbol{x}) + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & \hat{h}_k(\boldsymbol{x}) \leq 0. \end{cases}$$

This QP is feasible (by the Slater assumption), and its solution also solves

$$\text{minimize} \quad \hat{f}_k(\boldsymbol{x}) + \rho_k \max\{\hat{h}_k(\boldsymbol{x}),\, 0\} + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2\,,$$

for sufficiently large ρ_k. For the approach to work as the exact penalty method, ρ_k needs to stabilize eventually, which requires the Lagrange multiplier sequence $\{\lambda_k\}$ to be bounded, e.g., [20, Lemma 7.1]. Once the penalty parameter stabilizes at a value ρ, Theorem 12.4 applies to the function F. In particular, accumulation points of the sequence $\{\hat{\boldsymbol{x}}_k\}$ solve the constrained problem within an accuracy bound depending also on the asymptotic error made when estimating the penalty parameter at serious steps.

A specialized inexact proximal bundle method that does not require penalization is proposed in [78]. Therein, the authors deal with nonlinearly constrained convex programs by employing an improvement function that is slightly more general than the one considered in Sect. 12.3.

12.5 Doubly stabilized bundle method (DSBM)

Consider the problem (12.1) with no h-constraint. As seen in the previous sections, level bundle methods employ the model $\hat{f}_k$ of f in the subproblem's constraints:

$$\begin{cases} \text{minimize} & \|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & \boldsymbol{x} \in \mathcal{L}_k := \{\boldsymbol{x} \in G : \hat{f}_k(\boldsymbol{x}) \leq f_k^{lev}\}. \end{cases} \tag{12.27}$$

The proximal bundle methods use the model in the objective function of the subproblem:

$$\begin{cases} \text{minimize} & \hat{f}_k(\boldsymbol{x}) + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & \boldsymbol{x} \in G. \end{cases} \tag{12.28}$$

It is worth to mention that the choice of the parameter t_k in the proximal variant is quite a delicate task. Although the simplest choice $t_k = t > 0$ (for all k) is enough to prove theoretical convergence, it is well understood that for practical efficiency t_k must be properly updated along iterations. For level bundle algorithms a fixed level parameter f_k^{lev} is not possible as this may give infeasible level sets. But as seen in Sect. 12.3, there exist strategies that manage f_k^{lev} by simple explicit calculations. To simultaneously compute trial points and obtain a meaningful update for the prox-parameter t_k, the *doubly stabilized bundle method*—DSBM—of [18] combines level and proximal regularizations by adding to (12.28) the level constraint. More specifically, the method defines trial points by solving

$$\begin{cases} \text{minimize} & \hat{f}_k(\boldsymbol{x}) + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & \boldsymbol{x} \in G \ \text{ and } \ \hat{f}_k(\boldsymbol{x}) \le f_k^{lev}. \end{cases} \tag{12.29}$$

We first observe that the above doubly stabilized subproblem can be represented by

$$\begin{cases} \text{minimize} & r + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \boldsymbol{x} - \boldsymbol{x}_j \rangle \le r, \quad j \in \mathcal{J}_k, \\ & \boldsymbol{x} \in G\,, \ r \in \mathbb{R} \ \text{ and } \ r \le f_k^{lev}. \end{cases}$$

This QP has just one extra scalar bound constraint compared to the QP resulting from (12.28), and one more variable compared to (12.27). Thus, (12.29) is no harder (or at least, cannot be much harder) to solve than (12.28) or (12.27).

Overall, there seems to be some consensus that for solving unconstrained problems proximal bundle methods are very good choices (though updating t_k is not straightforward), while for constrained problems level bundle methods might be preferable. The doubly stabilized bundle method combines the attractive features of both approaches in a single algorithm [18]. One advantage when compared to the proximal approach is that the level set constraint provides a certain Lagrange multiplier, which is used to update the proximal parameter in a simple manner. When compared to level bundle methods, the objective function $\hat{f}_k(\boldsymbol{x}) + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2$ with proximal regularization allows for searching for good points inside of the level set $\mathcal{L}_k$ of (12.27), and not only on its boundary, as the level method does.

Another interesting feature of the doubly stabilized bundle method is that exact and inexact data are handled in the same manner, as is the case for the level variant and in contrast to the proximal.

12.5.1 Optimality Certificate and More

We first state some properties of the minimizer $\boldsymbol{x}_{k+1}$ in (12.29).

Proposition 12.5 *If $\mathcal{L}_k = \{x \in G : \hat{f}_k(\boldsymbol{x}) \leq f_k^{lev}\} \neq \emptyset$ then the problem (12.29) has the unique solution $\boldsymbol{x}_{k+1}$. In addition, if G is polyhedral or* ri $G \cap \{\boldsymbol{x} \in \mathbb{R}^n : \hat{f}_k(\boldsymbol{x}) \leqslant f_k^{lev}\} \neq \emptyset$ *then there exist $\boldsymbol{p}_k^f \in \partial \hat{f}_k(\boldsymbol{x}_{k+1})$ and $\boldsymbol{p}_k^G \in \partial i_G(\boldsymbol{x}_{k+1})$, and (scalar) Lagrange multipliers $\mu_k \geqslant 1$ and $\lambda_k \geqslant 0$ such that*

$$\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_k - t_k \mu_k \hat{\boldsymbol{\xi}}_k, \quad \text{with } \hat{\boldsymbol{\xi}}_k = \boldsymbol{p}_k^f + \frac{1}{\mu_k} \boldsymbol{p}_k^G, \tag{12.30}$$

$$\mu_k = \lambda_k + 1 \quad \text{and} \quad \lambda_k (\hat{f}_k(\boldsymbol{x}_{k+1}) - f_k^{lev}) = 0.$$

Furthermore, the aggregate linearization $\bar{f}_{k_a}(\cdot) := \hat{f}_k(\boldsymbol{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \cdot - \boldsymbol{x}_{k+1} \rangle$ satisfies $\bar{f}_{k_a}(\boldsymbol{x}) \leqslant \hat{f}_k(\boldsymbol{x}) \leqslant f(\boldsymbol{x}) + i_G(\boldsymbol{x}) + \eta$ for all $\boldsymbol{x} \in \mathbb{R}^n$, where we assume that (12.2) holds for the oracle.

Proof See [18, Proposition 1]. □

Interestingly, the Lagrange multiplier μ_k in Proposition 12.5 indicates that the solution $\boldsymbol{x}_{k+1}$ of (12.29) either solves the proximal master problem (12.28), or the level one (12.27).

Lemma 12.3 *For $t_k > 0$ and $f_k^{lev} \in \mathbb{R}$, let $\boldsymbol{x}^{prox} \in G$ and $\boldsymbol{x}^{lev} \in G$ be the (unique) solutions of problems (12.28) and (12.27), respectively. Let $\boldsymbol{x}_{k+1} \in G$ be the unique solution of the problem (12.29). Then it holds that*

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}^{prox}, & \text{if } \mu_k = 1, \\ \boldsymbol{x}^{lev}, & \text{if } \mu_k > 1, \end{cases}$$

where μ_k is the Lagrange multiplier defined in Proposition 12.5.

Proof The result follows by comparing the optimality conditions of the three strongly convex subproblems (12.28), (12.27) and (12.29). See [18, Lemma 1] for full details. □

It is thus clear that each iteration of the doubly stabilized algorithm makes either a step of the associated proximal bundle method, or of the level method. At every iteration, the algorithm makes this choice *automatically*.

Once the iterate $\boldsymbol{x}_{k+1}$ is computed, the oracle provides the new estimate function value $f_{\boldsymbol{x}_{k+1}}$. As in the proximal bundle methods, we shall change the stability center when the descent test (12.20) is verified:

$$f_{\boldsymbol{x}_{k+1}} \leqslant f_{\hat{\boldsymbol{x}}_k} - \kappa v_k \,.$$

As before, $\kappa \in (0,1)$ and $v_k = f_{\hat{\boldsymbol{x}}_k} - \hat{f}_k(\boldsymbol{x}_{k+1})$ is the decrease predicted by the model. It is worth to mention that the level parameter f_k^{lev} also provides *expected* decrease

$$v_k^{lev} := f_{\hat{\boldsymbol{x}}_k} - f_k^{lev} > 0,$$

where the inequality follows from the fact that level parameters are chosen to satisfy $f_k^{lev} < f_{\hat{\boldsymbol{x}}_k}$ for all k. The following result establishes helpful connections among the predicted decrease v_k, the expected decrease v_k^{lev} and the aggregate error

$$\hat{e}_k = f_{\hat{\boldsymbol{x}}_k} - \bar{f}_{k_a}(\hat{\boldsymbol{x}}_k).$$

Proposition 12.6 *It holds that*

$$v_k = \hat{e}_k + t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|^2 \geqslant v_k^{lev},$$

where μ_k is the Lagrange multiplier defined in Proposition 12.5. Moreover, if $\mu_k > 1$ then $v_k = v_k^{lev}$. Furthermore, it holds that

$$\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k + 2\eta)}[f(\hat{\boldsymbol{x}}_k) + i_G(\hat{\boldsymbol{x}}_k)].$$

Proof It follows from the definition of $\hat{e}_k$ and the left-most formula in (12.30) that

$$\begin{aligned}\hat{e}_k = f_{\hat{\boldsymbol{x}}_k} - \bar{f}_{k_a}(\hat{\boldsymbol{x}}_k) &= f_{\hat{\boldsymbol{x}}_k} - [\hat{f}_k(\boldsymbol{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1}\rangle] \\ &= v_k - t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|^2.\end{aligned}$$

In addition, since $\boldsymbol{x}_{k+1}$ is feasible in (12.29), we have that $\hat{f}_k(\boldsymbol{x}_{k+1}) \leqslant f_k^{lev} = f_{\hat{\boldsymbol{x}}_k} - v_k^{lev}$, which implies $v_k^{lev} \leqslant v_k$. Recall also that if $\mu_k > 1$ then $\lambda_k > 0$, in which case (12.30) implies $\hat{f}_k(\boldsymbol{x}_{k+1}) = f_k^{lev}$, so that $v_k^{lev} = v_k$. The inclusion $\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k + 2\eta)}[f(\hat{\boldsymbol{x}}_k) + i_G(\hat{\boldsymbol{x}}_k)]$ follows from the same steps in the proof of Lemma 12.2. □

As in the proximal bundle method, we can stop the method when both $\hat{e}_k$ and $\hat{\boldsymbol{\xi}}_k$ are small enough. An alternative stopping test is borrowed from the level variant: the management of the level parameter f_k^{lev} provides (inexact) lower bound f_k^{low} and thus the DSBM can stop when $f_{\hat{\boldsymbol{x}}_k} - f_k^{low}$ is small enough.

12.5.2 Doubly Stabilized Bundle Algorithm

We now present the inexact doubly stabilized algorithm of [18] for convex problems in the form (12.17).

Algorithm 12.4: DSBM

Data: Stopping tolerances $\delta_1^{tol}, \delta_2^{tol}, \delta_3^{tol} \geq 0$, descent parameters $\kappa, \gamma \in (0, 1)$, a stepsize $t_0 \geq \underline{t} > 0$, $\tau \in [\frac{1}{2}, 1)$ and $v_0^{lev} > 0$.

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_0 \in G$, set $\hat{\boldsymbol{x}}_0 := \boldsymbol{x}_0$, and call the oracle to compute $(f_{\boldsymbol{x}_0}, \boldsymbol{\xi}_{\boldsymbol{x}_0})$. Set $\mathcal{J}_0 := \{0\}$ and $k := 0$.

Step 1. (*First stopping criterion*) Set $\Delta_k := f_{\hat{\boldsymbol{x}}_k} - f_k^{low}$. If $\Delta_k \leq \delta_3^{tol}$, **stop**.

Step 2. (*Trial point computation*) Set $f_k^{lev} := f_{\hat{\boldsymbol{x}}_k} - v_k^{lev}$ and try to solve (12.29) to obtain $\boldsymbol{x}_{k+1}$ and a Lagrange multiplier λ_k associated to the level constraint $\hat{f}_k(\boldsymbol{x}) \leqslant f_k^{lev}$. Set $\mu_k := \lambda_k + 1$, $\hat{\boldsymbol{\xi}}_k := (\hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1})/(t_k \mu_k)$ and $\hat{e}_k := v_k - t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|^2$. If (12.29) is infeasible set $f_k^{low} := f_k^{lev}$, $v_k^{lev} := (1 - \gamma)\Delta_k$ and go back to Step 1.

Step 3. (*Second stopping criterion*) If $\|\boldsymbol{\xi}_k\| \leq \delta_1^{tol}$ and $\hat{e}_k \leq \delta_2^{tol}$, **stop**.

Step 4. (*Oracle call and descent test*) Call the inexact oracle (12.2) to compute $(f_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}_{\boldsymbol{x}_{k+1}})$. Set $f_{k+1}^{low} = f_k^{low}$.

- (*Serious step*) If the descent test (12.20) holds, then set $\hat{\boldsymbol{x}}_{k+1} := \boldsymbol{x}_{k+1}$, $t_{k+1} := \mu_k t_k$ and $v_{k+1}^{lev} := \min\{v_k^{lev}, (1 - \gamma)(f_{\hat{\boldsymbol{x}}_{k+1}} - f_{k+1}^{low})\}$.
- (*Null step*) Otherwise, set $\hat{x}_{k+1} := \hat{x}_k$. If $\mu_k > 1$ (level iterate) and $\hat{e}_k \geq -\tau t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|^2$, set $v_{k+1}^{lev} := \gamma v_k^{lev}$; otherwise set $v_{k+1}^{lev} := v_k^{lev}$. Choose $t_{k+1} \in [\underline{t}, t_k]$.

Step 5. (*Bundle management*) Same as Step 5 of Algorithm 12.2. Increase k by 1 and go to Step 1.

Observe that the lower bound f_k^{low} is only updated when the level set $\mathcal{L}_k$ is empty in Step 2. It thus follows from a similar analysis to the one presented in Lemma 12.1 that $f_k^{low} \leq f^* + \eta$ for all k as long as f_0^{low} is a valid lower bound. (Note that Algorithm 12.4 accepts the choice that $f_0^{low} := -\infty$.) Therefore, if the algorithm stops at Step 1, we have that

$$\delta_3^{tol} \geq f_{\hat{\boldsymbol{x}}_k} - f_k^{low} \geq f(\hat{\boldsymbol{x}}_k) - \eta - (f^* + \eta)\,,$$

i.e., $\hat{\boldsymbol{x}}_k$ is a $(\delta_3^{tol} + 2\eta)$-approximate solution to the problem (12.17).

We point out that $v_k^{lev} > 0$ as long as $\Delta_k > 0$. This property with Proposition 12.6 ensures that $v_k > 0$ for all k, i.e., the decent test (12.20) makes sense always. This is why Algorithm 12.4 does not need any additional procedure to handle inexact data, such as *noise attenuation* in Algorithm 12.2.

Step 5 increases the proximal parameter t_k only after descent steps resulting from level iterations ($\mu_k > 1$). On the other hand, t_k can be decreased only after null steps. A simple rule used in the numerical experiments of [18] is $t_{k+1} := \max\{\underline{t}, t_k v_k^{lev}/v_k\}$, which decreases the proximal parameter only after null steps resulting from proximal iterations ($v_k > v_k^{lev}$ is only possible when $\mu_k = 1$, see Proposition 12.6). In this manner, the level parameter f_k^{lev} and the multiplier μ_k

indicate how to update the proximal parameter t_k. This simple strategy has shown good performance in practice.

It is interesting to note that if at Step 2 (for all k) one sets $f_k^{lev} = \infty$, Algorithm 12.4 becomes a proximal bundle algorithm, c.f. Lemma 12.3.

12.5.3 Convergence Analysis

Convergence analysis of the DSBM has to account for all the possible combinations of level and proximal steps, whether null or descent, and the possibility of empty level sets. To that end the following three possible cases are considered:

(i) the level sets $\mathcal{L}_k$ are empty infinitely many times;
(ii) item (i) does not happen, and infinitely many descent steps are generated;
(iii) in the same situation, finitely many descent steps are generated.

In what follows, we assume that $\delta_1^{tol} = \delta_2^{tol} = \delta_3^{tol} = 0$ and that Algorithm 12.4 does not stop. If the algorithm stops for zero tolerances in Step 1 or Step 3, then the last descent iterate is a 2η-solution to the problem, as previously mentioned.

Proposition 12.7 (Infinitely Many Empty Level Sets) *Suppose the level set $\mathcal{L}_k$ is empty infinitely many times. Then $\Delta_k \to 0$.*

Proof Every time $\mathcal{L}_k$ is found to be empty, the lower bound is increased by $v_k^{lev} > 0$. Since $f^* + \eta \geq f_k^{low}$ for all k, then $v_k^{lev} \to 0$. The definition $v_k^{lev} := (1-\gamma)\Delta_k$ (and monotonicity of $\{f_{\hat{x}_k}\}$) yields $\Delta_k \to 0$. □

Consider now the case where $\mathcal{L}_k \neq \emptyset$ for all k large enough, and there is a finite number of descent steps.

Proposition 12.8 (Infinite Loop of Null Steps) *If the algorithm generates an infinite sequence of null steps after a last serious iterate, then* (12.21) *holds with* $\mathcal{K} = \{0, 1, \ldots\}$.

The proof of Proposition 12.8 is rather technical; see [18, Lemma 7].

Proposition 12.9 (Infinitely Many Serious Steps) *If the algorithm generates an infinite sequence of serious steps, then $\Delta_k \to 0$ or/and* (12.21) *holds for $\mathcal{K} = \{k \in \mathbb{N} :$ the descent test* (12.20) *is satisfied for $k\}$.*

Proof Analogously to the proof of Proposition 12.4, one obtains that $v_k \to 0$ as $\mathcal{K} \ni k \to \infty$. Proposition 12.6 then ensures that $v_k^{lev} \to 0$ as $\mathcal{K} \ni k \to \infty$. If there exists an infinite subset $\mathcal{K}' \subset \mathcal{K}$ such that

$$v_{k+1}^{lev} = \min\{v_k^{lev}, (1-\gamma)(f_{\hat{x}_{k+1}} - f_{k+1}^{low})\}$$

for all $k \in \mathcal{K}'$ in Step 2, then $\Delta_k \to 0$ as $\mathcal{K}' \ni k \to \infty$, as well as for $\mathcal{K} \ni k \to \infty$. Otherwise, v_k^{lev} is decreased during null steps. In the latter case, the condition $\hat{e}_k \geq -\tau t_k \mu_k \|\hat{\xi}_k\|^2$ holds. Then, as in the proof of Proposition 12.3, $v_k \to 0$ as $\mathcal{K} \ni k \to \infty$ implies (12.21). □

Combining all the cases considered above, we conclude the following.

Theorem 12.5 *Suppose that the problem* (12.17) *has bounded level sets, the inexact oracle satisfies* (12.2) *and that in Algorithm 12.4 one sets* $\delta_1^{tol} = \delta_2^{tol} = \delta_3^{tol} = 0$. *Then* $\Delta_k \to 0$ *or/and the optimality certificate* (12.21) *holds for some index set* $\mathcal{K}$. *Furthermore, every accumulation point* $\bar{\boldsymbol{x}}$ *of the sequence* $\{\hat{\boldsymbol{x}}_k\}$ *is a* 2η*-solution to the problem* (12.17).

12.6 Nonconvex Objective Functions

Consider the problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in G, \end{cases} \tag{12.31}$$

where the feasible set $G \subset \mathbb{R}^n$ is convex and compact, and the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is *proper* [65, p. 5], *regular* [65, Definition 7.25], and locally Lipschitz continuous (LLC) with full domain. Note that f need not be convex. In this situation, the Clarke subdifferential $\partial f(\boldsymbol{x})$ is well-defined for $\boldsymbol{x} \in \mathbb{R}^n$, and it can be considered as the convex hull of all possible limits of gradients at points of differentiablity, for all sequences of such points converging to $\boldsymbol{x}$. For alternative definitions of the subdifferential of regular functions, see [65, Chapter 8].

Again, we are interested in the situations where for a given point, only some inexact information about the function and subgradient values is available. For function values, this is modeled the same way as in the convex case above: for each given $\boldsymbol{x} \in \mathbb{R}^n$ an oracle returns some estimate $f_{\boldsymbol{x}}$ such that

$$f_{\boldsymbol{x}} = f(\boldsymbol{x}) - \eta_{\boldsymbol{x}}^v, \tag{12.32}$$

where the sign of errors $\eta_{\boldsymbol{x}}^v \in \mathbb{R}$ is not specified (so that the true function value can be either overestimated or underestimated).

As for approximate subgradients, we note that in the nonconvex case, a number of different interpretations of inexactness is possible. Here, we adopt the rather natural view of, e.g., [34, 73]. In particular, we consider that at a point $\boldsymbol{x} \in \mathbb{R}^n$, an element $\boldsymbol{\xi}_{\boldsymbol{x}} \in \mathbb{R}^n$ approximates within tolerance $\eta_{\boldsymbol{x}}^s \geq 0$ some subgradient of f at $\boldsymbol{x}$ if

$$\boldsymbol{\xi}_{\boldsymbol{x}} \in \partial f(\boldsymbol{x}) + \bar{B}(\mathbf{0}; \eta_{\boldsymbol{x}}^s), \tag{12.33}$$

where $\bar{B}(\mathbf{0}; \eta_{\boldsymbol{x}}^s)$ is the closed (Euclidean) ball of radius $\eta_{\boldsymbol{x}}^s$ centered at the origin. This means that there exists some exact subgradient $\boldsymbol{z} \in \partial f(\boldsymbol{x})$ such that $\|\boldsymbol{\xi}_{\boldsymbol{x}} - \boldsymbol{z}\| \leq \eta_{\boldsymbol{x}}^s$, i.e., $\boldsymbol{\xi}_{\boldsymbol{x}}$ approximates *this* subgradient $\boldsymbol{z}$ with the tolerance $\eta_{\boldsymbol{x}}^s \geq 0$.

The error terms in (12.32) and (12.33) are assumed to be uniformly bounded on the feasible set: for all $\boldsymbol{x} \in G$, it holds that

$$|\eta_{\boldsymbol{x}}^v| \leq \eta^v \quad \text{and} \quad 0 \leq \eta_{\boldsymbol{x}}^s \leq \eta^s. \tag{12.34}$$

However, the error terms themselves, or even their bounds η^v and η^s, are usually unknown.

12.6.1 Some Examples of Inexact Nonconvex Oracles

Next, we briefly discuss some situations which naturally lead to the setting of (12.32), (12.33) and (12.34).

Example 12.6 (Derivative-Free Optimization) Suppose f is twice continuously differentiable (generally nonconvex), but only its (exact) function values are accessible, with no derivatives information available. This is the framework of *derivative-free optimization*; see [4, 10]. In this setting, the gradients are approximated by finite differences, linear interpolation, or other techniques. Suitable error bounds can be given for a good number of such techniques; see [10, Sections 2–5] for some examples. Similar error bounds exist also for a variety of other (sub-)gradient approximation methods [5, 30–32, 48]. In general, these error bounds involve the Lipschitz constant of the true gradient, the geometry of the sample set of points used to create the approximation, and the diameter of the sample set. As the sample set is created by the user, its geometry and diameter are controlled. The compactness of G, in turn, yields a bound on the Lipschitz constant. One has then the error bound for the approximated gradients, but the value of the bound is unknown, of course.

Example 12.7 (H_∞-Control) In H_∞-control [1, 62], certain nonconvex functions can be locally approximated by using the support function of a compact set. A detailed explanation of this approximation technique is given in [62, Subsection 1.9]. An error bound of the form (12.32), (12.33) and (12.34), is provided in [1, Lemma 2.1]. Moreover, the function in question is lower-$\mathcal{C}^2$ (and therefore locally Lipschitz) [62, Lemma 9].

Example 12.8 (Stochastic Simulations) When the objective function is provided through stochastic simulation, the errors in the function and subgradient evaluations are understood through probability distribution functions. This encompasses Example 12.5 above, where also some details are given.

12.6.2 Models: Handling Nonconvexity and Inexactness

Bundle methods for nonconvex functions using *exact* information have been considered in [1, 27, 33, 43, 46, 52, 53, 56, 76]. Note that because of nonconvexity, even when exact function and subgradient values are used to build the cutting-plane model of f, this model can cut the graph of the function (and thus its minima). In other words, unlike in the convex case, the so-called *linearization errors* can be negative, even when the data is exact. To deal with this issue, most methods "downshift" the model, i.e., negative linearization errors are made nonnegative by "brute force" of downshifting. The method of [33] also tilts the slopes of the cutting planes, in addition to downshifting.

In the nonconvex case with *inexact* oracles, negative linearization errors may be caused by nonconvexity of the function, by inexact data, or by both. This presents additional challenges. To the best of our knowledge, the only works which deal with inexact information in bundle methods for nonconvex functions are [34] and [62]. The method of [62] employs the downshift mechanism that modifies linearization errors if they are negative. The method of [34] uses the ideas of [33]; in particular, it both downshifts the cutting-planes and tilts their slopes. We next outline the algorithm of [34].

As in the usual proximal bundle methods, at iteration k one has available information computed at some previous iterations: in our case, $f_{\boldsymbol{x}_j}$ and $\boldsymbol{\xi}_{\boldsymbol{x}_j}$, $j \in \mathcal{J}_k$ ($j \leq k$), which are the approximate function and subgradient values satisfying the relations (12.32), (12.33), and (12.34) written for $\boldsymbol{x} = \boldsymbol{x}_j$. Among the previous iterates, one is designated as the stability center $\hat{\boldsymbol{x}}_k \in G$ (the "best" point computed so far, with the approximate function value $f_{\hat{\boldsymbol{x}}_k}$).

Note that the usual cutting-plane model of the objective function f obtained from this information would be given by

$$\max_{j \in \mathcal{J}_k} \{f_{\boldsymbol{x}_j} + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \cdot - \boldsymbol{x}_j \rangle\} = f_{\hat{\boldsymbol{x}}_k} + \max_{j \in \mathcal{J}_k} \{-e_j^k + \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \cdot - \hat{\boldsymbol{x}}_k \rangle\},$$

where

$$e_j^k = f_{\hat{\boldsymbol{x}}_k} - f_{\boldsymbol{x}_j} - \langle \boldsymbol{\xi}_{\boldsymbol{x}_j}, \hat{\boldsymbol{x}}_k - \boldsymbol{x}_j \rangle \tag{12.35}$$

are the linearization errors. These linearization errors would have been nonnegative in the convex case with exact data, but not in our setting. We thus introduce the following *modified* cutting-plane model:

$$\hat{f}_k(\boldsymbol{x}) := f_{\hat{\boldsymbol{x}}_k} + \max_{j \in \mathcal{J}_k} \{-c_j^k + \langle \boldsymbol{s}_j^k, \boldsymbol{x} - \hat{\boldsymbol{x}}_k \rangle\}, \tag{12.36}$$

where in each affine piece both the intercept c_j^k and the slope $\boldsymbol{s}_j^k$ correspond, respectively, to the linearization error and an approximate subgradient of the "locally convexified" function $f(\cdot) + \frac{\beta_k}{2}\|\cdot - \hat{\boldsymbol{x}}_k\|^2$. The convexification parameter $\beta_k > 0$

is adjusted dynamically along iterations, and is taken sufficiently large to make the intercept c_j^k nonnegative. Accordingly, each affine piece has a shifted nonnegative intercept

$$0 \le c_j^k := e_j^k + b_j^k, \quad \text{where} \quad b_j^k := \frac{\beta_k}{2}\|\boldsymbol{x}_j - \hat{\boldsymbol{x}}_k\|^2, \tag{12.37}$$

with e_j^k given by (12.35), and a modified slope

$$\boldsymbol{s}_j^k := \boldsymbol{\xi}_{\boldsymbol{x}_j} + \beta_k(\boldsymbol{x}_j - \hat{\boldsymbol{x}}_k), \tag{12.38}$$

which results from tilting the given approximate subgradient $\boldsymbol{\xi}_{\boldsymbol{x}_j}$ at $\boldsymbol{x}_j$ by means of the convexification parameter β_k. Any choice of β_k that makes c_k^j in (12.37) nonnegative is acceptable; we take

$$\beta_k \ge \max\left\{\max_{j\in\mathcal{J}_k,\, \boldsymbol{x}_j\neq\hat{\boldsymbol{x}}_k} \frac{-2e_j^k}{\|\boldsymbol{x}_j - \hat{\boldsymbol{x}}_k\|^2}, 0\right\} + \gamma\,, \tag{12.39}$$

for a (small) positive parameter γ.

Once the cutting-plane model is set up, choosing a prox-parameter $t_k > 0$, the new iterate is given by $\boldsymbol{x}_{k+1}$, the solution of the usual subproblem of the proximal bundle method:

$$\begin{cases} \text{minimize} & \hat{f}_k(\boldsymbol{x}) + \frac{1}{2t_k}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_k\|^2 \\ \text{subject to} & \boldsymbol{x} \in G\,. \end{cases} \tag{12.40}$$

12.6.3 Key Relations and the Stationarity Certificate

We next discuss relations between the objects the algorithm constructs based on solving (12.40), with the model satisfying (12.36)–(12.39), and how these objects can be related to (approximate) stationarity for the problem (12.31).

First, as (12.40) has the structure of the usual proximal bundle method subproblem, it holds (just as in the convex case), that

$$\boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_k - t_k\hat{\boldsymbol{\xi}}_k, \quad \text{with} \quad \hat{\boldsymbol{\xi}}_k := \boldsymbol{p}_k^f + \boldsymbol{p}_k^G,$$

where $\boldsymbol{p}_k^G \in \partial i_G(\boldsymbol{x}_{k+1})$, $\boldsymbol{p}_k^f \in \partial\hat{f}_k(\boldsymbol{x}_{k+1})$, and

$$\boldsymbol{p}_k^f := \sum_{j\in\mathcal{J}_k} \alpha_j^k \boldsymbol{s}_j^k, \quad \sum_{j\in\mathcal{J}_k} \alpha_j^k = 1,\ \alpha_j^k \ge 0\ \ , j \in \mathcal{J}_k\,.$$

Then the *aggregate linearization* is given as previously, i.e., $\bar{f}_{k_a}(\boldsymbol{x}) := \hat{f}_k(\boldsymbol{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \boldsymbol{x} - \boldsymbol{x}_{k+1} \rangle$, and the *aggregate error* by

$$\hat{e}_k := \hat{f}_k(\hat{\boldsymbol{x}}_k) - \hat{f}_k(\boldsymbol{x}_{k+1}) - \langle \boldsymbol{p}_k^f, \hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1} \rangle \geq 0, \tag{12.41}$$

where the inequality is by the fact that $\boldsymbol{p}_k^f \in \partial \hat{f}_k(\boldsymbol{x}_{k+1})$. It can be further seen that $f_{\hat{\boldsymbol{x}}_k} = \hat{f}_k(\hat{\boldsymbol{x}}_k)$, and that

$$\hat{e}_k = f_{\hat{\boldsymbol{x}}_k} - \bar{f}_{k_a}(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{p}_k^f, \boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_k \rangle = \sum_{j \in \mathcal{J}_k} \alpha_j^k c_j^k. \tag{12.42}$$

For the aggregate linearization, it holds that

$$\bar{f}_{k_a}(\boldsymbol{x}) = f_{\hat{\boldsymbol{x}}_k} + \sum_{j \in \mathcal{J}_k} \alpha_j^k (-c_j^k + \langle \boldsymbol{s}_j^k, \boldsymbol{x} - \hat{\boldsymbol{x}}_k \rangle) = f_{\hat{\boldsymbol{x}}_k} - \hat{e}_k + \langle \boldsymbol{p}_k^f, \boldsymbol{x} - \hat{\boldsymbol{x}}_k \rangle,$$

where we have used (12.42). The following result (Lemma 12.4 below) shows what the algorithm should aim for, in order to compute (approximate) stationary points of the problem (12.31).

We note that the proof of Lemma 12.4 uses the following assumption: "The number of active indices, i.e., of $j \in \mathcal{J}_k$ such that $\alpha_j^k > 0$, is uniformly bounded in k". As a practical matter, this can be readily achieved if D is polyhedral (the typical case), and an active-set QP solver is employed to solve subproblems (12.40) (this is because active-set QP solvers choose linearly independent bases).

The last assertion in Lemma 12.4 refers to lower-$\mathcal{C}^1$ functions, introduced in [74]. It is a broad class of LLC functions that contains lower-$\mathcal{C}^2$ functions. One of the equivalent characterizations of f being a lower-$\mathcal{C}^1$ function consists in f being semismooth (see, e.g., [55]) and regular.

Lemma 12.4 *Suppose the cardinality of the set* $\{j \in \mathcal{J}_k : \alpha_j^k > 0\}$ *is uniformly bounded in* k. *If* $\hat{e}^k \to 0$ *as* $k \to \infty$ *and, for some subset* $\mathcal{K} \subset \{1, 2, \ldots\}$, $\hat{\boldsymbol{x}}_k \to \bar{\boldsymbol{x}}$, $\hat{\boldsymbol{\xi}}_k \to 0$ *as* $\mathcal{K} \ni k \to \infty$, *with* $\{\beta_k : k \in \mathcal{K}\}$ *bounded, then*

$$\mathbf{0} \in \partial f(\bar{\boldsymbol{x}}) + \partial i_G(\bar{\boldsymbol{x}}) + \bar{B}(\mathbf{0}; \eta^s). \tag{12.43}$$

If, in addition, f *is lower-*$\mathcal{C}^1$, *then for each* $\varepsilon > 0$ *there exists* $\rho > 0$ *such that for all* $\boldsymbol{y} \in G \cap \bar{B}(\bar{\boldsymbol{x}}; \rho)$, *it holds that*

$$f(\boldsymbol{y}) \geq f(\bar{\boldsymbol{x}}) - (\eta^s + \varepsilon)\|\boldsymbol{y} - \bar{\boldsymbol{x}}\| - 2\eta^f. \tag{12.44}$$

Proof See [34, Lemma 5]. □

The result above indicates that to find an approximate stationary point of the problem (12.31), the algorithm should drive $\hat{e}^k$ and $\hat{\boldsymbol{\xi}}_k$ to zero along the iterations.

12.6.4 Inexact Nonconvex Proximal Bundle Algorithm

Naturally, $\hat{e}^k$ and $\|\hat{\boldsymbol{\xi}}_k\|$ are driven to zero by means of a sufficient descent condition for the objective function with respect to its value at the stability center (recall, however, that both are inexact values in our setting). Specifically, the new iterate $\boldsymbol{x}_{k+1}$ computed by solving (12.40) is accepted as the new stability center if

$$f_{\boldsymbol{x}_{k+1}} \leq f_{\hat{\boldsymbol{x}}_k} - \kappa v_k, \tag{12.45}$$

where $\kappa \in (0, 1)$ and

$$v_k := \hat{e}^k + t_k \|\hat{\boldsymbol{\xi}}_k\|^2. \tag{12.46}$$

If (12.45) does not hold, then a null step is declared and the stability center is not changed.

Algorithm 12.5: Inexact proximal bundle method for nonconvex functions

Data: Stopping tolerance $\delta^{tol} \geq 0$, a descent parameter $\kappa \in (0, 1)$, convexification safeguarding parameter $\gamma > 0$.

Step 0. (*Initialization*) Choose a starting point $\boldsymbol{x}_0 \in G$, set $\hat{\boldsymbol{x}}_0 := \boldsymbol{x}_0$, and call the oracle to compute $(f_{\boldsymbol{x}_0}, \boldsymbol{\xi}_{\boldsymbol{x}_0})$. Set $\mathcal{J}_0 := \{0\}$, $k := 0$.

Step 1. (*Trial point computation*) Find $\boldsymbol{x}_{k+1}$ solving (12.40), and let α_j^k, $j \in \mathcal{J}_k$, denote the corresponding optimal simplicial Lagrange multipliers. Compute $\hat{\boldsymbol{\xi}}_k = (\hat{\boldsymbol{x}}_k - \boldsymbol{x}_{k+1})/t_k$, $\hat{e}_k$ by (12.42), and v_k by (12.46).

Step 2. (*Stopping criterion*) If $v_k \leq \delta^{tol}$, **stop**.

Step 3. (*Oracle call and descent test*) Call the inexact oracle to compute $(f_{\boldsymbol{x}_{k+1}}, \boldsymbol{\xi}_{\boldsymbol{x}_{k+1}})$. If the descent test (12.45) holds, then declare the iterate serious: set $\hat{\boldsymbol{x}}_{k+1} := \boldsymbol{x}_{k+1}$. Otherwise, declare the iterate null: set $\hat{\boldsymbol{x}}_{k+1} := \hat{\boldsymbol{x}}_k$.

Step 4. (*Bundle management*) Set $\mathcal{J}_{k+1} := \{j \in \mathcal{J}^k : \alpha_j^k \neq 0\} \cup \{k+1\}$.

Step 5. (*Parameters updating and loop*). If the iterate was declared serious, select $t_{k+1} > 0$. If the iterate was declared null, select $0 < t_{k+1} \leq t_k$. Increase k by 1. Compute the convexification parameter by (12.39), and the new intercepts c_j^k and slopes $\boldsymbol{s}_j^k$ by (12.37) and (12.38), respectively. Go to Step 1.

It is worth noting that, contrary to many nonconvex bundle methods endowed with a linesearch, e.g., [43, 46, 53], Algorithm 12.5 does not employ linesearch. In [34, Section 5] some explanations are given as to why linesearch is not required here.

12.6.5 Convergence Results

Depending on the assumptions about the prox-parameters t_k, it can be proven that the approximate optimality conditions stated in Lemma 12.4 hold for some accumulation point of $\{\hat{\boldsymbol{x}}_k\}$; or for all accumulation points of this sequence; or for the last serious iterate generated. Naturally, it is assumed that $\delta^{tol} = 0$ and an infinite sequence $\{\boldsymbol{x}_k\}$ is generated. Convergence results below assume that the sequence of convexification parameters $\{\beta_k\}$ is bounded. This had been shown true for the related algorithm with exact data [33]. In the inexact setting, no proof is available at this time. The numerical results in [34] indicate that the assumption is reasonable and appears to hold in computation.

Theorem 12.6 (Infinitely Many Serious Iterates) *Suppose Algorithm 12.5 generates an infinite number of serious steps. Then $v_k \to 0$ as $k \to \infty$. Let the sequence $\{\beta_k\}$ be bounded. If $\sum_{k=1}^{\infty} t_k = +\infty$, then as $k \to \infty$ it holds that $\hat{e}_k \to 0$, and there exist $\mathcal{K} \subset \{1, 2, \ldots\}$ and $\bar{\boldsymbol{x}}$ such that $\hat{\boldsymbol{x}}_k \to \bar{\boldsymbol{x}}$ and $\hat{\boldsymbol{\xi}}_k \to 0$ as $\mathcal{K} \ni k \to \infty$. In particular, if the cardinality of the set $\{j \in \mathcal{J}_k : \alpha_j^k > 0\}$ is uniformly bounded in k, then the conclusions of Lemma 12.4 hold for $\bar{\boldsymbol{x}}$. If $\liminf_{k\to\infty} t_k > 0$, then these assertions hold for all accumulation points of the sequence $\{\hat{\boldsymbol{x}}_k\}$.*

Proof See [34, Theorem 6]. □

Theorem 12.7 (Finite Serious Steps Followed by Infinitely Many Null Steps) *Suppose Algorithm 12.5 generates a finite number of serious iterates, the last being $\bar{\boldsymbol{x}} := \hat{\boldsymbol{x}}_{k_0}$, followed by infinite null steps. Let the sequence $\{\beta_k\}$ be bounded and let $\liminf_{k\to\infty} t_k > 0$. Then $\boldsymbol{x}_k \to \bar{\boldsymbol{x}}$, $v_k \to 0$, $\hat{e}_k \to 0$ and $\hat{\boldsymbol{\xi}}_k \to 0$. In particular, if the cardinality of the set $\{j \in \mathcal{J}_k : \alpha_j^k > 0\}$ is uniformly bounded in k, then the conclusions of Lemma 12.4 hold for $\bar{\boldsymbol{x}}$.*

Proof See [34, Theorem 7]. □

We conclude by noting that the convergence results for Algorithm 12.5 are quite similar to the ones for the algorithm in [62]. In the latter reference, the objective function f is either ε-convex or lower-$\mathcal{C}^1$, and bounded lower level sets for f are assumed (instead of boundedness of the feasible set G). Some details of comparisons between Algorithm 12.5 and [62] are given in [34, Section 5].

12.7 Concluding Remarks and Research Perspectives

We see several possible directions to develop further or broaden optimization techniques discussed in this work.

One research direction would be extending level bundle methods to deal with nonconvex objective and/or constraint functions. While proximal bundle methods have been adapted to deal with nonconvexity (both in the exact and inexact settings, see Sect. 12.6), no level or doubly stabilized variants exist for this setting.

Another topic worth to investigate is the handling of inexact oracles in optimization problems involving *Difference-of-Convex* (DC) functions. Although practical applications of DC programming involving hard-to-evaluate functions are known in the literature (see, e.g., [76, Section 5]), currently available bundle methods for DC programs [13, 28, 40] do not handle noisy oracles. Extending DC bundle methods to inexact data (with or without on-demand accuracy) is, in our opinion, a relevant topic for research. Besides, existent DC bundle methods are all of the proximal type. Investigating level bundle methods for DC programming may give rise to algorithms possessing dimension independent iteration complexity, both in the exact and inexact settings.

Finally, we close this chapter with some words and references on bundle methods software. Due to a large number of possible stability functions, rules to set parameters, descent and stopping tests, and oracle types, finding the most appropriate (inexact) bundle method for a given application is not a trivial task for a practitioner, especially when without very specific knowledge and skills. It therefore would be essential to make available open-source software (as generic as possible) implementing several variants of bundle methods. A practitioner could then choose, among many possibilities and upon trying some test problems, the most suitable bundle algorithm and its parameter settings, to solve her/his problem. Admittedly, there is a number of bundle software packages that have been around for a while by now. Still, making available a wide range of options is something that has not happened as of yet. That said, as many practical problems can be cast in specific classes of nonsmooth optimization problems such as Lagrangian relaxation, two-stage stochastic programs and dual decomposition of multistage stochastic programs, some functional and robust computational codes have been appearing relatively recently, in the last years. `Matlab` implementations of some of the algorithms discussed here are freely available on the first author's homepage. Other freely available implementations of bundle methods are:

- `C++` codes by Antonio Frangioni can be downloaded from his homepage http://pages.di.unipi.it/frangio;
- `Fortran` codes of a variety of bundle methods (convex, nonconvex, multiobjective, DC) by Napsu Karmitsa and her collaborators can be found at the link http://napsu.karmitsa.fi;
- `Julia` codes of several implementations of bundle methods (including the doubly stabilized bundle method discussed in Sect. 12.5) are available in the open-source and parallel package DSP https://github.com/Argonne-National-Laboratory/DSP, by Kibaek Kim and Victor M. Zavala.

Acknowledgements The authors acknowledge Wim van Ackooij and Antonio Frangioni for useful suggestions that helped improve both the contents and the presentation of this work. The first author acknowledges the partial financial support of PGMO (Gaspard Monge Program for Optimization and Operations Research) of the Hadamard Mathematics Foundation, through the project "Models for planning energy investment under uncertainty". The second author is supported in part by CNPq Grant 303724/2015-3, by FAPERJ Grant 203.052/2016, and by the Russian Foundation for Basic Research grant 19-51-12003 NNIOa.

References

1. Apkarian, P., Noll, D., Prot, O.: A proximity control algorithm to minimize nonsmooth and nonconvex semi-infinite maximum eigenvalue functions. J. Convex Anal. **16**(3–4), 641–666 (2009)
2. Arnold, T., Henrion, R., Möller, A., Vigerske, S.: A mixed-integer stochastic nonlinear optimization problem with joint probabilistic constraints. Pac. J. Optim. **10**(1), 5–20 (2014)
3. Astorino, A., Frangioni, A., Fuduli, A., Gorgone, E.: A nonmonotone proximal bundle method with (potentially) continuous step decisions. SIAM J. Optim. **23**(3), 1784–1809 (2013)
4. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham (2017)
5. Bagirov, A.M., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**(2), 317–334 (2008)
6. Ben-Tal, A., Nemirovski, A.: Non-Euclidean restricted memory level method for large-scale convex optimization. Math. Program. **102**, 407–456 (2005)
7. Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Theoretical and Practical Aspects. Universitext, 2nd edn., xiv+490 pp. Springer, Berlin (2006)
8. Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C.: Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment. IEEE Trans. Power Syst. **18**, 313–323 (2003)
9. Charnes, A., Cooper, W.: Chance-constrained programming. Manag. Sci. **6**, 73–79 (1959–1960)
10. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-free Optimization. MPS/SIAM Series on Optimization, vol. 8. Society for Industrial and Applied Mathematics (SIAM)/Mathematical Programming Society (MPS), Philadelphia (2009)
11. de Oliveira, W.: Regularized optimization methods for convex MINLP problems. TOP **24**(3), 665–692 (2016)
12. de Oliveira, W.: Target radius methods for nonsmooth convex optimization. Oper. Res. Lett. **45**(6), 659–664 (2017)
13. de Oliveira, W.: Proximal bundle methods for nonsmooth DC programming. J. Global Optim. **75**, 523–563 (2019). https://doi.org/10.1007/s10898-019-00755-4
14. de Oliveira, W., Eckstein, J.: A bundle method for exploiting additive structure in difficult optimization problems. Technical report (2015)
15. de Oliveira, W., Sagastizábal, C.: Bundle methods in the XXI century: a birds'-eye view. Pesquisa Operacional **34**(3), 647–670 (2014)
16. de Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on demand accuracy. Optim. Methods Softw. **29**(6), 1180–1209 (2014)
17. de Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM J. Optim. **21**(2), 517–544 (2011)
18. de Oliveira, W., Solodov, M.: A doubly stabilized bundle method for nonsmooth convex optimization. Math. Program. **156**(1), 125–159 (2016)
19. de Oliveira, W., Tcheou, M.P.: An inertial algorithm for DC programming. Set-Valued Var. Anal. **27**, 895–919 (2019). https://doi.org/10.1007/s11228-018-0497-0
20. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Program. **148**, 241–277 (2014)
21. Emiel, G., Sagastizábal, C.: Incremental-like bundle methods with application to energy planning. Comput. Optim. Appl. **46**, 305–332 (2010)
22. Fábián, C.I.: Bundle-type methods for inexact data. Central Eur. J. Oper. Res. **8**, 35–55 (2000)
23. Fábián, C.I., Wolf, C., Koberstein, A., Suhl, L.: Risk-averse optimization in two-stage stochastic models: computational aspects and a study. SIAM J. Optim. **25**(1), 28–52 (2015)
24. Fischer, I., Gruber, G., Rendl, F., Sotirov, R.: Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. Math. Program. **105**(2), 451–469 (2006)

25. Floudas, C.A.: Generalized Benders Decomposition, 2nd edn. Springer, Berlin (2009)
26. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **13**(1), 117–156 (2002)
27. Fuduli, A., Gaudioso, M., Giallombardo, G.: Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. SIAM J. Optim. **14**(3), 743–756 (2004)
28. Gaudioso, M., Giallombardo, G., Miglionico, G., Bagirov, A.M.: Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. J. Global Optim. **71**, 37–55 (2018)
29. Genz, A., Bretz, F.: Computation of multivariate normal and t probabilities. No. 195 in Lecture Notes in Statistics. Springer, Dordrecht (2009)
30. Gupal, A.M.: A method for the minimization of almost differentiable functions. Cybernetics **13**(1), 115–117 (1977)
31. Hare, W., Macklem, M.: Derivative-free optimization methods for finite minimax problems. Optim. Methods Softw. **28**(2), 300–312 (2013)
32. Hare, W., Nutini, J.: A derivative-free approximate gradient sampling algorithm for finite minimax problems. Comput. Optim. Appl. **56**(1), 1–38 (2013)
33. Hare, W., Sagastizábal, C.: A redistributed proximal bundle method for nonconvex optimization. SIAM J. Optim. **20**(5), 2442–2473 (2010)
34. Hare, W., Sagastizábal, C., Solodov, M.: A proximal bundle method for nonsmooth nonconvex functions with inexact information. Comput. Optim. Appl. **63**(1), 1–28 (2016)
35. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**(3), 673–696 (2000)
36. Hintermüller, M.: A proximal bundle method based on approximate subgradients. Comput. Optim. Appl. **20**, 245–266 (2001)
37. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms I, 2nd edn., No. 305 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin (1996)
38. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II, 2nd edn., No. 306 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin (1996)
39. Iutzeler, F., Malick, J., de Oliveira, W.: Asynchronous level bundle methods. Math. Program. (2019). https://doi.org/10.1007/s10107-019-01414-y
40. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. J. Global Optim. **68**(3), 501–535 (2017)
41. Karmitsa, N., Gaudioso, M., Joki, K.: Diagonal bundle method with convex and concave updates for large-scale nonconvex and nonsmooth optimization. Optim. Methods Softw. **34**(2), 363–382 (2019)
42. Kiwiel, K.C.: An aggregate subgradient method for nonsmooth convex minimization. Math. Program. **27**(3), 320–341 (1983)
43. Kiwiel, K.: A linearization algorithm for nonsmooth minimization. Math. Oper. Res. **10**(2), 185–194 (1985)
44. Kiwiel, K.: Exact penalty functions in proximal bundle methods for constrained convex nondifferentiable minimization. Math. Program. **52**(2), 285–302 (1991)
45. Kiwiel, K.C.: Approximations in proximal bundle methods and decomposition of convex programs. J. Optim. Theory Appl. **84**, 529–548 (1995)
46. Kiwiel, K.C.: Restricted step and Levenberg-Marquardt techniques in proximal bundle methods for nonconvex nondifferentiable optimization. SIAM J. Optim. **6**(1), 227–249 (1996)
47. Kiwiel, K.: A proximal bundle method with approximate subgradient linearizations. SIAM J. Optim. **16**(4), 1007–1023 (2006)
48. Kiwiel, K.C.: A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **20**(4), 1983–1994 (2010)
49. Lemaréchal, C.: An extension of Davidon methods to nondifferentiable problems. Math. Program. Study **3**, 95–109 (1975)
50. Lemaréchal, C.: Lagrangian relaxation. In: Computational combinatorial optimization (Schloß Dagstuhl, 2000). Lecture Notes in Computer Science, vol. 2241, pp. 112–156. Springer, Berlin (2001)

51. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**(1), 111–147 (1995)
52. Lukšan, L., Vlček, J.: A bundle-Newton method for nonsmooth unconstrained minimization. Math. Program. **83**(3), 373–391 (1998)
53. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization. Analysis and Algorithms with Applications to Optimal Control. World Scientific, River Edge (1992)
54. Malick, J., de Oliveira, W., Zaourar, S.: Uncontrolled inexact information within bundle methods. EURO J. Comput. Optim. **5**(1), 5–29 (2017)
55. Mifflin, R.: An algorithm for constrained optimization with semismooth functions. Math. Oper. Res. **2**, 191–207 (1977)
56. Mifflin, R.: A modification and extension of Lemarechal's algorithm for nonsmooth minimization. In: Sorensen D.C., Wets R.J.B. (eds) Nondifferential and Variational Techniques in Optimization. Mathematical Programming Studies, vol. 17, pp. 77–90. Springer, Berlin (1982)
57. Mifflin, R.: A quasi-second-order proximal bundle algorithm. Math. Program. **73**(1), 51–72 (1996)
58. Mifflin, R., Sagastizábal, C.: A VU-algorithm for convex minimization. Math. Program. **104**(2–3), 583–608 (2005)
59. Miller, S.: Inexact bundle method for solving large structured linear matrix inequalities. Ph.D. Thesis, University of California, Santa Barbara (2001)
60. Montonen, O., Karmitsa, N., Mäkelä, M.M.: Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization. Optimization **67**(1), 139–158 (2018)
61. Nasri, A., Kazempour, S.J., Conejo, A.J., Ghandhari, M.: Network-constrained AC unit commitment under uncertainty: a Benders' decomposition approach. IEEE Trans. Power Syst. **31**(1), 412–422 (2016)
62. Noll, D.: Bundle method for non-convex minimization with inexact subgradients and function values. In: Computational and Analytical Mathematics. Springer Proceedings in Mathematics and Statistics, vol. 50, pp. 555–592. Springer, New York (2013)
63. Noll, D., Apkarian, P.: Spectral bundle methods for non-convex maximum eigenvalue functions: first-order methods. Math. Program. **104**(2), 701–727 (2005)
64. Ouorou, A.: A proximal cutting plane method using Chebychev center for nonsmooth convex optimization. Math. Program. **119**(2), 239–271 (2009)
65. Rockafellar, R.T., Wets, R.J.B.: Variational Analysis. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 317. Springer, Berlin (1998)
66. Ruszczyński, A.: Decomposition Methods. Handbooks in Operations Research and Management Science, vol. 10. Elsevier, Amsterdam (2003)
67. Sagastizábal, C.: Divide to conquer: decomposition methods for energy optimization. Math. Program. **134**(1), 187–222 (2012)
68. Sagastizábal, C.: On Lagrangian decomposition for energy optimization. In: Proceedings of the 8th International Congress on Industrial and Applied Mathematics, pp. 289–310. Higher Ed. Press, Beijing (2015)
69. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**(1), 146–169 (2005)
70. Solodov, M.V.: On approximations with finite precision in bundle methods for nonsmooth optimization. J. Optim. Theory Appl. **119**(1), 151–165 (2003)
71. Solodov, M.V.: A bundle method for a class of bilevel nonsmooth convex minimization problems. SIAM J. Optim. **18**(1), 242–259 (2007)
72. Solodov, M.V.: Constraint Qualifications. Wiley Encyclopedia of Operations Research and Management Science (2011). Wiley, Hoboken. https://doi.org/10.1002/9780470400531.eorms0978
73. Solodov, M.V., Zavriev, S.K.: Error stability properties of generalized gradient-type algorithms. J. Optim. Theory Appl. **98**(3), 663–680 (1998)
74. Spingarn, J.E.: Submonotone subdifferentials of Lipschitz functions. Trans. Am. Math. Soc. **264**(1), 77–89 (1981)

75. van Ackooij, W., de Oliveira, W.: Level bundle methods for constrained convex optimization with various oracles. Comput. Optim. Appl. **57**(3), 555–597 (2014)
76. van Ackooij, W., de Oliveira, W.: Nonsmooth and nonconvex optimization via approximate difference-of-convex decompositions. J. Optim. Theory Appl. **182**, 49–80 (2019). https://doi.org/10.1007/s10957-019-01500-3
77. van Ackooij, W., Frangioni, A.: Incremental bundle methods using upper models. SIAM J. Optim. **28**(1), 379–410 (2018)
78. van Ackooij, W., Sagastizábal, C.: Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. SIAM J. Optim. **24**(2), 733–765 (2014)
79. van Ackooij, W., Henrion, R., Möller, A., Zorgati, R.: Joint chance constrained programming for hydro reservoir management. Optim. Eng. **15**, 509–531 (2014)
80. van Ackooij, W., Cruz, J.B., de Oliveira, W.: A strongly convergent proximal bundle method for convex minimization in Hilbert spaces. Optimization **65**(1), 145–167 (2016)
81. van Ackooij, W., Frangioni, A., de Oliveira, W.: Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support. Comput. Optim. Appl. **65**(3), 637–669 (2016)
82. Wolfe, P.: A method of conjugate subgradients for minimizing nondifferentiable functions. Math. Program. Stud. **3**, 145–173 (1975)

Chapter 13
New Multiobjective Proximal Bundle Method with Scaled Improvement Function

Marko M. Mäkelä and Outi Montonen

Abstract Improvement functions are used in nonsmooth optimization both for constraint handling and scalarization of multiple objectives. In the multiobjective case the improvement function possesses, for example the nice property that a descent direction for the improvement function improves all the objectives of the original problem. However, the numerical experiments have shown that the standard improvement function is rather sensitive for scaling. For this reason we present here a new scaled version of the improvement function capable not only for linear but also for polynomial, logarithmic, and exponential scaling for both objective and constraint functions. In order to be convinced about the usability of the scaled improvement function, we develop a new version of the multiobjective proximal bundle method utilizing the scaled improvement function. This new method can be proved to produce weakly Pareto stationary solutions. In addition, under some generalized convexity assumptions the solutions are guaranteed to be globally weakly Pareto optimal. Furthermore, we illustrate the affect of the scaling with some numerical examples.

13.1 Introduction

Besides nonsmoothness, many practical problems, for example in economics [28], engineering [19], mechanics [26], bioinformatics [5] and machine learning [7], have also a multiobjective nature. If the problem involves more than one goal, it is more relevant to treat it as a multiobjective problem than model it as a single-objective one [36]. Indeed, if we select only one objective to be optimized, we may obtain a solution being arbitrary bad respect to the other goals. Thus, nature of the multiobjective solution is different from single-objective optimization: instead of finding the best solution for one goal, we are searching a good compromise solution

M. M. Mäkelä (✉) · O. Montonen
Department of Mathematics and Statistics, University of Turku, Turku, Finland
e-mail: makela@utu.fi; outi.montonen@utu.fi

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_13

between several goals such that all the goals have as good outcome as possible. Compared with the single-objective case, the multiobjective problem hardly ever has a unique solution.

Some basic solution approaches developed from this point of view includes strategies to select one of the multiple objectives to be optimized and others transformed into constraints or adding up all the objectives. There exists a wide range of methods to transform a multiobjective problem into a single objective one and these are called scalarization techniques [22]. Due to the multiobjective nature, the decision maker has much more responsibility of the final solution than in the single-objective optimization. For example, in interactive methods [21, 27, 33, 34], at each iteration the decision maker can affect on the solution such that the final solution would fulfill the decision maker's wishes. For the survey of different kind of multiobjective methods classified based on the role of the decision maker is given in [20]. Here we are particularly focused on descent methods achieving solutions by improving all the objectives simultaneously. Thus, the expertise and personal opinions given by the decision maker are not needed and the obtained solutions are more neutral.

Despite the large variety in multiobjective optimization methods, the methods designed from the nonsmooth basis are more rare (see, e.g., [4, 24, 25, 30]). Among the approaches to nonsmooth multiobjective optimization, we are concentrating on the technique called the *improvement function*. This technique was initially used in single-objective smooth [29, 37] and nonsmooth [1, 8, 9, 12, 14, 23, 31] optimization for constraint handling by reformulating the constrained problem to an unconstrained single-objective problem. In order to get even more benefit from the improvement function, it can be utilized also in the multiobjective framework as is done in [10, 15, 18, 21, 24, 35]. In this case, the original multiobjective constrained problem is reformulated to a single-objective unconstrained one with the improvement function as its objective. In addition to constraint handling, the improvement function applied to multiobjective optimization has two other benefits. First, the descent direction for the improvement function improves all the objectives of the original problem. Second, we get a useful relation between the optimal solutions of the original and the reformulated problems.

While in theory the improvement function works perfectly, in practice the numerical experiments have shown that the standard improvement function is sensitive for scaling. Indeed, in some cases the scaling of the improvement function can significantly reduce the amount of the computational effort needed. For example in the multiobjective double bundle method for DC optimization (see [24] and Chap. 14), the different magnitudes of the objectives may lead to a situation where one of the DC components dominates the others hiding their effects and causing numerical difficulties. In [24], this issue was successfully handled by using a primitive linear scaling procedure. In this chapter we generalize this approach and present a new scaled version of the improvement function capable not only for linear but also for polynomial, logarithmic, and exponential scaling for both objective and constraint functions.

In order to be convinced about the usability of the scaled improvement function, we develop a new version of the multiobjective proximal bundle method (MPB) [15, 18, 21] by using the scaled improvement function instead of the standard improvement function. MPB is a generalization of the proximal bundle method [11, 14, 32] for nonconvex multiobjective problems with inequality constraints. The idea in brief is to apply the proximal bundle approach to solve the problem with the scaled improvement function as the objective. The new method can be proved to produce weakly Pareto stationary solutions and under some generalized convexity assumptions the solutions are guaranteed to be globally weakly Pareto optimal.

The remaining of the chapter is organized as follows. Section 13.2 summarizes some essential preliminary material on nonsmooth optimization. The scaled improvement function is presented in Sect. 13.3 with details. We move on to apply the scaled improvement function by giving an outline of the multiobjective proximal bundle method in Sect. 13.4 and stating the convergence results in Sect. 13.5. In Sect. 13.6 we illustrate the affect of the scaling with numerical examples and finally, the chapter is concluded in Sect. 13.7.

13.2 Preliminaries

We consider a nonsmooth multiobjective optimization problem of the form

$$\begin{cases} \text{minimize} & \{f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\} \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases} \tag{13.1}$$

where the feasible region is defined by

$$S = \{\boldsymbol{x} \in \mathbb{R}^n : g_l(x) \leq 0,\ l = 1, \ldots, m\}.$$

The objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$ for all $i \in \mathcal{I} = \{1, \ldots, k\}$ and the constraint functions $g_l : \mathbb{R}^n \to \mathbb{R}$ for all $l \in \mathcal{L} = \{1, \ldots, m\}$ are supposed to be LLC.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *f°-pseudoconvex* [6], if it is LLC and for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$

$$f(\boldsymbol{y}) < f(\boldsymbol{x}) \quad \text{implies} \quad f^\circ(\boldsymbol{x}; \boldsymbol{y} - \boldsymbol{x}) < 0$$

and *f°-quasiconvex*, if

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) \quad \text{implies} \quad f^\circ(\boldsymbol{x}; \boldsymbol{y} - \boldsymbol{x}) \leq 0.$$

Note, that a convex function is always f°-pseudoconvex, which again is f°-quasiconvex (see, e.g., [17]). We recall that if a LLC function attains its local

minimum at $\boldsymbol{x}^*$, then

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*). \tag{13.2}$$

For the f°-pseudoconvex function we state the following result.

Theorem 13.1 ([6]) *An f°-pseudoconvex function f attains its global minimum at $\boldsymbol{x}^*$, if and only if*

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*).$$

For LLC functions we can derive the following subderivation rules. The proofs of the result can be found, for example, in [2, 3].

Theorem 13.2 ([3]) *Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be LLC at $\boldsymbol{x}$ for all $i = 1, \ldots, s$. Then the function*

$$f(\boldsymbol{x}) = \max\{f_i(\boldsymbol{x}) : i = 1, \ldots, s\}$$

is also LLC at $\boldsymbol{x}$ and

$$\partial f(\boldsymbol{x}) \subseteq \operatorname{conv}\left\{\bigcup_{i=1}^{s} \partial f_i(\boldsymbol{x}) : f_i(\boldsymbol{x}) = f(\boldsymbol{x})\right\}. \tag{13.3}$$

In addition, if f_i is f°-pseudoconvex (f°-quasiconvex) for all $i = 1, \ldots, s$, then f is also f°-pseudoconvex (f°-quasiconvex).

Theorem 13.3 ([3]) *Let $f : \mathbb{R}^n \to \mathbb{R}$ be such that $f = h \circ g$, where $g : \mathbb{R}^n \to \mathbb{R}$ is LLC at $\boldsymbol{x} \in \mathbb{R}^n$ and $h : \mathbb{R} \to \mathbb{R}$ is LLC at $g(\boldsymbol{x}) \in \mathbb{R}$. Then f is LLC at $\boldsymbol{x}$ and*

$$\partial f(\boldsymbol{x}) \subseteq \operatorname{conv}\left\{\partial h\big(g(\boldsymbol{x})\big)\partial g(\boldsymbol{x})\right\}. \tag{13.4}$$

The equality holds in (13.4) if in addition

(i) *h is continuously differentiable at $g(\boldsymbol{x})$, or*
(ii) *g is subdifferentially regular at $\boldsymbol{x}$, h is subdifferentially regular at $g(\boldsymbol{x})$ and $\xi \geq 0$ for all $\xi \in \partial h\big(g(\boldsymbol{x})\big)$.*

Theorem 13.4 ([2]) *Let $f : \mathbb{R}^n \to \mathbb{R}$ be such that $f = h \circ g$, where $g : \mathbb{R}^n \to \mathbb{R}$ is f°-pseudoconvex (f°-quasiconvex) and $h : \mathbb{R} \to \mathbb{R}$ is f°-pseudoconvex (f°-quasiconvex) and strictly increasing (increasing), then f is also f°-pseudoconvex (f°-quasiconvex).*

Lemma 13.1 ([2]) *A LLC function $h : \mathbb{R} \to \mathbb{R}$ is f°-pseudoconvex (f°-quasiconvex) and strictly increasing (increasing), if and only if $\xi > 0$ ($\xi \geq 0$) for all $\xi \in \partial h(z)$ and $z \in \mathbb{R}$.*

A LLC function $h : \mathbb{R} \to \mathbb{R}$ is said to be *sign preserving* if sign $(h(z)) = \text{sign}(z)$, in other words

$$h(z) \begin{cases} < 0, & \text{when } z < 0, \\ = 0, & \text{when } z = 0, \\ > 0, & \text{when } z > 0. \end{cases} \tag{13.5}$$

The following elementary result is easy to prove.

Lemma 13.2 *If $A, B \subset \mathbb{R}^n$, then*

$$\text{conv } A \cup \text{conv } B \subseteq \text{conv}(A \cup B).$$

Next we discuss about the solution of the multiobjective problem (13.1). A vector $\boldsymbol{x}^*$ is said to be a *global Pareto optimum* of (13.1), if there does not exist $\boldsymbol{x} \in S$ such, that

$$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*) \text{ for all } i \in \mathcal{I} \quad \text{and} \quad f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}^*) \text{ for some } j.$$

Vector $\boldsymbol{x}^*$ is said to be a *global weak Pareto optimum* of (13.1), if there does not exist $\boldsymbol{x} \in S$ such, that

$$f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*) \quad \text{for all} \quad i \in \mathcal{I}.$$

Vector $\boldsymbol{x}^*$ is a *local (weak) Pareto optimum* of (13.1), if there exists $\delta > 0$ such, that $\boldsymbol{x}^*$ is a global (weak) Pareto optimum on $B(\boldsymbol{x}^*; \delta) \cap S$. Trivially every Pareto optimal point is weakly Pareto optimal.

A set $C \subset \mathbb{R}^n$ is a *cone* if $\lambda \boldsymbol{x} \in C$ for all $\lambda \geq 0$ and $\boldsymbol{x} \in C$. We also denote

$$\text{ray } S = \{\lambda \boldsymbol{x} : \lambda \geq 0,\ \boldsymbol{x} \in S\} \quad \text{and} \quad \text{cone } S = \text{ray conv } S.$$

In other words, ray S is the smallest cone containing S and the *conic hull* cone S is the smallest convex cone containing S. The *contingent cone* of a set $S \in \mathbb{R}^n$ at point $\boldsymbol{x}$ was defined in Definition 1.12 by

$$K_S(\boldsymbol{x}) = \left\{\boldsymbol{d} \in \mathbb{R}^n : \text{there exist } t_i \downarrow 0 \text{ and } \boldsymbol{d}_i \to \boldsymbol{d} \text{ with } \boldsymbol{x} + t_i \boldsymbol{d}_i \in S\right\}$$

and the *polar cone* in Definition 1.13 by

$$S^{\leq} = \{\boldsymbol{d} \in \mathbb{R}^n : \boldsymbol{s}^T \boldsymbol{d} \leq 0, \text{ for all } \boldsymbol{s} \in S\}.$$

Furthermore, let

$$F(\boldsymbol{x}) = \bigcup_{i \in \mathcal{I}} \partial f_i(\boldsymbol{x})$$

and

$$G(\boldsymbol{x}) = \bigcup_{l \in L(\boldsymbol{x})} \partial g_l(\boldsymbol{x}), \text{ where } L(\boldsymbol{x}) = \{l : g_l(\boldsymbol{x}) = 0\}.$$

For the optimality condition we pose the following *constraint qualification*

$$G^{\leq}(\boldsymbol{x}) \subseteq K_S(\boldsymbol{x}). \tag{13.6}$$

Now we can present the following generalized KKT optimality conditions.

Theorem 13.5 ([17]) *If $\boldsymbol{x}^*$ is a local weak Pareto optimum of (13.1) and the constraint qualification (13.6) is valid, then*

$$\mathbf{0} \in \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{cl} \operatorname{cone} G(\boldsymbol{x}^*). \tag{13.7}$$

Moreover, if f_i are f°-pseudoconvex for all $i \in \mathcal{I}$ and g_l are f°-quasiconvex for all $l \in \mathcal{L}$, then the condition (13.7) is sufficient for $\boldsymbol{x}^*$ to be a global weak Pareto optimum of (13.1).

A feasible point $\boldsymbol{x}^* \in S$ is called *weakly Pareto stationary* for problem (13.1), if it satisfies the necessary optimality condition (13.7).

13.3 Scaled Improvement Function

In what follows we consider nonsmooth multiobjective optimization methods not employing any scalarizing function [22] explicitly. Some kind of inherent scalarization is, however, needed in deriving the minimization method for all the objective functions. Theoretically, we utilize the *scaled improvement function* $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$H(\boldsymbol{x}, \boldsymbol{y}) := \max \left\{ \mu_i\big(f_i(\boldsymbol{x})\big) - \mu_i\big(f_i(\boldsymbol{y})\big),\ \delta_l\big(g_l(\boldsymbol{x})\big) : i \in \mathcal{I},\ l \in \mathcal{L} \right\},$$

where μ_i and δ_l are sign preserving (see (13.5)) scaling functions for all $i \in \mathcal{I}$ and $l \in \mathcal{L}$. Moreover, μ_i is supposed to be strictly increasing for all $i \in \mathcal{I}$.

Notice that $H(\boldsymbol{x}, \boldsymbol{y})$ is a generalized version of the standard improvement function used, for example in [10, 15, 18, 21, 35], where we have

$$\mu_i(z) = \delta_l(z) = z \quad \text{for all } i \in \mathcal{I}, l \in \mathcal{L} \text{ and } z \in \mathbb{R}.$$

The standard improvement function has proved to be very sensitive to scaling in several numerical tests (see e.g. [18, 24]). For this reason our aim is to stabilize its numerical behavior by using the scaling functions μ_i and δ_l.

Next we show that the scaled improvement function preserves the following important optimality condition.

Theorem 13.6 *Suppose, that μ_i and δ_l are sign preserving and μ_i is strictly increasing for all $i \in \mathcal{I}$ and $l \in \mathcal{L}$. A necessary condition for $\boldsymbol{x}^* \in \mathbb{R}^n$ to be a global weak Pareto optimum of (13.1) is that*

$$\boldsymbol{x}^* = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^n} H(\boldsymbol{x}, \boldsymbol{x}^*). \tag{13.8}$$

Moreover, if f_i and μ_i are f°-pseudoconvex for all $i \in \mathcal{I}$, g_l and δ_l are f°-quasiconvex and δ_l is increasing for all $l \in \mathcal{L}$ and the constraint qualification (13.6) is valid, then the condition (13.8) is sufficient for $\boldsymbol{x}^*$ to be a global weak Pareto optimum of (13.1).

Proof Suppose first, that $\boldsymbol{x}^* \in \mathbb{R}^n$ is a global weak Pareto optimum of (13.1). Since $\boldsymbol{x}^* \in S$ we have $g_l(\boldsymbol{x}^*) \leq 0$ and since the functions δ_l are sign preserving we have $\delta_l\big(g_l(\boldsymbol{x}^*)\big) \leq 0$ for all $l \in \mathcal{L}$. Moreover, $\mu_i\big(f_i(\boldsymbol{x}^*)\big) - \mu_i\big(f_i(\boldsymbol{x}^*)\big) = 0$, thus we have $H(\boldsymbol{x}^*, \boldsymbol{x}^*) = 0$. If $\boldsymbol{x}^*$ would not be a global minimizer of $H(\cdot, \boldsymbol{x}^*)$, there would exist $\boldsymbol{y}^* \in \mathbb{R}^n$ such that

$$H(\boldsymbol{y}^*, \boldsymbol{x}^*) < H(\boldsymbol{x}^*, \boldsymbol{x}^*) = 0.$$

Then we have $\delta_l\big(g_l(\boldsymbol{y}^*)\big) < 0$ implying $g_l(\boldsymbol{y}^*) < 0$ for all $l \in \mathcal{L}$, in other words, $\boldsymbol{y}^* \in S$. Furthermore, we have $\mu_i\big(f_i(\boldsymbol{y}^*)\big) < \mu_i\big(f_i(\boldsymbol{x}^*)\big)$. Since the functions μ_i are strictly increasing this means that $f_i(\boldsymbol{y}^*) < f_i(\boldsymbol{x}^*)$ for all $i \in \mathcal{I}$, which contradicts the global weak Pareto optimality of $\boldsymbol{x}^*$.

Suppose next that (13.8) holds true. Suppose also that (13.6) is valid, f_i and μ_i are f°-pseudoconvex for all $i \in \mathcal{I}$, and g_l and δ_l are f°-quasiconvex and δ_l is increasing for all $l \in \mathcal{L}$.

Due to Theorems 13.3 and 13.2 function $H(\cdot, \boldsymbol{x}^*)$ is LLC. Since $\boldsymbol{x}^*$ is a global minimizer of $H(\cdot, \boldsymbol{x}^*)$ we have $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$ by (13.2). Then it follows from Theorems 13.3 and 13.2 that

$$\mathbf{0} \in \operatorname{conv} \left\{ \bigcup_{i \in \mathcal{I}} \operatorname{conv} \partial \mu_i\big(f_i(\boldsymbol{x}^*)\big) \partial f_i(\boldsymbol{x}^*) \cup \bigcup_{l \in L(\boldsymbol{x}^*)} \operatorname{conv} \partial \delta_l\big(g_l(\boldsymbol{x}^*)\big) \partial g_l(\boldsymbol{x}^*) \right\}.$$

This means that by Lemma 13.1 there exist $\zeta_{\mu_i} \in \partial \mu_i\big(f_i(\boldsymbol{x}^*)\big)$ and $\zeta_{\delta_l} \in \partial \delta_l\big(g_l(\boldsymbol{x}^*)\big)$ such that for all $i \in \mathcal{I}$ and $l \in L(\boldsymbol{x}^*)$ we have $\zeta_{\mu_i} > 0$, $\zeta_{\delta_l} \geq 0$ and

$$\mathbf{0} \in \operatorname{conv} \left\{ \bigcup_{i \in \mathcal{I}} \operatorname{conv} \zeta_{\mu_i} \partial f_i(\boldsymbol{x}^*) \cup \bigcup_{l \in L(\boldsymbol{x}^*)} \operatorname{conv} \zeta_{\delta_l} \partial g_l(\boldsymbol{x}^*) \right\}.$$

Then by using Lemma 13.2, Lemma 2.10 in [16] and choosing $\zeta_\mu := \max_{i\in\mathcal{I}} \zeta_{\mu_i} > 0$ and $\zeta_\delta := \max_{l\in L(\boldsymbol{x}^*)} \zeta_{\delta_l} \geq 0$ we get

$$\begin{aligned}
\mathbf{0} &\in \operatorname{conv}\left\{\operatorname{conv}\bigcup_{i\in\mathcal{I}} \zeta_{\mu_i}\partial f_i(\boldsymbol{x}^*) \cup \operatorname{conv}\bigcup_{l\in L(\boldsymbol{x}^*)} \zeta_{\delta_l}\partial g_l(\boldsymbol{x}^*)\right\} \\
&\subseteq \operatorname{conv}\left\{\zeta_\mu \operatorname{conv}\bigcup_{i\in\mathcal{I}} \partial f_i(\boldsymbol{x}^*) \cup \zeta_\delta \operatorname{conv}\bigcup_{l\in L(\boldsymbol{x}^*)} \partial g_j(\boldsymbol{x}^*)\right\} \\
&= \operatorname{conv}\left\{\zeta_\mu \operatorname{conv} F(\boldsymbol{x}^*) \cup \zeta_\delta \operatorname{conv} G(\boldsymbol{x}^*)\right\} \\
&= \left\{\lambda\zeta_\mu \operatorname{conv} F(\boldsymbol{x}^*) + (1-\lambda)\zeta_\delta \operatorname{conv} G(\boldsymbol{x}^*) : \lambda \in [0,1]\right\}.
\end{aligned}$$

This means that there exists $\lambda^* \in [0, 1]$ such that the condition (13.7) is valid. Indeed, if $\lambda^* \in (0, 1]$ we have

$$\begin{aligned}
\mathbf{0} &\in \operatorname{conv} F(\boldsymbol{x}^*) + \frac{(1-\lambda^*)\zeta_\delta}{\lambda^*\zeta_\mu} \operatorname{conv} G(\boldsymbol{x}^*) \\
&\subseteq \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{ray}\operatorname{conv} G(\boldsymbol{x}^*) \\
&= \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{cone} G(\boldsymbol{x}^*) \\
&\subseteq \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{cl}\operatorname{cone} G(\boldsymbol{x}^*).
\end{aligned}$$

On the other hand, if $\lambda^* = 0$, we observe

$$\begin{aligned}
\mathbf{0} &\in \zeta_\delta \operatorname{conv} G(\boldsymbol{x}^*) \subseteq \operatorname{ray}\operatorname{conv} G(\boldsymbol{x}^*) = \operatorname{cone} G(\boldsymbol{x}^*) \\
&\subseteq \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{cl}\operatorname{cone} G(\boldsymbol{x}^*).
\end{aligned}$$

Thus, Theorem 13.5 implies that $\boldsymbol{x}^*$ is a global weak Pareto optimum of (13.1). □

Notice, that for any $\lambda > 0$ and $p \in \mathbb{N}_+$ the functions

$$\mu(z) = \lambda z \tag{13.9}$$

$$\mu(z) = \lambda \operatorname{sign}(z)\left((|z|+1)^p - 1\right) \tag{13.10}$$

$$\mu(z) = \lambda \operatorname{sign}(z)\ln(|z|+1) \tag{13.11}$$

$$\mu(z) = \lambda \operatorname{sign}(z)(e^{|z|} - 1) \tag{13.12}$$

are sign preserving, f°-pseudoconvex and strictly increasing. Thus, according to Theorem 13.6 besides *linear scaling* (13.9), we can use *polynomial* (13.10), *logarithmic* (13.11) or *exponential scaling* (13.12). These functions are illustrated in

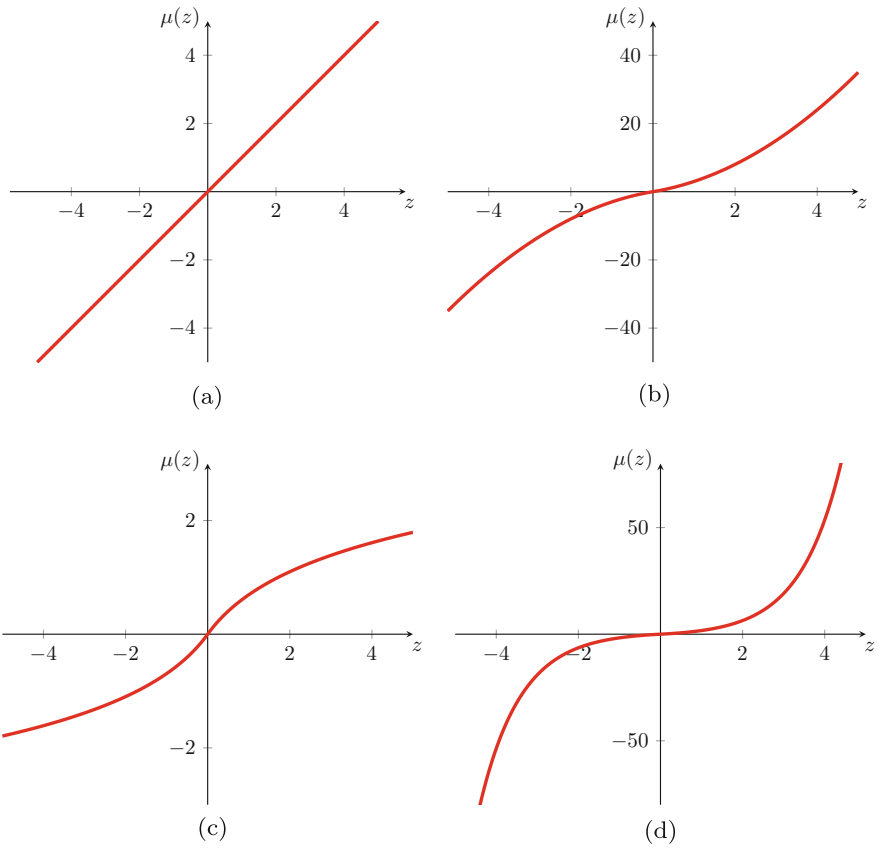

Fig. 13.1 Illustration of the scaling functions (13.9)–(13.12). (**a**) Linear scaling, $\lambda = 1$. (**b**) Polynomial scaling, $\lambda = 1$, $p = 2$. (**c**) Logarithmic scaling, $\lambda = 1$. (**d**) Exponential scaling, $\lambda = 1$

Fig. 13.1. Furthermore, for constraint functions, together with the functions (13.9)–(13.12), we can also use the polynomial scaling function

$$\delta(z) = \lambda \operatorname{sign}(z)|z|^p \tag{13.13}$$

being f°-quasiconvex but not f°-pseudoconvex ($0 \in \partial\delta(0)$ although 0 is not a global minimum, cf. Theorem 13.1). Notice also, that all the above scaling functions are continuously differentiable, and thus subdifferentially regular.

13.4 Multiobjective Proximal Bundle Method

This section is devoted to derive the version of the multiobjective proximal bundle method (MPB) using the scaled improvement function. This version modifies MPB [18, 21] where the idea of the standard improvement function [10, 21, 35] is utilized. MPB is a multiobjective counterpart of the single-objective proximal bundle method for nonconvex constrained problems presented in [14] which in its turn generalizes the original convex proximal bundle method [11].

The method presented here is suitable for multiobjective nonconvex constraint problems. As the result of the utilization of either the standard or the scaled improvement function, the method is a descent one, in other words, it improves all the objectives simultaneously at every iteration.

In what follows, we suppose that all the scaling functions μ_i and δ_l are sign preserving and subdifferentially regular for all $i \in \mathcal{I}$ and $l \in \mathcal{L}$. Furthermore μ_i is supposed to be f°-pseudoconvex and strictly increasing for all $i \in \mathcal{I}$ and δ_l is f°-quasiconvex and increasing for all $l \in \mathcal{L}$.

Direction Finding At each iteration h, we denote the current solution by $\boldsymbol{x}^h$ and auxiliary points from the previous iterations by $\boldsymbol{y}^j \in \mathbb{R}^n$ for $j \in \mathcal{J}^h = \{1, \ldots, h\}$. Moreover, we assume that we can evaluate subgradients $\boldsymbol{\xi}_{f_i}^j \in \partial f_i(\boldsymbol{y}^j)$ and $\zeta_{\mu_i}^j \in \partial \mu_i\big(f_i(\boldsymbol{y}^j)\big)$ for $j \in \mathcal{J}^h$, $i \in \mathcal{I}$, and $\boldsymbol{\xi}_{g_l}^j \in \partial g_l(\boldsymbol{y}^j)$ and $\zeta_{\delta_l}^j \in \partial \delta_l\big(g_l(\boldsymbol{y}^j)\big)$ for $j \in \mathcal{J}^h, l \in \mathcal{L}$.

As Theorem 13.6 suggests, the search direction of MPB is found by solving the following nonsmooth problem:

$$\begin{cases} \text{minimize} & H(\boldsymbol{x}^h + \boldsymbol{d}, \boldsymbol{x}^h) \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{13.14}$$

First, we focus on the convex case by assuming for a while that the objectives f_i and the constraints g_l for all $i \in \mathcal{I}$ and $l \in \mathcal{L}$ in the problem (13.1) are convex. Since the objective and constraint functions are convex, they are subdifferentially regular. Then, due to the subdifferential regularity and f°-quasiconvexity of the scaling functions, Lemma 13.1 and Theorem 13.3 (ii) guarantee that

$$\zeta_{\mu_i}^j \boldsymbol{\xi}_{f_i}^j \in \partial(\mu_i \circ f_i)(\boldsymbol{y}^j) \quad \text{for all} \quad i \in \mathcal{I},\ j \in \mathcal{J}^h, \quad \text{and} \tag{13.15}$$

$$\zeta_{\delta_l}^j \boldsymbol{\xi}_{g_l}^j \in \partial(\delta_l \circ g_l)(\boldsymbol{y}^j) \quad \text{for all} \quad l \in \mathcal{L},\ j \in \mathcal{J}^h. \tag{13.16}$$

Now we can linearize the scaled objective and the scaled constraint functions at the point $\boldsymbol{y}^j$ by

$$\overline{\mu f}_{i,j}(\boldsymbol{x}) := \mu_i\big(f_i(\boldsymbol{y}^j)\big) + \zeta^j_{\mu_i}(\boldsymbol{\xi}^j_{f_i})^T(\boldsymbol{x}-\boldsymbol{y}^j) \quad \text{for all} \quad i \in \mathcal{I},\ j \in \mathcal{J}^h, \quad \text{and}$$
$$\overline{\delta g}_{l,j}(\boldsymbol{x}) := \delta_l\big(g_l(\boldsymbol{y}^j)\big) + \zeta^j_{\delta_l}(\boldsymbol{\xi}^j_{g_l})^T(\boldsymbol{x}-\boldsymbol{y}^j) \quad \text{for all} \quad l \in \mathcal{L},\ j \in \mathcal{J}^h.$$

Then we define a convex piecewise linear approximation to the scaled improvement function by

$$\hat{H}^h(\boldsymbol{x}) := \max\left\{\overline{\mu f}_{i,j}(\boldsymbol{x}) - \mu_i\big(f_i(\boldsymbol{x}^h)\big),\ \overline{\delta g}_{l,j}(\boldsymbol{x}) : i \in \mathcal{I},\ l \in \mathcal{L},\ j \in \mathcal{J}^h\right\}$$

and get an approximation to the direction finding problem (13.14) by

$$\begin{cases} \text{minimize} & \hat{H}^h(\boldsymbol{x}^h+\boldsymbol{d}) + \frac{1}{2}u^h\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{13.17}$$

As usual, the stabilizing term $\frac{1}{2}u^h\|\boldsymbol{d}\|^2$ with the positive proximity parameter u^h is added to guarantee the existence and uniqueness of a solution to (13.17) and to keep the approximation local enough. Notice that (13.17) is still a nonsmooth problem, but due to its minmax-nature it is equivalent to the following (smooth) quadratic problem

$$\begin{cases} \text{minimize} & v + \frac{1}{2}u^h\|\boldsymbol{d}\|^2 \\ \text{subject to} & -\alpha^h_{\mu f_i,j} + \zeta^j_{\mu_i}(\boldsymbol{\xi}^j_{f_i})^T\boldsymbol{d} \le v,\ i \in \mathcal{I},\ j \in \mathcal{J}^h, \\ & -\alpha^h_{\delta g_l,j} + \zeta^j_{\delta_l}(\boldsymbol{\xi}^j_{g_l})^T\boldsymbol{d} \le v,\ l \in \mathcal{L},\ j \in \mathcal{J}^h, \end{cases} \tag{13.18}$$

where

$$\alpha^h_{\mu f_i,j} := \mu_i\big(f_i(\boldsymbol{x}^h)\big) - \overline{\mu f}_{i,j}(\boldsymbol{x}^h), \quad i \in \mathcal{I},\ j \in \mathcal{J}^h, \quad \text{and}$$
$$\alpha^h_{\delta g_l,j} := -\overline{\delta g}_{l,j}(\boldsymbol{x}^h), \quad l \in \mathcal{L},\ j \in \mathcal{J}^h,$$

are standard positive *linearization errors*.

In the nonconvex case the linearization errors are not necessarily positive anymore, and thus we replace them by *subgradient locality measures*

$$\beta^h_{\mu f_i,j} := \max\left\{|\alpha^h_{\mu f_i,j}|\,, \gamma_{f_i}\|\boldsymbol{x}^h-\boldsymbol{y}^j\|^2\right\},$$
$$\beta^h_{\delta g_l,j} := \max\left\{|\alpha^h_{\delta g_l,j}|\,, \gamma_{g_l}\|\boldsymbol{x}^h-\boldsymbol{y}^j\|^2\right\},$$

where $\gamma_{f_i} \geq 0$ for $i \in \mathcal{I}$ and $\gamma_{g_l} \geq 0$ for $l \in \mathcal{L}$, ($\gamma_{f_i} = 0$ if f_i is convex and $\gamma_{g_l} = 0$ if g_l is convex).

In the nonconvex case the formulas (13.15) and (13.16) are not necessarily valid anymore. Then, we have to assume the subdifferential regularity of the objective and constraint functions. Another possibility is to use smooth (continuously differentiable) scaling functions like (13.9)–(13.13), because then Theorem 13.3 (i) guarantees the equality in (13.4) and thus (13.15) and (13.16) are valid again.

Line Search Once the solution $(\boldsymbol{d}^h, v^h)$ of the problem (13.18) is obtained, we perform the two-point line search strategy [14] detecting the discontinuities in the gradients of the scaled objectives. Let $m_L \in (0, \frac{1}{2})$, $m_R \in (m_L, 1)$, and $\bar{t} \in (0, 1]$ be fixed line search parameters. We begin by determining the largest number $t_L^h \in [0, 1]$ such that

$$\max\left\{\mu_i\big(f_i(\boldsymbol{x}^h + t_L^h \boldsymbol{d}^h)\big) - \mu_i\big(f_i(\boldsymbol{x}^h)\big) : i \in \mathcal{I}\right\} \leq m_L t_L^h v^h, \quad \text{and}$$

$$\max\left\{\delta_l\big(g_l(\boldsymbol{x}^h + t_L^h \boldsymbol{d}^h)\big) : l \in \mathcal{L}\right\} \leq 0.$$

If $t_L^h \geq \bar{t}$, we take a *long serious step*:

$$\boldsymbol{x}^{h+1} = \boldsymbol{x}^h + t_L^h \boldsymbol{d}^h \quad \text{and} \quad \boldsymbol{y}^{h+1} = \boldsymbol{x}^{h+1},$$

if $0 < t_L^h < \bar{t}$, then we take a *short serious step*:

$$\boldsymbol{x}^{h+1} = \boldsymbol{x}^h + t_L^h \boldsymbol{d}^h \quad \text{and} \quad \boldsymbol{y}^{h+1} = \boldsymbol{x}^h + t_R^h \boldsymbol{d}^h$$

and if $t_L^h = 0$, we take a *null step*:

$$\boldsymbol{x}^{h+1} = \boldsymbol{x}^h \quad \text{and} \quad \boldsymbol{y}^{h+1} = \boldsymbol{x}^h + t_R^h \boldsymbol{d}^h,$$

where $t_R^h > t_L^h$ is such that

$$-\beta_{\mu f_i, h+1}^{h+1} + \zeta_{\mu_i}^{h+1} (\boldsymbol{\xi}_{f_i}^{h+1})^T \boldsymbol{d}^h \geq m_R v^h.$$

The iteration is terminated when

$$-\tfrac{1}{2} v^h < \varepsilon_s,$$

where $\varepsilon_s > 0$ is an accuracy parameter supplied by the user.

Algorithm To conclude this section, we summarize the previous subsections by presenting the algorithm of MPB using the scaled improvement function.

Algorithm 13.1: MPB

Data: A final accuracy tolerance $\varepsilon_s > 0$, an initial weight $u^1 > 0$, line search parameters $m_L \in (0, \frac{1}{2})$, $m_R \in (m_L, 1)$ and $\bar{t} \in (0, 1]$, and distance measure parameters $\gamma_{f_i} \geq 0$ for $i \in \mathcal{I}$ and $\gamma_{g_l} \geq 0$ for $l \in \mathcal{L}$.

Step 1. *(Initialization)* Select a feasible starting point $\boldsymbol{x}^1 \in S$. Set $h := 1$, $\boldsymbol{y}^1 := \boldsymbol{x}^1$ and calculate $\boldsymbol{\xi}^1_{f_i} \in \partial f_i(\boldsymbol{y}^1)$ and $\zeta^1_{\mu_i} \in \partial \mu_i\big(f_i(\boldsymbol{y}^1)\big)$ for $i \in \mathcal{I}$, and $\boldsymbol{\xi}^1_{g_l} \in \partial g_l(\boldsymbol{y}^1)$ and $\zeta^1_{\delta_l} \in \partial \delta_l\big(g_l(\boldsymbol{y}^1)\big)$ for $l \in \mathcal{L}$.

Step 2. *(Direction finding)* Solve the problem (13.18) in order to get the solution $(\boldsymbol{d}^h, v^h)$.

Step 3. *(Stopping criterion)* If $-\frac{1}{2}v^h < \varepsilon_s$, then **stop**.

Step 4. *(Line search)* Find the step sizes $t^h_L \in [0, 1]$ and $t^h_R \in [t^h_L, 1]$ to take either a *null step* ($t^h_L = 0$) or a *serious step* ($t^h_L > 0$). Set

$$\boldsymbol{x}^{h+1} = \boldsymbol{x}^h + t^h_L \boldsymbol{d}^h \quad \text{and} \quad \boldsymbol{y}^{h+1} = \boldsymbol{x}^h + t^h_R \boldsymbol{d}^h.$$

Step 5. *(Updating)* Set $h := h + 1$, calculate $\boldsymbol{\xi}^h_{f_i} \in \partial f_i(\boldsymbol{y}^h)$ and $\zeta^h_{\mu_i} \in \partial \mu_i\big(f_i(\boldsymbol{y}^h)\big)$ for $i \in \mathcal{I}$, and $\boldsymbol{\xi}^h_{g_l} \in \partial g_l(\boldsymbol{y}^h)$ and $\zeta^h_{\delta_l} \in \partial \delta_l\big(g_l(\boldsymbol{y}^h)\big)$ for $l \in \mathcal{L}$. Choose $\mathcal{J}^h \subseteq \{1, \ldots, h\}$ and update the weight u^h. Go to Step 2.

Some remarks about the algorithm are in order. In practice, the size of the index set $\mathcal{J}^h$ needs to be bounded, and that is why the subgradient aggregation strategy [9] is applied. For a suitable distance measure parameter selection, set $\gamma_{f_i} = 0$ if f_i is convex and similarly $\gamma_{g_l} = 0$ if g_l is convex. In Step 4, the step sizes $t^h_L \in [0, 1]$ and $t^h_R \in [t^h_L, 1]$ are generated by the line search strategy described in the previous subsection. Finally in Step 5, the weight u^h is updated by the procedure given in [11].

13.5 Convergence Analysis

The following convergence analysis is divided into two results. The first result claims that under some generalized convexity assumptions a global Pareto optimum can be guaranteed while the second result focuses on more general case when a weak Pareto stationary solution is obtained.

Theorem 13.7 *Let the constraint qualification* (13.6) *be valid and for all* $i \in \mathcal{I}$ *and* $l \in \mathcal{L}$ *let* f_i, μ_i, g_l *and* δ_l *be* f°*-pseudoconvex functions such that* $H(\cdot, \boldsymbol{x}^h)$ *is weakly semismooth. Moreover, let* μ_i *and* δ_l *be sign preserving, subdifferentially regular and strictly increasing functions. If the MPB algorithm stops with a finite*

number of iterations, then the solution is a global weak Pareto optimum of (13.1) *if in addition*

(i) μ_i *and* δ_l *are continuously differentiable, or*
(ii) f_i *and* g_l *are subdifferentially regular.*

On the other hand, any accumulation point of an infinite sequence of solutions generated by the MPB algorithm is a global weak Pareto optimum of (13.1).

Proof Due to Theorems 13.4 and 13.2, the improvement function $H(\cdot, \boldsymbol{x}^h)$ is f°-pseudoconvex. Moreover, in both cases (i) and (ii) Theorem 13.3 guarantees that

$$\zeta_{\mu_i}^j \boldsymbol{\xi}_{f_i}^j \in \partial(\mu_i \circ f_i)(\boldsymbol{y}^j) \quad \text{for all} \quad i \in \mathcal{I},\ j \in \mathcal{J}^h, \quad \text{and}$$
$$\zeta_{\delta_l}^j \boldsymbol{\xi}_{g_l}^j \in \partial(\delta_l \circ g_l)(\boldsymbol{y}^j) \quad \text{for all} \quad l \in \mathcal{L},\ j \in \mathcal{J}^h,$$

meaning that the linearizations $\overline{\mu f}_{i,j}(\boldsymbol{x})$ and $\overline{\delta g}_{l,j}(\boldsymbol{x})$ are well-defined.

The formulation of the MPB algorithm implies, that it is equivalent to the proximal bundle algorithm applied to unconstraint single objective optimization of H. According to the convergence analysis of the standard proximal bundle algorithm (see, e.g., [11, 32]) if it stops with a finite number of iterations, then the solution $\boldsymbol{x}^h$ is a stationary point of a weakly semismooth H, in other words $\mathbf{0} \in \partial H(\boldsymbol{x}^h, \boldsymbol{x}^h)$. Then by Theorem 13.1 function H attains its global minimum at $\boldsymbol{x}^h$. Since every f°-pseudoconvex function is also f°-quasiconvex, due to Theorem 13.6 $\boldsymbol{x}^h$ is a global weak Pareto optimum of (13.1). The proof of the case, when MPB generates an infinite sequence of solutions, goes similarly. □

In order to utilize Theorem 13.6 in the proof of Theorem 13.7, the scaled improvement function $H(\cdot, \boldsymbol{x}^h)$ must be f°-pseudoconvex. That is why the constraint functions g_l and the scaling functions δ_l are assumed to be f°-pseudoconvex while in Theorem 13.6 only the f°-quasiconvexity was required. Similarly, δ_l is supposed to be strictly increasing function instead of increasing.

Finally we state, that in a more general case weakly Pareto stationary points of the problem (13.1) are obtained.

Theorem 13.8 *Let the constraint qualification* (13.6) *be valid and for all* $i \in \mathrm{i}$ *and* $l \in \mathcal{L}$ *let* f_i *and* g_l *be weakly semismooth, and let* μ_i *and* δ_l *be sign preserving and subdifferentially regular functions. Moreover, let* μ_i *be* f°*-pseudoconvex and strictly increasing, and let* δ_l *be* f°*-quasiconvex and increasing functions. If the MPB algorithm stops with a finite number of iterations, then the solution is a weakly Pareto stationary point (i.e. it satisfies the necessary optimality condition* (13.7)*). On the other hand, any accumulation point of an infinite sequence of solutions generated by the MPB algorithm is weakly Pareto stationary.*

Proof The proof is analogous to that of Theorem 13.7. □

13.6 Numerical Examples

In general, MPB has turned out to be efficient and reliable solver both in the single-objective [2] and in the multiobjective [18] case. Here we focus on exemplifying the impact of the scaling with the following test problem presented in [18]

$$\begin{cases} \text{minimize} & f_1(\boldsymbol{x}) = \sqrt{\|\boldsymbol{x}\| + 2} \\ & f_2(\boldsymbol{x}) = \max\left\{-x_1 - x_2,\ -x_1 - x_2 + x_1^2 + x_2^2 - 1\right\} \\ \text{subject to} & g(\boldsymbol{x}) = \max\left\{x_1^2 + x_2^2 - 10, 3x_1 + x_2 + 1.5\right\} \leq 0, \end{cases}$$

where f_1 is f°-pseudoconvex and f_2 and g are convex.

In this example, the starting point is $\boldsymbol{x}^1 = (-0.5, -0.5)$ and the parameters selected are the accuracy tolerance $\varepsilon_s = 10^{-5}$, the line search parameters $m_L = 0.01$, $m_R = 0.5$, and $\bar{t} = 0.01$, the distance measure parameters $\gamma_{f_1} = 0.5$ and, $\gamma_{f_2} = \gamma_g = 0$. The weight u_1 is initialized by

$$u^1 = \frac{1}{k}\sum_{i=1}^{k} \|\xi^1_{f_i}\|.$$

Lastly, in order to perform the tests, a slightly modified version of the Fortran 77 implementation of MPB described in [15] utilizing the quadratic solver presented in [13] is used under Linux Ubuntu system with `gfortran` as a compiler.

At first, we illustrate the use of different weights for objectives and constraints in the scaled improvement function. For that purpose, we use the linear scaling function (13.9) for f_1, f_2 and g and different values for λ. In Table 13.1, the first three columns tell the values of λ used for f_1, f_2 and g, respectively. The next two columns represent the solution obtained and the last two columns describe the computational effort by specifying the number of iterations and function evaluations. It can be seen from the results that the weighting of the functions clearly has an influence both on the solution obtained and on the computational efficiency of the method. Especially the constraint function seems to be very sensitive for scaling.

Table 13.1 Results with linear scaling varying the weights

λ_{f_1}	λ_{f_2}	λ_g	$f_1(\boldsymbol{x}^*)$	$f_2(\boldsymbol{x}^*)$	Iterations	Func. calls
1	1	1	1.57349	0.57595	5	6
0.1	1	1	1.57301	0.60030	5	8
10	1	1	1.58256	0.49134	12	13
10	1	0.1	1.59020	0.45234	59	61
0.1	0.1	10	1.57405	0.56477	3	4

Table 13.2 Results with different scaling functions

μ_1	μ_2	δ_1	$f_1(\boldsymbol{x}^*)$	$f_2(\boldsymbol{x}^*)$	Iterations	Func. calls
–	–	–	1.57349	0.57595	5	6
(13.9)	(13.9)	–	1.57314	0.58763	4	5
(13.10)	(13.9)	(13.11)	1.57901	0.52853	12	13
(13.11)	(13.10)	–	1.57307	0.60900	15	16
(13.12)	(13.9)	(13.9)	1.57539	0.54644	10	11

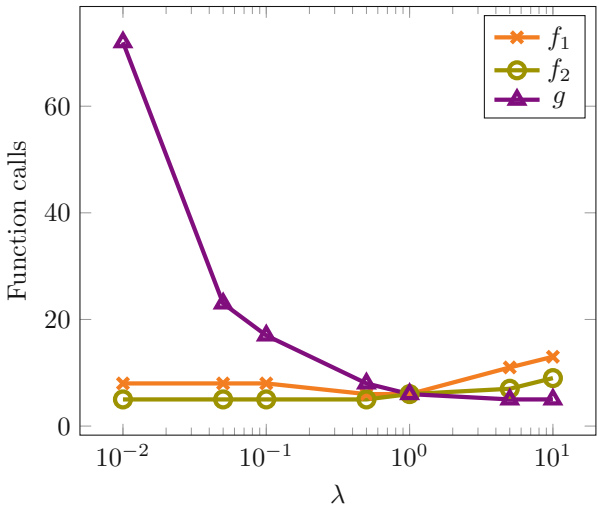

Fig. 13.2 Illustration of different weights λ in the linear scaling

In Table 13.2, five examples with different scaling functions are given and the first line describes the solution without any scaling. The first three columns of this table indicate the scaling functions (13.9)–(13.12) with $\lambda = 0.5$ and $p = 2$ used for each objective and constraint function respectively. Again, the rest of the columns are devoted to the solution obtained and the computational effort. The results indicate that also the change of the scaling function influences the final solution and the number of function evaluations needed, but the influence is not so remarkable than when changing the weight λ (see Table 13.1).

In Figs. 13.2 and 13.3 it is illustrated what happens with respect to function calls when only one function is scaled. In both figures, the line with a cross refers to f_1, the line with a circle to f_2 and the line with a triangle to g. In Fig. 13.2, only the linear scaling is used and the weights λ are from the interval 0.01–10 whereas in Fig. 13.3, different scaling functions are used for each function. Like in Tables 13.1 and 13.2 the constraint function seems to be the most sensitive for linear scaling while the change of the scaling function itself does not influence on the results so much.

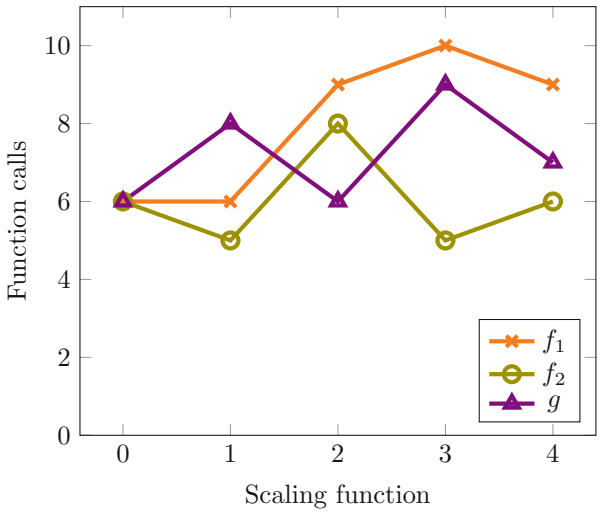

Fig. 13.3 Illustration of different scaling functions: (0)—no scaling, (1)—linear scaling, (2)—polynomial scaling, (3)—logarithmic scaling, (4)—exponential scaling

13.7 Conclusions

We have derived a scaled version of the improvement function capable to linear, polynomial, logarithmic and exponential scaling for both objective and constraint functions. We have also developed a new version of the multiobjective proximal bundle method utilizing this scaled improvement function. The method is globally convergent and descent. Moreover, it can be proved to produce weakly Pareto stationary solutions and under some generalized convexity assumptions the solutions are guaranteed to be globally weakly Pareto optimal. The effects of the scaling need still more numerical testing and the question what are the minimal assumptions of the convergence theorems will be an interesting topic for future work.

Acknowledgements This work was financially supported by the University of Turku. The authors want to thank Prof. Dominikus Noll for the idea given during the HCM Workshop "Nonsmooth Optimization and its Applications" in Bonn, May 2017.

References

1. Auslender, A.: Numerical methods for nondifferentiable convex optimization. Math. Program. Study **30**, 102–126 (1987)
2. Bagirov, A., Karmitsa, N., Mäkelä, M.M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, Cham, Heidelberg (2014)
3. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)

4. Da Cruz Neto, J.X., Da Silva, G.J.P., Ferreira, O.P., Lopes, J.O.: A subgradient method for multiobjective optimization. Comput. Optim. Appl. **54**(3), 461–472 (2013)
5. Handl, J., Kell, D.B., Knowles, J.D: Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**(2), 279–292 (2007)
6. Hiriart-Urruty, J.B.: New concepts in nondifferentiable programming. Bull. de la Soc. Math. de France Mémoires **60**, 57–85 (1979)
7. Jin, Y. (ed.) Multi-Objective Machine Learning. Studies in Computational Intelligence, vol. 16. Springer, Berlin (2006)
8. Karas, E., Ribeiro, A., Sagastizábal, C., Solodov, M.: A bundle-filter method for nonsmooth convex constrained optimization. Math. Program. **116**, 297–320 (2009)
9. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics, vol. 1133. Springer, Berlin (1985)
10. Kiwiel, K.C.: A descent method for nonsmooth convex multiobjective minimization. Large Scale Syst. **8**(2), 119–129 (1985)
11. Kiwiel, K.C: Proximity control in bundle methods for convex nondifferentiable optimization. Math. Program. **46**, 105–122 (1990)
12. Lemaréchal, C., Nemirovskii, A., Nesterov, Yu.: New variants of bundle methods. Math. Program. **69**, 111–148 (1995)
13. Lukšan, L.: Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minimax approximation. Kybernetika **20**(6), 445–457 (1984)
14. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific, Singapore (1992)
15. Mäkelä, M.M.: Multiobjective Proximal Bundle Method for Nonconvex Nonsmooth Optimization: Fortran Subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B, Scientific computing, B 13/2003. University of Jyväskylä, Jyväskylä (2003)
16. Mäkelä, M.M., Eronen, V.-P., Karmitsa, N.: On Nonsmooth Optimality Conditions with Generalized Convexities. TUCS Technical Reports 1056. Turku Centre for Computer Science, Turku (2012)
17. Mäkelä, M.M., Eronen, V.-P., Karmitsa, N.: On nonsmooth multiobjective optimality conditions with generalized convexities. In: Rassias, Th.M., Floudas, C.A., Butenko, S. (eds.) Optimization in Science and Engineering: In Honor of the 60th Birthday of Panos M. Pardalos, pp. 333–357. Springer, New York (2014)
18. Mäkelä, M.M., Karmitsa, N., Wilppu, O.: Proximal bundle method for nonsmooth and nonconvex multiobjective optimization. In: Neittaanmäki, P., Repin, S., Tuovinen, T. (eds.) Mathematical Modeling and Optimization of Complex Structures, pp. 191–204, Springer, Cham (2016)
19. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Struct. Multidiscip. Optim. **26**(6), 369–395 (2004)
20. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
21. Miettinen, K., Mäkelä, M.M.: Interactive bundle-based method for nondifferentiable multiobjective optimization: NIMBUS. Optimization **34**, 231–246 (1995)
22. Miettinen, K., Mäkelä, M.M.: On scalarizing functions in multiobjective optimization. OR Spectr. **24**(2), 193–213 (2002)
23. Mifflin, R.: An algorithm for constrained optimization with semismooth functions. Math. Oper. Res. **2**, 191–207 (1977)
24. Montonen, O., Joki, K.: Bundle-based descent method for nonsmooth multiobjective DC optimization with inequality constraints. J. Glob. Optim. **72**(3), 403–429 (2018)
25. Montonen, O., Karmitsa, N., Mäkelä, M.M.: Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization. Optimization **67**(1), 139–158 (2018)
26. Moreau, J.J., Panagiotopoulos, P.D., Strang, G. (eds.) Topics in Nonsmooth Mechanics. Birkhäuser, Basel (1988)

27. Mukai, H.: Algorithms for multicriterion optimization. IEEE Trans. Autom. Control **25**(2), 177–186 (1980)
28. Outrata, J., Kočvara, M., Zowe, J.: Nonsmooth approach to optimization problems with equilibrium constraints. In: Theory, Applications and Numerical Results. Kluwer Academic Publishers, Dordrecht (1998)
29. Pironneau, O., Polak, E.: Rate of convergence of a class of methods of feasible directions. SIAM J. Numer. Anal. **10**, 161–174 (1973)
30. Qu, S., Liu, C., Goh, M., Li, Y., Ji, Y.: Nonsmooth multiobjective programming with quasi-Newton methods. Eur. J. Oper. Res. **235**(3), 503–510 (2014)
31. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**, 146–169 (2005)
32. Schramm, H., Zowe, J.: A Version of the bundle idea for minimizing a nonsmooth functions: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**, 121–152 (1992)
33. Shin, W.S., Ravindran, A.: Interactive multiple objective optimization: survey I—continuous case. Comput. Oper. Res. **18**(1), 97–114 (1991)
34. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation, and Applications. Wiley, New York (1986)
35. Wang, S.: Algorithms for multiobjective and nonsmooth optimization. In: Kleinschmidt, P., Radermacher, F.J, Schweitzer, W., Wildermann, H. (eds.) Methods of Operations Research, vol. 58, pp. 131–142. Athenäum, Frankfurt am Main (1989)
36. Wu, W., Maier, H.R., Simpson, A.R.: Single-objective versus multiobjective optimization of water distribution systems accounting for greenhouse gas emissions by carbon pricing. J. Water Resour. Plan. Manag. **136**(5), 555–565 (2010)
37. Zoutendijk, G.: Methods of Feasible Directions: A Study in Linear and Nonlinear Programming. Elsevier, Amsterdam (1960)

Chapter 14
Multiobjective Double Bundle Method for DC Optimization

Outi Montonen and Kaisa Joki

Abstract We discuss about the multiobjective double bundle method for nonsmooth multiobjective optimization where objective and constraint functions are presented as a difference of two convex (DC) functions. By utilizing a special technique called the improvement function, we are able to handle several objectives and constraints simultaneously. The method improves every objective at each iteration and the improvement function preserves the DC property of the objectives and constraints. Once the improvement function is formed, we can approximate it by using a special cutting plane model capturing the convex and concave behaviour of a DC function. We solve the problem with a modified version of the single-objective double bundle method using the cutting plane model as an objective. The multiobjective double bundle method is proved to be finitely convergent to a weakly Pareto stationary solution under mild assumptions. Moreover, the applicability of the method is considered.

14.1 Introduction to Multiobjective DC Optimization

The vast range of practical optimization problems involve several goals. Usually, these goals are conflicting and compromises have to be made. Thus, it is sensible to study multiobjective optimization [8, 28]. The real-life application areas for multiobjective optimization are for example chemical engineering [36], cancer treatment planning [6], and humanitarian aid [12]. Compared with single-objective optimization, in multiobjective optimization various objectives need to be handled simultaneously. One fundamental idea to solve a multiobjective problem is the scalarization [8, 28], where several objectives are transformed into one objective, and then some efficient single-objective method can be applied. A typical feature of scalarization is that we have to take a stand on the relative importance of objectives.

O. Montonen (✉) · K. Joki
Department of Mathematics and Statistics, University of Turku, Turku, Finland
e-mail: outi.montonen@utu.fi; kaisa.joki@utu.fi

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_14

As many of the practical applications have nonsmooth nature as well, we focus here on nonsmooth multiobjective optimization. Instead of the scalarization, we want to treat the objectives as they are, and in particular, to investigate descent methods. We classify a method to be of a descent one if at each iteration it improves every objective, and in that sense, every objective is considered to be equally important. Some examples of descent methods can be found in the literature for convex (see e.g. [3, 4, 21, 31, 33]) and for nonconvex (see e.g. [27, 29, 35, 41]) problems.

Here we limit our study to DC functions (i.e. functions which can be represented as a difference of convex functions, so-called DC components) forming a wide subclass of nonconvex functions. In addition to the wideness, the class of DC functions has another unquestionable advantage compared to general nonconvex functions. Due to the convex DC components, we are able to utilize the convex analysis. Unfortunately, it may be hard to define DC components for a function even if the function is known to be a DC function. Moreover, due to the fact that a DC decomposition is not unique, the DC decomposition found may not be the most suitable one. However, there exist many practical applications where the objectives are in the explicit DC form, like in clustering [2], spherical separability problems [10], production-transportation planning [15], wireless sensor network planning [1], and data visualization [5]. It is worth noticing that these applications usually either model the problem directly as a single-objective problem or scalarize a bi-objective problem, even if they have multiobjective nature. Additionally, in [17] a probabilistic lot sizing model is solved as a multiobjective DC problem.

In this chapter, we are aiming to solve a DC problem of the form

$$\begin{cases} \text{minimize} & \{f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\} \\ \text{subject to} & \boldsymbol{x} \in X, \end{cases} \tag{14.1}$$

where $X = \{\boldsymbol{x} \in \mathbb{R}^n : g_l(\boldsymbol{x}) \leq 0,\ l \in \mathcal{L}\}$ and $\mathcal{L} = \{1, \ldots, m\}$. The objectives $f_i = p_i - q_i : \mathbb{R}^n \to \mathbb{R}$ for all $i \in \mathcal{I}$, such that the set $\mathcal{I} = \{1, \ldots, k\}$ denotes the indices of the objectives, and the constraints $g_l = r_l - s_l : \mathbb{R}^n \to \mathbb{R}$ for all $l \in \mathcal{L}$ are assumed to be at least partially conflicting and DC functions. Thus, the DC components p_i, q_i, r_l, s_l for all $i \in \mathcal{I}, l \in \mathcal{L}$ are convex.

The theory of DC functions has been widely studied in the past few decades (see e.g. [13, 14, 40]). Additionally, various methods have been developed for single-objective DC problems like DCA [23, 24, 32], proximal point based methods [38], special cutting plane based bundle methods [11, 19, 20], and branch-and-bound and outer approximation algorithms [16]. However, a little research devoted to multiobjective DC optimization focuses mainly on optimality conditions like in [9, 34, 39]. Moreover, the exact and inexact proximal point methods in [17, 18] have lately been introduced.

Our contribution to the field of multiobjective DC optimization is the multiobjective double bundle method for DC optimization (MDBDC) originally presented in [30]. MDBDC generalizes the single-objective double bundle method (DBDC) [20]

(see also Chap. 8) to the multiobjective optimization by utilizing the improvement function presented in [21, 27, 41]. MDBDC has some distinctive features. First, MDBDC can handle several DC objectives together with inequality DC constraints. Second, under mild assumptions MDBDC converges finitely to a weakly Pareto stationary solution. Last, MDBDC is a descent type method. In addition, MDBDC has proved to be usable also in practice, and when compared to a general nonconvex solver, MDBDC can obtain better solutions (i.e. every objective has a better value) by taking into account the DC structure of the problem [30].

We say some words about the comparison between MDBDC and the exact proximal point algorithm developed in [18]. Both of these methods base their ability to handle multiple objectives to the fact that the optimum of some single-objective problem is known to be a weak Pareto optimum, and they both utilize convex single-objective subproblems. However, in MDBDC the subgradients for both DC components needs to be evaluated while in the proximal point algorithm only the subgradient of the second DC component needs to be known. The first clear difference between the methods is that MDBDC can handle DC constraints while the proximal point algorithm is designed only for convex constraints. In addition, MDBDC is a descent type method improving every objective at each iteration. In the proximal point algorithm, the descent direction is provided only for a function $\sum_{i\in\mathcal{I}} \lambda_i f_i(\boldsymbol{x})$ such that $\sum_{i\in\mathcal{I}} \lambda_i = 1$ and $\lambda_i \geq 0$ for all $i \in \mathcal{I}$. Thus, every objective is not necessarily improved. Finally, MDBDC produces weakly Pareto stationary solutions while the proximal point algorithm yields only Pareto critical solutions. This theoretical difference is discussed with details in Sect. 14.2.

The rest of this chapter is organized as follows. First, in Sect. 14.2, we consider the optimality in multiobjective optimization and exemplify the difference between Pareto critical and weakly Pareto stationary solutions. Section 14.3 compresses all you need to know about MDBDC into a sketch of the method and proves its finite convergence. In Sect. 14.4, the numerical behaviour of MDBDC is discussed and argued in favour of to utilize a method specially designed for DC functions instead of a general nonconvex method. Finally, some concluding remarks are given in Sect. 14.5.

14.2 Critical Versus Stationary Solution in Multiobjective DC Optimization

We begin by saying some words about optimality in multiobjective optimization. A solution $\boldsymbol{x}^* \in X$ is a *global Pareto optimum* for the problem (14.1) if there does not exist another solution $\boldsymbol{x} \in X$ such that $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$ for all $i \in \mathcal{I}$ and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}^*)$ for at least one $j \in \mathcal{I}$. This means that we cannot improve any objective without deteriorating some other objective simultaneously. If we have a solution such that there does not exist any other solution yielding better values for every objective, we call the solution a *global weak Pareto optimum*. In other words, a

solution $\boldsymbol{x}^* \in X$ is a global weak Pareto optimum for the problem (14.1) if there does not exist another solution $\boldsymbol{x} \in X$ such that $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$ for all $i \in \mathcal{I}$. Moreover, a solution $\boldsymbol{x}^*$ is a *local (weak) Pareto optimum* for the problem (14.1) if there exists $\varepsilon > 0$ such that $\boldsymbol{x}^*$ is a global (weak) Pareto optimum on $X \cap B(\boldsymbol{x}^*; \varepsilon)$. To conclude, every Pareto optimum is also a weak Pareto optimum and instead of only one optimum we usually have several (weak) Pareto optima.

We start with defining weak Pareto stationary and critical solutions and after that the difference between these two concepts is exemplified. To simplify notations, we denote by

$$F(\boldsymbol{x}) = \bigcup_{i \in \mathcal{I}} \partial f_i(\boldsymbol{x}) \text{ and } G(\boldsymbol{x}) = \bigcup_{l \in L(\boldsymbol{x})} \partial g_l(\boldsymbol{x}),$$

where $\partial f_i(\boldsymbol{x})$ and $\partial g_l(\boldsymbol{x})$ are subdifferentials of f_i and g_l at $\boldsymbol{x} \in \mathbb{R}^n$, respectively and $L(\boldsymbol{x}) = \{l \in \mathcal{L} : g_l(\boldsymbol{x}) = 0\}$. Next, we give a necessary condition for a solution to be a local weak Pareto optimum of the problem (14.1) when objectives and constraints are general nonconvex functions. Recall that for a set $S \subseteq \mathbb{R}^n$ we denote by $K_S(\boldsymbol{x})$ and $S^{\leq}$ a contingent cone at $\boldsymbol{x} \in \mathbb{R}^n$ and a polar cone, respectively (see Definitions 1.12 and 1.13).

Theorem 14.1 ([26]) *If $\boldsymbol{x}^* \in X$ is a local weak Pareto optimum for the problem (14.1) with LLC objective and constraint functions, and the constraint qualification $G^{\leq}(\boldsymbol{x}^*) \subseteq K_X(\boldsymbol{x}^*)$ holds, then*

$$\mathbf{0} \in \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{cl} \operatorname{cone} G(\boldsymbol{x}^*). \tag{14.2}$$

Proof See [26, Theorem 15]. □

We say that the point $\boldsymbol{x}^*$ satisfying the condition (14.2) is *weakly Pareto stationary*. In general, nonconvex multiobjective methods find a solution $\boldsymbol{x}^* \in X$ being weakly Pareto stationary.

A suitable necessary condition for the problem (14.1) can be derived also by assuming that the objectives and constraints are DC functions. The condition like this is given in Theorem 3.1 in [34]. For our purposes to illustrate the properties of different conditions, it is enough to consider the unconstrained case of this condition given in [18]: If $\boldsymbol{x}' \in \mathbb{R}^n$ is a local weak Pareto optimum for the problem (14.1), where $X = \mathbb{R}^n$ and the objectives are DC functions, then

$$\operatorname{conv}\{\partial q_i(\boldsymbol{x}') : i \in \mathcal{I}\} \subseteq \operatorname{conv}\{\partial p_i(\boldsymbol{x}') : i \in \mathcal{I}\}.$$

However, this condition is hard to verify in practice, and thus in [18], the condition

$$\mathbf{0} \in \operatorname{conv}\{\partial p_i(\boldsymbol{x}') - \partial q_i(\boldsymbol{x}') : i \in \mathcal{I}\} \tag{14.3}$$

is validated and the solution $\boldsymbol{x}'$ satisfying this condition is called *Pareto critical*. Clearly, weakly Pareto stationary solution $\boldsymbol{x}^*$ is also Pareto critical, since

$$\mathbf{0} \in \operatorname{conv}\{\partial f_i(\boldsymbol{x}^*) : i \in \mathcal{I}\} \subseteq \operatorname{conv}\{\partial p_i(\boldsymbol{x}^*) - \partial q_i(\boldsymbol{x}^*) : i \in \mathcal{I}\}.$$

To argue why the inverse does not necessarily hold, we give an example.

Example 14.1 Consider the unconstrained bi-objective case of the problem (14.1) and let the DC components be $p_1(x) = \max\{-x, 2x\}$, $q_1(x) = \max\{-2x, x\}$, $p_2(x) = \max\{x^2, x\}$, and $q_2(x) = \max\{0.5x^2, -x\}$, where $x \in \mathbb{R}$. We consider the point $x' = 0$ and to verify its Pareto criticality we investigate the intersection

$$\lambda\partial p_1(x') + (1-\lambda)\partial p_2(x') \cap \lambda\partial q_1(x') + (1-\lambda)\partial q_2(x').$$

At the point x', neither of the objectives is differentiable, and the intersection is of the form $\lambda[-1, 2] + (1-\lambda)[0, 1] \cap \lambda[-2, 1] + (1-\lambda)[-1, 0]$. For instance, with $\lambda = 1$ this intersection equals $[-1, 1]$ being a nonempty set. Thus, the condition (14.3) is valid and x' is Pareto critical. However, x' is not weakly Pareto stationary, since $0 \notin \operatorname{conv}\{\partial f_1(x'),\ \partial f_2(x')\} = \{1\}$. The similar observation can be made in the single-objective case as well, as was exemplified in Chap. 8.

A natural approach towards solving the multiobjective DC problem would be to verify that the condition (14.3) is satisfied. However, in order to reduce the number of possible non-optimal solutions in our set of feasible solutions, we want to ensure that the solution produced is weakly Pareto stationary. To obtain a stationary solution is not a trivial task even in the single-objective case and to obtain weak Pareto stationarity in the multiobjective case, we introduce the *improvement function* $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ [21, 41] defined by

$$H(\boldsymbol{x}, \boldsymbol{y}) = \max\{f_i(\boldsymbol{x}) - f_i(\boldsymbol{y}),\ g_l(\boldsymbol{x}) : i \in \mathcal{I}, l \in \mathcal{L}\}. \tag{14.4}$$

One reason for the utility of the improvement function in the case of multiobjective DC optimization raises from the fact that it is a DC function. Indeed, since the objectives f_i for all $i \in \mathcal{I}$ and the constraints g_l for all $l \in \mathcal{L}$ are assumed to be DC functions, then $H(\cdot\, \boldsymbol{y})$ is a DC function as a maximum of DC functions [14]. Moreover, the improvement function has three useful elementary properties stated in the following theorem legitimating the use of it (see also Chap. 13).

Theorem 14.2 ([27, 41]) *The improvement function $H(\cdot\,, \boldsymbol{y})$ (14.4) has the following properties:*

(i) *If $H(\boldsymbol{x}, \boldsymbol{y}) < H(\boldsymbol{y}, \boldsymbol{y})$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in X$ then $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$ for all $i \in \mathcal{I}$ and $g_l(\boldsymbol{x}) < 0$ for all $l \in \mathcal{L}$.*

(ii) *If the solution* $\boldsymbol{x}^* \in X$ *is a global weak Pareto optimum of the problem* (14.1)*, then*

$$\boldsymbol{x}^* = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^n} H(\boldsymbol{x}, \boldsymbol{x}^*).$$

(iii) *If* $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$*, then the solution* $\boldsymbol{x}^* \in X$ *of the problem* (14.1) *is weakly Pareto stationary.*

Proof See [30, Theorem 2] and also Theorem 13.6 in Chap. 13. □

Based on the above theorem, we obtain a weakly Pareto stationary solution $\boldsymbol{x}^*$ for the problem (14.1) if we find a *Clarke stationary solution* for $H(\cdot, \boldsymbol{x}^*)$ (i.e. $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$). Therefore, we are able to utilize some single-objective DC method to find a Clarke stationary solution. However, many of these methods produce only critical solutions. Since we are able to use the DC structure of the improvement function as an advantage, we will apply the escape procedure presented in [20] being able to test whether an approximate Clarke stationary condition is valid. Additionally, if this condition is not satisfied, the procedure generates a new search direction.

14.3 Multiobjective Double Bundle Method for DC Optimization

The multiobjective double bundle method for DC optimization (MDBDC) collects ideas from three different bundle type methods and combines them into one package such that a method for multiobjective DC optimization with inequality DC constraints is created. These three methods are the multiobjective proximal bundle method (MPB) ([27, 29] and Chap. 13), the proximal bundle method for DC optimization (PBDC) ([19] and Chap. 8), and the double bundle method for DC optimization (DBDC) ([20] and Chap. 8).

The idea how to apply the improvement function with a single-objective bundle-based method is absorbed from MPB being a bundle-based method to solve nonconvex constrained multiobjective problems. In MPB, the technique utilizing the improvement function [21, 27, 41] was combined with the single-objective proximal bundle method [22]. Besides the ability to handle several objectives simultaneously, the improvement function gives a way to handle constraints as well. Additionally, the descent property of MDBDC is the direct consequence of the properties of the improvement function.

While MPB gives ideas to treat multiple objectives, PBDC gives core ingredients for the algorithm of MDBDC related to the single-objective part. The cutting plane model used in MDBDC for the improvement function bases on the one used in PBDC. The peculiarity of this model is that it captures both the convex and concave behaviour of a DC function by utilizing explicitly the DC decomposition.

DBDC is an unconstrained single-objective sibling of MDBDC. In order to avoid Pareto critical solutions in MDBDC, we take an advantage of the escape procedure being part of DBDC. DBDC is developed from PBDC by adding the escape procedure giving an ability to avoid critical solutions. If we end up to a critical point, the procedure either gives a new descent search direction or ensures that the approximate Clarke stationary condition is valid.

MDBDC is designed to solve the problem (14.1) with DC objectives and constraints. The method improves all the objectives simultaneously meaning that MDBDC is a method of descent type. The idea of MDBDC in its simplicity is to apply the improvement function such that instead of a constrained multiobjective problem we can solve an unconstrained single-objective one. This new problem is then solved by using a special cutting plane model and the modified version of DBDC. As a result of this process, we end up to a weakly Pareto stationary solution. Here, we describe the general idea of MDBDC and the more detailed description can be found in [30].

Improvement Function and Cutting Plane Model Since the cutting plane model used in MDBDC utilizes the DC decomposition of the objective, we discuss first about the DC decomposition of the improvement function. As it was previously mentioned, the improvement function formed by DC functions is a DC function, and thus, the DC decomposition exists. This decomposition can be obtained as in [14]. For example, we can rewrite the objectives $f_i = p_i - q_i$ for all $i \in \mathcal{I}$ and the constraints $g_l = r_l - s_l$ for all $l \in \mathcal{L}$ as

$$f_i(\boldsymbol{x}) = p_i(\boldsymbol{x}) + \sum_{\substack{j \in \mathcal{I} \\ j \neq i}} q_j(\boldsymbol{x}) + \sum_{t \in \mathcal{L}} s_t(\boldsymbol{x}) - \sum_{j \in \mathcal{I}} q_j(\boldsymbol{x}) - \sum_{t \in \mathcal{L}} s_t(\boldsymbol{x}),$$

$$g_l(\boldsymbol{x}) = r_l(\boldsymbol{x}) + \sum_{\substack{t \in \mathcal{L} \\ t \neq l}} s_t(\boldsymbol{x}) + \sum_{j \in \mathcal{I}} q_j(\boldsymbol{x}) - \sum_{j \in \mathcal{I}} q_j(\boldsymbol{x}) - \sum_{t \in \mathcal{L}} s_t(\boldsymbol{x}).$$

In order to simplify the presentation, we denote

$$A_i(\boldsymbol{x}, \boldsymbol{y}) = p_i(\boldsymbol{x}) + \sum_{\substack{j \in \mathcal{I} \\ j \neq i}} q_j(\boldsymbol{x}) + \sum_{t \in \mathcal{L}} s_t(\boldsymbol{x}) - f_i(\boldsymbol{y}) \quad \text{and}$$

$$B_l(\boldsymbol{x}) = r_l(\boldsymbol{x}) + \sum_{\substack{t \in \mathcal{L} \\ t \neq l}} s_t(\boldsymbol{x}) + \sum_{j \in \mathcal{I}} q_j(\boldsymbol{x}).$$

Now the DC decomposition of the improvement function is of the form

$$H(\boldsymbol{x}, \boldsymbol{y}) = H_1(\boldsymbol{x}, \boldsymbol{y}) - H_2(\boldsymbol{x}),$$

where

$$H_1(\boldsymbol{x}, \boldsymbol{y}) = \max\{A_i(\boldsymbol{x}, \boldsymbol{y}), B_l(\boldsymbol{x}) : i \in \mathcal{I}, l \in \mathcal{L}\} \text{ and} \tag{14.5}$$
$$H_2(\boldsymbol{x}) = \sum_{j \in \mathcal{I}} q_j(\boldsymbol{x}) + \sum_{t \in \mathcal{L}} s_t(\boldsymbol{x}).$$

Both DC components H_1 and H_2 are convex with respect to $\boldsymbol{x}$ and the vector $\boldsymbol{y}$ is treated as a constant.

As the name of MDBDC suggests, we collect information from the previous iterations into bundles. In the following, the index h denotes the h-th iteration and $\boldsymbol{x}_h$ is the current iteration point. We assume that at each auxiliary point the function value and one arbitrary subgradient of p_i, q_i, r_l and s_l can be evaluated. From these, we can compose values for functions $A_i(\cdot, \boldsymbol{x}_h)$, B_l, $H_1(\cdot, \boldsymbol{x}_h)$, and H_2 and their subgradients $\boldsymbol{a}_i$, $\boldsymbol{b}_l$, $\boldsymbol{h}_1$, and $\boldsymbol{h}_2$, respectively.

In MDBDC, we collect information into two separate bundles $\mathcal{B}_1^h$ and $\mathcal{B}_2^h$ for $H_1(\cdot, \boldsymbol{y})$ and H_2, respectively. The bundles consist of triplets formed by an auxiliary point $\boldsymbol{y}_j$, a corresponding function value, and a subgradient. Here the index j is an element of the index set $\mathcal{J}_1^h$ or $\mathcal{J}_2^h$ depending on whether the bundle is for $H_1(\cdot, \boldsymbol{x}_h)$ or H_2. In practice, the bundle $\mathcal{B}_1^h$ is formed by having separate bundles for each $A_i(\cdot, \boldsymbol{x}_h)$, $i \in \mathcal{I}$ and B_l, $l \in \mathcal{L}$ and taking the union of them.

In order to find a search direction, we approximate the improvement function by utilizing the special cutting plane model which is based on the one proposed in [19]. Therefore, we linearize the convex DC components separately by utilizing the classical cutting plane model [22, 25, 37]. This way, we can capture both the convex and concave behaviour of the improvement function. To form an approximation for $H_1(\cdot, \boldsymbol{x}_h)$ and H_2, we linearize all the components $A_i(\cdot, \boldsymbol{x}_h)$ and B_l of $H_1(\cdot, \boldsymbol{x}_h)$ and H_2. We begin by giving the linearization for $A_i(\cdot, \boldsymbol{x}_h)$:

$$\hat{A}_i^h(\boldsymbol{x}) = \max_{j \in \mathcal{J}_1^h} \left\{ A_i(\boldsymbol{x}_h, \boldsymbol{x}_h) + \boldsymbol{a}_{i,j}^T(\boldsymbol{x} - \boldsymbol{x}_h) - \alpha_{i,j}^A \right\},$$

where $\boldsymbol{a}_{i,j} \in \partial A_i(\boldsymbol{y}_j, \boldsymbol{x}_h)$ for $j \in \mathcal{J}_1^h$. The linearization errors evaluated at $\boldsymbol{x}_h$ for all $j \in \mathcal{J}_1^h$ are

$$\alpha_{i,j}^A = A_i(\boldsymbol{x}_h, \boldsymbol{x}_h) - A_i(\boldsymbol{y}_j, \boldsymbol{x}_h) - \boldsymbol{a}_{i,j}^T(\boldsymbol{x}_h - \boldsymbol{y}_j) \text{ for all } i \in \mathcal{I}.$$

Note that due to the convexity, all the linearization errors are nonnegative.

Similarly, we can linearize functions B_l for all $l \in \mathcal{L}$ and H_2 and denote these approximations by $\hat{B}_l^h(\boldsymbol{x})$ and $\hat{H}_2^h(\boldsymbol{x})$, respectively. Thus, we obtain the cutting plane model for $H_1(\cdot, \boldsymbol{x}_h)$ of the form

$$\hat{H}_1^h(\boldsymbol{x}) = \max\{\hat{A}_i^h(\boldsymbol{x}),\ \hat{B}_l^h(\boldsymbol{x}) : i \in \mathcal{I}, l \in \mathcal{L}\}. \tag{14.6}$$

Finally, by utilizing the above approximations, we obtain the following piecewise linear nonconvex DC cutting plane model of $H(\cdot, \boldsymbol{x}_h)$:

$$\hat{H}^h(\boldsymbol{x}) = \hat{H}_1^h(\boldsymbol{x}) - \hat{H}_2^h(\boldsymbol{x}).$$

From the definition of the cutting plane model, it follows that $\hat{H}_1^h(\boldsymbol{x}_h + \boldsymbol{d}) \leq H_1(\boldsymbol{x}_h + \boldsymbol{d}, \boldsymbol{x}_h)$ and $\hat{H}_2^h(\boldsymbol{x}_h + \boldsymbol{d}) \leq H_2(\boldsymbol{x}_h + \boldsymbol{d})$ [30].

Direction Finding By bearing in mind Theorem 14.2, we are motivated to find a solution $\boldsymbol{x}^* \in X$ such that $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$. Therefore, we define a search direction as a solution of the problem

$$\begin{cases} \text{minimize} & H(\boldsymbol{x}_h + \boldsymbol{d}, \boldsymbol{x}_h) \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{14.7}$$

By utilizing the model of $H(\cdot, \boldsymbol{x}_h)$, we can estimate the problem (14.7) with a nonsmooth nonconvex DC problem

$$\begin{cases} \text{minimize} & P^h(\boldsymbol{d}) = \hat{H}_1^h(\boldsymbol{x}_h + \boldsymbol{d}) - \hat{H}_2^h(\boldsymbol{x}_h + \boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{14.8}$$

where $t > 0$ is a proximity parameter. The solution of this problem is denoted by $\boldsymbol{d}_t^h$.

We use the solution approach described in [23, 24, 32] to find a global solution of the problem (14.8). This approach can be applied, since the second DC component of P^h is $\hat{H}_2^h(\boldsymbol{x}_h + \boldsymbol{d})$ and it is polyhedral convex see (8.5) in Chap. 8. The objective of the problem (14.8) can now be rewritten as

$$P^h(\boldsymbol{d}) = \min_{j \in \mathcal{J}_2^h} \left\{ P_j^h(\boldsymbol{d}) = \hat{H}_1^h(\boldsymbol{x}_h + \boldsymbol{d}) - H_2(\boldsymbol{x}_h) - \boldsymbol{h}_{2,j}^T \boldsymbol{d} + \alpha_{2,j}^H + \frac{1}{2t}\|\boldsymbol{d}\|^2 \right\}.$$

This enables us to change the order of the minimization in the problem (14.8). Thus, we end up to solve $|\mathcal{J}_2^h|$ convex subproblems

$$\begin{cases} \text{minimize} & P_j^h(\boldsymbol{d}) = \hat{H}_1^h(\boldsymbol{x}_h + \boldsymbol{d}) - H_2(\boldsymbol{x}_h) - \boldsymbol{h}_{2,j}^T \boldsymbol{d} + \alpha_{2,j}^H + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{subject to} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{14.9}$$

where $j \in \mathcal{J}_2^h$. The solution of the individual subproblem $j \in \mathcal{J}_2^h$ is denoted by $\boldsymbol{d}_t^h(j)$, and the global solution $\boldsymbol{d}_t^h$ of the problem (14.8) is $\boldsymbol{d}_t^h = \boldsymbol{d}_t^h(j^*)$, where the index $j^* = \operatorname{argmin} \left\{ P_j^h\left(\boldsymbol{d}_t^h(j)\right) : j \in \mathcal{J}_2^h \right\}$. In practice, the amount of computation can be controlled, since the size of the bundle $\mathcal{B}_2^h$ can be freely chosen such that $|\mathcal{J}_2^h| \geq 1$.

If $\|\boldsymbol{d}_t^h\| < \delta$, where $\delta > 0$ is a fixed stopping tolerance, we either generate a new descent direction or Clarke stationarity is achieved. In order to test Clarke stationarity, we need some information about the subdifferential $\partial H(\boldsymbol{x}_h, \boldsymbol{x}_h)$. Unfortunately, by calculating arbitrary subgradients $\boldsymbol{h}_1 \in \partial H_1(\boldsymbol{x}_h, \boldsymbol{x}_h)$ and $\boldsymbol{h}_2 \in \partial H_2(\boldsymbol{x}_h)$, we cannot guarantee that $\boldsymbol{h}_1 - \boldsymbol{h}_2 \in \partial H(\boldsymbol{x}_h, \boldsymbol{x}_h)$. Thus, we are justified to use the escape procedure (see Algorithm 8.2 in Chap. 8) having the ability to select $\boldsymbol{h}_1^* \in \partial H_1(\boldsymbol{x}, \boldsymbol{x})$, $\boldsymbol{h}_2^* \in \partial H_2(\boldsymbol{x})$ for the objective function $H = H_1 - H_2$ at any $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{h}_1^* - \boldsymbol{h}_2^* \in \partial H(\boldsymbol{x}, \boldsymbol{x})$ is ensured [20].

At the j-th iteration of the escape procedure, we approximate the Goldstein ε-subdifferential $\partial_\varepsilon^G H(\boldsymbol{x}_h, \boldsymbol{x}_h)$ with a set U_j consisting of subgradients calculated as the difference of subgradients of the DC components. Thus, the new search direction can be found by calculating $\boldsymbol{d}_{j+1} = -\bar{\boldsymbol{u}}_j / \|\bar{\boldsymbol{u}}_j\|$, where $\bar{\boldsymbol{u}}_j$ is a solution of the problem

$$\begin{cases} \text{minimize} & \frac{1}{2}\|\boldsymbol{u}\|^2 \\ \text{subject to} & \boldsymbol{u} \in U_j. \end{cases}$$

If this direction is not descent or Clarke stationarity is not achieved, then the approximation of $\partial_\varepsilon^G H(\boldsymbol{x}_h, \boldsymbol{x}_h)$ is improved with a new subgradient. To conclude, in order to exit from the escape procedure, we either find a new descent search direction or $\|\bar{\boldsymbol{u}}_j\| \leq \delta$ meaning that the approximate Clarke stationary condition is satisfied and the algorithm of MDBDC is terminated.

Algorithm In this section, we give a general idea of the method with the simplified version of MDBDC presented in Algorithm 14.1. More detailed description of the MDBDC algorithm is given in [30].

We make some remarks about the algorithm. First, in Step 5 we execute the escape procedure given in Algorithm 8.2 in Chap. 8. Note that in this case, the procedure is executed for the current iteration point $\boldsymbol{x}_h$ with the improvement function $H(\cdot, \boldsymbol{x}_h)$ as its objective. Second, we utilize the proximity parameter $t \in [t_{\min}, t_{\max}]$ in Algorithm 14.1, which can be either decreased in Steps 7 and 8 or updated in Step 10 by utilizing the updating procedure inspired by the weighting update method in [22]. During the latter update, the proximity parameter may either increase or decrease. The update of the proximity parameter yields an improvement for the model in the both cases.

As was mention, we give here a more general outline of the algorithm working well in theory. In practice, we can improve the numerical behaviour of MDBDC significantly. For example, in the initialization phase of Algorithm 14.1, the scaling procedure [30] may be executed. The positive affect of scaling has its roots in the DC decomposition of the improvement function. If the objective functions have different magnitudes, one of the DC components may dominate the others and hide their effects. Even if the scaling is executed, it does not affect the optimality of the solution, since the modified objectives have the same optima than the unmodified original objectives. Other possible numerical improvements are, for instance, to

Algorithm 14.1: MDBDC

Data: The stopping tolerance $\delta \in (0, 1)$, the enlargement parameter $\theta > 0$, the decrease parameters $r, c \in (0, 1)$, the increase parameter $R > 1$, and the descent parameter $m \in (0, 1)$.

Step 1. (*Initialization*) Select $\boldsymbol{x}_0 \in X$ and calculate $\boldsymbol{h}_1(\boldsymbol{x}_0) \in \partial H_1(\boldsymbol{x}_0, \boldsymbol{x}_0)$ and $\boldsymbol{h}_2(\boldsymbol{x}_0) \in \partial H_2(\boldsymbol{x}_0)$. Initialize $\mathcal{B}_1^0$ and $\mathcal{B}_2^0$, $t = 0$, $\boldsymbol{h}_{2,\max} = \boldsymbol{0}$, and $h = 0$.

Step 2. (*Criticality*) If $\|\boldsymbol{h}_1(\boldsymbol{x}_h) - \boldsymbol{h}_2(\boldsymbol{x}_h)\| < \delta$, then $\boldsymbol{d}_t = \boldsymbol{0}$ and go to Step 5.

Step 3. (*Proximity parameter*) If $\|\boldsymbol{h}_2(\boldsymbol{x}_h)\| > \|\boldsymbol{h}_{2,\max}\|$, then $\boldsymbol{h}_{2,\max} = \boldsymbol{h}_2(\boldsymbol{x}_h)$. Set

$$t_{\min} = \frac{r\theta}{2(\|\boldsymbol{h}_1(\boldsymbol{x}_h)\| + \|\boldsymbol{h}_{2,\max}\|)} \tag{14.10}$$

and $t_{\max} = Rt_{\min}$. If $t \notin [t_{\min}, t_{\max}]$, then select $t \in [t_{\min}, t_{\max}]$.

Step 4. (*Search direction*) Calculate the search direction $\boldsymbol{d}_t$ as a solution of (14.8).

Step 5. (*Escape procedure*) If $\|\boldsymbol{d}_t\| < \delta$, then execute Algorithm 8.2 presented in Chap. 8 for the point $\boldsymbol{x}_h$ to obtain $\boldsymbol{x}^+$. Set $\boldsymbol{x}_{h+1} = \boldsymbol{x}^+$ and go to Step 9.

Step 6. (*Descent test*) Set $\boldsymbol{y} = \boldsymbol{x}_h + \boldsymbol{d}_t$. If

$$H(\boldsymbol{y}, \boldsymbol{x}_h) - H(\boldsymbol{x}_h, \boldsymbol{x}_h) \leq m\big(\hat{H}^h(\boldsymbol{y}) - H(\boldsymbol{x}_h, \boldsymbol{x}_h)\big),$$

then set $\boldsymbol{x}_{h+1} = \boldsymbol{y}$ and go to Step 9.

Step 7. (*Parameter update*) If $f_i(\boldsymbol{y}) > f_i(\boldsymbol{x}_0)$ for any $i \in \mathcal{I}$ and $\|\boldsymbol{d}_t\| > \theta$, then set $t = t - c(t - t_{\min})$ and go to Step 4.

Step 8. (*Bundle update*) Decrease t if necessary, and update $\mathcal{B}_1^h$ and $\mathcal{B}_2^h$. If a new subgradient $\boldsymbol{h}_2 \in \partial H_2(\boldsymbol{y})$ satisfies $\|\boldsymbol{h}_2\| > \|\boldsymbol{h}_{2,\max}\|$, then set $\boldsymbol{h}_{2,\max} = \boldsymbol{h}_2$ and update $t_{\min}$ using (14.10). Go to Step 4.

Step 9. (*Clarke stationarity*) If $\boldsymbol{x}_{h+1} = \boldsymbol{x}_h$, then Clarke stationarity is achieved and **stop** with $\boldsymbol{x}^* = \boldsymbol{x}_h$ as the final solution.

Step 10. (*Model update*) Update t and the bundles $\mathcal{B}_1^{h+1} \subseteq \mathcal{B}_1^h$ and $\mathcal{B}_2^{h+1} \subseteq \mathcal{B}_2^h$ selected. Calculate $\boldsymbol{h}_1(\boldsymbol{x}_{h+1}) \in \partial H_1(\boldsymbol{x}_{h+1}, \boldsymbol{x}_{h+1})$ and $\boldsymbol{h}_2(\boldsymbol{x}_{h+1}) \in \partial H_2(\boldsymbol{x}_{h+1})$. Set $h = h + 1$ and go to Step 2.

execute the escape procedure when the decrease in the model is nearly non-existing or to utilize more sophisticated update procedure for the proximity parameter in Step 8.

Lastly, some words about the bundles. Obviously, in practice the size of the bundles must be limited. The size of the bundle $\mathcal{B}_1^h$ has to be selected such that information regarding both $A_i(\cdot, \boldsymbol{x}_h)$ and B_l for all $i \in \mathcal{I}$ and $l \in \mathcal{L}$ is included.

Thus, $|\mathcal{J}_1^h| \geq k + m$. For the bundle $\mathcal{B}_2^h$, the size of the bundle $|\mathcal{J}_2^h| \geq 1$, since the bundle element associated to the current iteration point must be included. As was mention previously, via restriction of $|\mathcal{J}_2^h|$ the number of subproblems solved can be controlled.

Convergence Analysis The following convergence analysis is divided as follows: In Theorem 14.3, we state that MDBDC stops after a finite number of iterations at the point which satisfies the approximate Clarke stationary condition. In order to prove that, we need Lemma 14.1 to guarantee that the loop between Steps from 4 to 8 is finite and Theorem 8.6 in Chap. 8 to give the finite maximum number of iterations needed in the execution of the escape procedure in Step 5. Finally, Theorem 14.4 considers weak Pareto stationarity of the solution.

Throughout the convergence analysis, we assume that the following assumptions are valid:

Assumption 14.1 *The subdifferentials $\partial H_1(\boldsymbol{x}, \boldsymbol{y})$ and $\partial H_2(\boldsymbol{x})$ are polytopes for each $\boldsymbol{x} \in \mathbb{R}^n$.*

Assumption 14.2 *The level set $\mathcal{F}_0 = \{\boldsymbol{x} \in X : f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}_0)$ for all $i \in \mathcal{I}\}$ is compact.*

We begin our convergence analysis by investigating the loop between Steps from 4 to 8. If this loop is infinite, it would lead to a contradiction as is seen in the proof of Lemma 14.1.

Lemma 14.1 ([30]) *Let Assumption 14.2 be valid. For any $\delta \in (0, 1)$, Algorithm 14.1 cannot pass infinitely through the sequence of Steps from 4 to 8.*

Proof The proof is similar to the one given in Proposition 8.4 in Chap. 8. □

Now we show the finite convergence of Algorithm 14.1 to a solution satisfying the approximate Clarke stationary condition for the improvement function.

Theorem 14.3 ([30]) *Let Assumptions 14.1 and 14.2 be valid. For any $\delta \in (0, 1)$ and $\varepsilon > 0$, the execution of Algorithm 14.1 stops after a finite number of iterations at the point $\boldsymbol{x}^*$ satisfying the approximate Clarke stationary condition $\|\boldsymbol{\xi}^*\| \leq \delta$, where $\boldsymbol{\xi}^* \in \partial_\varepsilon^G H(\boldsymbol{x}^*, \boldsymbol{x}^*)$.*

Proof The execution of Algorithm 14.1 stops only if the Clarke stationary point $\boldsymbol{x}^*$ is found in Step 9 meaning that the approximate Clarke stationary condition is satisfied in the escape procedure in Algorithm 8.2. Assume, that Algorithm 14.1 is executed infinitely, and thus, the stopping condition in Step 9 is never satisfied.

Similarly to the proof of Theorem 8.7 in Chap. 8, we can deduce that after each iteration we have $H(\boldsymbol{x}_{h+1}, \boldsymbol{x}_h) - H(\boldsymbol{x}_h, \boldsymbol{x}_h) \leq -\sigma < 0$, where

$$\sigma = \min\left\{\hat{m}\varepsilon\delta, \ \frac{m\delta^3}{Rr\theta}\right\} > 0$$

and $m, r \in (0,1)$, $R > 1$ and $\theta > 0$ are the parameters of Algorithm 14.1. Additionally, $\varepsilon > 0$ and $\hat{m} \in (0,1)$ are the parameters of the escape procedure in Algorithm 8.2. Due to the definition of $H(\cdot, \boldsymbol{x}_h)$ in (14.4), $H(\boldsymbol{x}_h, \boldsymbol{x}_h) = 0$ yielding that $H(\boldsymbol{x}_{h+1}, \boldsymbol{x}_h) \leq -\sigma$. Especially,

$$f_i(\boldsymbol{x}_{h+1}) - f_i(\boldsymbol{x}_h) < -\sigma < 0 \text{ for all } i \in \mathcal{I},$$

and after the h-th iteration,

$$f_i(\boldsymbol{x}_h) - f_i(\boldsymbol{x}_0) \leq -h\sigma \text{ for all } i \in \mathcal{I}.$$

By passing to the limit $h \to \infty$, we obtain

$$\lim_{h \to \infty} f_i(\boldsymbol{x}_h) - f_i(\boldsymbol{x}_0) \leq -\infty \text{ for all } i \in \mathcal{I}$$

yielding a contradiction, since based on Assumption 14.2 and the fact that DC functions are LLC, every $f_i, i \in \mathcal{I}$ must be bounded from below. □

Finally, we are interested in to argue that a Clarke stationary solution for the improvement function yields a weakly Pareto stationary solution for the original multiobjective problem. In order to prove this, the properties of the improvement function described in Theorem 14.2 are applied.

Theorem 14.4 ([30]) *Let f_i and g_l be DC functions for all $i \in \mathcal{I}$ and $l \in \mathcal{L}$. Suppose that Assumptions 14.1 and 14.2 are valid. Then, MDBDC stops after a finite number of iterations with the solution $\boldsymbol{x}^* \in X$ being a weakly Pareto stationary point for the problem (14.1).*

Proof Consider minimization of an improvement function (14.4). By Theorem 14.3, after a finite number of iterations MDBDC finds a solution $\boldsymbol{x}^* \in \mathbb{R}^n$ such that $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$. According to Theorem 14.2 (iii), the solution $\boldsymbol{x}^* \in X$ is weakly Pareto stationary for the problem (14.1). □

14.4 Numerical Behaviour of MDBDC

We discuss about the numerical behaviour of MDBDC by utilizing the computational experiments provided in [30], where the performances of MDBDC and MPB are compared. MPB has been selected as a reference method due to its somehow similar structure, but unlike MDBDC, it is designed for a general nonconvex problem. The 53 test problems are formed such that they all have either two or three objectives and the objectives are collected from academic single-objective DC problems. Some of the problems also include either a DC or concave constraint. The dimension of the problems varies from 2 to 500 such that 37 of them are small

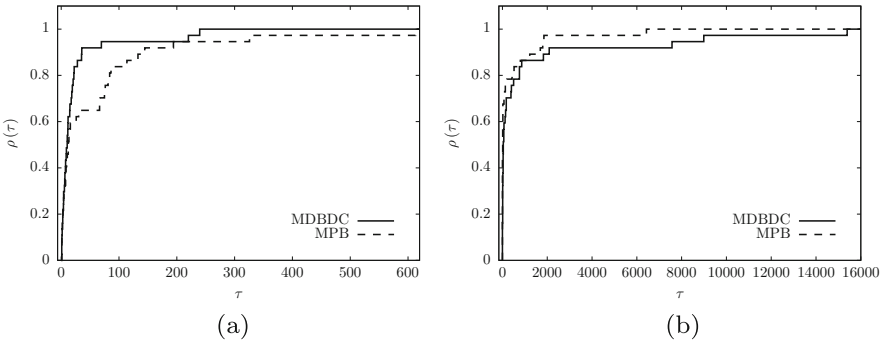

Fig. 14.1 Small test problems ($2 \leq n \leq 100$). (**a**) Subgradient evaluations. (**b**) CPU

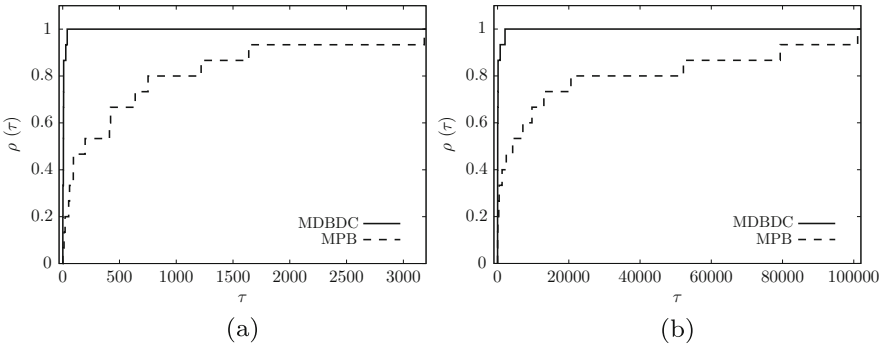

Fig. 14.2 Large test problems ($250 \leq n \leq 500$). (**a**) Subgradient evaluations. (**b**) CPU

($2 \leq n \leq 100$) and 16 are large ($250 \leq n \leq 500$). Since MDBDC and MPB both failed at one large test problem, we exclude this case from the discussion.

The numerical performance is illustrated in Figs. 14.1 and 14.2 where the performance profiles [7] for small and large test problems are given, respectively. In the performance profiles, subgradient evaluations and CPU times are compared. In the small test problems, MDBDC wins MPB in terms of subgradient evaluations but in terms of CPU times MPB performs slightly better than MDBDC. Nevertheless, in the larger test problems, MDBDC beats MPB both in subgradient evaluations and CPU times. Therefore, we can conclude that, in the computational point of view, MDBDC is a good alternative for MPB in the case of DC problems.

Not only to compare the execution of the algorithms, another main goal in the numerical experiments in [30] was to emphasize the difference in the solutions obtained. To compare solutions, we say that a solution is better than the other if it has better function values for every objective. In practice, it is possible that one method finds a better solution than the other even if they both find theoretically equally good solutions, namely weak Pareto stationary points. This is due to the fact that both local and global optima satisfy the condition (14.2) and the feasible

set in the objective space is nonconvex. An interesting observation is that by taking into account the DC structure of the problem, MDBDC finds a better solution than MPB around 30% of the tests performed in [30], even though both methods find theoretically equally good solutions. Moreover, in half of the cases where MPB uses less computational efforts, MDBDC finds a better solution. This shows that the model used in MPB is more easily attracted to local optima.

14.5 Conclusions

We have discussed about the multiobjective double bundle method for DC optimization (MDBDC) being a method for multiobjective DC problems with inequality DC constraints. The method is descent and under mild assumptions it is proved to be finitely convergent to a weakly Pareto stationary solution. MDBDC has shown to behave well numerically and it is observed to be profitable to use a method taking into account a DC structure instead of a general nonconvex method.

MDBDC can be used in several ways. First, it can be used to solve only one weakly Pareto stationary solution, or execute it with different starting points to obtain an approximation of the set of local weak Pareto optima. Due to the descent property, the starting point is projected to the set of local weak Pareto optima in the decision space such that the solution obtained lies in the negative orthant from the starting point. Another possibility is to use MDBDC as a component of some interactive method as MPB was used in [29]. In addition, MDBDC is suitable to solve single-objective DC problems with DC constraints.

Acknowledgement This work was financially supported by the University of Turku and the Academy of Finland (Project No. 294002).

References

1. Astorino, A., Miglionico, G.: Optimizing sensor cover energy via DC programming. Optim. Lett. **10**(2), 355–368 (2016)
2. Bagirov, A., Yearwood, J.: A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. Eur. J. Oper. Res. **170**(2), 578–596 (2006)
3. Bello Cruz, J.Y., Iusem, A.N.: A strongly convergent method for nonsmooth convex minimization in Hilbert spaces. Numer. Funct. Anal. Optim. **32**(10), 1009–1018 (2011)
4. Bonnel, H., Iusem, A.N., Svaiter, B.F.: Proximal methods in vector optimization. SIAM J. Optim. **15**(4), 953–970 (2005)
5. Carrizosa, E., Guerrero, V., Romero Morales, D.: Visualizing data as objects by DC (difference of convex) optimization. Math. Program. **169**(1), 119–140 (2018)
6. Craft, D., Halabi, T., Shih, H.A., Bortfeld, T.: An approach for practical multiobjective IMRT treatment planning. Int. J. Radiat. Oncol. Biol. Phys. **69**(5), 1600–1607 (2007)
7. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)

8. Ehrgott, M.: Multicriteria Optimization, 2nd edn. Springer, Berlin (2005)
9. Gadhi, N., Metrane, A.: Sufficient optimality condition for vector optimization problems under D.C. data. J. Global Optim. **28**(1), 55–66 (2004)
10. Gaudioso, M., Gruzdeva, T.V., Strekalovsky, A.S.: On numerical solving the spherical separability problem. J. Global Optim. **66**(1), 21–34 (2016)
11. Gaudioso, M., Giallombardo, G., Miglionico, G., Bagirov, A.: Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. J. Global Optim. **71**(1), 37–55 (2018)
12. Gutjahr, W.J., Nolz, P.C.: Multicriteria optimization in humanitarian aid. Eur. J. Oper. Res. **252**(2), 351–366 (2016)
13. Hartman, P.: On functions representable as a difference of convex functions. Pac. J. Math. **9**(3), 707–713 (1959)
14. Hiriart-Urruty, J-.B.: Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. In: Ponstein, J. (ed.) Convexity and Duality in Optimization, vol. 256, pp. 37–70. Springer, Berlin (1985)
15. Holmberg, K., Tuy, H.: A production-transportation problem with stochastic demand and concave production costs. Math. Program. **85**(1), 157–179 (1999)
16. Horst, R., Thoai, N.V.: DC programming: Overview. J. Optim. Theory Appl. **103**(1), 1–43 (1999)
17. Ji, Y., Qu, S.: Proximal point algorithms for vector DC programming with applications to probabilistic lot sizing with service levels. Discret. Dyn. Nat. Soc. **2017**, 5675183 (2017). https://doi.org/10.1155/2017/5675183
18. Ji, Y., Goh, M., De Souza, R.: Proximal point algorithms for multi-criteria optimization with the difference of convex objective functions. J. Optim. Theory Appl. **169**(1), 280–289 (2016)
19. Joki, K., Bagirov, A., Karmitsa, N., Mäkelä, M.M.: A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. J. Global Optim. **68**(3), 501–535 (2017)
20. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M., Taheri, S.: Double bundle method for finding Clarke stationary points in nonsmooth DC programming. SIAM J. Optim. **28**(2), 1892–1919 (2018)
21. Kiwiel, K.C.: A descent method for nonsmooth convex multiobjective minimization. Large Scale Syst. **8**(2), 119–129 (1985)
22. Kiwiel, K.C.: Proximity control in bundle methods for convex nondifferentiable optimization. Math. Program. **46**(1–3), 105–122 (1990)
23. Le Thi, H.A., Pham Dinh, T.: Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. J. Global Optim. **11**(3), 253–285 (1997)
24. Le Thi, H.A., Pham Dinh, T.: The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann. Oper. Res. **133**(1–4), 23–46 (2005)
25. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: analysis and algorithms with applications to optimal control. World Scientific, Singapore (1992)
26. Mäkelä, M.M., Eronen, V.-P., Karmitsa, N.: On nonsmooth multiobjective optimality conditions with generalized convexities. In: Rassias, T.M., Floudas, C.A., Butenko, S. (eds.) Optimization in Science and Engineering, pp. 333–357. Springer, Berlin (2014)
27. Mäkelä, M.M., Karmitsa, N., Wilppu, O.: Proximal bundle method for nonsmooth and nonconvex multiobjective optimization. In: Tuovinen, T., Repin, S., Neittaanmäki, P. (eds.) Mathematical Modeling and Optimization of Complex Structures. Computational Methods in Applied Sciences, vol. 40, pp. 191–204. Springer, Berlin (2016)
28. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
29. Miettinen, K., Mäkelä, M.M.: Interactive bundle-based method for nondifferentiable multiobjective optimization: NIMBUS. Optimization **34**(3), 231–246 (1995)
30. Montonen, O., Joki, K.: Bundle-based descent method for nonsmooth multiobjective DC optimization with inequality constraints. J. Global Optim. **72**(3), 403–429 (2018)
31. Montonen, O., Karmitsa, N., Mäkelä, M.M.: Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization. Optimization **67**(1), 139–158 (2018)

32. Pham Dinh, T., Le Thi, H.A.: Convex analysis approach to DC programming: theory, algorithms and applications. Acta Math. Vietnam. **22**(1), 289–355 (1997)
33. Poirion, F., Mercier, Q., Désidéri, J.-A.: Descent algorithm for nonsmooth stochastic multiobjective optimization. Comput. Optim. Appl. **68**(2), 317–331 (2017)
34. Qu, S., Goh, M., Wu, S.-Y., De Souza, R.: Multiobjective DC programs with infinite convex constraints. J. Global Optim. **59**(1), 41–58 (2014)
35. Qu, S., Liu, C., Goh, M., Li, Y., Ji, Y.: Nonsmooth multiobjective programming with quasi-Newton methods. Eur. J. Oper. Res. **235**(3), 503–510 (2014)
36. Rangaiah, G.P.: Multi-Objective Optimization: Techniques and Applications in Chemical Engineering. Advances in Process Systems Engineering. World Scientific, Singapore (2009)
37. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (1992)
38. Sun, W.Y., Sampaio, R.J.B., Candido, M.A.B.: Proximal point algorithm for minimization of DC functions. J. Comput. Math. **21**(4), 451–462 (2003)
39. Taa, A.: Optimality conditions for vector optimization problems of a difference of convex mappings. J. Global Optim. **31**(3), 421–436 (2005)
40. Toland, J.F.: On subdifferential calculus and duality in nonconvex optimization. Mémoires de la Société Mathématique de France **60**, 177–183 (1979)
41. Wang, S.: Algorithms for Multiobjective and Nonsmooth Optimization. In: Kleinschmidt, P., Radermacher, F.J., Sweitzer, W., Wildermann, H. (eds.) Methods of Operations Research, 58, pp. 131–142. Athenaum Verlag, Frankfurt (1989)

Chapter 15
Mixed-Integer Linear Optimization: Primal–Dual Relations and Dual Subgradient and Cutting-Plane Methods

Ann-Brith Strömberg, Torbjörn Larsson, and Michael Patriksson

Abstract This chapter presents several solution methodologies for mixed-integer linear optimization, stated as mixed-binary optimization problems, by means of Lagrangian duals, subgradient optimization, cutting-planes, and recovery of primal solutions. It covers Lagrangian duality theory for mixed-binary linear optimization, a problem framework for which ultimate success—in most cases—is hard to accomplish, since strong duality cannot be inferred. First, a simple conditional subgradient optimization method for solving the dual problem is presented. Then, we show how ergodic sequences of Lagrangian subproblem solutions can be computed and used to recover mixed-binary primal solutions. We establish that the ergodic sequences accumulate at solutions to a convexified version of the original mixed-binary optimization problem. We also present a cutting-plane approach to the Lagrangian dual, which amounts to solving the convexified problem by Dantzig–Wolfe decomposition, as well as a two-phase method that benefits from the advantages of both subgradient optimization and Dantzig–Wolfe decomposition. Finally, we describe how the Lagrangian dual approach can be used to find near optimal solutions to mixed-binary optimization problems by utilizing the ergodic sequences in a Lagrangian heuristic, to construct a core problem, as well as to guide the branching in a branch-and-bound method. The chapter is concluded with a section comprising notes, references, historical downturns, and reading tips.

Keywords Mixed-binary linear optimization · Convexified problem · Lagrange dual · Non-smooth convex function · Subgradient method · Ergodic sequences · Cutting planes · Column generation · Dantzig–Wolfe decomposition · Core problem

A.-B. Strömberg (✉) · M. Patriksson
Chalmers University of Technology and the University of Gothenburg, Göteborg, Sweden
e-mail: anstr@chalmers.se; mipat@chalmers.se

T. Larsson
Linköping University, Linköping, Sweden
e-mail: torbjorn.larsson@liu.se

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_15

15.1 Introduction and Motivation

The aim of this chapter is to provide theory and methodology for Lagrangian dual approaches for solving mixed-integer linear optimization problems, when stated as mixed-binary problems.[1] It covers Lagrangian duality theory for mixed-binary linear optimization, generalizations of classical dual subgradient algorithms, cutting-plane methods, and (fractional) approximations of primal solutions from ergodic sequences, as well as the recovery of primal integer solutions. The chapter summarizes a research stream on subgradient methods for nondifferentiable optimization applications accomplished by the authors over two decades. While being based on a series of articles by the authors and co authors, some of the results presented here are, however, hitherto unpublished.

A strong motive for using Lagrangian dual methods in many applications of discrete and combinatorial optimization is that such problems may often be viewed as relatively easily solvable problems (being commonly highly structured) to which a number of complicating side constraints are added. An example is the (asymmetric) traveling salesperson problem, which can be viewed as that of finding a least cost trip assignment subject to subtour eliminating side constraints. In a Lagrangian dual method appropriate prices are assigned to the side constraints which then are included in the objective function. A solution to the resulting simpler problem yields a lower bound on the optimal value of the original problem, but does usually not fulfill the relaxed constraints. The prices are iteratively improved by means of some updating rule in an attempt to find prices such that the relaxed constraints become 'optimally fulfilled', that is, such that an optimal solution to the original problem is obtained. In discrete optimization, however, such prices usually do not exist.

Lagrangian dual methods are nevertheless increasingly popular tools in discrete optimization. Among their merits are their flexibility and applicability to many different problem structures and their ease of implementation. They also often produce higher lower bounds on the optimal value than does a continuous (linear programming) relaxation.[2] Lagrangian dual methods are most often used successfully in combination with other discrete optimization methodologies, such as branch-and-bound algorithms—within which they provide lower bounds —, local search methods, and primal heuristics. Among the latter, a popular combination is Lagrangian heuristics, which combine Lagrangian dual schemes with manipulations of primal infeasible solutions, aiming at producing near optimal and primal feasible solutions.

[1] Note that any mixed-integer optimization problem can be transformed into a mixed-binary optimization problem with a finite number of binary variables, provided that the feasible region for the original integer variables is bounded.

[2] One must add, however, that it requires solving a convex and nondifferentiable optimization problem, which may be quite nontrivial.

We consider a Lagrangian dual approach for solving mixed-binary linear optimization problems. We outline the properties of the Lagrangian dual problem, its relations to the primal mixed-binary problem, and its optimality conditions. Further, primal–dual optimality conditions are presented, both for a convexified primal problem and for the mixed-binary problem.

It is described how the dual problem can be solved by means of a simple conditional subgradient optimization method. For this general method, we provide a convergence result for an ergodic (averaged) sequence of solutions to the Lagrangian subproblems. The ergodic sequence is constructed in the same manner as done in [44, 60] for the case of convex optimization, and which establishes that the ergodic sequences in the limit produce optimal solutions to the original problem. Here, however, we establish that the sequences accumulate at solutions to a convexified version of the original mixed-binary optimization problem.

We further present a cutting-plane approach to the Lagrangian dual problem; it amounts to solving the convexified problem utilizing Dantzig–Wolfe decomposition, that is, column generation with the columns being associated with the solutions to the Lagrangian subproblems. Then we describe an approach to generate high quality initial columns, obtained from subproblem solutions in a subgradient optimization method applied—in a prediction phase—to the Lagrangian dual.

In the following section, we present a Lagrangian dual approach to find feasible and near optimal solutions to mixed-binary optimization problems utilizing

(1) a Lagrangian heuristic based on the ergodic sequences,
(2) a core problem, which is constructed based on information from the ergodic sequences, and
(3) the ergodic sequences to guide the branching in a branch-and-bound method.

The chapter is then concluded with an extensive section with notes, references, historical downturns and further reading tips.

15.2 Mixed-Binary Linear Optimization and Its Convexified Counterpart

We consider a general mixed-binary linear optimization problem. In our presentation and the derivation of methods to follow, the feasible set is described as the intersection of two sets. One set is characterized by general, explicit linear inequality constraints, which are to be *Lagrangian relaxed*. The other set is implicit and may be a Cartesian product set, resulting in one or several separable subproblems in the solution procedure(s); our description is general in that each subproblem may contain solely continuous, solely binary, or mixed variables.

Our mixed-binary linear optimization problem is defined as

$$z^* := \min_{\boldsymbol{x}_\mathrm{b}, \boldsymbol{x}_\mathrm{c}} \boldsymbol{c}_\mathrm{b}^\top \boldsymbol{x}_\mathrm{b} + \boldsymbol{c}_\mathrm{c}^\top \boldsymbol{x}_\mathrm{c} \tag{15.1a}$$

$$\text{subject to } A_\mathrm{b}\boldsymbol{x}_\mathrm{b} + A_\mathrm{c}\boldsymbol{x}_\mathrm{c} \geq \boldsymbol{b}, \tag{15.1b}$$

$$(\boldsymbol{x}_\mathrm{b}^\top, \boldsymbol{x}_\mathrm{c}^\top)^\top \in X, \tag{15.1c}$$

where z^* denotes the optimal value, $\boldsymbol{c}_\mathrm{b} \in \mathbb{R}^{n_\mathrm{b}}$, $\boldsymbol{c}_\mathrm{c} \in \mathbb{R}^{n_\mathrm{c}}$, $A_\mathrm{b} \in \mathbb{R}^{m \times n_\mathrm{b}}$, $A_\mathrm{c} \in \mathbb{R}^{m \times n_\mathrm{c}}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $n_\mathrm{b}, n_\mathrm{c}, m \in \mathbb{Z}_+$. The set $X := \left\{ (\boldsymbol{x}_\mathrm{b}^\top, \boldsymbol{x}_\mathrm{c}^\top)^\top \in \{0,1\}^{n_\mathrm{b}} \times \mathbb{R}_+^{n_\mathrm{c}} \mid D_\mathrm{b}\boldsymbol{x}_\mathrm{b} + D_\mathrm{c}\boldsymbol{x}_\mathrm{c} \geq \boldsymbol{d} \right\}$, where $D_\mathrm{b} \in \mathbb{R}^{k \times n_\mathrm{b}}$, $D_\mathrm{c} \in \mathbb{R}^{k \times n_\mathrm{c}}$, $\boldsymbol{d} \in \mathbb{R}^k$, and $k \in \mathbb{Z}_+$, is assumed to be nonempty and bounded. By defining $\boldsymbol{x} := (\boldsymbol{x}_\mathrm{b}^\top, \boldsymbol{x}_\mathrm{c}^\top)^\top$, $\boldsymbol{c} := (\boldsymbol{c}_\mathrm{b}^\top, \boldsymbol{c}_\mathrm{c}^\top)^\top$, $A := (A_\mathrm{b}, A_\mathrm{c})$, $D := (D_\mathrm{b}, D_\mathrm{c})$, and $n := n_\mathrm{b} + n_\mathrm{c}$, the optimization problem (15.1) can be equivalently expressed as[3]

$$z^* := \min_{\boldsymbol{x}} \boldsymbol{c}^\top \boldsymbol{x}, \tag{15.2a}$$

$$\text{subject to } A\boldsymbol{x} \geq \boldsymbol{b}, \tag{15.2b}$$

$$\boldsymbol{x} \in X, \tag{15.2c}$$

where $X = \left\{ \boldsymbol{x} \in \{0,1\}^{n_\mathrm{b}} \times \mathbb{R}_+^{n_\mathrm{c}} \mid D\boldsymbol{x} \geq \boldsymbol{d} \right\}$. We generally assume that the mixed-binary linear optimization problem is feasible, that is, that $\{ \boldsymbol{x} \in X \mid A\boldsymbol{x} \geq \boldsymbol{b} \} \neq \emptyset$ holds, and denote by $X^* := \operatorname{argmin}_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \mid A\boldsymbol{x} \geq \boldsymbol{b} \right\}$ its nonempty solution set.

By denoting an *extreme point* of the *convex hull* of the set X with $\boldsymbol{x}^q$ and letting $\mathcal{Q}$ be an index set for all such points, the convex hull can be expressed as

$$X_\mathrm{conv} := \operatorname{conv} X = \operatorname*{conv}_{q \in \mathcal{Q}} \left\{ \boldsymbol{x}^q \right\}. \tag{15.3}$$

Any extreme point to X_conv can be expressed as $\boldsymbol{x}^q = ((\boldsymbol{x}_\mathrm{b}^q)^\top, (\boldsymbol{x}_\mathrm{c}^q)^\top)^\top$, where $\boldsymbol{x}_\mathrm{b}^q \in \{0,1\}^{n_\mathrm{b}}$ and $\boldsymbol{x}_\mathrm{c}^q$ is an extreme point to the nonempty polyhedral set

$$X_\mathrm{c}(\boldsymbol{x}_\mathrm{b}^q) := \left\{ \boldsymbol{x}_\mathrm{c} \in \mathbb{R}_+^{n_\mathrm{c}} \;\middle|\; D_\mathrm{c}\boldsymbol{x}_\mathrm{c} \geq \boldsymbol{d} - D_\mathrm{b}\boldsymbol{x}_\mathrm{b}^q \right\}, \qquad q \in \mathcal{Q}.$$

The linear programming (LP) relaxation of the set X is expressed as

$$X_\mathrm{LP} := \left\{ (\boldsymbol{x}_\mathrm{b}^\top, \boldsymbol{x}_\mathrm{c}^\top)^\top \in [0,1]^{n_\mathrm{b}} \times \mathbb{R}_+^{n_\mathrm{c}} \mid D_\mathrm{b}\boldsymbol{x}_\mathrm{b} + D_\mathrm{c}\boldsymbol{x}_\mathrm{c} \geq \boldsymbol{d} \right\} \tag{15.4a}$$

$$= \left\{ \boldsymbol{x} \in [0,1]^{n_\mathrm{b}} \times \mathbb{R}_+^{n_\mathrm{c}} \mid D\boldsymbol{x} \geq \boldsymbol{d} \right\}. \tag{15.4b}$$

[3]The notation in (15.1) and (15.2) will be used interchangeably throughout this chapter.

It holds that $X \subseteq X_{\text{conv}} \subseteq X_{\text{LP}}$. Replacing the set X in (15.2c) with its convex hull X_{conv} results in the linear optimization problem defined as

$$z^*_{\text{conv}} := \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \;\middle|\; A\boldsymbol{x} \geq \boldsymbol{b};\ \boldsymbol{x} \in X_{\text{conv}} \right\} \leq z^*. \tag{15.5}$$

By assumption, the set $\{\, \boldsymbol{x} \in X_{\text{conv}} \,|\, A\boldsymbol{x} \geq \boldsymbol{b} \,\}$ is nonempty and bounded. We let X^*_{conv} denote the solution set to the optimization problem (15.5). Replacing the set X in (15.2c) by X_{LP} yields

$$z^*_{\text{LP}} := \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \;\middle|\; A\boldsymbol{x} \geq \boldsymbol{b};\ \boldsymbol{x} \in X_{\text{LP}} \right\} \leq z^*_{\text{conv}}, \tag{15.6}$$

and we let X^*_{LP} denote the solution set to the optimization problem (15.6).

For the case when the set X_{LP} possesses the *integrality property* with respect to the binary variables, that is, when all its extreme points have only integer valued variables $\boldsymbol{x}_{\text{b}}$, the equality $X_{\text{conv}} = X_{\text{LP}}$ holds, implying the equality $z^*_{\text{conv}} = z^*_{\text{LP}}$.

Remark 15.1 In many applications, the set X is a Cartesian product set, here denoted by $X := Y_1 \times Y_2 \times \ldots \times Y_S = \bigtimes_{s\in\mathcal{S}} Y_s$, where $\mathcal{S} = \{1, \ldots, S\}$. It is then assumed that each set $Y_s \subset \{0,1\}^{n_{\text{b},s}} \times \mathbb{R}^{n_{\text{c},s}}$ is defined over binary and/or continuous variables $\boldsymbol{y}_s$, such that $(\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_S^\top)^\top \equiv \boldsymbol{x}$, and such that the relations $n_{\text{b},s}, n_{\text{c},s} \in \mathbb{Z}_+$, $s \in \mathcal{S}$, $\sum_{s\in\mathcal{S}} n_{\text{b},s} = n_{\text{b}}$, and $\sum_{s\in\mathcal{S}} n_{\text{c},s} = n_{\text{c}}$ hold. The optimization problem (15.2) is then expressed as

$$z^* := \min_{\boldsymbol{y}_s, s\in\mathcal{S}} \sum_{s\in\mathcal{S}} \boldsymbol{c}_s^\top \boldsymbol{y}_s \tag{15.7a}$$

$$\text{subject to} \sum_{s\in\mathcal{S}} A_s \boldsymbol{y}_s \geq \boldsymbol{b}, \tag{15.7b}$$

$$\boldsymbol{y}_s \in Y_s,\ s \in \mathcal{S}, \tag{15.7c}$$

where $\boldsymbol{c}_s \in \mathbb{R}^{n_s}$, $A_s \in \mathbb{R}^{m\times n_s}$, and $n_s = n_{\text{b},s} + n_{\text{c},s}$, $s \in \mathcal{S}$. The constraints (15.7b) are said to be *coupling*, since relaxing them will result in a Lagrangian subproblem that separates into one minimization problem for each $s \in \mathcal{S}$. The integrality property may be considered for each of the sets Y_s, as needed/being relevant.

15.2.1 The Lagrangian Dual

The *Lagrange function* $L : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ with respect to the relaxation of the constraints (15.1b) by means of the *price* vector $\boldsymbol{u} \in \mathbb{R}^m_+$, also called *dual variables*

or (*Lagrangian* or *dual*) *multipliers*, is defined as

$$L(\boldsymbol{x}, \boldsymbol{u}) := \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{u}^\top (\boldsymbol{b} - A\boldsymbol{x}).$$

Its minimization over $\boldsymbol{x} \in X$ determines the *Lagrangian dual function* $h : \mathbb{R}^m \mapsto \mathbb{R}$, as defined by

$$\begin{aligned} h(\boldsymbol{u}) := \min_{\boldsymbol{x} \in X} L(\boldsymbol{x}, \boldsymbol{u}) &= \boldsymbol{b}^\top \boldsymbol{u} + \min_{\boldsymbol{x} \in X} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u}\right)^\top \boldsymbol{x} \right\} \\ &= \boldsymbol{b}^\top \boldsymbol{u} + \min_{q \in \mathcal{Q}} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u}\right)^\top \boldsymbol{x}^q \right\}. \end{aligned} \tag{15.8}$$

The function h is formed by the point wise minimum over $|\mathcal{Q}|$ affine functions, and it is therefore piecewise affine and concave, hence continuous but generally not everywhere differentiable. Letting $X(\boldsymbol{u})$ and $\mathcal{Q}(\boldsymbol{u})$ denote the optimal set to the respective inner minimization problem—the so-called *Lagrangian subproblem* or *Lagrangian relaxed problem*—in (15.8), the following relations hold:

$$X(\boldsymbol{u}) = \operatorname*{argmin}_{\boldsymbol{x} \in X} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u}\right)^\top \boldsymbol{x} \right\} = \left\{ \boldsymbol{x}^q \right\}_{q \in \mathcal{Q}(\boldsymbol{u})}; \tag{15.9a}$$

$$X_{\text{conv}}(\boldsymbol{u}) := \operatorname{conv} X(\boldsymbol{u}) = \operatorname*{argmin}_{\boldsymbol{x} \in X_{\text{conv}}} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u}\right)^\top \boldsymbol{x} \right\} = \operatorname*{conv}_{q \in \mathcal{Q}(\boldsymbol{u})} \left\{ \boldsymbol{x}^q \right\}. \tag{15.9b}$$

The expression for $X(\boldsymbol{u})$ in (15.9a) can always be replaced by its *convexified* version $X_{\text{conv}}(\boldsymbol{u})$ in (15.9b), since for any linear objective there is an optimal extreme point to the set X_{conv} that is also optimal with respect to the set X.

By *weak duality*, the inequality $h(\boldsymbol{u}) \leq \boldsymbol{c}^\top \boldsymbol{x}$ holds whenever $\boldsymbol{u} \in \mathbb{R}^m_+$ and $\boldsymbol{x} = (\boldsymbol{x}_{\text{b}}^\top, \boldsymbol{x}_{\text{c}}^\top)^\top$ is feasible in (15.2) [and, consequently, in (15.1)]. In order to find the best possible underestimate of z^*, the prices $\boldsymbol{u}$ should be chosen as to maximize the Lagrangian dual function, that is, to solve the *Lagrangian dual problem* defined as

$$h^* := \max_{\boldsymbol{u} \in \mathbb{R}^m_+} h(\boldsymbol{u}). \tag{15.10}$$

The problem (15.10) is a *convex* optimization problem having a concave and generally *nondifferentiable* objective function. By the assumption that the polyhedron $\{\, \boldsymbol{x} \in X_{\text{conv}} \mid A\boldsymbol{x} \geq \boldsymbol{b} \,\}$ is nonempty, also the optimal set of (15.10)—denoted U^*—is nonempty and polyhedral. Thus, by weak duality, the inequality $h^* \leq z^*$ holds. For most mixed-binary linear optimization problems, however, it holds that $h^* < z^*$, that is, the *duality gap* $z^* - h^*$ is nonzero.

Using the transformations in (15.8) along with a LP dualization, the Lagrangian dual (15.10) can be reformulated according to

$$h^* = \max_{\boldsymbol{u}} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + \min_{q \in \mathcal{Q}} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u} \right)^\top \boldsymbol{x}^q \right\} \,\middle|\, \boldsymbol{u} \in \mathbb{R}^m_+ \right\} \tag{15.11a}$$

$$= \max_{\boldsymbol{u},\, v} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + v \,\middle|\, \left(A\boldsymbol{x}^q \right)^\top \boldsymbol{u} + v \le \boldsymbol{c}^\top \boldsymbol{x}^q,\ q \in \mathcal{Q};\ \boldsymbol{u} \in \mathbb{R}^m_+;\ v \in \mathbb{R} \right\} \tag{15.11b}$$

$$= \min_{\boldsymbol{\lambda}} \left\{ \sum_{q \in \mathcal{Q}} \left(\boldsymbol{c}^\top \boldsymbol{x}^q \right) \lambda_q \,\middle|\, \sum_{q \in \mathcal{Q}} \left(A\boldsymbol{x}^q \right) \lambda_q \ge \boldsymbol{b};\ \sum_{q \in \mathcal{Q}} \lambda_q = 1;\ \lambda_q \ge 0,\ q \in \mathcal{Q} \right\} \tag{15.11c}$$

$$= \min_{\boldsymbol{x},\, \boldsymbol{\lambda}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \,\middle|\, A\boldsymbol{x} \ge \boldsymbol{b};\ \boldsymbol{x} = \sum_{q \in \mathcal{Q}} \lambda_q \boldsymbol{x}^q;\ \sum_{q \in \mathcal{Q}} \lambda_q = 1;\ \lambda_q \ge 0,\ q \in \mathcal{Q} \right\} \tag{15.11d}$$

$$= \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \,\middle|\, A\boldsymbol{x} \ge \boldsymbol{b};\ \boldsymbol{x} \in X_{\text{conv}} \right\} = z^*_{\text{conv}}. \tag{15.11e}$$

In summary, the above derivations of primal–dual connections imply the following *weak* and *strong duality* relations for the problems (15.1), (15.2), (15.5), (15.6), and (15.10):

$$z^* \ge z^*_{\text{conv}} = h^* \ge z^*_{\text{LP}}.$$

For typical—as well as the most interesting—applications of Lagrangian dualization the straightforward continuous relaxation of the set X does, however, not result in a set X_{LP} [defined in (15.4)] having integer extreme points. Hence, although the equality $z^*_{\text{conv}} = z^*_{\text{LP}}$ may hold, typically $X_{\text{conv}} \subset X_{\text{LP}}$ holds [X_{conv} being defined in (15.3)], which—in practice—most often implies the strict inequality $z^*_{\text{conv}} > z^*_{\text{LP}}$.

15.2.2 Optimality Conditions for the Convexified Problem

In order to derive optimality conditions for the convexified optimization problem (15.5) and, eventually, also for the original optimization problem (15.1) [or (15.2)], we first define the *subdifferential* of the concave function h at $\boldsymbol{u} \in \mathbb{R}^m$ as

$$\partial h(\boldsymbol{u}) := \left\{ \boldsymbol{\gamma} \in \mathbb{R}^m \,\middle|\, h(\boldsymbol{v}) \le h(\boldsymbol{u}) + \boldsymbol{\gamma}^\top (\boldsymbol{v} - \boldsymbol{u}),\ \boldsymbol{v} \in \mathbb{R}^m \right\},$$

the elements of which are called *subgradients*. The following characterization of the subdifferential holds for any Lagrangian dual objective function.

Proposition 15.1 (Subdifferential of the Dual Objective Function) *For each* $\boldsymbol{u} \in \mathbb{R}^m$*, it holds that* $\partial h(\boldsymbol{u}) = \{\, \boldsymbol{b} - A\boldsymbol{x} \mid \boldsymbol{x} \in X_{\mathrm{conv}}(\boldsymbol{u}) \,\} = \mathrm{conv}\{\, \boldsymbol{b} - A\boldsymbol{x}^q \mid q \in \mathcal{Q}(\boldsymbol{u}) \,\}$*. The function* h *is differentiable at* $\boldsymbol{u}$ *if and only if* $\partial h(\boldsymbol{u})$ *is a singleton set, that is, if and only if* $\boldsymbol{b} - A\boldsymbol{x}$ *is constant on* $X_{\mathrm{conv}}(\boldsymbol{u})$*, in which case* $\nabla h(\boldsymbol{u}) = \boldsymbol{b} - A\boldsymbol{x}$ *for any* $\boldsymbol{x} \in X_{\mathrm{conv}}(\boldsymbol{u})$*.*[4]

The *normal cone* to the set $\mathbb{R}^m_+$ at $\boldsymbol{u} \in \mathbb{R}^m$ is defined as

$$N_{\mathbb{R}^m_+}(\boldsymbol{u}) := \begin{cases} \left\{\, \boldsymbol{v} \in \mathbb{R}^m \mid \boldsymbol{v}^\top(\boldsymbol{v} - \boldsymbol{u}) \le 0,\ \boldsymbol{v} \in \mathbb{R}^m_+ \,\right\} & \\ \qquad\qquad = \left\{\, \boldsymbol{v} \in \mathbb{R}^m_- \mid \boldsymbol{v}^\top \boldsymbol{u} = 0 \,\right\}, & \boldsymbol{u} \in \mathbb{R}^m_+, \\ \emptyset\,, & \boldsymbol{u} \notin \mathbb{R}^m_+. \end{cases}$$

Letting $\boldsymbol{e}^i$ denote the i-th unit column, $N_{\mathbb{R}^m_+}(\boldsymbol{u}) = \mathrm{cone}\,\{\, -\boldsymbol{e}^i \mid u_i = 0,\ i \in \{\,1, \ldots, m\,\}\,\}$ holds for $\boldsymbol{u} \in \mathbb{R}^m_+$. The *conditional subdifferential* of h at $\boldsymbol{u} \in \mathbb{R}^m_+$, the elements of which will be referred to as *conditional subgradients*, is in our Lagrangian dual setting then defined as

$$\begin{aligned} \partial^{\mathbb{R}^m_+} h(\boldsymbol{u}) :&= \left\{\, \boldsymbol{\gamma} \in \mathbb{R}^m \;\middle|\; h(\boldsymbol{v}) \le h(\boldsymbol{u}) + \boldsymbol{\gamma}^\top(\boldsymbol{v} - \boldsymbol{u}),\ \boldsymbol{v} \in \mathbb{R}^m_+ \,\right\} \\ &= \partial h(\boldsymbol{u}) - N_{\mathbb{R}^m_+}(\boldsymbol{u}) \qquad (15.12) \\ &= \mathrm{conv}\left\{\, \boldsymbol{b} - A\boldsymbol{x}^q \mid q \in \mathcal{Q}(\boldsymbol{u}) \right\} + \mathrm{cone}\left\{\, \boldsymbol{e}^i \mid u_i = 0,\ i \in \{\,1, \ldots, m\,\} \right\}. \end{aligned}$$

Clearly, $\partial^{\mathbb{R}^m_+} h(\boldsymbol{u}) \supseteq \partial h(\boldsymbol{u})$ holds for all $\boldsymbol{u} \in \mathbb{R}^m_+$. The next proposition is immediate.

Proposition 15.2 (Properties of the Conditional Subdifferential) *The conditional subdifferential* $\partial^{\mathbb{R}^m_+} h(\boldsymbol{u})$ *is nonempty, closed and convex for all* $\boldsymbol{u} \in \mathbb{R}^m_+$*. Further,* $\partial^{\mathbb{R}^m_+} h(\boldsymbol{u})$ *is unbounded whenever* $\boldsymbol{u} \in \mathrm{bd}\,\mathbb{R}^m_+$*.*

Proposition 15.3 (Properties of the Lagrangian Dual) *The following statements are equivalent.*

(i) *the Lagrangian dual problem* (15.10) *has a bounded optimal solution;*
(ii) $\boldsymbol{0} \in \bigcup_{\boldsymbol{u} \in \mathbb{R}^m_+} \partial^{\mathbb{R}^m_+} h(\boldsymbol{u})$*;*
(iii) $\{\, \boldsymbol{x} \in X_{\mathrm{conv}} \mid A\boldsymbol{x} \ge \boldsymbol{b} \,\} \ne \emptyset$*.*

The optimality conditions for the Lagrangian dual (15.10) are expressed as follows.

Proposition 15.4 (Optimality Conditions for the Lagrangian Dual Problem) *A dual vector* $\boldsymbol{u} \in \mathbb{R}^m_+$ *is optimal, that is,* $\boldsymbol{u} \in U^*$*, if and only if* $\partial h(\boldsymbol{u}) - N_{\mathbb{R}^m_+}(\boldsymbol{u}) \ni \boldsymbol{0}$*, or equivalently,* $\partial h(\boldsymbol{u}) \cap N_{\mathbb{R}^m_+}(\boldsymbol{u}) \ne \emptyset$*, that is, if and only if there exists a* $\boldsymbol{\gamma} \in \partial h(\boldsymbol{u})$ *such that* $\boldsymbol{\gamma} \le \boldsymbol{0}$ *and* $\boldsymbol{u}^\top \boldsymbol{\gamma} = 0$ *hold.*

[4] While Proposition 15.1 implies that the function h is differentiable at $\boldsymbol{u} \in \mathbb{R}^m$ if the set $X_{\mathrm{conv}}(\boldsymbol{u})$ is a singleton, the opposite, however, does not hold in general.

Further, subgradients that verify the optimality of dual solutions correspond to optimal solutions to the convexified primal problem (15.11e), as expressed below.

Proposition 15.5 (Primal–Dual Optimality Conditions for the Convexified Problem) *Let* $(\boldsymbol{x}, \boldsymbol{u}) \in X_{\text{conv}} \times \mathbb{R}^m_+$. *Then* $(\boldsymbol{x}, \boldsymbol{u}) \in X^*_{\text{conv}} \times U^*$ *if and only if* $\boldsymbol{x} \in X_{\text{conv}}(\boldsymbol{u})$ *and* $\boldsymbol{b} - A\boldsymbol{x} \in \partial h(\boldsymbol{u}) \cap N_{\mathbb{R}^m_+}(\boldsymbol{u})$ *hold, that is, if and only if* $\boldsymbol{x} \in X_{\text{conv}}(\boldsymbol{u})$, $A\boldsymbol{x} \geq \boldsymbol{b}$, *and* $\boldsymbol{u}^\top(\boldsymbol{b} - A\boldsymbol{x}) = 0$ *hold.*

As is well known, a primal–dual pair $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}}) \in X_{\text{conv}} \times \mathbb{R}^m_+$ satisfies the conditions of the proposition exactly when it is a *saddle point* of the Lagrange function L with respect to $X_{\text{conv}} \times \mathbb{R}^m_+$:

$$L(\overline{\boldsymbol{x}}, \boldsymbol{u}) \leq L(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}}) \leq L(\boldsymbol{x}, \overline{\boldsymbol{u}}), \qquad \boldsymbol{u} \in \mathbb{R}^m_+, \quad \boldsymbol{x} \in X_{\text{conv}}. \tag{15.13}$$

The mapping $\partial h \cap N_{\mathbb{R}^m_+}$ is constant on the dual solution set U^*. Hence, irrespective of the choice of dual solution $\boldsymbol{u}^* \in U^*$ the solution set to the primal problem (15.11e) may be expressed as

$$X^*_{\text{conv}} = \left\{ \boldsymbol{x} \in X_{\text{conv}}(\boldsymbol{u}^*) \;\middle|\; A\boldsymbol{x} \geq \boldsymbol{b};\ \left(\boldsymbol{u}^*\right)^\top (A\boldsymbol{x} - \boldsymbol{b}) = 0 \right\}. \tag{15.14}$$

In the typical situation, the subproblem solution set $X_{\text{conv}}(\boldsymbol{u}^*)$ is *not* a singleton, the dual objective function is nonsmooth on U^*, and finding a subgradient that verifies dual optimality is computationally expensive. Further, since not all points in $X_{\text{conv}}(\boldsymbol{u}^*)$ are optimal in the primal problem,[5] finding a point $\boldsymbol{x} \in X^*_{\text{conv}}$ is nontrivial; this phenomenon is referred to as *non-coordinability*, and is relevant both when the original problem is an LP or a mixed-binary linear optimization problem.

15.2.3 Conditions for Optimality and Near Optimality of Mixed-Binary Linear Optimization Problems

The optimality conditions in Proposition 15.5 and the characterization in (15.14) can be constructed because of the convexity of the problem (15.5), which yields that strong duality, that is, $h^* = z^*_{\text{conv}}$ holds. For the *nonconvex* mixed-binary linear optimization problem (15.2), for which $h^* \leq z^*$ (and typically $h^* < z^*$) holds, these conditions can be generalized through a well-defined relaxation.

[5] Usually, some points in $X_{\text{conv}}(\boldsymbol{u}^*)$ are infeasible in (15.2), while others are feasible but non-optimal.

Proposition 15.6 (Primal–Dual Optimality Conditions for Mixed-Binary Linear Optimization) *Let* $(\boldsymbol{x}, \boldsymbol{u}) \in X \times \mathbb{R}^m_+$. *Then* $(\boldsymbol{x}, \boldsymbol{u}) \in X^* \times U^*$ *if and only if the following system is consistent:*

$$A\boldsymbol{x} \geq \boldsymbol{b}, \tag{15.15a}$$

$$\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{u}^\top (\boldsymbol{b} - A\boldsymbol{x}) \leq h(\boldsymbol{u}) + \varepsilon, \tag{15.15b}$$

$$-\boldsymbol{u}^\top (\boldsymbol{b} - A\boldsymbol{x}) \leq \delta, \tag{15.15c}$$

$$\varepsilon + \delta \leq z^* - h^*, \tag{15.15d}$$

$$\varepsilon, \delta \geq 0. \tag{15.15e}$$

The conditions stated in Proposition 15.6 characterize primal–dual optimal solutions through five properties:

(i) primal feasibility [$\boldsymbol{x} \in X$ and (15.15a)];
(ii) dual feasibility [$\boldsymbol{u} \in \mathbb{R}^m_+$];
(iii) Lagrangian ε-optimality [(15.15b)];
(iv) a relaxed (δ-)complementarity [(15.15c)];
(v) a bounded sum of nonnegative perturbations [(15.15d)–(15.15e)].

The combination of the inequalities in (15.15) with the condition $(\boldsymbol{x}, \boldsymbol{u}) \in X \times \mathbb{R}^m_+$ leads to the equality $\varepsilon + \delta = z^* - h^*$ being fulfilled. Further, the system (15.15) is consistent for the primal–dual pair $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}}) \in X \times \mathbb{R}^m_+$ if and only if the inequalities

$$L(\overline{\boldsymbol{x}}, \boldsymbol{u}) - (z^* - h^*) + \varepsilon \;\leq\; L(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}}) \;\leq\; L(\boldsymbol{x}, \overline{\boldsymbol{u}}) + \varepsilon \tag{15.16}$$

hold for all $\boldsymbol{u} \in \mathbb{R}^m_+$ and all $\boldsymbol{x} \in X$, and for some $\varepsilon \in [0, z^* - h^*]$, meaning that $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}})$ is a *near saddle point* to the Lagrange function L. For a zero duality gap, that is, when $z^* = h^*$, the conditions (15.15) reduce to those stated in Proposition 15.5 while the near saddle point property (15.16) reduces to that of an ordinary saddle point, that is, (15.13).

Using the conditions (15.15) and an optimal Lagrangian dual solution $\boldsymbol{u}^* \in U^*$, the optimal set of the mixed-binary linear optimization problem (15.2) is given by

$$X^* = \bigcup_{\substack{\delta = z^* - h^* - \varepsilon \\ \varepsilon \in [0,\, z^* - h^*]}} \left\{ \boldsymbol{x} \in X \;\middle|\; A\boldsymbol{x} \geq \boldsymbol{b};\; \boldsymbol{c}^\top \boldsymbol{x} - h^* - \varepsilon \leq -(\boldsymbol{u}^*)^\top (\boldsymbol{b} - A\boldsymbol{x}) \leq \delta \right\}. \tag{15.17}$$

We define, for any $\boldsymbol{u} \in \mathbb{R}^m_+$, the sets of ε-optimal [cf. (15.9a)] and δ-complementary solutions to the Lagrangian subproblem by the following expressions:

$$X^{\text{opt}}_\varepsilon(\boldsymbol{u}) := \left\{ \boldsymbol{x} \in X \;\middle|\; \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{u}^\top (\boldsymbol{b} - A\boldsymbol{x}) \leq h(\boldsymbol{u}) + \varepsilon \right\}, \qquad \varepsilon \geq 0,$$

$$X_{\delta}^{\text{comp}}(\boldsymbol{u}) := \left\{ \boldsymbol{x} \in X \,\middle|\, -\boldsymbol{u}^{\top}(\boldsymbol{b} - A\boldsymbol{x}) \leq \delta \right\}, \qquad \delta \geq 0.$$

The set X^*, as formulated in (15.17), can then for any $\boldsymbol{u}^* \in U^*$ be expressed as

$$X^* = \bigcup_{\varepsilon \in [0,\, z^* - h^*]} \left\{ \boldsymbol{x} \in X_{\varepsilon}^{\text{opt}}(\boldsymbol{u}^*) \,\middle|\, A\boldsymbol{x} \geq \boldsymbol{b};\ -(\boldsymbol{u}^*)^{\top}(\boldsymbol{b} - A\boldsymbol{x}) \leq z^* - h^* - \varepsilon \right\} \tag{15.18a}$$

$$= \bigcup_{\varepsilon \in [0,\, z^* - h^*]} \left\{ \boldsymbol{x} \in X_{\varepsilon}^{\text{opt}}(\boldsymbol{u}^*) \cap X_{z^* - h^* - \varepsilon}^{\text{comp}}(\boldsymbol{u}^*) \,\middle|\, A\boldsymbol{x} \geq \boldsymbol{b} \right\}, \tag{15.18b}$$

where (15.18a) is derived as a generalization of (15.14).

Given an optimal dual solution, the conditions in the expression (15.14) can, in principle, be used to calculate an optimal solution to the convexified problem (15.5). In contrast, given an optimal dual solution, the conditions in the expressions (15.17) and (15.18) are in general not instrumental for finding optimal solutions to the problem (15.2), because the size of the duality gap $z^* - h^*$ is unknown.

The characterizations (15.17) and (15.18) can be generalized to allow for non-optimal choices of $\boldsymbol{u} \in \mathbb{R}_+^m$ and to describe near optimal solutions to the problem (15.2). For this purpose, we introduce the functions $\varepsilon, \delta : X \times \mathbb{R}_+^m \mapsto \mathbb{R}$, defined by

$$\varepsilon(\boldsymbol{x}, \boldsymbol{u}) := \boldsymbol{c}^{\top}\boldsymbol{x} + \boldsymbol{u}^{\top}(\boldsymbol{b} - A\boldsymbol{x}) - h(\boldsymbol{u}), \tag{15.19a}$$

and

$$\delta(\boldsymbol{x}, \boldsymbol{u}) := -\boldsymbol{u}^{\top}(\boldsymbol{b} - A\boldsymbol{x}). \tag{15.19b}$$

It holds that $\varepsilon(\boldsymbol{x}, \boldsymbol{u}) \geq 0$ when $\boldsymbol{x} \in X$, and that $\delta(\boldsymbol{x}, \boldsymbol{u}) \geq 0$ when $\boldsymbol{u} \in \mathbb{R}_+^m$ and $A\boldsymbol{x} \geq \boldsymbol{b}$. For any primal–dual pair $(\boldsymbol{x}, \boldsymbol{u}) \in X \times \mathbb{R}_+^m$, (15.19) can be used to characterize the ε-optimality and the δ-complementarity of a solution $\boldsymbol{x}^{\text{appr}}(\boldsymbol{u})$ that is near optimal in the Lagrangian subproblem (15.9a) and feasible in the primal problem (15.2), that is, values of $\varepsilon \geq 0$ and $\delta \geq 0$ such that $\boldsymbol{x}^{\text{appr}}(\boldsymbol{u})$ is included in the sets $X_{\varepsilon}^{\text{opt}}(\boldsymbol{u}) = \{ \boldsymbol{x} \in X \mid \varepsilon(\boldsymbol{x}, \boldsymbol{u}) \leq \varepsilon \}$ and $X_{\delta}^{\text{comp}}(\boldsymbol{u}) = \{ \boldsymbol{x} \in X \mid \delta(\boldsymbol{x}, \boldsymbol{u}) \leq \delta \}$, respectively. Then, for $\boldsymbol{u} \in \mathbb{R}_+^m$ and $\beta \geq 0$, the set of β-optimal solutions to the problem (15.2) can be expressed as

$$X_{\beta}^* := \left\{ \boldsymbol{x} \in X \,\middle|\, A\boldsymbol{x} \geq \boldsymbol{b};\ \boldsymbol{c}^{\top}\boldsymbol{x} \leq z^* + \beta \right\} \tag{15.20a}$$

$$= \left\{ \boldsymbol{x} \in X \,\middle|\, A\boldsymbol{x} \geq \boldsymbol{b};\ \varepsilon(\boldsymbol{x}, \boldsymbol{u}) + \delta(\boldsymbol{x}, \boldsymbol{u}) \leq z^* - h(\boldsymbol{u}) + \beta \right\} \tag{15.20b}$$

$$= \left\{ \boldsymbol{x} \in X_{\varepsilon}^{\text{opt}}(\boldsymbol{u}) \cap X_{\delta}^{\text{comp}}(\boldsymbol{u}) \,\middle|\, A\boldsymbol{x} \geq \boldsymbol{b};\ \varepsilon + \delta \leq z^* - h(\boldsymbol{u}) + \beta;\ \varepsilon, \delta \geq 0 \right\}. \tag{15.20c}$$

In particular, it holds that $X_0^* = X^*$. Further, for $\beta = 0$ and any $\boldsymbol{u} \in U^*$, the characterizations (15.17)–(15.18) are recovered.

Neither of the characterizations in (15.20) are practically useful for solving the problem (15.2), since the optimal value z^* is unknown, but it suggests a heuristic principle for searching for near optimal solutions. Given any value of $\boldsymbol{u} \in \mathbb{R}_+^m$ [preferably being (near) optimal in the Lagrangian dual (15.10)], a point $\boldsymbol{x} \in X$ such that $A\boldsymbol{x} \geq \boldsymbol{b}$ holds and such that the values of $\varepsilon(\boldsymbol{x}, \boldsymbol{u})$ and $\delta(\boldsymbol{x}, \boldsymbol{u})$ are both small, should be sought, that is, a primal feasible solution that is ε-optimal and δ-complementary, with small values of $\varepsilon \geq 0$ and $\delta \geq 0$. A natural starting point for such a heuristic search is a Lagrangian subproblem solution $\boldsymbol{x}(\boldsymbol{u}) \in X_0^{\text{opt}}(\boldsymbol{u}) = X(\boldsymbol{u})$ [for which $\varepsilon(\boldsymbol{x}(\boldsymbol{u}), \boldsymbol{u}) = 0$ holds], and the search should typically be restricted to the set X. The search should then strive for primal feasibility with respect to $A\boldsymbol{x} \geq \boldsymbol{b}$, while maintaining small values of $\varepsilon(\boldsymbol{x}, \boldsymbol{u})$ and $\delta(\boldsymbol{x}, \boldsymbol{u})$. This heuristic principle has shown to be effective for finding near optimal solutions for applications where the problem (15.2) possesses specific structures that can be exploited in a search which strive for primal feasibility.

Remark 15.2 In the case when the constraints (15.2b) are instead equalities, $\delta(\boldsymbol{x}, \boldsymbol{u}) = 0$ holds for any primal feasible solution. The characterization (15.20b) then reduces to

$$X_\beta^* = \left\{ \boldsymbol{x} \in X \;\middle|\; A\boldsymbol{x} = \boldsymbol{b};\ \varepsilon(\boldsymbol{x}, \boldsymbol{u}) \leq z^* - h(\boldsymbol{u}) + \beta \right\},$$

that is, primal feasibility and ε-optimality in the Lagrangian relaxed problem. In particular, then $X^* = \{ \boldsymbol{x} \in X \mid A\boldsymbol{x} = \boldsymbol{b};\ \varepsilon(\boldsymbol{x}, \boldsymbol{u}) \leq z^* - h(\boldsymbol{u}) \}$. It follows that $A\boldsymbol{x} \neq \boldsymbol{b}$ must hold for any $\boldsymbol{x} \in X$ such that $\varepsilon(\boldsymbol{x}, \boldsymbol{u}) < z^* - h(\boldsymbol{u})$. Hence, a solution $\boldsymbol{x} \in X$ is optimal in (15.2) if and only if it is the most near optimal solution in the Lagrangian subproblem that is fulfills the relaxed constraints. This observation implies that (15.2) can be solved by enumerating elements of X with respect to increasing values of $\varepsilon(\boldsymbol{x}, \boldsymbol{u})$, whence the first one found that fulfills $A\boldsymbol{x} = \boldsymbol{b}$ is therefore also optimal.

For the case of inequality constraints in the problem (15.2), only

$$X^* \subseteq \left\{ \boldsymbol{x} \in X \;\middle|\; A\boldsymbol{x} \geq \boldsymbol{b};\ \varepsilon(\boldsymbol{x}, \boldsymbol{u}) \leq z^* - h(\boldsymbol{u}) \right\}$$

holds, and there is no guarantee that a feasible solution with a minimal value of $\varepsilon(\boldsymbol{x}, \boldsymbol{u})$ is optimal, since the corresponding value of $\delta(\boldsymbol{x}, \boldsymbol{u})$ may be large [while an optimal solution may have a large value of $\varepsilon(\boldsymbol{x}, \boldsymbol{u})$ and a small value of $\delta(\boldsymbol{x}, \boldsymbol{u})$]. If, however, an upper bound $\overline{z} \geq z^*$ is at hand, by enumerating *all* elements in the set $\{ \boldsymbol{x} \in X \mid A\boldsymbol{x} \geq \boldsymbol{b};\ \varepsilon(\boldsymbol{x}, \boldsymbol{u}) \leq \overline{z} - h(\boldsymbol{u}) \}$ an optimal solution to (15.2) will be found.

15.3 Conditional Subgradient Optimization

This section presents a generalization of subgradient optimization for solving a Lagrangian dual problem to utilize *conditional subgradients*. We discuss a particular choice of conditional subgradients, obtained through Euclidean projections, which leads to an easily implementable modification of traditional subgradient optimization schemes. Computational experiments have shown that the resulting subgradient projection method performs better than traditional subgradient optimization; in some cases the difference is considerable. This generalization of subgradient optimization is especially advantageous in the context of Lagrangian duals possessing many nonnegativity constraints, onto which Euclidean projections are simple. The section further presents a computationally cheap scheme for generating ergodic sequences of subproblem solutions and which is shown to converge to the optimal set of the convexified problem (15.5). This scheme is then enhanced, in terms of faster convergence to the optimal set. The section is then concluded by a heuristic scheme for the finite generation of feasible solutions that are ε-optimal, for an arbitrary $\varepsilon > 0$.

15.3.1 Basic Convergence in the Lagrangian Dual Problem

We consider solving the Lagrangian dual problem (15.10) by the *conditional subgradient optimization* method, which is given by the following. Choose a starting solution $\boldsymbol{u}^0 \in \mathbb{R}^m_+$ and compute iterates $\boldsymbol{u}^t$, $t = 0, 1, \ldots$, according to

$$\boldsymbol{u}^{t+\frac{1}{2}} = \boldsymbol{u}^t + \alpha_t\big(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t)\big), \quad \boldsymbol{u}^{t+1} = \big[\boldsymbol{u}^{t+\frac{1}{2}}\big]_+, \tag{15.21}$$

where $\boldsymbol{x}(\boldsymbol{u}^t) \in X(\boldsymbol{u}^t)$ solves the Lagrangian subproblem in (15.9) at $\boldsymbol{u}^t \in \mathbb{R}^m_+$, so that $\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) \in \partial h(\boldsymbol{u}^t)$ is a subgradient to h at $\boldsymbol{u}^t$, $\boldsymbol{v}(\boldsymbol{u}^t) \in N_{\mathbb{R}^m_+}(\boldsymbol{u}^t)$ is an element of the normal cone of $\mathbb{R}^m_+$ at $\boldsymbol{u}^t$, $\alpha_t > 0$ is the step length chosen at iteration t, and $[\cdot]_+$ denotes the Euclidean projection onto the nonnegative orthant $\mathbb{R}^m_+$. Note that $\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \in \partial^{\mathbb{R}^m_+} h(\boldsymbol{u}^t)$, that is, the step direction belongs to the conditional subdifferential, defined in (15.12).

If $\{\, \boldsymbol{v}(\boldsymbol{u}^t) \,\} := \{\boldsymbol{0}\}$, then the method (15.21) reduces to the traditional subgradient optimization method.

The choices $\boldsymbol{v}(\boldsymbol{u}^t) := \operatorname{argmin}_{\boldsymbol{v} \in N_{\mathbb{R}^m_+}(\boldsymbol{u}^t)} \| \boldsymbol{v} - (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t)) \|$, where $\| \cdot \|$ denotes the Euclidean norm, define a special case of the method (15.21) called the *subgradient projection method*, which uses a *feasible direction* from every $\boldsymbol{u}^t \in \mathbb{R}^m_+$, as

$$b_i - \boldsymbol{A}_i\boldsymbol{x}(\boldsymbol{u}^t) - v_i(\boldsymbol{u}^t) = \begin{cases} \big[b_i - \boldsymbol{A}_i\boldsymbol{x}(\boldsymbol{u}^t)\big]_+, & \text{if } u_i^t = 0, \\ b_i - \boldsymbol{A}_i\boldsymbol{x}(\boldsymbol{u}^t), & \text{if } u_i^t > 0, \end{cases} \quad i = 1, \ldots, m, \tag{15.22}$$

here $b_i/\boldsymbol{A}_i/v_i(\cdot)$ denotes the i-th element/row of the vector/matrix $\boldsymbol{b}/A/\boldsymbol{v}(\cdot)$.

Due to the nondifferentiability of the Lagrangian dual objective function, subgradient based optimization methods cannot rely on a one dimensional maximization for determining the step length in each iteration. Instead, to ensure convergence of the sequence of dual iterates to optimality, the step lengths must be computed according to a (typically predefined) rule. We next present convergence results for the method (15.21) under different step length rules. The first assures that the sequence of dual iterates tends to the set of solutions to the Lagrangian dual.

Theorem 15.1 (Convergence to the Solution Set by Divergent Series Step Lengths) *Let the method* (15.21) *be applied to the problem* (15.10)*, with the sequence* $\{\alpha_t\}$ *of step lengths fulfilling the divergent series conditions*

$$\alpha_t > 0, \qquad t = 0, 1, 2, \ldots, \tag{15.23a}$$

$$\alpha_t \to 0, \qquad as \quad t \to \infty, \tag{15.23b}$$

and

$$\left\{ \sum_{s=0}^{t-1} \alpha_s \right\} \to \infty, \qquad as \quad t \to \infty. \tag{15.23c}$$

If the sequence $\{\boldsymbol{v}(\boldsymbol{u}^t)\}$ *is bounded then it holds that* $\{h(\boldsymbol{u}^t)\} \to h^*$ *and* $\left\{ \min_{\boldsymbol{u} \in U^*} \| \boldsymbol{u} - \boldsymbol{u}^t \| \right\} \to 0$ *as* $t \to \infty$.

Proof We show that the iterates will eventually belong to an arbitrarily small neighbourhood of the set of solutions to (15.10).

Let $\delta > 0$ be arbitrary and define $B^\delta := \{ \boldsymbol{u} \in \mathbb{R}^m \mid \|\boldsymbol{u}\| \leq \delta \}$. Since the function h is piecewise affine and concave, the set $\mathbb{R}^m_+$ is nonempty, closed and convex, and the set U^* is nonempty and polyhedral, there exists an $\varepsilon > 0$ such that the level set $U^\varepsilon := \{ \boldsymbol{u} \in \mathbb{R}^m_+ \mid h(\boldsymbol{u}) \geq h^* - \varepsilon \}$ fulfills $U^\varepsilon \subseteq U^* + B^{\delta/2}$. Further, the sequence $\{ \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \}$ is bounded and $\alpha_t \to 0$. Hence, there exists an N_δ such that $\alpha_t \| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \|^2 \leq \varepsilon$ and $\alpha_t \| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \| \leq \delta/2$ for all $t \geq N_\delta$.

The sequel of the proof is based on induction. First we show that there exists a $t_\delta \geq N_\delta$ such that $\boldsymbol{u}^{t_\delta} \in U^* + B^\delta$. Then, we establish that if the inclusion $\boldsymbol{u}^t \in U^* + B^\delta$ holds for some value $t \geq N_\delta$, then also $\boldsymbol{u}^{t+1} \in U^* + B^\delta$.

For an arbitrary $\boldsymbol{u}^* \in U^*$, in each iteration t of the method (15.21) the relations

$$\begin{aligned} \left\| \boldsymbol{u}^* - \boldsymbol{u}^{t+1} \right\|^2 &= \left\| \boldsymbol{u}^* - \left[\boldsymbol{u}^t + \alpha_t \big(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \big) \right]_+ \right\|^2 \\ &\leq \left\| \boldsymbol{u}^* - \boldsymbol{u}^t - \alpha_t \big(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \big) \right\|^2 \\ &= \left\| \boldsymbol{u}^* - \boldsymbol{u}^t \right\|^2 - 2\alpha_t \big(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \big)^\top \big(\boldsymbol{u}^* - \boldsymbol{u}^t \big) \\ &\quad + \alpha_t^2 \left\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \right\|^2 \end{aligned} \tag{15.24}$$

hold, where the inequality follows from the projection property.[6] Now suppose that, for all $s \geq N_\delta$, the inequality

$$2\left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s)\right)^\top \left(\boldsymbol{u}^* - \boldsymbol{u}^s\right) - \alpha_s \left\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s)\right\|^2 > \varepsilon \tag{15.25}$$

holds. Then by telescoping (15.24), we obtain that for every $t \geq N_\delta$, the inequality

$$\left\| \boldsymbol{u}^* - \boldsymbol{u}^{t+1} \right\|^2 < \left\| \boldsymbol{u}^* - \boldsymbol{u}^{N_\delta} \right\|^2 - \varepsilon \sum_{s=N_\delta}^{t} \alpha_s,$$

holds. Then, from (15.23c) it follows that the right-hand-side of this inequality tends to $-\infty$ as $t \to \infty$, which is clearly impossible. Therefore, the inequality

$$2\left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t)\right)^\top \left(\boldsymbol{u}^* - \boldsymbol{u}^t\right) - \alpha_t \left\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \right\|^2 \leq \varepsilon \tag{15.26}$$

must hold for at least one iteration $t = t_\delta \geq N_\delta$. By definition of N_δ the inequality $\alpha_{t_\delta} \| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^{t_\delta}) - \boldsymbol{v}(\boldsymbol{u}^{t_\delta}) \|^2 \leq \varepsilon$ holds, which together with (15.26) implies the inequality $(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^{t_\delta}) - \boldsymbol{v}(\boldsymbol{u}^{t_\delta}))^\top (\boldsymbol{u}^* - \boldsymbol{u}^{t_\delta}) \leq \varepsilon$. Since $\boldsymbol{u}^*, \boldsymbol{u}^{t_\delta} \in \mathbb{R}^m_+$, the definition (15.12) implies the inequality $h(\boldsymbol{u}^*) - h(\boldsymbol{u}^{t_\delta}) \leq (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^{t_\delta}) - \boldsymbol{v}(\boldsymbol{u}^{t_\delta}))^\top (\boldsymbol{u}^* - \boldsymbol{u}^{t_\delta})$. Hence, it holds that $h(\boldsymbol{u}^{t_\delta}) \geq h^* - \varepsilon$, that is, $\boldsymbol{u}^{t_\delta} \in U^\varepsilon \subseteq U^* + B^{\delta/2} \subset U^* + B^\delta$.

Now, suppose that $\boldsymbol{u}^t \in U^* + B^\delta$ for some $t \geq N_\delta$. If (15.25) holds, then (15.24) implies the inequality $\| \boldsymbol{u}^* - \boldsymbol{u}^{t+1} \| < \| \boldsymbol{u}^* - \boldsymbol{u}^t \|$ for any $\boldsymbol{u}^* \in U^*$. Defining the Euclidean projection of $\boldsymbol{u}^t$ onto U^* as $\overline{\boldsymbol{u}}^t_{\text{proj}} := \operatorname{argmin}_{\boldsymbol{u} \in U^*} \|\boldsymbol{u} - \boldsymbol{u}^t\|$ then yields the inequalities

$$\left\| \overline{\boldsymbol{u}}^{t+1}_{\text{proj}} - \boldsymbol{u}^{t+1} \right\| \leq \left\| \overline{\boldsymbol{u}}^{t}_{\text{proj}} - \boldsymbol{u}^{t+1} \right\| < \left\| \overline{\boldsymbol{u}}^{t}_{\text{proj}} - \boldsymbol{u}^{t} \right\| \leq \delta, \qquad t = 0, 1, \ldots,$$

which imply the inclusion $\boldsymbol{u}^{t+1} \in U^* + B^\delta$. Otherwise, (15.26) must hold and, using the same arguments as above, we obtain the inequality $h(\boldsymbol{u}^t) \geq h^* - \varepsilon$, that is, $\boldsymbol{u}^t \in U^\varepsilon \subseteq U^* + B^{\delta/2}$. Since the relations

$$\begin{aligned} \left\| \boldsymbol{u}^{t+1} - \boldsymbol{u}^t \right\| &= \left\| \left[\boldsymbol{u}^t + \alpha_t \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t)\right)\right]_+ - \boldsymbol{u}^t \right\| \\ &\leq \left\| \boldsymbol{u}^t + \alpha_t \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t)\right) - \boldsymbol{u}^t \right\| \\ &= \alpha_t \left\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \right\| \leq \frac{\delta}{2} \end{aligned}$$

hold whenever $t \geq N_\delta$, it follows that $\boldsymbol{u}^{t+1} \in U^* + B^{\delta/2} + B^{\delta/2} = U^* + B^\delta$. We conclude that, in either case, whenever $t \geq N_\delta$, $\boldsymbol{u}^t \in U^* + B^\delta$ implies that $\boldsymbol{u}^{t+1} \in U^* + B^\delta$.

[6] The projection property states that for any vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^m$ and any convex set $U \subseteq \mathbb{R}^m$, the inequality $\left\| \operatorname{argmin}_{\boldsymbol{u} \in U} \|\boldsymbol{u} - \boldsymbol{v}\| - \operatorname{argmin}_{\boldsymbol{u} \in U} \|\boldsymbol{u} - \boldsymbol{w}\| \right\| \leq \|\boldsymbol{v} - \boldsymbol{w}\|$ holds.

By induction with respect to $t \geq t_\delta$, it follows that $\boldsymbol{u}^t \in U^* + B^\delta$ for all $t \geq t_\delta$. Since this holds for arbitrarily small values of $\delta > 0$ and since the function h is continuous, the theorem follows. □

By requiring also that the sum of squared step lengths is convergent it is ensured that the sequence of dual iterates accumulates at a point in the dual solution set.

Theorem 15.2 (Convergence to a Solution by Quadratically Convergent Divergent Series Step Lengths) *Let the method* (15.21) *be applied to the problem* (15.10)*, with the step lengths* α_t *fulfilling the conditions* (15.23) *as well as*

$$\left\{ \sum_{s=0}^{t-1} \alpha_s^2 \right\} \to p, \quad as \quad t \to \infty, \tag{15.27}$$

where $p < \infty$*. If the sequence* $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ *is bounded, then* $\{ \boldsymbol{u}^t \} \to \boldsymbol{u}^\infty \in U^*$.

Proof Let $\boldsymbol{u}^* \in U^*$ be arbitrary and let $t \geq 1$. Telescoping (15.24) yields the inequality

$$\begin{aligned} \| \boldsymbol{u}^* - \boldsymbol{u}^t \|^2 \leq \| \boldsymbol{u}^* - \boldsymbol{u}^0 \|^2 - 2 \sum_{s=0}^{t-1} \alpha_s \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \right)^\top \left(\boldsymbol{u}^* - \boldsymbol{u}^s \right) \\ + \sum_{s=0}^{t-1} \alpha_s^2 \| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \|^2. \end{aligned} \tag{15.28}$$

Since $\boldsymbol{u}^* \in U^*$, $\boldsymbol{u}^s \in \mathbb{R}^m_+$, and $\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \in \partial^{\mathbb{R}^m_+} h(\boldsymbol{u}^s)$ for all $s \geq 0$ we obtain the inequalities

$$h(\boldsymbol{u}^s) \leq h(\boldsymbol{u}^*) \leq h(\boldsymbol{u}^s) + \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \right)^\top \left(\boldsymbol{u}^* - \boldsymbol{u}^s \right), \tag{15.29}$$

and hence that the inequality $\left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \right)^\top (\boldsymbol{u}^* - \boldsymbol{u}^s) \geq 0$ holds for all $s \geq 0$. We define $c := \sup_t \{ \| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \| \}$, so that the inequality $\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \| \leq c$ holds for all $s \geq 0$. From (15.27) and (15.23a) follow that $\sum_{s=0}^{t-1} \alpha_s^2 < p$ for all $t \geq 1$. By inserting this in (15.28), we then conclude that $\| \boldsymbol{u}^* - \boldsymbol{u}^t \|^2 < \| \boldsymbol{u}^* - \boldsymbol{u}^0 \|^2 + pc^2$ for any $t \geq 1$; it follows that the sequence $\{ \boldsymbol{u}^t \}$ is bounded.

Assume now that there is no subsequence $\mathcal{T}$ such that $\{ (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t))^\top (\boldsymbol{u}^* - \boldsymbol{u}^t) \}_{t \in \mathcal{T}} \to 0$. Then, there exists an $\varepsilon > 0$ and a $t_\varepsilon > 1$ such that the inequality $\left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \right)^\top (\boldsymbol{u}^* - \boldsymbol{u}^s) \geq \varepsilon$ for all $s \geq t_\varepsilon$. By (15.28) and (15.23c) then follow that $\{ \| \boldsymbol{u}^* - \boldsymbol{u}^t \| \} \to -\infty$, which is clearly impossible. Therefore, there is a subsequence $\mathcal{T}$ such that $\{ (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t))^\top (\boldsymbol{u}^* - \boldsymbol{u}^t) \}_{t \in \mathcal{T}} \to 0$. From (15.29) then follows that $\{ h(\boldsymbol{u}^t) \}_{t \in \mathcal{T}} \to h^*$.

The boundedness of the sequence $\{ \boldsymbol{u}^t \}$ implies the existence of an accumulation point of the subsequence $\{ \boldsymbol{u}^t \}_{t \in \mathcal{T}}$, say $\boldsymbol{u}^\infty$, and by the continuity of h then follows that $\boldsymbol{u}^\infty \in U^*$.

To show that $\boldsymbol{u}^\infty$ is the only accumulation point of the sequence $\{ \boldsymbol{u}^t \}$, let $\delta > 0$ and choose an $N_\delta \geq 0$ such that the inequalities $\| \boldsymbol{u}^\infty - \boldsymbol{u}^{N_\delta} \|^2 \leq \delta/2$ and $\sum_{s=N_\delta}^{\infty} \alpha_s^2 \leq \delta/(2c^2)$ hold. Consider any $t > N_\delta$. Analogously to the derivation of (15.28), and using (15.29), we then obtain that

$$\begin{aligned} \| \boldsymbol{u}^\infty - \boldsymbol{u}^t \|^2 &\leq \left\| \boldsymbol{u}^\infty - \boldsymbol{u}^{N_\delta} \right\|^2 + \sum_{s=N_\delta}^{t-1} \alpha_s^2 \| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) - \boldsymbol{v}(\boldsymbol{u}^s) \|^2 \\ &< \frac{\delta}{2} + \frac{\delta}{2c^2} c^2 = \delta. \end{aligned}$$

Since this holds for arbitrarily small values of $\delta > 0$, the theorem follows. □

In subgradient optimization, the important *Polyak step length* rule has documented practical usefulness. Convergence to an optimal solution with this step length rule relies on the optimal objective value h^*. This result extends to the case of the conditional subgradient optimization method (15.21), for which the Polyak step length formula is defined by

$$\alpha_t := \frac{\theta_t \left(h^* - h(\boldsymbol{u}^t) \right)}{\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \|^2}, \quad 0 < \epsilon_1 \leq \theta_t \leq 2 - \epsilon_2 < 2, \quad t = 0, 1, 2, \ldots \tag{15.30}$$

Proposition 15.7 (Convergence to a Solution by Polyak Step Lengths) *Let the method* (15.21) *be applied to the problem* (15.10)*, with the step lengths* α_t *fulfilling the conditions* (15.30)*. If the sequence* $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ *is bounded, then* $\{ h(\boldsymbol{u}^t) \} \to h^*$ *and* $\{ \boldsymbol{u}^t \} \to \boldsymbol{u}^\infty \in U^*$.

Remark 15.3 With step lengths defined by (15.30), the case of subgradient projection according to (15.22) yields actual steps in (15.21) (that is, $\boldsymbol{u}^{t+1} - \boldsymbol{u}^t$) that are longer, as compared with the case of plain subgradients, that is, when $\{ \boldsymbol{v}^t \} \equiv \{ \boldsymbol{0} \}$ holds.

For an $\varepsilon > \overline{h} - h^* \geq 0$, finite convergence to ε-optimality can be achieved by replacing h^* in (15.30) by an upper bound $\overline{h} \geq h^*$ and letting $\theta_t \equiv 1$.

Proposition 15.8 (Finite ε-Optimality by Polyak Step Lengths) *Let the method* (15.21) *be applied to the problem* (15.10)*, with the step lengths* $\{ \alpha_t \}$ *defined by*

$$\alpha_t := \frac{\theta_t \left(\overline{h} - h(\boldsymbol{u}^t) \right)}{\| \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) - \boldsymbol{v}(\boldsymbol{u}^t) \|^2}, \qquad t = 0, 1, 2, \ldots, \tag{15.31}$$

where $\theta_t \equiv 1$ *and* $\overline{h} > h^*$. *If the sequence* $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ *is bounded, then for any* $\varepsilon > 0$, *there is a* $t_\varepsilon > 0$ *such that* $h(\boldsymbol{u}^{t_\varepsilon}) \geq 2h^* - \overline{h} - \varepsilon$.

Our results to follow in Sect. 15.3.2—on ergodic convergence to a primal solution—rely on the divergent series conditions (15.23), (15.27) on the step lengths in the method (15.21). By ensuring that the sequence $\{ \alpha_t \}$ fulfills the conditions

$$\frac{\underline{a}}{b+t} \leq \alpha_t \leq \frac{\overline{a}}{b+t}, \qquad 0 < \underline{a} \leq \overline{a}, \quad b > 0, \quad t = 0, 1, \ldots, \tag{15.32}$$

convergence in terms of Theorem 15.2 can, however, be shown.[7]

Corollary 15.1 (Convergence to a Solution by Generalized Harmonic Series Step Lengths) *Let the method* (15.21), *where the sequence* $\{ \alpha_t \}$ *fulfills* (15.32), *be applied to the problem* (15.10). *If the sequence* $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ *is bounded, then* $\{ \boldsymbol{u}^t \} \to \boldsymbol{u}^\infty \in U^*$ *hold.*

15.3.2 Ergodic Convergence in the Primal Problem

The conditional subgradient optimization method (15.21) constructs a sequence $\{ \boldsymbol{x}(\boldsymbol{u}^t) \}$ of solutions to the Lagrangian subproblem (15.9). Due to the non-coordinability of the Lagrangian subproblem (see Sect. 15.2.2) this sequence is, however, not convergent. We propose a scheme for generating an *ergodic sequence of subproblem solutions*, which is shown to converge to the solution set X^*_{conv}. The generation of the ergodic sequence is computationally cheap and its storage requires a relatively small amount of memory. The sequence is defined by convexity weights that are proportional to the step lengths α_t; the latter requirement is then generalized and improved.

From Propositions 15.2 and 15.4 follow that the set $\partial h(\boldsymbol{u}^\infty) \cap N_{\mathbb{R}^m_+}(\boldsymbol{u}^\infty)$ is nonempty. The next proposition establishes that the sequence $\{ \boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) \}$ of subgradients to the dual objective function converges in an ergodic sense to an element that verifies optimality of the Lagrangian dual, in terms of Proposition 15.4.

Proposition 15.9 (Ergodic Subgradients Converge to the Optimality-Verifying Set) *Apply the method* (15.21), (15.23) *to the problem* (15.10) *and define the sequence* $\left\{ \overline{\boldsymbol{g}}^t \right\}$ *as*

$$\overline{\boldsymbol{g}}^t := \frac{1}{\sum_{s=0}^{t-1} \alpha_s} \sum_{s=0}^{t-1} \alpha_s \big(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^s) \big), \qquad t = 1, 2, \ldots$$

If the sequence $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ *is bounded, then* $\left\{ \min_{\boldsymbol{\gamma} \in \partial h(\boldsymbol{u}^\infty) \cap N_{\mathbb{R}^m_+}(\boldsymbol{u}^\infty)} \| \boldsymbol{\gamma} - \overline{\boldsymbol{g}}^t \| \right\} \to 0$.

[7] For any sequence $\{ \alpha_t \}$ of step lengths as defined in (15.31) [or according to (15.23), (15.27)], there exists constants $\underline{a}$, $\overline{a}$, and b such that the conditions (15.32) are fulfilled.

The ergodic sequence $\{\overline{\boldsymbol{x}}^t\}$ is then defined as weighted averages (that is, convex combinations) of the subproblem solutions found up to iteration t of (15.21), as

$$\overline{\boldsymbol{x}}^t := \frac{1}{\sum_{s=0}^{t-1}\alpha_s}\sum_{s=0}^{t-1}\alpha_s \boldsymbol{x}(\boldsymbol{u}^s), \qquad t = 1, 2, \ldots \tag{15.33}$$

The convergence of the sequence $\{\overline{\boldsymbol{x}}^t\}$ to the set X^*_{conv}, as expressed in (15.14), is then established in terms of fulfilment of the optimality conditions in Proposition 15.5.

Theorem 15.3 ($\{\overline{\boldsymbol{x}}^t\}$ Converges to the Optimal Set of the Convexified Problem) *Let the method* (15.21), (15.23) *be applied to the problem* (15.10)*, the set X^*_{conv} and the sequence $\{\overline{\boldsymbol{x}}^t\}$ be given by* (15.14) *and* (15.33)*, respectively, and suppose that the sequence $\{\boldsymbol{v}(\boldsymbol{u}^t)\}$ is bounded. Then, $\{\min_{\boldsymbol{x}\in X^*_{\text{conv}}} \|\boldsymbol{x} - \overline{\boldsymbol{x}}^t\|\} \to 0$.*

Efficient updates of the ergodic iterates $\overline{\boldsymbol{x}}^t$ requires only the previous ergodic iterate $\overline{\boldsymbol{x}}^{t-1}$ and subproblem solution $\boldsymbol{x}(\boldsymbol{u}^{t-1})$, according to the convex combination

$$\overline{\boldsymbol{x}}^t := \frac{\sum_{s=0}^{t-1}\alpha_s - \alpha_{t-1}}{\sum_{s=0}^{t-1}\alpha_s}\overline{\boldsymbol{x}}^{t-1} + \frac{\alpha_{t-1}}{\sum_{s=0}^{t-1}\alpha_s}\boldsymbol{x}(\boldsymbol{u}^{t-1}), \qquad t = 2, 3, \ldots \tag{15.34}$$

with $\overline{\boldsymbol{x}}^1 := \boldsymbol{x}(\boldsymbol{u}^0)$.

15.3.3 Enhanced Primal Ergodic Convergence

The convergence of the ergodic sequence of subproblem solutions according to Theorem 15.3 is, however, typically very slow. Efforts have therefore been put into enhancing the convergence speed, by exploiting more information from later subproblem solutions than from earlier ones. We next present a more general pattern for constructing the ergodic sequences; the ergodic sequence $\{\tilde{\boldsymbol{x}}^t\}$ is defined by

$$\tilde{\boldsymbol{x}}^t := \sum_{s=0}^{t-1}\mu_s^t \boldsymbol{x}(\boldsymbol{u}^s); \qquad \sum_{s=0}^{t-1}\mu_s^t = 1; \qquad \mu_s^t \geq 0, \quad s = 0, \ldots, t-1, \tag{15.35}$$

where the convexity weights μ_s^t are defined as

$$\mu_s^t := \gamma_s^t \alpha_s, \quad s = 0, \ldots, t-1, \quad t = 1, 2, \ldots, \tag{15.36a}$$

and the parameters γ_s^t fulfil the requirements ($N > 0$ being a constant)

$$\gamma_s^t \geq \gamma_{s-1}^t, \qquad s = 1, \ldots, t-1,\ t = 2, 3, \ldots, \tag{15.36b}$$

$$\left\{ \max_{s \in \{1,\ldots,t-1\}} \left\{ \gamma_s^t - \gamma_{s-1}^t \right\} \right\} \to 0, \qquad \text{as } t \to \infty, \tag{15.36c}$$

$$\gamma_0^t \to 0, \qquad \text{as } t \to \infty, \tag{15.36d}$$

$$\gamma_{t-1}^t \leq N, \qquad t = 1, 2, \ldots \tag{15.36e}$$

The requirement (15.36b), together with the definition (15.36a), implies the inequality $\mu_s^t \, (\mu_{s-1}^t)^{-1} \geq \alpha_s \, (\alpha_{s-1})^{-1}$, which means that the ratio of any two consecutive convexity weights should be no less than that of the corresponding step lengths. The requirement (15.36c) implies that the difference between consecutive pairs of subsequent convexity weights tends to zero as t increases, meaning that no primal iterate may be completely neglected. The requirement (15.36d) implies, however, that the weight (that is, μ_0^t) of the first iterate [that is, $\boldsymbol{x}(\boldsymbol{u}^0)$] tends to zero when t increases. The requirements (15.36e), (15.36d), and (15.36b) assure that, for decreasing step lengths α_s, the convexity weights μ_s^t decrease at a rate not slower than that of the step lengths. Efficient updates of the ergodic iterates $\tilde{\boldsymbol{x}}^t$ can be made, analogously to (15.34).

Theorem 15.4 ($\{\tilde{\boldsymbol{x}}^t\}$ Converges to the Optimal Set of the Convexified Problem) *Apply the method* (15.21), (15.23), (15.27) *to the problem* (15.10) *and define the set* X_{conv}^* *and the sequence* $\{\tilde{\boldsymbol{x}}^t\}$ *by* (15.14) *and* (15.35), (15.36), *respectively. If the sequence* $\{\boldsymbol{v}(\boldsymbol{u}^t)\}$ *is bounded, then* $\left\{ \min_{\boldsymbol{x} \in X_{\text{conv}}^*} \| \boldsymbol{x} - \tilde{\boldsymbol{x}}^t \| \right\} \to 0$.

Remark 15.4 For any fixed value of $s \in \{0, \ldots, t-1\}$, the requirements (15.36b)–(15.36e) imply that $\gamma_s^t \leq \gamma_0^t + s \cdot \max_{r \in \{1,\ldots,t-1\}} \left\{ \gamma_r^t - \gamma_{r-1}^t \right\} \to 0$ as $t \to \infty$. This yields that $\mu_s^t = \gamma_s^t \alpha_s \to 0$ as $t \to \infty$, since $\alpha_s < \infty$, $s = 0, 1, \ldots$

Remark 15.5 The ergodic sequence $\{\overline{\boldsymbol{x}}^t\}$, defined in (15.33), is equivalent to the special case of (15.35), (15.36) defined by $\gamma_s^t := \left(\sum_{r=0}^{t-1} \alpha_r \right)^{-1}$ (being independent of s). For this choice of $\{\gamma_s^t\}$, the requirements (15.36b)–(15.36e) are fulfilled with $N := (\alpha_0)^{-1}$.

Remark 15.6 For the special case of (15.35), (15.36) given by (15.32) with $a := \underline{a} = \overline{a}$, that is, *modified harmonic series* step lengths

$$\alpha_s := \frac{a}{b+s}, \qquad a > 0, \quad b > 0, \quad s = 0, 1, \ldots, t-1, \tag{15.37}$$

and choosing $\gamma_s^t := (t\alpha_s)^{-1}$, the convexity weights become $\mu_s^t = t^{-1}$ (then, $\tilde{\boldsymbol{x}}^t$ equals a simple average of the subproblem solutions found). For these choices of

$\{\gamma_s^t\}$ and $\{\alpha_s\}$, the requirements (15.36b)–(15.36e) are fulfilled, with $N := a^{-1} \cdot \max\{b, 1\}$.

By choosing the step lengths α_t according to (15.37) and letting the sequence $\{\mu_s^t\}$ of convexity weights fulfil the requirements

$$\mu_s^t \geq \mu_{s-1}^t, \quad s = 1, \ldots, t-1, \; t = 2, 3, \ldots, \tag{15.38a}$$

$$\left\{ t \cdot \max_{s \in \{1,\ldots,t-1\}} \{\mu_s^t - \mu_{s-1}^t\} \right\} \to 0, \qquad \text{as } t \to \infty, \tag{15.38b}$$

$$t\,\mu_{t-1}^t \leq M, \qquad t = 1, 2, \ldots, \tag{15.38c}$$

where $M > 0$ is a constant, it can be shown that also the requirements (15.36b)–(15.36e) are fulfilled, with $N = a^{-1}M \cdot \max\{b, 1\}$.

The so-called *s^k-rule*, for which the corresponding convexity weights, μ_s^t, fulfill the requirements (15.38), is defined by

$$\mu_s^t := \frac{(s+1)^k}{\sum_{r=0}^{t-1}(r+1)^k}, \qquad s = 0, \ldots, t-1, \quad t = 1, 2, \ldots, \quad k \geq 0. \tag{15.39}$$

For $k > 0$, the s^k-rule results in an ergodic sequence (15.35) in which the later iterates are assigned higher weights than the earlier ones. For larger values of k, the weights are shifted towards later iterates. Given that step lengths α_t according to (15.37) are utilized in the method (15.21) applied to the Lagrangian dual (15.10), it can be shown that weights according to (15.39) fulfill the requirements (15.38), so that convergence for the resulting primal ergodic sequence $\{\tilde{\boldsymbol{x}}^t\}$ to the optimal set X_{conv}^* can be established.

Remark 15.7 Note that the s^0-rule yields $\mu_s^t = t^{-1}$ [cf. Remark 15.6, where $\gamma_s^t = (t\alpha_s)^{-1}$]. For $k > 0$, the s^k-rule results in an ergodic sequence in which later iterates are assigned higher weights than earlier ones. For larger values of k, the weights are shifted towards increasingly later iterates.

Remark 15.8 Since the conditional subgradient optimization method (15.21) is memory-less, without loss of generality the computation of the ergodic sequences $\{\overline{\boldsymbol{x}}^t\}$, $\{\overline{\boldsymbol{g}}^t\}$, and $\{\tilde{\boldsymbol{x}}^t\}$ may be postponed a finite number of iterations. If the postponement is "long enough", each $\overline{\boldsymbol{x}}^t$ (or $\tilde{\boldsymbol{x}}^t$) will be a solution to the Lagrangian subproblem at the optimal dual point $\boldsymbol{u}^\infty$, as defined in Theorem 15.2.

15.3.4 Finite Primal Feasibility and Finite ε-Optimality

Theorem 15.3 establishes optimality in the limit for the sequence $\{\overline{\boldsymbol{x}}^t\}$. While dual feasibility holds for the sequence $\{\boldsymbol{u}^t\}$, in general neither primal feasibility (that is, $A\overline{\boldsymbol{x}}^t \geq \boldsymbol{b}$) nor complementarity [that is, $(\boldsymbol{u}^t)^\top(\boldsymbol{b} - A\overline{\boldsymbol{x}}^t) = 0$] will be finitely satisfied. Eventually, however, $\overline{\boldsymbol{x}}^t$ will be both near feasible and near complementary. Whenever finite primal feasibility is required, a procedure can be applied that converts any $\overline{\boldsymbol{x}}^t$ into a feasible solution to (15.5), for example, by computing the Euclidean projection of $\overline{\boldsymbol{x}}^t$ onto the feasible set, as

$$\overline{\boldsymbol{x}}^t_{\text{proj}} := \underset{\boldsymbol{x} \in X_{\text{conv}}}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{x} - \overline{\boldsymbol{x}}^t \right\| \,\middle|\, A\boldsymbol{x} \geq \boldsymbol{b} \right\}. \tag{15.40}$$

Solving (15.40) regularly may, however, be computationally too expensive. Instead, we consider a heuristic procedure, preferably exploiting the structure of the set $\{\boldsymbol{x} \in X_{\text{conv}} \mid A\boldsymbol{x} \geq \boldsymbol{b}\}$ in the search for a feasible and near optimal solution to (15.40). Let the function $\delta : \mathbb{R}_+ \mapsto \mathbb{R}_+$ be continuous and such that $\delta(\beta) > 0$ whenever $\beta > 0$ and $\lim_{\beta \to 0^+} \delta(\beta) = 0$. Define a *heuristic projection* $\overline{\boldsymbol{x}}^t_{\text{heur}}$ of $\overline{\boldsymbol{x}}^t \in X_{\text{conv}}$ by the inclusion

$$\overline{\boldsymbol{x}}^t_{\text{heur}} \in \left\{ \boldsymbol{x} \in X_{\text{conv}} \mid A\boldsymbol{x} \geq \boldsymbol{b} \right\}, \tag{15.41a}$$

and such that

$$\left\| \overline{\boldsymbol{x}}^t_{\text{heur}} - \overline{\boldsymbol{x}}^t_{\text{proj}} \right\| \leq \delta(\beta) \quad \text{whenever} \quad \left\| \overline{\boldsymbol{x}}^t - \overline{\boldsymbol{x}}^t_{\text{proj}} \right\| \leq \beta. \tag{15.41b}$$

Theorem 15.5 (Convergence to Primal Optimality by Heuristic Projections) *Let the method* (15.21), (15.23), (15.27) *be applied to the problem* (15.10). *Let the set* X^*_{conv} *and the sequences* $\{\overline{\boldsymbol{x}}^t\}$, $\{\overline{\boldsymbol{x}}^t_{\text{proj}}\}$, *and* $\{\overline{\boldsymbol{x}}^t_{\text{heur}}\}$ *be defined by* (15.14), (15.33), (15.40), *and* (15.41a), *respectively. If the sequence* $\{\boldsymbol{v}(\boldsymbol{u}^t)\}$ *is bounded and the conditions* (15.41b) *hold, then* $\left\{ \min_{\boldsymbol{x} \in X^*_{\text{conv}}} \| \boldsymbol{x} - \overline{\boldsymbol{x}}^t_{\text{heur}} \| \right\} \to 0$.

We can now construct an algorithm employing heuristic projections and yielding convergence to the optimal value in both the primal and dual procedures.

Corollary 15.2 (Finite Termination at ε-Optimality) *Given the assumptions of Theorem 15.5, for every* $\varepsilon > 0$ *there is a* $t_\varepsilon > 0$ *such that* $\boldsymbol{c}^\top \overline{\boldsymbol{x}}^t_{\text{heur}} - h(\boldsymbol{u}^t) \leq \varepsilon$ *holds for all* $t \geq t_\varepsilon$.

This projection principle is thus a way to recover primal feasibility, and eventually also optimality, in an otherwise purely dual method.

15.4 Dual Cutting-Planes: Dantzig–Wolfe Decomposition

Assuming that a nonempty subset $\overline{\mathcal{Q}} \subseteq \mathcal{Q}$ is given, the Lagrangian dual function is outer approximated by the function $\overline{h} : \mathbb{R}^m \mapsto \mathbb{R}$ as defined by

$$\overline{h}(\boldsymbol{u}) := \boldsymbol{b}^\top \boldsymbol{u} + \min_{q \in \overline{\mathcal{Q}}} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u}\right)^\top \boldsymbol{x}^q \right\}. \tag{15.42}$$

This function is formed by the point-wise minimum of $\left| \overline{\mathcal{Q}} \right|$ affine functions. Hence, it is piecewise affine and concave, as is the Lagrangian dual function h. Clearly, $\overline{h}(\boldsymbol{u}) \geq h(\boldsymbol{u})$ holds for any $\boldsymbol{u} \in \mathbb{R}^m$. Hence, the problem

$$\overline{h}^* := \max_{\boldsymbol{u} \in \mathbb{R}^m_+} \overline{h}(\boldsymbol{u}), \tag{15.43}$$

is a *relaxation* of the Lagrangian dual problem (15.10), so that $\overline{h}^* \geq h^*$ holds. The subdifferential of the function $\overline{h}$ at $\boldsymbol{u}$ is given by

$$\partial \overline{h}(\boldsymbol{u}) := \operatorname*{conv}_{q \in \overline{\mathcal{Q}}(\boldsymbol{u})} \left\{ \boldsymbol{b} - A\boldsymbol{x}^q \right\}, \qquad \boldsymbol{u} \in \mathbb{R}^m_+,$$

where $\overline{\mathcal{Q}}(\boldsymbol{u})$ denotes the optimal index set of the inner minimization in (15.42), and its conditional subdifferential is given by

$$\partial^{\mathbb{R}^m_+} \overline{h}(\boldsymbol{u}) = \partial \overline{h}(\boldsymbol{u}) - N_{\mathbb{R}^m_+}(\boldsymbol{u}), \qquad \boldsymbol{u} \in \mathbb{R}^m_+.$$

The relaxed dual problem (15.43) has a bounded optimal solution if and only if

$$\boldsymbol{0} \in \bigcup_{\boldsymbol{u} \in \mathbb{R}^m_+} \partial^{\mathbb{R}^m_+} \overline{h}(\boldsymbol{u}). \tag{15.44}$$

Remark 15.9 A sufficient condition for the relaxed Lagrangian dual problem (15.43) to have a bounded optimal solution is that some point $\boldsymbol{x}^{\overline{q}}$, $\overline{q} \in \overline{\mathcal{Q}}$, is feasible in the original mixed-binary linear optimization problem (15.2), since this implies that

$$\begin{aligned} \overline{h}^* &= \max_{\boldsymbol{u} \in \mathbb{R}^m_+} \left\{ \min_{q \in \overline{\mathcal{Q}}} \left\{ \boldsymbol{c}^\top \boldsymbol{x}^q + \boldsymbol{u}^\top \left(\boldsymbol{b} - A\boldsymbol{x}^q\right) \right\} \right\} \\ &\leq \max_{\boldsymbol{u} \in \mathbb{R}^m_+} \left\{ \boldsymbol{c}^\top \boldsymbol{x}^{\overline{q}} + \boldsymbol{u}^\top \left(\boldsymbol{b} - A\boldsymbol{x}^{\overline{q}}\right) \right\} \\ &= \boldsymbol{c}^\top \boldsymbol{x}^{\overline{q}} \end{aligned}$$

holds, where the second equality holds because $A\boldsymbol{x}^{\overline{q}} \geq \boldsymbol{b}$.

Assuming that the condition (15.44) holds, we find a $\overline{\boldsymbol{u}} \in \mathbb{R}^m_+$ such that the inclusion $\mathbf{0} \in \partial^{\mathbb{R}^m_+} \overline{h}(\overline{\boldsymbol{u}})$ holds, implying that $\overline{\boldsymbol{u}}$ is optimal in (15.42). Hence, $\overline{h}^* = \overline{h}(\overline{\boldsymbol{u}})$ holds. The value of the Lagrangian dual function at this point is

$$h(\overline{\boldsymbol{u}}) = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \min_{\boldsymbol{x} \in X} \left\{ \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x} \right\} = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \min_{q \in \mathcal{Q}} \left\{ \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^q \right\}.$$

From the Lagrangian dual problem (15.10) we conclude that $h(\overline{\boldsymbol{u}}) \leq h^*$. To summarize, the relations $h(\overline{\boldsymbol{u}}) \leq h^* \leq \overline{h}^*$ hold, that is, the values $\overline{h}^*$ and $h(\overline{\boldsymbol{u}})$ provide upper and lower bounds, respectively, on the optimal value h^*.

If the equality $h(\overline{\boldsymbol{u}}) = \overline{h}^*$ holds, it follows that $h(\overline{\boldsymbol{u}}) = h^*$, implying that $\overline{\boldsymbol{u}}$ is optimal in (15.10). In the case when $h(\overline{\boldsymbol{u}}) < \overline{h}^*$ holds, we consider any solution to the minimization in (15.8) for $\boldsymbol{u} = \overline{\boldsymbol{u}}$, which we denote by $\boldsymbol{x}^{\overline{q}}$, where $\overline{q} \in \mathcal{Q}(\overline{\boldsymbol{u}})$, such that $h(\overline{\boldsymbol{u}}) = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^{\overline{q}}$ holds. Then, the relations

$$\boldsymbol{b}^\top \overline{\boldsymbol{u}} + \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^{\overline{q}} < \overline{h}^* = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \min_{q \in \overline{\mathcal{Q}}} \left\{ \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^q \right\}$$

hold, which yields that $\overline{q} \notin \overline{\mathcal{Q}}$. By then augmenting the set $\overline{\mathcal{Q}}$ with the index $\overline{q}$, an improved outer approximation of the Lagrangian dual function is obtained. By resolving the problem (15.43) and repeating, an iterative procedure for solving the Lagrangian dual problem (15.10) is obtained. Its convergence is finite, since the set $\mathcal{Q}$ is finite and since a point $\boldsymbol{x}^{\overline{q}}$ can never be regenerated.

The above procedure is commonly described as a *cutting-plane* or *constraint generation* procedure for the LP formulation (15.11b) of the Lagrangian dual problem. This formulation can be restated as

$$h^* = \max_{(\boldsymbol{u}, v) \in \mathbb{R}^m_+ \times \mathbb{R}} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + v \,\middle|\, \boldsymbol{c}^\top \boldsymbol{x}^q - \left(A\boldsymbol{x}^q\right)^\top \boldsymbol{u} - v \geq 0,\ q \in \mathcal{Q} \right\}. \tag{15.45}$$

Let $(\overline{\boldsymbol{u}}, \overline{v}) \in \mathbb{R}^m_+ \times \mathbb{R}$ be optimal in the relaxed problem

$$\overline{h}^* = \max_{(\boldsymbol{u}, v) \in \mathbb{R}^m_+ \times \mathbb{R}} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + v \,\middle|\, \boldsymbol{c}^\top \boldsymbol{x}^q - \left(A\boldsymbol{x}^q\right)^\top \boldsymbol{u} - v \geq 0,\ q \in \overline{\mathcal{Q}} \right\}. \tag{15.46}$$

Then $\overline{h}^* = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \overline{v}$. The question then is if all constraints in the problem (15.45) are satisfied or not at the point $(\boldsymbol{u}, v) = (\overline{\boldsymbol{u}}, \overline{v})$. This is determined by finding the *most violated* constraint, if any, and amounts to solving

$$\begin{aligned} \min_{q \in \mathcal{Q}} \left\{ \boldsymbol{c}^\top \boldsymbol{x}^q - \left(A\boldsymbol{x}^q\right)^\top \overline{\boldsymbol{u}} - \overline{v} \right\} &= \min_{q \in \mathcal{Q}} \left\{ \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^q - \overline{h}^* \right\} \\ &= \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^{\overline{q}} - \overline{h}^* \\ &= h(\overline{\boldsymbol{u}}) - \overline{h}^*, \end{aligned} \tag{15.47}$$

where $\boldsymbol{x}^{\overline{q}} \in X(\overline{\boldsymbol{u}})$ solves the minimization in (15.8) at $\boldsymbol{u} = \overline{\boldsymbol{u}}$. If the minimum value $h(\overline{\boldsymbol{u}}) - \overline{h}^*$ is negative, then a most violated constraint has been identified. Otherwise, $h(\overline{\boldsymbol{u}}) = \overline{h}^*$ holds and $\overline{\boldsymbol{u}} \in \mathbb{R}^m_+$ solves the dual problem (15.10).

The dual cutting-plane procedure sketched above is dually equivalent to applying *Dantzig–Wolfe decomposition* to the convexified problem (15.5), which by the reformulation (15.11c) is equivalent to the *complete Dantzig–Wolfe master problem*

$$\min_{\lambda} \left\{ \sum_{q \in \mathcal{Q}} \left(\boldsymbol{c}^\top \boldsymbol{x}^q \right) \lambda_q \;\middle|\; \sum_{q \in \mathcal{Q}} \left(A \boldsymbol{x}^q \right) \lambda_q \geq \boldsymbol{b};\; \sum_{q \in \mathcal{Q}} \lambda_q = 1;\; \lambda_q \geq 0,\; q \in \mathcal{Q} \right\}, \tag{15.48}$$

while also being the LP dual of the problem (15.45). Since

$$X_{\text{conv}} = \left\{ \boldsymbol{x} = \sum_{q \in \mathcal{Q}} \lambda_q \boldsymbol{x}^q \;\middle|\; \sum_{q \in \mathcal{Q}} \lambda_q = 1;\; \lambda_q \geq 0,\; q \in \mathcal{Q} \right\}$$

holds, the variables λ_q in (15.48) are commonly referred to as *convexity variables* and the equality $\sum_{q \in \mathcal{Q}} \lambda_q = 1$ as a *convexity constraint*. In Dantzig–Wolfe decomposition, this problem is solved using the linear programming technique called *column generation*, in which, for this application, a column corresponds to the data of a convexity variable, that is, the values of the scalar $\boldsymbol{c}^\top \boldsymbol{x}^q$, the vector $A\boldsymbol{x}^q$, and a 1.

Using the analogous reformulation as in (15.11), we obtain

$$\overline{h}^* = \max_{\boldsymbol{u} \in \mathbb{R}^m_+} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + \min_{q \in \overline{\mathcal{Q}}} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u} \right)^\top \boldsymbol{x}^q \right\} \right\} \tag{15.49a}$$

$$= \min_{\lambda} \left\{ \sum_{q \in \overline{\mathcal{Q}}} (\boldsymbol{c}^\top \boldsymbol{x}^q) \lambda_q \;\middle|\; \sum_{q \in \overline{\mathcal{Q}}} \left(A \boldsymbol{x}^q \right) \lambda_q \geq \boldsymbol{b};\; \sum_{q \in \overline{\mathcal{Q}}} \lambda_q = 1;\; \lambda_q \geq 0,\; q \in \overline{\mathcal{Q}} \right\} \tag{15.49b}$$

$$= \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \;\middle|\; A\boldsymbol{x} \geq \boldsymbol{b};\; \boldsymbol{x} \in \operatorname*{conv}_{q \in \overline{\mathcal{Q}}} \left\{ \boldsymbol{x}^q \right\} \right\}. \tag{15.49c}$$

The problem (15.49b) is the *restricted* Dantzig–Wolfe master problem, referring to the fact that this problem includes the variables λ_q, with the corresponding columns $\left(\boldsymbol{c}^\top \boldsymbol{x}^q, (A\boldsymbol{x}^q)^\top, 1 \right)^\top$, only for $q \in \overline{\mathcal{Q}} \subseteq \mathcal{Q}$, which is clearly equivalent to imposing the restrictions $\lambda_q = 0$ for $q \in \mathcal{Q} \setminus \overline{\mathcal{Q}}$ in (15.48).

Let $(\overline{\boldsymbol{u}}, \overline{v}) \in \mathbb{R}^m_+ \times \mathbb{R}$ be an optimal solution to the LP dual of the restricted master problem. Using strong duality for LP, it then holds that $\overline{h}^* = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \overline{v}$. The LP reduced cost for any variable λ_q, $q \in \mathcal{Q}$, in the complete master problem can

then be expressed as

$$\overline{c}^q := \boldsymbol{c}^\top \boldsymbol{x}^q - \overline{\boldsymbol{u}}^\top A \boldsymbol{x}^q - \overline{v} = \boldsymbol{b}^\top \overline{\boldsymbol{u}} + \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^q - \overline{h}^*.$$

The problem of finding the variable λ_q, $q \in \mathcal{Q}$, with the most negative reduced cost reduces to find

$$\min_{q \in \mathcal{Q}} \left\{ \left(\boldsymbol{c} - A^\top \overline{\boldsymbol{u}}\right)^\top \boldsymbol{x}^q \right\} = \min_{\boldsymbol{x} \in X} \left\{ \left(\boldsymbol{c} - A^\top \boldsymbol{u}\right)^\top \boldsymbol{x} \right\}, \tag{15.50}$$

which is in this context known as a *column generation problem*, or *Dantzig–Wolfe subproblem*. If the solution to (15.50) corresponds to a negative reduced cost $\overline{c}^{\overline{q}} < 0$, $\overline{q} \in \mathcal{Q}$,[8] then $\overline{\mathcal{Q}} := \overline{\mathcal{Q}} \cup \left\{ \overline{q} \right\}$ and the problem (15.49b) is resolved. When no more columns having negative reduced costs can be found, which clearly occurs finitely, the convexified problem (15.5) has been solved.

Note that the problem (15.50) is equivalent to that of finding the most violated cutting-plane in the dual space [cf. (15.47)], since each constraint in the cutting-plane formulation of the Lagrangian dual corresponds to a convexity variable in the master problem. Note also that relaxing the dual problem (15.45) into (15.46) is dually equivalent to restricting the complete Dantzig–Wolfe master problem (15.48) into (15.49b).

Remark 15.10 Many textbooks derive the Dantzig–Wolfe decomposition method for an LP problem with two sets of explicit affine constraints, in our notation corresponding to $A\boldsymbol{x} \geq \boldsymbol{b}$ and $\boldsymbol{x} \in X_{\text{conv}}$. The method can, however, be applied also when no explicit representation of the set X_{conv} in terms of affine constraints is available, as long as the subproblem (15.50) can be solved by some means.

Remark 15.11 If the mixed-binary linear optimization problem (15.2) has a Cartesian product structure, as in Remark 15.1, then a separate set of convexity variables and a convexity constraint can be defined for each of the sets in the product. Letting $\{\boldsymbol{y}_s^q\}_{q \in \mathcal{Q}_s}$ denote the extreme points of conv Y_s, $s \in \mathcal{S}$, the complete Dantzig–Wolfe master problem for the problem (15.7) becomes

$$z^* = \min_{\lambda_{sq},\, q \in \mathcal{Q}_s,\, s \in \mathcal{S}} \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{Q}_s} \left(\boldsymbol{c}_s^\top \boldsymbol{y}_s^q\right) \lambda_{sq} \tag{15.51a}$$

$$\text{subject to} \quad \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{Q}_s} \left(A_s \boldsymbol{y}_s^q\right) \lambda_{sq} \geq \boldsymbol{b}, \tag{15.51b}$$

[8] It then holds that $\overline{q} \in \mathcal{Q} \setminus \overline{\mathcal{Q}}$, since $\overline{c}^q \geq 0$ holds for all $q \in \overline{\mathcal{Q}}$.

$$\sum_{q \in \mathcal{Q}_s} \lambda_{sq} = 1, \quad s \in \mathcal{S}, \tag{15.51c}$$

$$\lambda_{sq} \geq 0, \quad q \in \mathcal{Q}_s, \, s \in \mathcal{S}. \tag{15.51d}$$

When applying column generation to the master problem (15.51), there will be one Dantzig–Wolfe subproblem for each $s \in \mathcal{S}$.

Since each vector $\boldsymbol{y}_s$, $s \in \mathcal{S}$, is here described by a separate set of convexity variables, the master problem (15.51) is based on a *disaggregate representation*, while the master problem (15.48) is based on an *aggregate representation*. Whenever the feasible set of the original problem (15.2) is described by a Cartesian product set and coupling constraints, as in (15.7), a disaggregate representation is typically favourable. This is due to (1) that the disaggregate and aggregate complete master problems contain $\sum_{s \in \mathcal{S}} |\mathcal{Q}_s|$ and $\prod_{s \in \mathcal{S}} |\mathcal{Q}_s|$ convexity variables, respectively, and (2) that a disaggregate restricted master problem is a relaxation of the corresponding aggregate restricted master problem, and hence the former typically provides stronger bounds on z^*.

15.5 A Two-Phase Method: Subgradient Optimization and Dantzig–Wolfe Decomposition

Subgradient optimization and Dantzig–Wolfe decomposition as means for solving the Lagrangian dual problem (15.10) possess their distinct advantages and disadvantages, which are consequences of the nondifferentiability of the Lagrangian dual function and the inherent properties of these solution principles.

In *subgradient optimization*, the update of the dual iterate is inexpensive once the Lagrangian relaxed problem has been solved. The method lacks, however, a good termination criterion, and in practice it is therefore often run for a preset number of iterations. Further, primal feasible solutions are typically not easily obtained, neither to the mixed-binary linear optimization problem (15.2) nor to its convexified version (15.5). It is, however, quite common to use a Lagrangian heuristic—tailored to each specific application—to convert Lagrangian subproblem solutions into feasible solutions to the original mixed-binary problem.

Dantzig–Wolfe decomposition converges finitely and produces feasible solutions to the convexified problem (15.5). The latter property allows early termination when the upper and lower bounds on the optimal value of (15.5) are close enough. Each iteration of the method is, however, expensive, since the next dual iterate is found by reoptimizing the LP restricted master problem. Further, due to the inherent instability of cutting-plane approaches, the method typically shows a poor convergence behaviour, in the sense that successive dual iterates may be very far apart. This phenomenon is commonly prevented by the introduction of a stabilization mechanism, such as *trust regions* in the dual space.

Another way to improve a Dantzig–Wolfe decomposition scheme is to heuristically generate an initial set of columns of high quality. We here describe a specific means to generate such columns, leading to a *two-phase method* that benefits from the advantages of both subgradient optimization and Dantzig–Wolfe decomposition. A first *prediction phase* employs subgradient optimization, in which Lagrangian subproblem solutions are stored. At termination, a lower bound on the optimal value of the convexified problem as well as a number of Lagrangian subproblem solutions are at hand. A second *solution phase* uses these solutions to construct an initial restricted master problem, whereafter the Dantzig–Wolfe method is employed.

The prediction phase aims at setting up a high quality initial restricted master problem, such that the solution phase can attain and verify optimality in fewer column generations. If the prediction phase works perfectly, then it is enough to solve one restricted master problem and make one column generation in order to reach and verify optimality. Hence, the solution phase can alternatively be viewed as an evaluation of the outcome of the prediction phase, and if needed compensate for its shortcoming. The rationale for the prediction phase is that the subgradient optimization method asymptotically finds columns that are optimal in the complete master problem, if the step lengths are appropriately chosen, as demonstrated below.

We consider solving the problem (15.10) by using the conditional subgradient method (15.21), with a sequence $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ that is bounded and with step lengths α_t fulfilling the conditions (15.23) and (15.27), such that the assumptions of Theorem 15.2 are fulfilled. This method will then in the limit find a dual solution $\boldsymbol{u}^\infty$, such that $(\boldsymbol{u}^\infty, v^\infty)$ is optimal in the LP dual (15.45) of the complete master problem (15.48).

Define the index sets

$$\mathcal{T}_q := \left\{ t \in \mathbb{Z}_+ \,\middle|\, \boldsymbol{x}(\boldsymbol{u}^t) = \boldsymbol{x}^q \right\}, \qquad q \in \mathcal{Q},$$

and

$$\widehat{\mathcal{Q}} := \left\{ q \in \mathcal{Q} \,\middle|\, \mathcal{T}_q \text{ is infinite} \right\} \subseteq \mathcal{Q},$$

that is, $\widehat{\mathcal{Q}}$ contains the indices $q \in \mathcal{Q}$ such that $\boldsymbol{x}^q$ solves the Lagrangian subproblem an infinite number of times. Consider then the corresponding restricted master problem

$$\widehat{h}^* := \min_{\lambda} \left\{ \sum_{q \in \widehat{\mathcal{Q}}} \left(\boldsymbol{c}^\top \boldsymbol{x}^q \right) \lambda_q \,\middle|\, \sum_{q \in \widehat{\mathcal{Q}}} \left(A\boldsymbol{x}^q \right) \lambda_q \geq \boldsymbol{b};\ \sum_{q \in \widehat{\mathcal{Q}}} \lambda_q = 1;\ \lambda_q \geq 0,\ q \in \widehat{\mathcal{Q}} \right\}. \tag{15.52}$$

Theorem 15.6 (Asymptotic Generation of Optimal Columns) *Let the method* (15.21) *be applied to the problem* (15.10)*, with the step lengths* $\{ \alpha_t \}$ *fulfilling the conditions* (15.23) *and* (15.27)*, and assume that the sequence* $\{ \boldsymbol{v}(\boldsymbol{u}^t) \}$ *is bounded. Then* $\widehat{h}^* = h^*$ *holds.*

Further, let $\boldsymbol{u}^\infty \in U^*$ be the limit point for the sequence $\{\boldsymbol{u}^t\}$ (cf. Theorem 15.2) and let $v = v^\infty$ be optimal in (15.45) given that $\boldsymbol{u} = \boldsymbol{u}^\infty$, so that $(\boldsymbol{u}^\infty, v^\infty)$ is an optimal dual solution to the complete master problem (15.48). For each Lagrangian subproblem solution $\boldsymbol{x}(\boldsymbol{u}^t)$, $t \in \mathbb{Z}_+$, define the reduced cost $\bar{c}_t := \boldsymbol{c}^\top \boldsymbol{x}(\boldsymbol{u}^t) - (\boldsymbol{u}^\infty)^\top A\boldsymbol{x}(\boldsymbol{u}^t) - v^\infty$. Then $\bar{c}_t = 0$ holds for every t that is sufficiently large.

Proof Consider the ergodic sequence $\{\overline{\boldsymbol{x}}^t\}$ of subproblem solutions given by (15.33), that is, $\overline{\boldsymbol{x}}^t = \left(\sum_{s=0}^{t-1} \alpha_s\right)^{-1} \sum_{s=0}^{t-1} \alpha_s \boldsymbol{x}(\boldsymbol{u}^s)$, $t = 1, 2, \ldots$ Define the convexity weights

$$\lambda_q^t := \frac{1}{\sum_{s=0}^{t-1} \alpha_s} \sum_{s \in \mathcal{T}_q^t} \alpha_s, \qquad q \in \mathcal{Q}, \quad t = 1, 2, \ldots, \tag{15.53}$$

where $\mathcal{T}_q^t := \{s \in \{0, 1, \ldots, t-1\} \mid \boldsymbol{x}(\boldsymbol{u}^s) = \boldsymbol{x}^q\}$, $q \in \mathcal{Q}$, $t = 1, 2, \ldots$ Clearly, $\lambda_q^t \geq 0$, $q \in \mathcal{Q}$, and $\sum_{q \in \mathcal{Q}} \lambda_q^t = 1$ hold for $t = 1, 2, \ldots$ Then, the ergodic solution $\overline{\boldsymbol{x}}^t$ can alternatively be expressed as the convex combination

$$\overline{\boldsymbol{x}}^t = \sum_{q \in \mathcal{Q}} \lambda_q^t \boldsymbol{x}^q, \qquad t = 1, 2, \ldots \tag{15.54}$$

The sequence $\{\boldsymbol{\lambda}^t\}_{t=1}^\infty$, where $\boldsymbol{\lambda}^t := \left(\lambda_q^t\right)_{q \in \mathcal{Q}}$, is contained in the unit simplex in $\mathbb{R}^{|\mathcal{Q}|}$, in which it thus has some accumulation point, say $\overline{\boldsymbol{\lambda}}$. The sequence $\{\overline{\boldsymbol{x}}^t\}$ then has an accumulation point at $\overline{\boldsymbol{x}} := \sum_{q \in \mathcal{Q}} \overline{\lambda}_q \boldsymbol{x}^q$. From Theorem 15.3 follows that $\overline{\boldsymbol{x}} \in X^*_{\text{conv}}$ must hold, which implies that $\overline{\boldsymbol{\lambda}}$ is optimal in the complete master problem (15.48).

For any $q \notin \widehat{\mathcal{Q}}$, the set $\mathcal{T}_q$ is finite and the formula (15.53) together with the divergent series conditions (15.23) yield that $\overline{\lambda}_q = 0$ holds. Hence, $\overline{\boldsymbol{\lambda}}$ is feasible in the restricted master problem (15.52). It follows that $\widehat{h}^* = h^*$ holds.

From the optimality of $(\boldsymbol{u}^\infty, v^\infty)$ in the LP dual (15.45) of the complete master problem (15.48) follows that

$$h^* = \boldsymbol{b}^\top \boldsymbol{u}^\infty + v^\infty. \tag{15.55}$$

Since the Lagrangian dual function h is polyhedral, for large enough values of t, $\boldsymbol{x}(\boldsymbol{u}^t)$ solves the Lagrangian subproblem at $\boldsymbol{u}^t$ as well as at $\boldsymbol{u}^\infty$. Hence, it holds that $\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t) \in \partial h(\boldsymbol{u}^t) \cap \partial h(\boldsymbol{u}^\infty)$, and it follows that the inequalities

$$h(\boldsymbol{u}^\infty) \leq h(\boldsymbol{u}^t) + \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t)\right)^\top \left(\boldsymbol{u}^\infty - \boldsymbol{u}^t\right)$$

and

$$h(\boldsymbol{u}^t) \leq h(\boldsymbol{u}^\infty) + \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^t)\right)^\top \left(\boldsymbol{u}^t - \boldsymbol{u}^\infty\right)$$

hold. By combining these inequalities and simplifying we obtain the equalities

$$\begin{aligned} h(\boldsymbol{u}^{\infty}) &= h(\boldsymbol{u}^{t}) + \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^{t})\right)^{\top}\left(\boldsymbol{u}^{\infty} - \boldsymbol{u}^{t}\right) \\ &= \boldsymbol{c}^{\top}\boldsymbol{x}(\boldsymbol{u}^{t}) + \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^{t})\right)^{\top}\boldsymbol{u}^{t} + \left(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{u}^{t})\right)^{\top}\left(\boldsymbol{u}^{\infty} - \boldsymbol{u}^{t}\right) \\ &= \boldsymbol{c}^{\top}\boldsymbol{x}(\boldsymbol{u}^{t}) - \left(A\boldsymbol{x}(\boldsymbol{u}^{t})\right)^{\top}\boldsymbol{u}^{\infty} + \boldsymbol{b}^{\top}\boldsymbol{u}^{\infty}, \end{aligned}$$

which together with the equality (15.55) and $h^* = h(\boldsymbol{u}^{\infty})$ yield that

$$\boldsymbol{c}^{\top}\boldsymbol{x}(\boldsymbol{u}^{t}) - \left(A\boldsymbol{x}(\boldsymbol{u}^{t})\right)^{\top}\boldsymbol{u}^{\infty} - v^{\infty} = 0$$

holds when t is large enough. □

According to Theorem 15.6, all columns that are needed to solve the complete master problem (15.48) are eventually found in the subgradient optimization. In case of nondegeneracy at a dual optimum, late iterations will provide exactly the optimal basic columns of the complete master problem.[9] Further, columns found in the early iterations are never indispensable. These properties justify the use of subgradient optimization for predicting optimal basic columns before applying column generation.

In practice, the two-phase method works as follows. In the prediction phase, $\bar{t}$ subgradient optimization iterations are performed, with step lengths fulfilling the conditions (15.23) and (15.27). Lagrangian subproblem solutions are collected and columns in the restricted master problem are constructed from iteration $\underline{t} \leq \bar{t}$. Upon termination of this phase, an initial restricted master problem is available. Thereafter, Dantzig–Wolfe decomposition is used as usual. In order to benefit from the two-phase method, the Lagrangian relaxed problem must be computationally cheap in comparison with the restricted master problems, since the prediction phase involves a relatively large number of subgradient optimization iterations.

Theorem 15.3 suggests an ergodic sequence computed within the subgradient optimization method (15.21) that asymptotically solves the problem (15.5), and hence also (15.48). Finitely generated ergodic solutions are, however, typically infeasible. In contrast, a Dantzig–Wolfe restricted master problem—finitely generated within the prediction phase—can be used to find feasible near optimal solutions to (15.5). The key difference is that in the former approach, the primal solution is defined by (15.54) with the convexity weights given a priori by (15.53), while in the latter approach the values of the convexity weights are optimized by the restricted master problem.

[9] In a degenerate dual optimum, nonbasic columns with zero reduced costs can also be obtained.

15.6 Recovery of Primal Integer Solutions by Means of Lagrangian Dual Methods

As an application of Theorem 15.3 we give a partial motivation for the success of Lagrangian heuristics in cases where the Lagrangian lower bound is strong, that is, when the difference $z^* - h^* \geq 0$ is small. We establish that the extent to which one may expect such heuristics to generate feasible solutions of high quality is governed by the same factors as those determining the quality of lower bounds. Hence, a solution strategy that yields a high quality Lagrangian lower bound $h^* = z^*_{\text{conv}} \leq z^*$ is also likely to yield a high quality upper bound $\overline{z} \geq z^*$.

15.6.1 Primal Integer Solutions From Lagrangian Heuristics

The basic idea behind Lagrangian heuristics is to use the information obtained from the Lagrangian dual problem (15.10) to construct feasible solutions to the original problem (15.2). A Lagrangian heuristic commonly works as follows. It is initiated at a solution that is feasible with respect to the non-relaxed constraints (15.2c). This solution is gradually adjusted through a finite number of steps that (1) use information from the Lagrangian dual problem, (2) retain feasible in the non-relaxed constraints (15.2c), and (3) strive for feasibility in the relaxed constraints (15.2b). If the heuristic is successful, the final solution is feasible in (15.2). Lagrangian heuristics are, however, often not guaranteed to find feasible solutions.

To comply with the required initial feasibility in the non-relaxed constraints (15.2c), Lagrangian heuristics are commonly initiated with (near) optimal solutions to the subproblem in (15.8). Appropriate adjustments of solutions are necessarily problem specific and range from simple roundings, via elaborate manipulations of solutions, to solving a mixed-integer linear optimization problem. The information used from the Lagrangian dual problem is typically a near optimal dual solution, obtained by, for example, subgradient optimization. The adjustments made when striving for feasibility in the relaxed constraints (15.2b) are often guided by a merit function defined by original costs or Lagrangian cost, such that the final solution, if feasible, is likely also near optimal. The heuristic may also continue with a local search or meta-heuristic search in the original problem, after feasibility is reached.

We distinguish between two types of Lagrangian heuristics: *conservative* and *radical*. The latter type allows the solution finally found to be far from optimal in the subproblem. A radical heuristic can, for example, solve a *restriction* of the original problem (e.g., a Benders subproblem), which yields a feasible solution.

In conservative heuristics, which are the more common, the initial solution is (near) optimal in the subproblem, the adjustments made are local and such that near optimality in the subproblem is retained, and the number of adjustments is required to be rather small. Due to these limitations, such a heuristic may produce very good

feasible solutions, or frequently fail to find feasible solutions, depending on the characteristics of the original problem and the details of the heuristic.

Initialize a conservative Lagrangian heuristic by a solution $\tilde{\boldsymbol{x}}$ to the subproblem (15.9). Then, the effect of the adjustments made can be characterized by the inclusion

$$\boldsymbol{x}_{\text{heur}}(\tilde{\boldsymbol{x}}) \in \{\, \boldsymbol{x} \in X \mid A\boldsymbol{x} \geq \boldsymbol{b};\ \text{“the distance } \| \boldsymbol{x} - \tilde{\boldsymbol{x}} \| \text{ is small”} \,\}. \tag{15.56}$$

To understand (at least partially) why conservative heuristics can produce very good feasible solutions, we first recall that $\tilde{\boldsymbol{x}}$ is *optimal* in a relaxation of the original problem. Further, in many applications a subproblem solution obtained using a near optimal Lagrangian dual solution is near feasible in the original problem. Moreover, if $\tilde{\boldsymbol{x}}$ is near feasible, the adjustments needed to reach feasibility are small, and a feasible solution $\boldsymbol{x}_{\text{heur}}(\tilde{\boldsymbol{x}})$, will indeed be close to the input $\tilde{\boldsymbol{x}}$. Hence, it appears likely that $\boldsymbol{x}_{\text{heur}}(\tilde{\boldsymbol{x}})$ can be near optimal in the original problem. The risk of failure in a conservative heuristic is due to the adjustments being only local, which may be insufficient to repair intricate infeasibilities.

As the dual sequence $\{\, \boldsymbol{u}^t \,\}$ approaches the set U^*, the ergodic sequence $\{\, \tilde{\boldsymbol{x}}^t \,\}$ will approach the set X^*_{conv}. If the Lagrangian lower bound h^* is strong, that is, if the duality gap $z^* - h^* \geq 0$ is small, the sets X^*_{conv} and X^* are expected to be close, in the sense that the value $\min\{\, \| \boldsymbol{x} - \boldsymbol{y} \| \,\big|\, \boldsymbol{x} \in X^*_{\text{conv}}, \boldsymbol{y} \in X^* \,\}$ is (relatively) small. Since the adjustments made to the ergodic primal iterates are quite small, a heuristic based on the inclusion (15.56) can thus be expected to produce solutions which are close to the set X^*. If, however, as in classical Lagrangian heuristics, the subproblem solutions $\boldsymbol{x}(\boldsymbol{u}^t)$ are used as inputs to the heuristic, the above arguments cannot be used to claim that the resulting point will be close to X^*_{conv} or to X^*. We thus propose to use the ergodic iterate $\tilde{\boldsymbol{x}}^t$ in place of the subproblem solution $\boldsymbol{x}(\boldsymbol{u}^t)$, as input to the Lagrangian heuristic of Algorithm 15.1; it is based on (15.56), that is, to find $\boldsymbol{x}_{\text{heur}}(\tilde{\boldsymbol{x}}^t)$.

Algorithm 15.1: Lagrangian heuristic for (15.2) utilizing ergodic sequences of subproblem solutions

Data: $\boldsymbol{u}^0 \in \mathbb{R}^m_+$; $t := 0$;
Result: a (heuristic) solution $\boldsymbol{x}_{\text{heur}}$ to (15.2);
repeat
  Compute $\boldsymbol{u}^{t+1}$ and $\boldsymbol{x}(\boldsymbol{u}^{t+1})$ according to (15.21), (15.23), (15.27);
  $t := t + 1$;
  Update $\tilde{\boldsymbol{x}}^{t+1}$ according to (15.35) and (15.36) [or (15.38)];
  Compute a solution $\boldsymbol{x}_{\text{heur}}(\tilde{\boldsymbol{x}}^{t+1})$ according to (15.56);
until *the best solution* $\boldsymbol{x}_{\text{heur}}$ *found is satisfactory in* (15.2);

15.6.2 Approximate Solutions to the Primal Problem via Core Problems

A common solution strategy for discrete optimization problems is to use some heuristic, problem adapted technique for predicting optimal values for a (relatively large) subset of the binary variables, and solve the (relatively small) restriction of the original problem to the nonfixed variables—the *core problem*—either exactly or approximately. The core problem should be constructed with the aim of making it feasible; an optimal solution to the core problem then yields a feasible and near optimal solution to the original problem. Whenever the core problem turns infeasible, previously fixed variables need to be relaxed and reinserted in the core.

Let $\mathcal{J}_0^* \subseteq \{1, \ldots, n_\mathrm{b}\}$ and $\mathcal{J}_1^* \subseteq \{1, \ldots, n_\mathrm{b}\} \setminus \mathcal{J}_0^*$ denote the sets of indices for the variables in $\boldsymbol{x}_\mathrm{b}$ which possess the value 0 and 1, respectively, in *every* optimal solution to the convexified problem (15.5). Further, let $\mathcal{J}_\mathrm{frac}^* := \{1, \ldots, n_\mathrm{b}\} \setminus (\mathcal{J}_0^* \cup \mathcal{J}_1^*)$ denote the complementary index set, corresponding to the variables in $\boldsymbol{x}_\mathrm{b}$ which possess a fractional optimal value in *at least one* optimal solution to (15.5).

For each iteration t in the method (15.21), let $\tilde{\boldsymbol{x}}^t$ denote the weighted average of the solutions to the Lagrangian subproblem (15.8) as defined in (15.35), with step lengths according to (15.23), (15.27), and the convexity weights fulfilling the conditions (15.36) or (15.38). For each $j \in \{1, \ldots, n_\mathrm{b}\}$ the value $\tilde{x}_{\mathrm{b},j}^t \in [0, 1]$ can then be interpreted as the weighted relative frequency by which the variable $x_{\mathrm{b},j}$ attains the value 1 in an optimal solution to the subproblem. The following result is immediate.

Proposition 15.10 (On the Weighted Relative Frequency of Binary Solutions) *It holds that* $\left\{ \tilde{x}_{\mathrm{b},j}^t \right\} \to 0$ *for all* $j \in \mathcal{J}_0^*$ *and* $\left\{ \tilde{x}_{\mathrm{b},j}^t \right\} \to 1$ *for all* $j \in \mathcal{J}_1^*$. *If the sequence* $\left\{ \tilde{x}_{\mathrm{b},j}^t \right\}$ *accumulates at a point in the open interval* $(0, 1)$, *then* $j \in \mathcal{J}_\mathrm{frac}^*$.

Proposition 15.10 motivates the use of the weighted relative frequency of the binary subproblem solutions as an indicator for the solution to the convexified optimization problem (15.5), as well as for the solution to the original problem (15.1) [or (15.2)]. For each variable $x_{\mathrm{b},j}$, $j = 1, \ldots, n_\mathrm{b}$, we thus define the two threshold values $\sigma_j^0, \sigma_j^1 \in (0, \frac{1}{2})$, which are used to define the two approximating sets

$$\mathcal{J}_0(\boldsymbol{\sigma}^0, \tilde{\boldsymbol{x}}^t) := \left\{ j \in \{1, \ldots, n_\mathrm{b}\} \,\middle|\, \tilde{x}_{\mathrm{b},j}^t \leq \sigma_j^0 \right\}$$

and

$$\mathcal{J}_1(\boldsymbol{\sigma}^1, \tilde{\boldsymbol{x}}^t) := \left\{ j \in \{1, \ldots, n_\mathrm{b}\} \,\middle|\, \tilde{x}_{\mathrm{b},j}^t \geq 1 - \sigma_j^1 \right\}.$$

A sequence of core problems—to be utilized in Algorithm 15.2—is then defined by

$$z^*_{\text{core}}(\boldsymbol{\sigma}^0, \boldsymbol{\sigma}^1, \tilde{\boldsymbol{x}}^t) := \min_{\boldsymbol{x}_\text{b}, \boldsymbol{x}_\text{c}} \ \boldsymbol{c}_\text{b}^\top \boldsymbol{x}_\text{b} + \boldsymbol{c}_\text{c}^\top \boldsymbol{x}_\text{c} \tag{15.57a}$$

$$\text{subject to} \quad (\boldsymbol{x}_\text{b}^\top, \boldsymbol{x}_\text{c}^\top)^\top \in \{\, \boldsymbol{x} \in X \mid A\boldsymbol{x} \geq \boldsymbol{b} \,\}, \tag{15.57b}$$

$$x_{\text{b},j} = 0, \qquad j \in \mathcal{J}_0(\boldsymbol{\sigma}^0, \tilde{\boldsymbol{x}}^t), \tag{15.57c}$$

$$x_{\text{b},j} = 1, \qquad j \in \mathcal{J}_1(\boldsymbol{\sigma}^1, \tilde{\boldsymbol{x}}^t). \tag{15.57d}$$

We let $\boldsymbol{x}^t_{\text{core}}$ denote a feasible (optimal or approximate) solution to the core problem (15.57), the value of which—whenever feasible—is an upper bound on the optimal value, that is, the inequalities $\boldsymbol{c}^\top \boldsymbol{x}^t_{\text{core}} \geq z^*_{\text{core}}(\boldsymbol{\sigma}^0, \boldsymbol{\sigma}^1, \tilde{\boldsymbol{x}}^t) \geq z^*$ hold. We define $z^t_{\text{core}} := \min_{s=0,\ldots,t} \left\{ \boldsymbol{c}^\top \boldsymbol{x}^s_{\text{core}} \right\}$. Since lower bounds $h(\boldsymbol{u}^t) \leq h^*$ are given by the dual iterates $\boldsymbol{u}^t$, a termination criterion can be based on the differences

$$z^t_{\text{core}} - \overline{h}_t \geq z^* - h^* \geq 0, \tag{15.58}$$

where $\overline{h}_t := \max_{s=0,\ldots,t} \{\, h(\boldsymbol{u}^s) \,\}, \quad t = 1, 2, \ldots$

Algorithm 15.2: Approximate solution of (15.2) from a sequence of core problems

Data: $\tau \in \mathbb{Z}_+$; $\varepsilon > z^* - h^*$; $\boldsymbol{\sigma}^0, \boldsymbol{\sigma}^1 \in (0, \frac{1}{2})^{n_\text{b}}$; $\boldsymbol{u}^0 \in \mathbb{R}^m_+$; $t := 0$;
Result: a (approximate) solution $\boldsymbol{x}_{\text{core}}$ to (15.2) [or (15.1)];
repeat
  Perform τ iterations of the method (15.21), (15.23), (15.27);
  Compute $\overline{h}_\tau$ as defined in (15.58);
  Compute $\tilde{\boldsymbol{x}}^\tau$ as defined in (15.35) and (15.36) [or (15.38)];
  repeat
    Decrease the values of σ^0_j and σ^1_j, $j = 1, \ldots, n_\text{b}$;
    Generate the sets $\mathcal{J}_0(\boldsymbol{\sigma}^0, \tilde{\boldsymbol{x}}^\tau)$ and $\mathcal{J}_1(\boldsymbol{\sigma}^1, \tilde{\boldsymbol{x}}^\tau)$;
  until *the core problem* (15.57) *is feasible*;
  Compute a solution $\boldsymbol{x}^\tau_{\text{core}}$ (exact or approximate) to (15.57);
  Update z^τ_{core};
  Increase the values of σ^0_j and σ^1_j, $j = 1, \ldots, n_\text{b}$;
  $\boldsymbol{u}^0 := \boldsymbol{u}^\tau$, $t := 0$;
until $z^\tau_{\text{core}} - \overline{h}_\tau \leq \varepsilon$, *or the solution* $\boldsymbol{x}^\tau_{\text{core}}$ *is satisfactory in* (15.2);

15.6.3 Optimal Solutions via a Branch-and-Bound Framework

We next consider using ergodic sequences to obtain feasible solutions to the problem (15.2) within a branch-and-bound framework. A subgradient method (15.21) can be

applied to the Lagrange dual (15.10) of the local linear program corresponding to each branch-and-bound tree node, yielding

(1) a *lower bound* on z^* from a lower estimate of h^* (that is, from the value of an approximate solution to the Lagrangian dual of the node problem),
(2) an *upper bound* on z^* from feasible solutions to (15.2) constructed using the subproblem solutions obtained in (15.21), and
(3) a *branching decision* based on the (approximate) *reduced costs* obtained in the dual sequence $\{ \boldsymbol{u}^t \}$.

One drawback of this principle is that it seldom provides a fractional primal solution, since the subproblem solutions are integer valued. So when aiming for a breadth-first branching, deciding on which variable to branch on is nontrivial. We propose a procedure in which the upper bounding (2) and the branching rule (3) are replaced by measures based on the ergodic iterates $\tilde{\boldsymbol{x}}^t$. The upper bound is obtained by applying Algorithm 15.1, which provides feasible solutions to the problem (15.2). The branching rule is based on the ergodic iterate $\tilde{\boldsymbol{x}}^t$ obtained from the method (15.21), (15.23), (15.27), (15.35) and (15.36) [or (15.38)]. Branching can be done on the variables with values close to binary or close to $\frac{1}{2}$. The optimization problem addressed in node n of the branch-and-bound tree is then the problem (15.5), with the additional constraints $x_{\mathrm{b},j} = \ell$, $j \in \mathcal{I}_\ell^n$, $\ell \in \{0, 1\}$, where the set $\mathcal{I}_\ell^n$ contains the indices of the variables that have been fixed to ℓ in the parent node of node n. By defining the set $X_{\mathrm{conv}}^n := \mathrm{conv}\left\{ \boldsymbol{x} \in X \mid x_{\mathrm{b},j} = \ell,\ j \in \mathcal{I}_\ell^n,\ \ell \in \{0, 1\} \right\} \subseteq X_{\mathrm{conv}}$, this linear program can then be expressed as

$$z_n^* := \min_{\boldsymbol{x}} \quad \boldsymbol{c}^\top \boldsymbol{x} \tag{15.59a}$$

$$\text{subject to} \quad A\boldsymbol{x} \geq \boldsymbol{b}, \tag{15.59b}$$

$$\boldsymbol{x} \in X_{\mathrm{conv}}^n. \tag{15.59c}$$

The branching procedure of Algorithm 15.3 is then applied, where the dual starting point $\boldsymbol{u}^0$ in step 1 is often chosen as the final point $\boldsymbol{u}^\tau$ obtained from the subgradient scheme for the parent node. The search strategy for the branch-and-bound tree can be defined as, for example, depth-, breadth-, or best lower bound-first.

Algorithm 15.3: Branching decision based on ergodic primal iterates

Data:	$\tau \in \mathbb{Z}_+$; $\boldsymbol{u}^0 \in \mathbb{R}_+^m$; $t := 0$.
Step 1.	Apply τ iterations of Algorithm 15.1 to the linear optimization problem (15.59); this yields lower and upper bounds on z_n^*.
Step 2.	Based on the lower and upper bounds on z^*, decide whether or not branching should be performed.
Step 3.	Perform a branching based on the ergodic iterate $\tilde{\boldsymbol{x}}^\tau$: branch on a variable $x_{\mathrm{b},j}$ with $\tilde{x}_{\mathrm{b},j}^\tau$ close to $\frac{1}{2}$, or close to 0 or 1.

15.7 Notes and Further Reading

In this section we collect notes regarding the various results presented in the former sections and review important background material, along with references to the literature. While mentioning a few of the classic articles in the field of Lagrangian duality—chosen for their eloquence and pioneer status rather than for representing the state of the art—we also include a selection of more recent articles in the field—that we think tie in well with the classics.

Notes on Sect. 15.2.1: The Lagrangian Dual That the Lagrangian dual function h is piecewise affine and concave is shown in, for example, [11, Proposion 5.1.12]. That the dual optimal set U^* is nonempty and polyhedral when the feasible set of the convexified problem (15.5) is nonempty is verified in, for example, [11, Section 5]. When the feasible set of (15.5) is empty, however, the Lagrangian dual (15.10) is unbounded; conditions for the existence of an optimal Lagrangian dual set are detailed in [74].

In [31] Everett establishes a theorem for the Lagrangian function, which states that for a given vector of dual multipliers (not necessarily optimal), the minimization of the Lagrangian function over the primal variables yields a primal vector that is globally optimal for a primal problem whose available resources are identical to those utilized in the primal minimum of the Lagrangian function for the given multipliers. This result suggests that dual multipliers are near optimal when the minimum of the corresponding Lagrangian subproblem is near feasible. Brooks and Geoffrion extend in [15] the analysis in [31] to provide near optimal solutions also when the original problem may include, for example, integer variables.

In [46, 47] Held and Karp investigate approaches to the symmetric traveling salesperson problem, based on a 1-tree relaxation.[10] One seeks prices (i.e., multipliers) on the nodes (which, however, appear on the adjacent links) such that the cheapest 1-tree equals an optimal Hamiltonian cycle. Optimal prices are derived from the solution of the corresponding Lagrangian dual problem, which is to maximize the value of the 1-tree. The lower bounds obtained from the dual procedure are also utilized in a branch-and-bound procedure.

Guignard provides in [39] an extensive introduction to Lagrangean approaches for the exact or approximate solution of difficult combinatorial optimization problems. The theme is similar to ours, while it is aimed at less experienced readers.

A quite little studied form of Lagrangian relaxation—not covered by this chapter—is known as *Lagrangian decomposition*, or variable splitting; see [40, 53]. It can be applied to mixed-integer linear optimization problems with (at least) two sets of explicit constraints [in (15.1b), (15.2b), or (15.7b)], such that two different Lagrangian relaxations are possible. The starting point for this approach is a problem reformulation in which copies of (a subset of) the primal variables are introduced, in one of the sets of constraints, together with additional constraints

[10] A 1-tree is the union of a spanning tree and one additional link.

that ensure consistency between the original variables and the copies. The consistency constraints are then Lagrangian relaxed. The corresponding Lagrangian dual problem can yield stronger bounds than both of the two possible straightforward Lagrangian relaxations of the original problem, provided that neither of the LP relaxations of the resulting Lagrangian subproblems possesses the integrality property.

Notes on Sect. 15.2.2: Optimality Conditions for the Convexified Problem While some literature on nondifferentiable functions (naturally) draw a distinction between 'supdifferentials of concave functions' and 'subdifferentials of convex functions', we use the term subdifferential all through, relying on the readers' familiarity with the concept.

The subdifferential of a concave function is defined in [11, Definition 3.2.3]. Proposition 15.1 follows from [11, Theorem 6.3.7], the convexity of the set X_{conv}, and [64, Theorem 11]. Further, the differentiability property of the dual function h relying on the singleton property of the set $X_{\text{conv}}(\mathbf{u})$ is depicted in [11, Theorem 6.3.3]. For a proof of Proposition 15.4, see [11, Theorem 3.4.3]. The saddle point optimality in (15.13) is proven in [11, Theorem 6.2.4], while Proposition 15.5 follows from [11, Theorem 6.2.5]. In [17, Lemma 2] it is shown that the composite mapping $\partial h \cap N_{\mathbb{R}^m_+}$ is constant on the solution set U^*. The definition of the subdifferential of an objective function is in [24, 25] generalized to take the feasible set into account; in our terminology it is called the *conditional subdifferential*. The *non-coordinability* phenomenon is a consequence of the nondifferentiability of the Lagrangian dual function, which in turn is a consequence of the linearity of the primal problem. This phenomenon is of interest in economic systems which are described by linear models and where *constant prices* are used as a tool for steering decentralized decisions towards system optimality (e.g., [27]).

Turning to the question of how to exploit the Lagrangian dual problem (15.10) as a tool for finding an (near) optimal solution to the mixed-binary linear optimization problem (15.2), the fundamental observation is that the dual problem effectively solves the convexified primal problem (15.5), that is, finds an element of the solution set X^*_{conv}. Worth noting, however, is that whether such an element is directly available or not depends on the solution strategy employed for the dual problem.[11] If an (near) optimal, solution to the convexified problem (15.5) is at hand, a simple strategy to find a mixed-binary feasible solution is to employ *rounding*, that is, to systematically round the values of $\boldsymbol{x}_{\text{b}}$ in a solution to the convexified problem to binary values, and adjust the values of $\boldsymbol{x}_{\text{c}}$ accordingly, with the aim of reaching feasibility and near optimality in the original problem (15.2).

Simple rounding procedures are, however, in general inadequate for finding near optimal or even feasible solutions. A more general idea is *randomized rounding* (see

[11] For example, a cutting-plane scheme for the dual problem identifies an element of X^*_{conv} while a subgradient optimization scheme does not. The key difference is that the former scheme provides, at termination, an optimal dual solution to the Lagrangian dual problem, that is, an optimal primal solution, while this is not the case for the latter scheme.

[78]), which for certain applications works as approximation algorithms. A generic such technique entails the solution of a continuous relaxation of the original problem, followed by a randomization scheme to decide whether to round up or down. The article [12] provides a framework for finding approximate solutions to covering problems through generic heuristics, all based on rounding (deterministic—using primal and dual information—or randomized—with nonlinear rounding functions) of an optimal solution to a LP relaxation. The roundings are applied to several known, as well as to new, results for the set covering, facility location, general covering, network design, and cut covering problems. The follow-up article [13] describes a class of structure exploiting methods to round fractional solutions, by introducing dependencies in the process. The technique improves approximation bounds for several problems, including the min-k-SAT problem.

The reader may note that rounding and randomized rounding strategies have in common with the core problem principle (see Sect. 15.6.2) that they identify and elaborate with a subset of the original variables, based on primal information (e.g., variable values) or dual information (e.g., Lagrangian reduced costs).

Notes on Sect. 15.2.3: Conditions for Optimality and Near Optimality of Mixed-Binary Linear Optimization Problems The *global primal–dual optimality conditions* (15.15) and the equivalent *near saddle point condition* (15.16) were introduced in [57], in which also computational results are reported. For examples of solution methods related to the enumeration principle discussed in Remark 15.2, see [45] and [21] (constrained shortest path problems), and [20] (train timetabling).

Notes on Sect. 15.3: Conditional Subgradient Optimization Subgradient optimization methods for minimizing non-differentiable convex functions originate in a work by Shor from 1962; [83] reviews the early history of nonsmooth optimization. For the case of unconstrained optimization, Ermol'ev established in [30] the convergence of the method using step lengths according to a *divergent series*. Polyak extended in [76, 77] the method to the case of constrained convex optimization and presented additional convergence results; see also [82, Section 2]. These methods have been frequently and often successfully applied, particularly in connection with Lagrangian duality; see, for example, [32, 33, 36, 48]. Worth noting is that subgradient optimization methods are closely related to *relaxation methods* for solving systems of linear inequalities; see [37].

The important *Polyak step length* rule, which has proved to be very useful in computational practice, was presented in [77]. For the case when the optimal objective value h^* is known a priori, convergence to an optimal solution by the method (15.21) using the step lengths (15.30) was established: the restrictions on the step length parameter θ_t guarantee that the distance between the current iterate and the solution set decreases in every iteration. Finite convergence to a near optimal point was also established for the case when the optimal value h^* in (15.30) is replaced by an estimate $\overline{h} \geq h^*$; this generalization, expressed in (15.31), is the most commonly used in practice.

Notes on Sect. 15.3.1: Basic Convergence in the Lagrangian Dual Problem The *conditional subgradient* optimization method was presented in [58] (see also [24, 25]) and generalizes the subgradient optimization method; it includes as a special case the *subgradient projection* method, as given in (15.22), which has shown a better practical performance than traditional subgradient methods (see [58]). The convergence of the method (15.21) was established in [58] for the divergent series and the Polyak step length rules. The proofs of Theorems 15.1 and 15.2 are special cases of those given in [58, Theorems 2.6 and 2.7, respectively]. The special case of subgradient projection is detailed in [58, Section 3]. Proposition 15.7 can be proven analogously as [77, Theorem 1] adapted to a Lagrangian dual problem, while Proposition 15.8 follows from [77, Theorem 4]. The condition (15.32) is referred to as the *almost complete relaxation* strategy; see [26, Section 3.4] and [58, Corollary 2.8].

Notes on Sect. 15.3.2: Ergodic Convergence in the Primal Problem The principle of constructing *ergodic sequences* of primal subproblem solutions in subgradient optimization can be traced back a long time; see [82, pp. 117] and [2]. The results presented here are developed in a series of articles; see [56, 59, 60]. Proposition 15.9 and Theorem 15.3 are special cases of [60, Proposition 5 and Theorem 1, respectively]. The convergence of sequences of convex combinations in general relies on a result in [54] and which is described in [44, Lemma 3].

A relative to the principle of constructing ergodic primal solutions, and also to the two-phase method presented in Sect. 15.5, is the *volume algorithm* of Barahona and Anbil in [8]. Referencing the classic works of Held and Karp [46, 47] and Held et al. [48] on the search for good lower bounds for computationally challenging *large-scale optimization* problems, Barahona and Anbil identify the drawback of not considering the convergence characteristics in the primal space and of the lack of a natural stopping criterion. While at the outset admitting the lack of a complete convergence theory, the authors describe a dual scheme, which is similar to a classic conjugate subgradient method (see [93]), in tandem with a constructive primal heuristic that mimics the master problem in Dantzig–Wolfe decomposition. The volume algorithm is applied to large-scale linear optimization problems arising from continuous relaxations of set partitioning, set covering, airline crew scheduling, max-cut, and facility location problems. The conclusion from the numerical experiments is that the more favourable problem instances are those in which variables are bounded within the interval $[0, 1]$, constraint coefficients lie in the set $\{0, 1, -1\}$, and the pricing problem is solvable in linear time.

Notes on Sect. 15.3.3: Enhanced Primal Ergodic Convergence Ergodic sequences that exploit more information from later subproblem solutions (as compared to earlier ones) were first presented in 1996 by Sherali and Choi [81] for the case of LP; this construction of ergodic sequences was generalized in 2015 by Gustavsson et al. [44] to incorporate also general convex optimization, as well as refined to the so-called s^k-rule. Theorem 15.4 follows from [44, Theorem 1]. The result that the requirements (15.38) on the convexity weights μ_s^t combined with step

lengths α_t according to a modified harmonic series (15.37) imply the requirements (15.36b)–(15.36e) on the parameters γ_s^t is established in [44, Proposition 5]. Convergence analyses for the s^k-rule are presented in [44], while the delayed initialization of the ergodic sequences according to Remark 15.8 is detailed in [60, Section 3, Remark 1].

Notes on Sect. 15.3.4: Finite Primal Feasibility and Finite ε-Optimality Theorem 15.5 and the finiteness result in Corollary 15.2 are established in [60, Theorem 3 and Corollary 3, respectively]. Approximate solution of the Lagrangian subproblems yields directions being ε-subgradients; the resulting *conditional ε-subgradient* method is investigated and analysed in [61]. Finiteness results are also presented in [63], in which the ergodic sequences are generated within a *simplicial decomposition* framework for nondifferentiable convex optimization.

A characterization in the case of a possibly inconsistent primal problem (15.5) is carefully detailed in [74], for the more general case of convex optimization. Convergence is established of an ergodic sequence of subproblem solutions to a point in the primal space such that the Euclidean norm of the infeasibility in the relaxed primal constraints is minimized, while the sequence of dual iterates diverges along the feasible direction of steepest ascent for the Lagrangian dual function.

Notes on Sect. 15.4: Dual Cutting-Planes: Dantzig–Wolfe Decomposition The article "Decomposition principle for linear programs" published in 1960 by Dantzig and Wolfe [23] is a pioneering work on the subject of large-scale optimization and among the most influential publications in the history of operations research. The Dantzig–Wolfe decomposition principle is founded on the *representation theorem for polyhedra* (e.g., [65, Chapter 3, Theorem 3]), which states that a point belongs to a polyhedron if and only if it can be expressed as a convex combination of the polyhedron's extreme points plus a nonnegative linear combination of its extreme directions.

Dantzig–Wolfe decomposition is derived from the structure of the constraints of the linear program. It is assumed that the constraints can be partitioned into two sets, of which one is (relatively) computationally tractable. The other set of constraints is deemed complicating; hence these are Lagrangian relaxed—giving the method's subproblem—and instead dealt with in a separate linear program—the restricted master problem. For practical problems Dantzig–Wolfe decomposition typically exploits a block diagonal structure of the subproblem, which can then be solved as several separate subproblems. Assuming that the subproblems always have finite optima, the optimal solutions obtained for different price vectors are—in the restricted master problem—convex combined, such that the complicating constraints are optimally utilized, with respect to the problem's objective function.[12] Subject to the usual non-degeneracy assumptions (or means for dealing with degen-

[12]The case of unbounded subproblem solutions can also be handled, by finding feasible directions along which the objective value is unbounded and—in the restricted master problem—considering nonnegative linear combinations of such directions.

eracy), the convergence is finite; this follows directly from the finite convergence of the simplex method, of which Dantzig–Wolfe decomposition is, in essence, an application.

Plenty of variations of this computational scheme are possible. For example, each subproblem can be terminated at a near optimal solution, as long as this solution is good enough to ensure some progress in the restricted master problem. Further, the solution of the restricted master problem can be truncated, as long as some progress has been made. Further, convergence is ensured even if only the current basis is maintained in the restricted master problem, in a revised simplex manner.

The extension of the Dantzig–Wolfe decomposition principle to the case of nonconvex [e.g., (mixed-) integer] or non-linear optimization problems is known as *generalized LP*. In [68], a fundamental property of generalized LP is established. It is shown that for nearly all problems of practical importance, any limit point of the sequence of dual solutions produced by the algorithm is optimal in the Lagrangian dual problem of the given, primal, problem. This result holds even if the generalized LP algorithm does not solve the primal problem, which is typically the case whenever this problem is nonconvex.

Among the first applications of the Dantzig–Wolfe decomposition principle to a mixed-integer optimization problem is the work in [29] by Dzielinski and Gomory from 1965; see also [27, Section 7.2]. They consider a problem of production and inventory planning, which can be characterized as a time indexed, multi product economic lot size scheduling problem with common production resources. The problem is straightforward to model as a mixed-integer optimization problem, but they instead consider an approximate linear optimization model, which is based on the work [69] by Manne from 1958 and in which each variable corresponds to a complete production schedule for a single product. The approximate model can be regarded as a Dantzig–Wolfe master problem for a convexified version of a mixed-integer model of the problem. Since the number of possible production schedules can be huge, column generation is used. The column generation problem (i.e., the Dantzig–Wolfe subproblem) finds a production schedule for a single product and can be efficiently solved by the well-known Wagner–Whitin lot-sizing algorithm (which is a dynamic programming scheme), while the restricted master problem optimally combines the available production schedules with respect to overall cost and the common production resources.

In the pioneering two-part work [3, 4] on the application of column generation in the field of vehicle routing, Appelgren describes column generation approaches to a ship scheduling problem, obtained from a Swedish ship owning company. The first article applies the Dantzig–Wolfe decomposition method to the LP relaxation of the scheduling problem, which—probably thanks to the favourable matrix structure—achieves solutions that are near integer. In the second article, the column generation method is combined with a branch-and-bound algorithm, in which the branching is performed on one of the "essential" fractional variables and the bounds are obtained by the decomposition algorithm. This combined method was able to solve all problem instances tested, mostly with one branching only.

The work by Appelgren is an early predecessor to what is today known as *branch-and-price*. (The term price refers to the column generation, i.e., the pricing problem.) In the article [10] the authors first summarize the relations between *branch-and-cut* and branch-and-price for (mixed-) integer optimization problems. In branch-and-cut, classes of *valid inequalities*, preferably facets of the convex hull of the set of feasible solutions, are left out of the LP relaxation, because they comprise far too many constraints to generate and handle efficiently, and most of them will neither be binding in an optimal solution. Then, if an optimal solution to a LP relaxation is infeasible (with respect to integrality restrictions), a subproblem—called the separation problem—is solved in order to identify violated inequalities in a class. If one or more violated inequalities are found, some of them are added to the linear optimization problem, in order to cut off the infeasible solution, followed by its re optimization. Branching occurs only when no violated inequalities can be found. Branch-and-cut is thus a generalization of branch-and-bound with LP relaxations, which allows cutting to be applied throughout the branch-and-bound tree.

The philosophy of branch-and-price resembles that of branch-and-cut, except that the focus is on column generation instead of row generation. In fact, pricing and cutting are complementary procedures for tightening a LP relaxation. In branch-and-price, sets of columns are left out of the LP relaxation because they are far too many to be generated and handled efficiently, and most of the associated variables will anyway be zero valued in an optimal solution. To check the optimality of a LP solution, a subproblem—the *pricing problem*, which is a separation problem for the dual linear program—is solved in order to identify columns to enter the basis, and the linear program is re-optimized. Branching occurs when no columns price out to enter the basis and the linear programming solution does not satisfy the integrality conditions. branch-and-price is a generalization of branch-and-bound with linear programming relaxations, which allows column generation to be applied throughout the branch-and-bound tree.

While appearing contradictory at first, there are several reasons (see [10]) for considering formulations with huge numbers of variables. Not infrequently a mixed-integer optimization formulation with many variables has a better LP relaxation (with respect to bound quality). Further, a compact formulation—which is a formulation *not* involving a huge number of variables—of a mixed-integer optimization problem may possess structural symmetries that allows solutions being mathematically different but having indifferent real-life interpretations; this causes branch-and-bound perform poorly as the problem barely changes after branching. A reformulation with a huge number of variables may eliminate such symmetries. Further, column generation provides a decomposition of the problem into a master problem and one or more subproblems. This decomposition may have a natural interpretation in the problem context, thus allowing for the incorporation of additional important constraints. Finally, a formulation with a huge number of variables may be the only choice.

At first glance, it may seem that branch-and-price involves nothing more than combining well-known ideas for solving linear programs by column generation and traditional branch-and-bound. This is, however, not that straightforward, as

observed already many years ago by Appelgren in [3, 4]. The most fundamental difficulty arising is that the traditional single variable branching is no longer viable, the reason being that it leads to *regeneration* of (already available) columns. Appelgren resolved this by finding a best column among those that are not already available.

Today, the common countermeasure is to use other branching techniques, which are compatible with column generation by allowing branching restrictions to be transferred to the column generation problem without leading to essential changes of its properties. Vanderbeck analyzes in [88] the challenges in combining a branching scheme with column generation. The article presents a generic branching scheme in which the pricing oracle of the root node remains of use after branching, and which does not require an extended formulation of the original problem. It then recursively partitions the subproblem solution set. Branching constraints are enforced in the pricing problem, which is solved approximately by a limited number of calls to the pricing oracle. The scheme is illustrated on the cutting stock and bin packing problems; it is the first branch-and-price algorithm capable of solving such problems to integrality without modifying the subproblem or expanding its variable space.

An early and tidy application of branch-and-price is to the generalized assignment problem [80], which is decomposed into a set partitioning master problem and knapsack column generation problems. Another tidy application of branch-and-price is given in [87], which considers a time-indexed (i.e., time discretized) formulation of a machine scheduling problem. Such formulations are known to provide strong LP bounds, but they tend to be extremely large. The authors show how to (partly) alleviate this difficulty by means of Dantzig–Wolfe decomposition, leading to a reformulation with many more variables, but far fewer constraints. The central pricing problem is solved by dynamic programming in $O(nT)$ time, with T and n being the number of time steps and jobs, respectively. To find an integer optimum, the decomposition approach is embedded in a branch-and-bound scheme.

For general surveys of column generation and branch-and-price, see [66, 92]. A recent research trend in mixed-integer optimization is to develop effective and efficient solution methods by combining decomposition approaches with heuristic or metaheuristic principles, in order to exploit their respective advantages. For a general overview of metaheuristic methods based on decomposition principles, see [79]. In a column generation context, such a combined method would extend a heuristic search beyond the columns necessary for solving the LP master problem. Classes of column generation based primal heuristics for mixed-integer linear optimization are reviewed in [51], with the aim to extract generic classes of column generation methods for use as black-box primal heuristics across applications. One such class consists of the so-called *diving heuristics*, which perform depth first searches in a branch-and-price tree, gradually obtaining integer solutions by variable fixings according to branchings which priorities columns that are part of LP optimal solutions in the nodes. To escape from local optima, partial backtracking can be used. Examples of applications of diving heuristics are found in, for example, [35, 41].

An interesting topic for further studies is to use an enumeration principle of the type outlined in Remark 15.2 with the aim to find favourable columns for inclusion in a restricted master problem. The goal is then to construct a restricted master problem capable of identifying an (near) optimal solution to the original problem (15.2), rather than to directly find an optimal solution through enumeration. This strategy is reasonable if the original problem has a Cartesian product structure—see Remarks 15.1 and 15.10—such that columns from different sets in the Cartesian product can be combined in order to achieve overall (near) optimality. Solution methods related to this strategy are found in [86] (a production planning problem) and [7] (a vehicle routing problem, not having a Cartesian product structure but a related suitable structure.)

Another interesting topic for further studies within the field of column generation is the little studied *combination of Lagrangian decomposition* (see [40, 53]) *and Dantzig–Wolfe decomposition* (i.e., dual cutting-planes) for solving the Lagrangian dual problem; see [72, 75, 94] for examples of this combination. Both Lagrangian decomposition and Dantzig–Wolfe decomposition can separately provide strong lower bounds, and the synergy between these two bounding principles has the potential to provide even stronger lower bounds.

Notes on Sect. 15.5: A Two-Phase Method: Subgradient Optimization and Dantzig–Wolfe Decomposition The inherent instability of dual cutting-plane procedures is discussed in, for example, [49, Chapter 15]. The fundamental *dual box step stabilization* was introduced in [70]. Examples of applications of this type of stabilized dual cutting-plane method (i.e., stabilized Dantzig–Wolfe decomposition) are found in [62, 90]. More general stabilization techniques are given in [28]. The use of heuristically generated high quality starting columns in Dantzig–Wolfe decomposition is discussed in, for example, [66, Subsection 4.1.1].

The two-phase method was introduced in [91], which also reports successful computational experience from an application to large-scale multicommodity network flows. Some similar methods can be found in the literature; in contrast to the one presented, those methods are, however, not justified by theoretical results.

In [95], subgradient optimization is performed in a first phase, which—in each iteration—stores dual cuts and solves a Dantzig–Wolfe master problem; the objective value of the latter is used in the Polyak step length formula. If the same cut is found too many times, the method switches to a second phase: the Dantzig–Wolfe method. A drawback of this two-phase method is that a computationally demanding master problem is solved in each subgradient iteration of the first phase, although it is only the objective value of the master problem that is actually used. This two-phase method is further developed in [89], which studies different criteria for switching to the second phase, in which the effect of using a bundle method is also studied. Also [55] studies a combined subgradient optimization and bundle method.

In [9] a fixed number of subgradient iterations is run every few iterations of the Dantzig–Wolfe method, starting from the restricted master dual optimum. The columns found are then included in the master problem. Substantially shorter

computing times are reported, as compared to the standard Dantzig–Wolfe method. This line of research is continued in [50], which employs subgradient optimization also for finding approximate dual solutions to Dantzig–Wolfe master problems.

Notes on Sect. 15.6: Recovery of Primal Integer Solutions by Means of Lagrangian Dual Methods Besides our own stream of research, the recovery of primal solutions has been treated in, for example, [67, 81, 84].

Notes on Sect. 15.6.1: Primal Integer Solutions from Lagrangian Heuristics A classic reference on Lagrangian relaxation and *Lagrangian heuristics* in integer optimization is [32] by Fisher from 1981. In [57] the characteristics of the Lagrangian heuristic principle is described more formally and such heuristics are classified as *conservative* or *radical*, depending on their nature. The essential difference is that in a conservative Lagrangian heuristic, the goal is to keep the values of $\varepsilon(\boldsymbol{x}, \boldsymbol{u})$ and $\delta(\boldsymbol{x}, \boldsymbol{u})$ [defined in (15.19)] small, while in a radical heuristic they may be large. A conservative heuristic typically starts at $\boldsymbol{x}(\boldsymbol{u}) \in X(\boldsymbol{u})$ and makes small changes, while a radical often involves solving an auxiliary optimization problem over a subset of the original problem variables, while keeping the values of the other variables fixed. For examples of conservative heuristics for specific applications, see [18, 34, 38, 52]. Examples of radical heuristics are found in [16, 22, 73, 85]; all of these exploit auxiliary optimization problems, some of which, however, being trivially solved.

The presented Lagrangian heuristic methodology utilizing ergodic sequences is developed in [1, 43].

Notes on Sect. 15.6.2: Approximate Solutions to the Primal Problem via Core Problems The use of *core problems* was introduced by Balas and Zemel [6] in 1981, then applied to binary knapsack problems; a collection of improvements of their scheme is found in [71].

The core problem principle has also been applied to very large-scale set covering models arising in crew scheduling [19, 22]; the construction of the core problem is there based on near optimal LP reduced costs found by Lagrangian relaxation and subgradient optimization. Further applications of core problems include capacitated facility location [5] and fixed charge transportation [94].

Our procedure for constructing core problems using ergodic sequences of Lagrangian subproblem solutions is developed in [42, 43].

Notes on Sect. 15.6.3: Optimal Solutions via a Branch-and-Bound Framework The use of dual subgradient methods and Lagrangian heuristics as a means for obtaining branching rules and bounds in branch-and-bound schemes is well studied (e.g., [14, 32]). The utilization of ergodic sequences to guide the branching in a branch-and-bound framework is developed in [1, 43].

Acknowledgements This chapter relies heavily on research and earlier publications by the authors and their collaborators. The most important of these publications are—in chronological order—[56], [58], [60], [57], [44], [43], [91], and [1].

References

1. Aldenvik, P., Schierscher, M.: Recovery of primal solutions from dual subgradient methods for mixed binary linear programming; a branch-and-bound approach. Master's Thesis, University of Gothenburg, Göteborg (2015)
2. Anstreicher, K.M., Wolsey, L.A.: Two "well-known" properties of subgradient optimization. Math. Program. **120**(1), 213–220 (2009)
3. Appelgren, L.H.: A column generation algorithm for a ship scheduling problem. Transp. Sci. **3**(1), 53–68 (1969)
4. Appelgren, L.H.: Integer programming methods for a vessel scheduling problem. Transp. Sci. **5**(1), 64–78 (1971)
5. Avella, P., Boccia, M., Sforza, A., Vasil'ev, I.: An effective heuristic for large-scale capacitated facility location problems. J. Heuristics **15**(6), 597–615 (2009)
6. Balas, E., Zemel, E.: An algorithm for large zero-one knapsack problems. Oper. Res. **28**(5), 1130–1154 (1980)
7. Baldacci, R., Christofides, N., Mingozzi, A.: An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Math. Program. **115**(2), 351–385 (2008)
8. Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Math. Program. **87**(3), 385–399 (2000)
9. Barahona, F., Jensen, D.: Plant location with minimum inventory. Math. Program. **83**(1–3), 101–111 (1998)
10. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: column generation for huge integer programs. Oper. Res. **46**(3), 316–329 (1998)
11. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms, 2nd edn. Wiley, New York (2006)
12. Bertsimas, D., Vohra, R.: Rounding algorithms for covering problems. Math. Program. **80**, 63–89 (1998)
13. Bertsimas, D., Teo, C., Vohra, R.: On dependent randomized rounding algorithms. Oper. Res. Lett. **24**(3), 105–114 (1999)
14. Borchers, B., Mitchell, J.E.: An improved branch and bound algorithm for mixed integer nonlinear programs. Comput. Oper. Res. **21**(4), 359–367 (1994)
15. Brooks, R., Geoffrion, A.: Finding Everett's Lagrange multipliers by linear programming. Oper. Res. **14**(6), 1149–1153 (1966)
16. Brown, G.G., Geoffrion, A.M., Bradley, G.H.: Production and sales planning with limited shared tooling at the key operation. Manag. Sci. **27**(3), 247–259 (1981)
17. Burke, J.V., Ferris, M.C.: Characterization of solution sets of convex programs. Oper. Res. Lett. **10**(1), 57–60 (1991)
18. Campbell, G.M., Mabert, V.A.: Cyclical schedules for capacitated lot sizing with dynamic demands. Manag. Sci. **37**(4), 409–427 (1991)
19. Caprara, A., Fischetti, M., Toth, P.: A heuristic method for the set covering problem. Oper. Res. **47**(5), 730–743 (1999)
20. Caprara, A., Fischetti, M., Toth, P.: Modeling and solving the train timetabling problem. Oper. Res. **50**(5), 851–861 (2002)
21. Carlyle, W.M., Royset, J.O., Wood, R.K.: Lagrangian relaxation and enumeration for solving constrained shortest-path problems. Networks **52**(4), 256–270 (2008)
22. Ceria, S., Nobili, P., Sassano, A.: A Lagrangian-based heuristic for large-scale set covering problems. Math. Program. **81**(2), 215–228 (1998)
23. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Oper. Res. **8**(1), 101–111 (1960)
24. Dem'yanov, V.F., Shomesova, V.K.: Conditional subdifferentials of convex functions. Soviet Math. Doklady **19**, 1181–1185 (1978)

25. Dem'yanov, V.F., Shomesova, V.K.: Subdifferentials of functions on sets. Cybernetics **16**(1), 24–31 (1980)
26. Dem'yanov, V.F., Vasil'ev, L.V.: Nondifferentiable Optimization. Optimization Software, New York (1985)
27. Dirickx, Y.M.I., Jennergren, L.P.: System Analysis by Multilevel Methods: with Applications to Economics and Management. Wiley, Chichester (1979)
28. du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P.: Stabilized column generation. Discret. Math. **194**(1–3), 229–237 (1999)
29. Dzielinski, B.P., Gomory, R.E.: Optimal programming of lot sizes, inventory and labor allocations. Manag. Sci. **11**(9), 874–890 (1965)
30. Ermolev, Y.M.: Methods for solving nonlinear extremal problems. Cybernetics **2**(4), 1–14 (1966)
31. Everett, H.: Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper. Res. **11**(3), 399–417 (1963)
32. Fisher, M.L.: The Lagrangian relaxation method for solving integer programming problems. Manag. Sci. **27**(1), 1–18 (1981)
33. Fisher, M.L.: An applications oriented guide to Lagrangian relaxation. Interfaces **15**(2), 10–21 (1985)
34. Fisher, M.L.: Optimal solution of vehicle-routing problems using minimum K-trees. Oper. Res. **42**(4), 626–642 (1994)
35. Gamache, M., Soumis, F., Marquis, G., Desrosiers, J.: A column generation approach for large-scale aircrew rostering problems. Oper. Res. **47**(2), 247–263 (1999)
36. Geoffrion, A.M.: Lagrangian relaxation for integer programming. In: M.L. Balinski (ed.) Approaches to Integer Programming. Mathematical Programming Study, vol. 2, pp. 82–114 Springer, Berlin (1974)
37. Goffin, J.L.: Nondifferentiable optimization and the relaxation method. In: Lemaréchal, C., Mifflin, R. (eds.) Nonsmooth Optimization. Proceedings of a IIASA Workshop 1977, pp. 31–49. Pergamon Press, Oxford (1978)
38. Graves, S.C.: Using Lagrangean techniques to solve hierarchical production planning problems. Manag. Sci. **28**(3), 260–275 (1982)
39. Guignard, M.: Lagrangean relaxation. Top **11**(2), 151–228 (2003)
40. Guignard, M., Kim, S.: Lagrangean decomposition: a model yielding stronger Lagrangean bounds. Math. Program. **39**(2), 215–228 (1987)
41. Günlük, O., Kimbrel, T., Ladanyi, L., Schieber, B., Sorkin, G.B.: Vehicle routing and staffing for sedan service. Transp. Sci. **40**(3), 313–326 (2006)
42. Gustavsson, E.: Topics in convex and mixed binary linear optimization. Ph.D. Thesis, University of Gothenburg, Göteborg (2015)
43. Gustavsson, E., Larsson, T., Patriksson, M., Strömberg, A.-B.: Recovery of primal solutions from dual subgradient methods for mixed binary linear programming (2015). Preprint. Chalmers University of Technology and University of Gothenburg
44. Gustavsson, E., Patriksson, M., Strömberg, A.-B.: Primal convergence from dual subgradient methods for convex optimization. Math. Program. **150**(2), 365–390 (2015)
45. Handler, G.Y., Zang, I.: A dual algorithm for the constrained shortest path problem. Networks **10**(4), 293–310 (1980)
46. Held, M., Karp, R.M.: The traveling-salesman problem and minimum spanning trees. Oper. Res. **18**(6), 1138–1162 (1970)
47. Held, M., Karp, R.M.: The traveling-salesman problem and minimum spanning trees: part II. Math. Program. **1**(1), 6–25 (1971)
48. Held, M., Wolfe, P., Crowder, H.P.: Validation of subgradient optimization. Math. Program. **6**(1), 62–88 (1974)
49. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods. Springer, Berlin (1993)
50. Huisman, D., Jans, R., Peeters, M., Wagelmans, A.P.M.: Combining column generation and Lagrangian relaxation. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.) Column Generation, Chap. 9, pp. 247–270. Springer, New York (2005)

51. Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., Vanderbeck, F.: Column generation based primal heuristics. Electron Notes Discrete Math. **36**, 695–702 (2010)
52. Jonsson, Ö., Larsson, T., Värbrand, P.: A Lagrangean relaxation scheme for a scheduling problem. Asia-Pacific J. Oper. Res. **7**, 155–162 (1990)
53. Jörnsten, K., Näsberg, M.: A new Lagrangian relaxation approach to the generalized assignment problem. Eur. J. Oper. Res. **27**(3), 313–323 (1986)
54. Knopp, K.: Infinite Sequences and Series. Dover Publications, New York (1956)
55. Kohl, N., Madsen, O.B.G.: An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. Oper. Res. **45**(3), 395–406 (1997)
56. Larsson, T., Liu, Z.: A Lagrangean relaxation scheme for structured linear programs with application to multicommodity network flows. Optimization **40**(3), 247–284 (1997)
57. Larsson, T., Patriksson, M.: Global optimality conditions for discrete and nonconvex optimization—with applications to Lagrangian heuristics and column generation. Oper. Res. **54**(3), 436–453 (2006)
58. Larsson, T., Patriksson, M., Strömberg, A.-B.: Conditional subgradient optimization—Theory and applications. Eur. J. Oper. Res. **88**(2), 382–403 (1996)
59. Larsson, T., Patriksson, M., Strömberg, A.-B.: Ergodic results and bounds on the optimal value in subgradient optimization. In: Kleinschmidt, P., Bachem, A., Derigs, U., Fischer, D., Leopold-Wildburger, U., Möhring, R. (eds.) Operations Research Proceedings 1995, pp. 30–35. Springer, Berlin (1996)
60. Larsson, T., Patriksson, M., Strömberg, A.-B.: Ergodic, primal convergence in dual subgradient schemes for convex programming. Math. Program. **86**(2), 283–312 (1999)
61. Larsson, T., Patriksson, M., Strömberg, A.-B.: On the convergence of conditional ε-subgradient methods for convex programs and convex–concave saddle-point problems. Eur. J. Oper. Res. **151**(3), 461–473 (2003)
62. Larsson, T., Patriksson, M., Rydergren, C.: A column generation procedure for the side constrained traffic equilibrium problem. Transp. Res. B **38**(1), 17–38 (2004)
63. Larsson, T., Patriksson, M., Strömberg, A.-B.: Ergodic convergence in subgradient optimization—with application to simplicial decomposition of convex programs. In: S. Reich, A.J. Zaslavski (eds.) Optimization Theory and Related Topics. Contemporary Mathematics, vol. 568, pp. 159–186 American Mathematical Society and Bar-Ilan University (2012)
64. Lasdon, L.S.: Duality and decomposition in mathematical programming. IEEE Trans. Sys. Sci. Cybern. **4**(2), 86–100 (1968)
65. Lasdon, L.S.: Optimization Theory for Large Systems. Dover Publications, Mineola (2002)
66. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. Oper. Res. **53**(6), 1007–1023 (2005)
67. Ma, J.: Recovery of primal solution in dual subgradient schemes. Master's Thesis, Computation for Design and Optimization, School of Engineering, Massachusetts Institute of Technology, Cambridge (2007)
68. Magnanti, T.L., Shapiro, J.F., Wagner, M.H.: Generalized linear programming solves the dual. Manag. Sci. **22**(11), 1195–1203 (1976)
69. Manne, A.S.: Programming of economic lot sizes. Manag. Sci. **4**(2), 115–135 (1958)
70. Marsten, R.E., Hogan, W.W., Blankenship, J.W.: The boxstep method for large-scale optimization. Oper. Res. **23**(3), 389–405 (1975)
71. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester (1990)
72. Mingozzi, A., Roberti, R.: An exact algorithm for the fixed charge transportation problem based on matching source and sink patterns. Transp. Sci. **52**(2), 229–238 (2018)
73. Mulvey, J.M., Crowder, H.P.: Cluster analysis: an application of Lagrangian relaxation. Manag. Sci. **25**(4), 329–340 (1979)
74. Önnheim, M., Gustavsson, E., Strömberg, A.-B., Patriksson, M., Larsson, T.: Ergodic, primal convergence in dual subgradient schemes for convex programming, II: the case of inconsistent primal problems. Math. Program. **163**(1–2), 57–84 (2017)

75. Pimentel, C.M.O., Alvelos, F.P.e., Valério De Carvalho, J.M.: Comparing Dantzig–Wolfe decompositions and branch-and-price algorithms for the multi-item capacitated lotsizing problem. Optim. Methods Softw. **25**(2), 299–319 (2010)
76. Polyak, B.T.: A general method of solving extremum problems. Soviet Math. Doklady **8**(3), 593–597 (1967)
77. Polyak, B.T.: Minimization of unsmooth functionals. USSR Comput. Math. Math. Phys. **9**, 14–29 (1969)
78. Raghavan, P., Thompson, C.D.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica **7**(4), 365–374 (1987)
79. Raidl, G.R.: Decomposition based hybrid metaheuristics. Eur. J. Oper. Res. **244**(1), 66–76 (2015)
80. Savelsbergh, M.: A branch-and-price algorithm for the generalized assignment problem. Oper. Res. **45**(6), 831–841 (1997)
81. Sherali, H.D., Choi, G.: Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. Oper. Res. Lett. **19**(3), 105–113 (1996)
82. Shor, N.Z.: Minimization Methods for Non-Differentiable Functions. Springer, Berlin. Translated from the Russian by K.C. Kiwiel and A. Ruszczyński (1985)
83. Shor, N.Z.: The development of numerical methods for nonsmooth optimization in the USSR. In: Lenstra, J.K., Rinnoy Kan A.H.G., Schrijver,A. (eds.) History of Mathematical Programming: A Collection of Personal Reminiscences, pp. 135–139. North-Holland, Amsterdam (1991)
84. Simonetto, A., Jamali-Rad, H.: Primal recovery from consensus-based dual decomposition for distributed convex optimization. J. Optim. Theory Appl. **168**(1), 172–197 (2016)
85. Sridharan, R.: A Lagrangian heuristic for the capacitated plant location problem with single source constraints. Eur. J. Oper. Res. **66**(3), 305–312 (1993)
86. Sweeney, D.J., Murphy, R.A.: A method of decomposition for integer programs. Oper. Res. **27**(6), 1128–1141 (1979)
87. van den Akker, J.M., Hurkens, C.A.J., Savelsbergh, M.W.P.: Time-indexed formulations for machine scheduling problems: column generation. INFORMS J. Comput. **12**(2), 111–124 (2000)
88. Vanderbeck, F.: Branching in branch-and-price: a generic scheme. Math. Program. **130**(2), 249–294 (2011)
89. Vera, J.R., Weintraub, A., Koenig, M., Bravo, G., Guignard, M., Barahona, F.: A Lagrangian relaxation approach for a machinery location problem in forest harvesting. Pesquisa Oper. **23**(1), 111–128 (2003)
90. Westerlund, A., Göthe-Lundgren, M., Larsson, T.: A stabilized column generation scheme for the traveling salesman subtour problem. Discret. Appl. Math. **154**(15), 2212–2238 (2006)
91. Westerlund, A., Göthe-Lundgren, M., Larsson, T., Yuan, D.: Subgradient optimization prior to column generation—A means for predicting optimal columns. Inter. J. Pure Appl. Math. **103**(4), 797–818 (2015)
92. Wilhelm, W.E.: A technical review of column generation in integer programming. Optim. Eng. **2**(2), 159–200 (2001)
93. Wolfe, P.: A method of conjugate subgradients for minimizing nondifferentiable functions. In: M.L. Balinski, P. Wolfe (eds.) Nondifferentiable Optimization. Mathematical Programming Study, vol. 3, pp. 145–173 Springer, Berlin (1975)
94. Zhao, Y., Larsson, T., Rönnberg, E., Pardalos, P.M.: The fixed charge transportation problem: a strong formulation based on Lagrangian decomposition and column generation. J. Glob. Optim. **72**(3), 517–538 (2018)
95. Zhu, S.: Hybrid methods for solving Lagrangean duals and applications in mixed integer programming. Ph.D. Thesis, Operations and Information Management, University of Pennsylvania, Philadelphia (1995)

Chapter 16
On Mixed Integer Nonsmooth Optimization

Ville-Pekka Eronen, Tapio Westerlund, and Marko M. Mäkelä

Abstract In this chapter we review some deterministic solution methods for convex mixed integer nonsmooth optimization problems. The methods are branch and bound, outer approximation, extended cutting plane, extended supporting hyperplane and extended level bundle method. Nonsmoothness is taken into account by using Clarke subgradients as a substitute for the classical gradient. Ideas for convergence proofs are given as well as references where the details can be found. We also consider how some algorithms can be modified in order to solve nonconvex problems including f°-pseudoconvex functions or even f°-quasiconvex constraints.

16.1 Introduction

In mixed integer optimization some variables are continuous and some are integers. The difficulty in dealing with integer variables is that the feasible set is not necessarily connected nor convex. This causes finding descent direction and doing line searches less fruitful than in continuous optimization. The deterministic methods to solve *mixed integer nonlinear programming* (MINLP) problems with differentiable functions include branch and bound [10], outer approximation [16] and cutting plane [48] methods. An important special case of an MINLP problem is when integers are binary variables having only possible values $\{0, 1\}$. In this case the variables can model yes or no type decisions, making these kind of problems important in various applications. MINLP models have been used in many fields including chemical and mechanical engineering, physics, medicine and, for example, in design of water/gas network [2, 3, 28, 29, 39].

V.-P. Eronen (✉) · T. Westerlund · M. M. Mäkelä
Department of Mathematics and Statistics, University of Turku, Turku, Finland
e-mail: vpoero@utu.fi; twesterl@abo.fi; makela@utu.fi

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_16

In *nonsmooth optimization* (NSO) some functions are not necessarily continuously differentiable. A typical reason for nonsmoothness is functions like max, $|\cdot|$ or $\|\cdot\|$. We will only consider the case where functions are *locally Lipschitz continuous* (LLC). Lipschitz continuity implies that the set of nondifferentiable points has zero measure [1]. Furthermore, while we cannot calculate the classical gradient, a Clarke subgradient belonging to the Clarke subdifferential exists at any point [1, 7]. The problem in solving NSO is finding a descent direction, which is not as readily available as in the continuously differentiable case. In addition, identifying a minimum in which objective function is not differentiable may be problematic. Nonsmooth problems can be found in many applications for example in optimal control problems, data analysis and economics [1]. In general NSO problems can be solved with subgradient (see e.g. [43] and Chap. 2 in this book) and bundle methods (see e.g. [13, 34] and Chaps. 3–5). In addition, derivative free methods (see e.g. [41] and Chaps. 18 and 19) or gradient sampling methods (see e.g. Chap. 6) can be used to solve nonsmooth problems as well but we do not study these methods here.

In this chapter we review some deterministic methods that solve convex *mixed integer nonsmooth optimization problems* (MINSO) with help of the Clarke subdifferential. In these problems the objective and constraint functions are convex and the integer relaxed feasible set is convex. Furthermore, we assume that we can evaluate a subgradient of any function at any given point. Some of the methods can be generalized to solve problems with f°-pseudoconvex functions or even f°-quasiconvex constraint functions. Methods to solve MINSO problems have been studied recently in [11, 12, 18, 20, 46, 51], among others.

MINSO problems can be found for example in electrical engineering [6], gas network planning [42], statistics [52] and facility layout problems [5]. In these articles the problems were solved with metaheuristics and reformulation techniques. Other techniques, not covered here, are derivative free algorithms to solve MINSO problems containing black box functions. These methods can be found, for example, in [33, 36, 37] and references therein.

One reason for the lack of research on methods for mixed integer NSO is that typical nonsmooth functions, like max and $|\cdot|$, can be modelled with auxiliary variables leading to a smooth MINLP problem. However, transformation sometimes leads to a more difficult problem as was noticed in [51]. In that case transforming a mixed integer NSO problem by means of a pseudoconvex objective function resulted in a nonconvex MINLP problem. Another way to deal with function nonsmoothness is to use smoothing techniques. For example $|x| \approx \sqrt{x^2 + \tau}$ for a small $\tau > 0$ [42].

In Sect. 16.2 we present several deterministic algorithms to solve convex MINSO problems. Ideas of the convergence proofs are given as well as references to articles where more details can be found. In Sect. 16.3 the algorithms are illustrated by solving a simple example problem.

16.2 Deterministic Algorithms for Convex MINSO

The considered MINSO problem can be formulated as follows

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} & g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0,\ j \in \mathcal{J}, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in L, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{Z}^m, \end{cases} \tag{MINSO}$$

where $\mathcal{J} = \{1, 2, \ldots, J\}$. The set L is defined by linear constraints and it is assumed to be compact. Hence, we assume that there are finitely many feasible integer vectors $\boldsymbol{y}$. Define the finite set

$$Y = \left\{ \boldsymbol{y} \in \mathbb{Z}^m \mid (\boldsymbol{x}, \boldsymbol{y}) \in L, \boldsymbol{x} \in \mathbb{R}^n \right\}.$$

Sometimes it is convenient not to separate continuous and integer variables and due to this we define $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ and set

$$Z = \left\{ \boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{Z}^m \right\}.$$

Denote also

$$G(\boldsymbol{z}) = \max_{j \in \mathcal{J}} \left\{ g_j(\boldsymbol{z}) \right\} \text{ and } N = \{ \boldsymbol{z} \mid G(\boldsymbol{z}) \leq 0 \}.$$

With these notations we can formulate the problem (MINSO) as follows

$$\begin{cases} \text{minimize} & f(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{z} \in N \cap L \cap Z. \end{cases}$$

If not otherwise stated, the functions f and g_j, $j \in \mathcal{J}$ are assumed to be convex and thus LLC. Hence, the problem is convex: nonlinear functions are convex and the feasible set is convex when integer variables are relaxed to be continuous variables.

Some of the algorithms can be generalized to solve problems with f°-pseudoconvex functions and even f°-quasiconvex constraint functions.

Definition 16.1 A LLC function $f : \mathbb{R}^n \to \mathbb{R}$ is $f^\circ - pseudoconvex$ ($f^\circ - quasiconvex$) if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$

$$f(\boldsymbol{y}) < (\leq)\, f(\boldsymbol{x}) \qquad \text{implies} \qquad f^\circ(\boldsymbol{x}; \boldsymbol{y} - \boldsymbol{x}) < (\leq)\, 0.$$

f°-pseudoconvexity is a straightforward generalization of the classical pseudoconvexity to LLC functions. f°-quasiconvexity is slightly more restrictive than quasiconvexity for LLC functions [1]. A convex function is always f°-pseudoconvex, which in turn, is always f°-quasiconvex. Moreover, an f°-quasiconvex function is

always quasiconvex. An important feature of an f°-pseudoconvex function is that a local minimum is also a global one. Another important property, that holds for f°-quasiconvex functions as well, is that the level sets $\{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq a\}$ are convex for any $a \in \mathbb{R}$. From the definition of generalized directional derivative and Definition 16.1 we can see that inequality

$$\boldsymbol{\xi}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \leq 0,$$

holds true for any $\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_2)$ and $\boldsymbol{x}_1 \leq \boldsymbol{x}_2$ such that $f(\boldsymbol{x}_1) < f(\boldsymbol{x}_2)$ if f is f°-quasiconvex. This inequality will prove to be useful when studying problems with f°-quasiconvex constraint functions.

Articles [2, 24] describe several deterministic methods to solve the MINLP in case all the functions are continuously differentiable. The following deterministic methods for the nonsmooth case (MINSO) are mainly based on those. Most of the methods require that we are able to calculate the values of the nonlinear functions and an arbitrary subgradient at the points where the algorithm visits. These points are assumed to belong to the set L.

16.2.1 NSO Branch and Bound

The *branch and bound* (B&B) method for *mixed integer linear programming* (MILP) problems was developed in 1960 [31] and it is a general framework to deal with integer variables. The method was generalized for MINLP problems in [10]. In the B&B method only integer relaxed problems are solved, that is, problems where integer variables are treated as continuous ones. To cut off the previous non-integer solutions, bounds to the integer variables are added to the subsequent problems. The problems to be solved can also be seen as nodes of a tree.

The B&B method solves the problem (MINSO) as a sequence of continuous NSO subproblems. The first subproblem is

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} & g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0,\ j \in \mathcal{J}, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in L, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{R}^m. \end{cases} \tag{BB-NSO}$$

If at iteration k the solution of the subproblem is not integer feasible, *branching* will be applied. In branching we pick an integer variable y_i that was not integer at the solution point $(\boldsymbol{x}^k, \boldsymbol{y}^k)$. Then we create two subproblems with all constraints from the previously solved problem and add the constraint $y_i \leq \lfloor y_i^k \rfloor$ to one subproblem and $y_i \geq \lceil y_i^k \rceil$ to the other. Clearly, any of these constraints will cut off the previous solution point. The branching does not occur in three cases:

- the problem has no feasible solution;

- the solution is integer feasible. In this case the solution is feasible in the original MINSO problem and, thus, the objective function value at the solution is an upper bound;
- the solution is not integer feasible but the value of the objective function is greater than the current best upper bound.

In these cases the node will be pruned and it becomes a leaf of the tree. The B&B continues to solve previously created subproblems until none is left. As noted in [2] the subproblems can be uniquely identified by the bounds $(\boldsymbol{l}, \boldsymbol{u})$ given to the integer vectors. For example, problem NSO$(\boldsymbol{l}, \boldsymbol{u})$ corresponds to subproblem with bounds $\boldsymbol{l} \leq \boldsymbol{y} \leq \boldsymbol{u}$ and NSO$(-\infty, \infty)$ corresponds to the first subproblem or the root node. The NSO branch and bound algorithm is presented in Algorithm 16.1.

Algorithm 16.1: NSO branch and bound algorithm

Step 1.	Set upper bound $U = \infty$, $k = 0$ and initiate the list of unsolved subproblems $\mathcal{L} = \{\text{NSO}(-\infty, \infty)\}$.
Step 2.	If $\mathcal{L} = \emptyset$ the current upper bound is the global minimum. Otherwise, choose a subproblem NSO$(\boldsymbol{l}, \boldsymbol{u})$ from the list $\mathcal{L}$ and update $\mathcal{L} = \mathcal{L} \setminus \{\text{NSO}(\boldsymbol{l}, \boldsymbol{u})\}$.
Step 3.	Solve NSO$(\boldsymbol{l}, \boldsymbol{u})$. If NSO$(\boldsymbol{l}, \boldsymbol{u})$ is infeasible go to Step 2. Otherwise, denote $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ the solution of NSO$(\boldsymbol{l}, \boldsymbol{u})$ and set $k = k + 1$.
Step 4.	Suppose $\boldsymbol{y}^k$ is an integer vector. If $f(\boldsymbol{x}^k, \boldsymbol{y}^k) < U$ set $U = f(\boldsymbol{x}^k, \boldsymbol{y}^k)$. Go to Step 2.
Step 5.	Suppose that $\boldsymbol{y}^k$ is not an integer vector. If $f(\boldsymbol{x}^k, \boldsymbol{y}^k) > U$, go to Step 2. Otherwise, take an integer variable y_i that was not integer at $\boldsymbol{y}^k$. Set $\boldsymbol{l}^- = \boldsymbol{l}$, $l_i^- = \lceil y_i^k \rceil$, $\boldsymbol{u}^+ = \boldsymbol{u}$ and $u_i^+ = \lfloor y_i^k \rfloor$. Add two new subproblems to the list $\mathcal{L} = \mathcal{L} \cup \left\{\text{NSO}(\boldsymbol{l}^-, \boldsymbol{u}), \text{NSO}(\boldsymbol{l}, \boldsymbol{u}^+)\right\}$. Go to Step 2.

Theorem 16.1 *If the NSO subproblems are solved to a global minimizer, Algorithm 16.1 finds a global minimizer of* (MINSO) *after solving a finite number of NSO subproblems.*

Proof Since there are finitely many feasible integer vectors the tree will be finite as well. If every NSO subproblem is solved to a global minimum, the minimum of the original problem will be found. □

In the worst case the B&B method solves more NSO problems than there are integer vectors in Y. For example if Y consists of binary vectors $\{0, 1\}^m$ there are 2^m possible binary vectors but in the worst case the B&B method solves $2^{m+1} - 1$ NSO problems. Thus, the efficiency of the B&B is based on ruling out regions from the feasible set by a good upper bound or by finding integer feasible solutions from the relaxed problems.

There is a couple of degrees of freedom in Algorithm 16.1. The choice of the branching variable and the choice of the problem to be solved at Step 3 is discussed

for example in [2]. Good choices will ideally result in a smaller search tree. The choice of branching variable aims to find a variable causing the minimum of the resulting NSO problem to be greatest making it probable to prune it. The choice of NSO problem aims to find a good upper bound fast and then increase the lower bound in order to prove the optimality of the current upper bound.

The NSO algorithm itself can be freely chosen as long as it can solve the NSO problems to the global minimum. Nonsmooth algorithms like in [13, 34] could be used to solve the NSO problems we consider. However, to authors' knowledge there exists no implementation of these with the B&B. There exists, though, an implementation of the B&B with a derivative free NLP algorithm [21] that can deal with nonsmooth functions.

Being a general way to handle integer variables the B&B method can be used to solve also MILP problems. This can be done by using a linear programming solver instead of the NSO solver in Algorithm 16.1. In fact, many popular LP solvers, like CPLEX [8] and Gurobi [25], utilize sophisticated versions of the B&B and can solve MILP problems very efficiently. MILP is an important class of problems since the other methods we consider to solve the problem (MINSO) generate MILP subproblems as a part of their solution strategy.

16.2.2 Outer Approximation

The outer approximation method was developed in 1980s–1990s [16, 22]. The linear outer approximation method is not as readily applicable for nonsmooth problems as the NLP branch and bound. Hence, we will present the method for the smooth case first.

In the linear outer approximation method an MILP master problem is created by removing nonlinearities from the original MINLP problem. The nonlinear functions are taken into account by adding linear approximations of them to the MILP master problem. The points at which these approximations are made are found by solving NLP problems. Thus, the method alternates between solving NLP and MILP problems. At iteration k the NLP problem corresponds to the original MINLP problem where integer vector has a fixed value $\boldsymbol{y}^k$ obtained as an initial point or from the solution point of the MILP master problem. The NLP problem can be written as

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}, \boldsymbol{y}^k) \\ \text{subject to} & g_j(\boldsymbol{x}, \boldsymbol{y}^k) \leq 0,\ j \in \mathcal{J}, \\ & (\boldsymbol{x}, \boldsymbol{y}^k) \in L, \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \qquad \text{(OA-NLP}(\boldsymbol{y}^k)\text{)}$$

The solution of this problem will provide an upper bound for the original MINLP problem. Denote U^k the smallest upper bound found after solving the kth NLP or feasibility problem. If the problem (OA-NLP($\boldsymbol{y}^k$)) turns out to be infeasible we need

to solve a feasibility problem instead. One possible feasibility problem is

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & g_j(\boldsymbol{x}, \boldsymbol{y}^k) \le \mu,\ j \in \mathcal{J}, \\ & (\boldsymbol{x}, \boldsymbol{y}^k) \in L, \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{OA-F($\boldsymbol{y}^k$)}$$

In the both cases, linearizations of the nonlinear functions at the solution point $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ will be added to the MILP master problem

$$\begin{cases} \text{minimize} & \eta \\ \text{subject to} & \eta < U^k - \varepsilon, \\ & \eta \ge f(\boldsymbol{x}^i, \boldsymbol{y}^i) + \nabla f(\boldsymbol{x}^i, \boldsymbol{y}^i)^T \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^i \\ \boldsymbol{y} - \boldsymbol{y}^i \end{pmatrix}, i \in T^k, \\ & 0 \ge g_j(\boldsymbol{x}^i, \boldsymbol{y}^i) + \nabla g_j(\boldsymbol{x}^i, \boldsymbol{y}^i)^T \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^i \\ \boldsymbol{y} - \boldsymbol{y}^i \end{pmatrix}, i \in T^k \cup S^k, j \in \mathcal{J}, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in L, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{Z}^m, \eta \in \mathbb{R}. \end{cases} \tag{OA-MILP$_k$}$$

Here $\varepsilon > 0$ is a small user given parameter and the set T^k collects indices when (OA-NLP($\boldsymbol{y}^k$)) is feasible whereas S^k collects indices when it is not.

Algorithm 16.2: OA algorithm

Data:	Let feasible integer solution $\boldsymbol{y}^1 \in \mathbb{Z}^m$ be given.
Step 1.	Set $k = 1$, $T^0 = \emptyset$, $S^0 = \emptyset$ and $U^0 = \infty$.
Step 2.	Solve (OA-NLP($\boldsymbol{y}^k$)), or the feasibility problem (OA-F($\boldsymbol{y}^k$)), if (OA-NLP($\boldsymbol{y}^k$)) is infeasible, and let the solution be $\boldsymbol{x}^k$.
Step 3.	Linearize the objective and constraint functions on $(\boldsymbol{x}^k, \boldsymbol{y}^k)$. Set $T^k = T^{k-1} \cup \{k\}$ or $S^k = S^{k-1} \cup \{k\}$ as appropriate.
Step 4.	If (OA-NLP($\boldsymbol{y}^k$)) is feasible and $f(\boldsymbol{x}^k, \boldsymbol{y}^k) < U^{k-1}$ then update $U^k = f(\boldsymbol{x}^k, \boldsymbol{y}^k)$. Otherwise, set $U^k = U^{k-1}$
Step 5.	Solve (OA-MILP$_k$), giving a new integer vector $\boldsymbol{y}^{k+1}$. If (OA-MILP$_k$) is infeasible, **stop**: the current upper bound is the global minimum. Otherwise, set $k = k + 1$ and go to Step 2.

To guarantee the optimality of the solution we need to assume that KKT-conditions hold at the minimizer of (OA-NLP($\boldsymbol{y}^k$)) for each feasible $\boldsymbol{y}^k$. This assumption holds if an appropriate constraint qualification is satisfied at $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ one example being that the gradients of the active constraint functions are linearly independent [22]. Due to variable μ the gradients of constraint functions of (OA-F($\boldsymbol{y}^k$)) are never $\mathbf{0}$. Thus, the Cottle constraint qualification [1] holds at

any point, implying that the classical KKT-conditions hold at the minimum without any additional assumptions.

When the assumption holds, the linearizations made at $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ will make the integer value $\boldsymbol{y}^k$ infeasible in the subsequent MILP master problems. The convexity of f and g_j for all $j \in \mathcal{J}$ implies that no feasible point in which f attains smaller value than $U^k - \varepsilon$ will be cut off. The solution is found after a finite number of iterations since Y is finite. Then U^k is the global minimum as every (OA-NLP($\boldsymbol{y}^k$)) is solved to the global minimum.

Theorem 16.2 *Algorithm 16.2 solves the problem* (MINSO) *to a global minimizer when functions are convex and continuously differentiable and KKT-conditions hold at minimizer of* (OA-NLP($\boldsymbol{y}^k$)) *for every feasible* $\boldsymbol{y}^k$.

Proof The proof can be found in [22]. □

Outer approximation has been generalized to handle continuously differentiable quasiconvex constraints [4, 26]. With these constraints the integer relaxed feasible set will be convex. Three difficulties arises with this generalization. The first one is that the linearizations in (OA-MILP$_k$) may cut off feasible points when the corresponding functions are not convex. This is solved by doing linearizations from quasiconvex functions at point $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ only if they are active, that is, $g_j(\boldsymbol{x}^k, \boldsymbol{y}^k) = 0$. Then the linearization reduces to

$$\nabla g_j(\boldsymbol{x}^k, \boldsymbol{y}^k)^T \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^k \\ \boldsymbol{y} - \boldsymbol{y}^k \end{pmatrix} \leq 0. \tag{16.1}$$

Due to the properties of quasiconvexity linearization (16.1) will not cut off any points where g_j attains value less than or equal to $g_j(\boldsymbol{x}^k, \boldsymbol{y}^k) = 0$. The active constraints are the important ones when proving that the linearizations in (OA-MILP$_k$) will cut off the old solution $(\boldsymbol{x}^k, \boldsymbol{y}^k)$. Hence, this property holds when linearizations (16.1) are used in (OA-MILP$_k$) for quasiconvex functions.

Another difficulty is the feasibility problem (OA-F($\boldsymbol{y}^k$)) that will now have a nonquasiconvex constraint functions. Instead of the feasibility problem we may solve problem

$$\begin{cases} \text{minimize} & \theta(\boldsymbol{y}; \boldsymbol{y}^k) := \left\| \boldsymbol{y} - \boldsymbol{y}^k \right\|^2 \\ \text{subject to} & g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \ \ j \in \mathcal{J}, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in L, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{R}^m. \end{cases} \tag{FP-NLP($\boldsymbol{y}^k$))}$$

Note that $\boldsymbol{y}$ is considered continuous variable in this problem. (FP-NLP($\boldsymbol{y}^k$)) finds $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ that minimizes distance of $\boldsymbol{y}^k$ and the projection of the integer relaxed feasible set $N \cap L$ onto $\mathbb{R}^m$. Then hyperplane

$$\left\{ \boldsymbol{y} \mid \nabla\theta(\overline{\boldsymbol{y}}; \boldsymbol{y}^k)^T (\boldsymbol{y} - \overline{\boldsymbol{y}}) = 0 \right\}$$

separates convex sets $N \cap L$ and $\left\{ \boldsymbol{y} \mid \theta(\boldsymbol{y}; \boldsymbol{y}^k) \leq \theta(\overline{\boldsymbol{y}}; \boldsymbol{y}^k) \right\}$. More importantly, it separates $N \cap L$ and $\boldsymbol{y}^k$. Hence, the constraint

$$\nabla\theta(\overline{\boldsymbol{y}}; \boldsymbol{y}^k)^T (\boldsymbol{y} - \overline{\boldsymbol{y}}) = 2(\overline{\boldsymbol{y}} - \boldsymbol{y}^k)^T (\boldsymbol{y} - \overline{\boldsymbol{y}}) \geq 0 \tag{16.2}$$

will not cut off any points from $N \cap L$ but will make $\boldsymbol{y}^k$ infeasible. The linearization (16.2) will be added to the MILP master problem (OA-MILP$_k$).

In [26] an alternative to the constraint (16.2) was presented. Let $\hat{\mathcal{J}}$ be the set of indices of active constraints at point $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$. Then linearizations

$$\nabla g_j(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})^T \begin{pmatrix} \boldsymbol{x} - \overline{\boldsymbol{x}} \\ \boldsymbol{y} - \overline{\boldsymbol{y}} \end{pmatrix} \leq 0, \, j \in \hat{\mathcal{J}},$$

will cut off $\boldsymbol{y}^k$ but points in $N \cap L$ remain feasible. However, a constraint qualification must be satisfied at $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ in order to make the KKT-conditions hold. With this procedure the objective of the problem (FP-NLP($\boldsymbol{y}^k$)) may also be set to $\theta(\boldsymbol{y}; \boldsymbol{y}^k) = \left\| \boldsymbol{y} - \boldsymbol{y}^k \right\|_1$.

The last problem in this generalization is that NLP problems have quasiconvex constraints. These problems were assumed to be solved to global optimality in [4, 26]. With some additional assumptions this can be done, for example, with the supporting hyperplane algorithm presented later.

In [22] the outer approximation method was generalized to solve the following nonsmooth MINLP problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}, \boldsymbol{y}) + h(g(\boldsymbol{x}, \boldsymbol{y})) \\ \text{subject to} & (\boldsymbol{x}, \boldsymbol{y}) \in L, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{Z}^m. \end{cases} \tag{OA-MINSO}$$

Here the functions $f : \mathbb{R}^{n+m} \to \mathbb{R}$ and $g : \mathbb{R}^{n+m} \to \mathbb{R}^J$ are assumed to be convex and continuously differentiable, while $h : \mathbb{R}^J \to \mathbb{R}$ is assumed to be convex but nonsmooth. The function h is also assumed to be monotone in the sense that if $a_i \leq b_i$ for all $i = 1, 2, \ldots, n$ then $h(\boldsymbol{a}) \leq h(\boldsymbol{b})$. This kind of restrictions are met by functions like $\max_i \{a_i\}$ and $\sum_{i=1}^n |\max\{0, a_i\}|$. The problem (OA-MINSO) is a special case of the problem (MINSO). It can be used to solve the continuously differentiable case with the exact penalty function method [22].

When solving problem (OA-MINSO) for a fixed value $\boldsymbol{y}^k$ there will always be a feasible solution and no feasibility problems are needed to solve. At the solution $\boldsymbol{z}^k = (\boldsymbol{x}^k, \boldsymbol{y}^k)$ the linearization

$$\eta \geq f(\boldsymbol{z}^k) + \nabla f(\boldsymbol{z}^k)^T \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^k \\ \boldsymbol{y} - \boldsymbol{y}^k \end{pmatrix} + h\left(g(\boldsymbol{z}^k) + \nabla g(\boldsymbol{z}^k)^T \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^k \\ \boldsymbol{y} - \boldsymbol{y}^k \end{pmatrix} \right)$$

is added to the MILP problem. The convergence of the algorithm was proven in [22]. Although the proof utilized subgradients of h, no subgradients are needed to calculate in the algorithm.

OA method for MINSO problem was also studied in [18]. It was proved that even if we can solve NSO problems instead of NLP problems we cannot guarantee the convergence of the OA method by simply replacing gradients by arbitrary subgradients in Algorithm 16.2. The trouble is that the linearizations done after (OA-NLP($\boldsymbol{y}^k$)) or (OA-F($\boldsymbol{y}^k$)) problem may not cut off the previous integer vector $\boldsymbol{y}^k$, thus, resulting in potential infinite loop. The reason for this is that an arbitrary subgradient does not necessarily satisfy KKT-conditions as an equality. However, if we can find a subgradient that does this, the algorithm will work as intended. This was also studied in detail in [46]. Also, in [47] it was proved that it is sufficient that the functions are continuously differentiable with respect to $\boldsymbol{x}$. In this case we can use arbitrary subgradients in Algorithm 16.2. If the integer variables are binary, the infinite loop can also be avoided by using integer cuts presented in [16, 24].

Another special case of MINSO problem that can be solved with OA method can be found in [15]. In that paper a mixed integer second order conic programming (MISOCP) problem is considered. Due to conic constraints the problem is not smooth. With help of duality the authors were able to find subgradients satisfying KKT-conditions of NLP problem, and thus, formulate an OA based algorithm with proven convergence.

Recently, the OA method was successfully generalized to handle convex MINSO problems in [11]. NSO and the feasibility problem were solved with an exact penalization proximal bundle algorithm. This provides subgradients satisfying KKT-conditions along with the KKT-coefficients. Linearizations are done only from the constraints that are active at the solution $(\boldsymbol{x}^k, \boldsymbol{y}^k)$. The components of the subgradient that corresponds to integer variables can not be chosen arbitrarily but in a way that the resulting vector is a subgradient.

A straightforward generalization of the algorithm in [11] for problems with f°-quasiconvex constraint functions is not possible. The difficulty arises in NSO problems that have f°-quasiconvex constraint functions while the exact penalization proximal bundle algorithm requires convex functions. Otherwise, since the linearizations are created only from the active constraints the algorithm should be applicable for problems with f°-quasiconvex constraints if the feasibility problems are dealt in similar manner to [26]. Of course, similar restrictions are valid as in the problem classes studied in [26], that is, the KKT-conditions must be satisfied at points where the linearizations are created.

In [11] numerical tests included an academic hybrid robust/chance-constraint problem. While the problem was convex MINSO problem it could be formulated as a quadratic smooth nonconvex MINLP problem. However, it turned out to be easier to solve the nonsmooth formulation especially when the size of the problem was increased. The OA method was also tested against a variant of extended cutting-plane method and extended level bundle method, which are presented here later. In large problem instances the OA method outperformed these two methods which are

known to excel in cases where nonlinear function evaluations are time consuming. In the considered problem the nonlinear function could be evaluated fast. However, the extended cutting-plane method was the fastest algorithm in almost 35% of the problems.

LP/NLP-Based Branch and Bound LP/NLP-based branch and bound [40] can be seen as an improvement of the outer approximation method where only one MILP master problem is solved. The MILP master problem of the OA algorithm is solved, but, each time an integer solution $\boldsymbol{y}^k$ is obtained in the tree of LP problems, an (OA-NLP($\boldsymbol{y}^k$)) problem is solved. Note that this is different from OA method, where MILP master problems are solved to the minimum before solving any NLP problems. After solving (OA-NLP($\boldsymbol{y}^k$)) or the feasibility problem (OA-F($\boldsymbol{y}^k$)), cuts similar to OA method are generated to all remaining LP problems. Furthermore, the problem that yielded $\boldsymbol{y}^k$ is solved again with the additional constraints.

The algorithm uses similar cuts as the OA method and hence the same difficulties as with the OA method arises when solving MINSO problems. In addition, a similar solution to the difficulties applies and the method presented in [11] generalizes LP/NLP-based branch and bound for MINSO problems.

16.2.3 Extended Cutting-Plane Method

The *extended cutting-plane* (ECP) method was introduced in [48]. It generalizes Kelley's cutting-plane method [27] for smooth mixed integer case. The method was further generalized for nonsmooth functions by replacing gradients with arbitrary subgradients in [18]. For the ECP method it is necessary to transform the nonlinear objective function to a constraint as in (OA-MILP$_k$). This is done by introducing an auxiliary variable μ that will be minimized and adding the constraint

$$f(\boldsymbol{z}) - \mu \leq 0 \tag{16.3}$$

to the problem (MINSO). Since f is assumed to be convex so is $f - \mu$. Suppose in this subsection that variable μ is included in vector $\boldsymbol{z}$. As some MILP solvers can also solve mixed integer quadratic programming (MIQP) problem, this transformation is not necessary if the objective function is quadratic. Although, this will result in solving MIQP problems instead of MILP problems.

The ECP method is similar to the OA method in the sense that the method solves MILP subproblems. However, at the MILP solution point $\boldsymbol{z}^k$ the most violated nonlinear constraint function is linearized instead of solving an NSO problem. For the most violated constraint j_k, the equation $G(\boldsymbol{z}^k) = g_{j_k}(\boldsymbol{z}^k)$ holds. Then the linear constraint that will be added to the MILP problem is

$$l^k(\boldsymbol{z}) := g_{j_k}(\boldsymbol{z}^k) + \boldsymbol{\xi}_{j_k}^T(\boldsymbol{z} - \boldsymbol{z}^k) \leq 0, \tag{16.4}$$

where $\boldsymbol{\xi}_{j_k} \in \partial g_{j_k}(\boldsymbol{z}^k)$ is arbitrary. Hence, at iteration k the MILP problem is

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & l^i(\boldsymbol{z}) \leq 0,\ i = 1, 2, \ldots, k-1, \\ & \boldsymbol{z} \in L \cap Z. \end{cases} \qquad \text{(ECP-MILP}_k\text{)}$$

The problem (ECP-MILP$_1$) corresponds to the original problem without the nonlinear constraints. The process of solving an MILP problem and adding new linear constraint is repeated until the solution of the MILP problem satisfies also the nonlinear constraints. The ECP method is described in Algorithm 16.3.

Algorithm 16.3: ECP algorithm

Data:	Give the tolerance parameter $\varepsilon_g > 0$.
Step 1.	Set $k = 1$.
Step 2.	Solve the problem (ECP-MILP$_k$). Denote the solution by $(\boldsymbol{z}^k)$.
Step 3.	If $\boldsymbol{z}^k$ satisfies the nonlinear constraints, that is $G(\boldsymbol{z}^k) \leq \varepsilon_g$, **stop**, the point $\boldsymbol{z}^k$ is a global minimizer of the original MINSO problem.
Step 4.	Create a new problem (ECP-MILP$_{k+1}$) by adding in (ECP-MILP$_k$) the constraint $g_{j_k}(\boldsymbol{z}^k) + \boldsymbol{\xi}_{j_k}^T(\boldsymbol{z} - \boldsymbol{z}^k) \leq 0$, where $\boldsymbol{\xi}_{j_k} \in \partial g_{j_k}(\boldsymbol{z}^k)$ is arbitrary and $g_{j_k}(\boldsymbol{z}^k) = G(\boldsymbol{z}^k)$.
Step 5.	Set $k = k + 1$ and go to Step 2.

Unlike in the OA method, the cutting-plane (16.4) at iteration k does not cut off all solutions with integer values $\boldsymbol{y}^k$. However, it cuts off the previous solution $\boldsymbol{z}^k$. Due to the convexity of the constraint functions, the cutting-planes do not cut off any points from the feasible region. Then, since the objective function is linear, if an MILP solution is feasible in the nonlinear constraint functions, it is a global minimizer of (MINSO). If this is not obtained in a finite number of iterations, Algorithm 16.3 generates a sequence $\{\boldsymbol{z}^k\}$ of different points. The compactness of L and the Cauchy-Weierstrass Theorem imply that there exists an accumulation point. It turns out that any accumulation point of $\{\boldsymbol{z}^k\}$ is feasible in nonlinear constraint functions, and thus a global minimizer of (MINSO).

Theorem 16.3 *If $\varepsilon_g = 0$, Algorithm 16.3 finds a global minimizer of* (MINSO) *in a finite number of iterations or it generates a sequence with a global minimizer as an accumulation point.*

Proof The proof can be found in [18]. □

The advantage of the ECP over the NSO B&B and the OA methods is that no NSO problems are needed to solve. This usually results in a fewer number of nonlinear function evaluations, which is significant if the function evaluations are time consuming like in the chromatographic separation problem [17]. On the other

hand, if the bottleneck of the problem is hard MILP problems, the other methods are supposedly faster.

There are two modifications that may speed up the solving process of the ECP method. Instead of one violated constraint function we may linearize more, or even all, violated constraints at the current MILP solution z^k. While more linearizations should make algorithm solve less MILP problems, the MILP problems may become harder due to more constraints. Furthermore, we do not need to solve the MILP problems to optimality every time. Only the last MILP problem needs to be solved to optimality. We can, for example, stop MILP solving at the first integer feasible solution and create cutting-planes there. The last MILP problem must be solved to optimality. To ensure this, we can gradually increase the number of integer feasible solutions a MILP solver needs to find before stopping. The strategy may prove useful if the MILP problems are hard to solve. This ploy was presented in flow chart in [49] and applied in [5] (strategy 1) for the αECP algorithm that is presented in the next subsection.

αECP Method The ECP method was generalized to deal with pseudoconvex constraint functions and objective function in [38, 49, 50]. In [19] this was further generalized to deal with nonsmooth f°-pseudoconvex functions. The difficulty in applying the Algorithm 16.3 to the problems with f°-pseudoconvex constraint functions is that cutting-plane (16.4) may then cut off feasible points. To avoid this problem an α coefficient is added to the linearization (16.4) resulting in α-cutting-plane

$$g_{j_k}(z^k) + \alpha_{j_k}^k \boldsymbol{\xi}_{j_k}^T (z - z^k) \leq 0, \tag{16.5}$$

where $\boldsymbol{\xi}_{j_k} \in \partial g_{j_k}(z^k)$ is arbitrary. The coefficient $\alpha_{j_k}^k$ is first set to 1. The α-cutting-plane will cut off the previous solution point z^k from the MILP problem. As the constraint function is not convex, it may cut off some feasible points as well. However, since the constraint functions are f°-pseudoconvex and thus LLC, there exists $\hat{\alpha}$ such that (16.5) does not cut off any feasible point if $\alpha_{j_k}^k > \hat{\alpha}$ holds true [19]. The constant is not generally known beforehand (besides the convex case $\hat{\alpha} = 1$). In practice, we gradually increase the α coefficients until certain limit. If the MILP problem (ECP-MILP$_k$) is infeasible the α coefficients are increased by setting

$$\alpha_{j_k}^{k+1} = \beta \cdot \alpha_{j_k}^k, \tag{16.6}$$

where $\beta > 1$. The α coefficients are also increased if a feasible solution is found. In that case we use another parameter $\gamma > 1$ instead of β in (16.6). The coefficient $\alpha_{j_k}^k$ will be updated until

$$\alpha_{j_k}^k \geq \frac{g_{j_k}(z^k)}{\varepsilon_z \left\| \boldsymbol{\xi}_{j_k} \right\|}, \tag{16.7}$$

where $\varepsilon_z > 0$ is a user given parameter. To interpret this inequality, consider the case that relation (16.7) holds as an equality. By inserting the obtained $\alpha_{j_k}^k$ to the α-cutting-plane (16.5) we get

$$\frac{\boldsymbol{\xi}_{j_k}^T}{\|\boldsymbol{\xi}_{j_k}\|}(z - z^k) \leq -\varepsilon_z.$$

Thus, the points with distance less than ε_z from the hyperplane

$$H = \left\{ z \in \mathbb{R}^{m+n} \mid \frac{\boldsymbol{\xi}_{j_k}^T}{\|\boldsymbol{\xi}_{j_k}\|}(z - z^k) = 0 \right\}$$

are cut off. The criterion (16.7) can be interpreted as follows. The coefficient $\alpha_{j_k}^k$ is sufficiently large if none of the feasible points with greater distance than ε_z from the hyperplane H are cut off. The smaller the parameter ε_z the more we need to update α coefficients, but the smaller the feasible region is that may be cut off.

If an f°-pseudoconvex objective function is transformed into constraint (16.3), the constraint may not be f°-pseudoconvex. In fact the function $f(z) - \mu$ is quasiconvex only if f is convex [9]. To avoid this non-f°-pseudoconvex constraint, the f°-pseudoconvex objective function is taken into account by adding the *reduction constraint* $f(z) - f_r \leq 0$ and $\mu \leq f_r$ in the original MINSO problem and minimizing μ. The constant f_r is the current upper bound, which is updated whenever we find z such that $G(z) \leq \varepsilon_g$ and $f(z) < f_r$. To make the resulting MINSO problem meaningful we need to bound μ from below with help of f. This is done with linear constraints presented next.

We will add constraints

$$f_r + \boldsymbol{\xi}^T(z - z^k) \leq \mu, \tag{16.8}$$

where $\boldsymbol{\xi} \in \partial f(z^k)$, in the MINSO problem at any point z^k with $f(z^k) = f_r$. Linearization is also added at z^k that is ε_g-feasible in the MINSO problem. Then, the constraint $f(z) \leq f_r$ guarantees that $f(z^k) \leq f_r + \varepsilon_g$. Note that since constraints (16.8) are linear, they transfer to the MILP problems being solved in order to solve the MINSO problem.

Since the upper bound f_r changes whenever a new upper bound is found we need to update constraints (16.8). One option would be to omit the linearizations with the old f_r. This makes the convergence proof tricky requiring the additional assumption that the solution sequence has only one accumulation point [19]. Another option is to simply replace the old f_r values in (16.8) with the new one.

Linearizations (16.8) steer MILP solution away from z^k, and thus prevent infinite loops. They are also the only constraints that bounds the minimized variable μ from below besides the box constraints. If a feasible solution $\tilde{z}$ with $f(\tilde{z}) < f_r$ exists, f°-pseudoconvexity implies that $\boldsymbol{\xi}^T(\tilde{z} - z^k) < 0$. Thus, when minimizing μ we get $\mu^k < f_r$ whenever f_r is not the global minimum value. Eventually, the algorithm

stops when $f_r - \mu \leq \varepsilon_f$ is satisfied for given $\varepsilon_f > 0$. If $\varepsilon_f = 0$ it can be shown that the algorithm converges to an ε_g-feasible global minimum [19].

16.2.4 Extended Supporting Hyperplane Method

Supporting hyperplane method was introduced in [45] to solve NLP and MINLP problems. The method was recently implemented and studied in more detail in [30] where the name *extended supporting hyperplane* (ESH) was introduced. In [20] it was generalized for MINSO problems with f°-pseudoconvex constraint functions. In [51] it was further generalized for problems with f°-pseudoconvex objective function.

The ESH method is quite similar to the ECP method. The ESH method proceeds similarly to the ECP method, but instead of cutting-planes we create supporting hyperplanes to the nonlinear feasible set N. To find these hyperplanes we need an inner point z_{int} that strictly satisfies all nonlinear constraints, that is, $G(z_{\text{int}}) < 0$. Furthermore, we have to do a line search between the current MILP solution point z^k_{MILP} and z_{int} to find the point z^k such that $G(z^k) = 0$. The polyhedral approximation of N is enhanced by adding to the MILP problem the linearization

$$\boldsymbol{\xi}_{j_k}^T (z - z^k) \leq 0, \tag{16.9}$$

where $\boldsymbol{\xi}_{j_k} \in \partial g_{j_k}(z^k)$ is arbitrary and $g_{j_k}(z^k) = G(z^k)$. Notice that this is actually a cutting-plane since $g_{j_k}(z^k) = 0$. It also turns out that we do not need α coefficients for f°-pseudoconvex constraint functions. The constraint functions can be f°-quasiconvex with an additional restriction that $\mathbf{0} \notin \partial g_j(z^k)$ if linearization is created from g_j at z^k [51]. This condition is met if the Cottle constraint qualification is satisfied at z^k.

The convergence to the global minimum can be proven similarly to the ECP method. The supporting hyperplane (16.9) will cut off the previous solution point z^k_{MILP}. Furthermore, it does not cut off any feasible point. The compactness of L and the Cauchy-Weierstrass Theorem will guarantee that the solution sequence $\{z^k_{\text{MILP}}\}$ has an accumulation point and it can be proven to be feasible. The linearity of the objective function then implies that the accumulation point is also optimal. For more details see [20].

Above we required that $G(z_{\text{int}}) < 0$. Essentially this means that (MINSO) must satisfy the Slater constraint qualification. In the case some constraint functions are active at the inner point, it is possible that algorithm gets stuck to an infinite loop [20]. In practice, where we can assure feasibility only up to a tolerance $\varepsilon_g > 0$, we can relax this condition. If in the line search we find a point z^k such that $g_{j_k}(z^k) = \frac{\varepsilon_g}{2}$ we can still guarantee to find an ε_g-feasible point. In this case it is sufficient that $G(z_{\text{int}}) \leq 0$. This kind of point exists if (MINSO) has a feasible solution. We will use this ploy in the subsequent consideration.

We can deal with the f°-pseudoconvex objective function similarly as we did in the αECP method. However, the inner point should satisfy $f(z_{\text{int}}) - f_r \leq 0$, for any upper bound f_r. Eventually, f_r will be close to the optimal value, and thus, $f(z_{\text{int}})$ should be less than the global minimum value. This can be achieved if we first solve the original problem with integer variables relaxed to continuous variables and let z_{int} be the solution of this relaxed problem. Essentially this means we are solving the root node of NSO branch and bound method. Another way is to have separate inner points for the objective function and for the other constraints. The inner point for the objective function, denoted by z^f_{int}, can be updated, whenever new upper bound f_r is found.

In [51] MINLP problem with f°-pseudoconvex objective function is solved without using the reduction constraint $f(z) - f_r \leq 0$. The procedure relies on line search between obtained feasible point of the MINSO problem and point z^f_{int} such that $f(z^f_{\text{int}}) \leq f_r$. The point z^f_{int} can always be set to be the point where the upper bound f_r was obtained. Furthermore, if there are several points $z^1_f, z^2_f, \ldots, z^p_f$ satisfying $f(z^i_f) \leq f_r$ we can set

$$z^f_{\text{int}} = \sum_{i=1}^{p} \frac{1}{p} z^i_f. \tag{16.10}$$

The line search will find a point where the constraint (16.8) can be added. The line search was already studied for the αECP in [38], where it ended at a point from the set $\{z \mid f(z) = f_r\}$. It is sufficient to find a point from the region

$$\left\{z \mid f(z) = f_r + \varepsilon_g\right\}, \tag{16.11}$$

which allowed us to omit the reduction constraint, that in [38] prevented infinite loops.

Assuming that we update the upper bound f_r in constraints (16.8), the MILP problem solved at iteration k is

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & f_r + \boldsymbol{\xi}^T(z - z^k) \leq \mu,\ k \in I_f, \\ & \boldsymbol{\xi}^T_{j_k}(z - z^k) \leq 0,\ k \in I_g, \\ & z \in L \cap Z. \end{cases} \tag{ESH-MILP$_k$}$$

Here I_g represents iterations where MILP solution is not feasible in nonlinear constraints and hence a supporting hyperplane is added. Correspondingly, I_f denotes iterations where a feasible point is found and a line search is done to find a point from region (16.11).

Algorithm 16.4: ESH algorithm

Data: Give the tolerance parameters $\varepsilon_g, \varepsilon_f > 0$ and an inner point $z_{\text{int}} \in N \cap L$.

Step 1. Set $f_r = \infty$ and $k = r = 1$.

Step 2. Solve the problem (ESH-MILP$_k$). Denote the solution by $(z^k_{\text{MILP}}, \mu^k)$.

Step 3. If $\mu^k \geq f_r - \varepsilon_f$, then **stop**: f_r is the optimal value.

Step 4. If $G(z^k_{\text{MILP}}) > \varepsilon_g$, do a line search between z_{int} and z^k_{MILP} to find z^k such that $G(z^k) = \frac{\varepsilon_g}{2}$. Add to (ESH-MILP$_{k+1}$) the linear constraint $\xi^T_{j_k}(z - z^k) \leq 0$, where $\xi_{j_k} \in \partial g_j(z^k)$ and $g_j(z^k) = G(z^k)$.

Step 5. If $G(z^k_{\text{MILP}}) \leq \varepsilon_g$ then

Step 5.1. If $f(z^k_{\text{MILP}}) < f_r$, update $r = r + 1$. Set $z^k = z^k_{\text{MILP}}$ and $f_r = f(z^k)$. Update the constraints of type (16.8) by using the new value f_r.

Step 5.2. If $f_r \leq f(z^k_{\text{MILP}}) \leq f_r + \varepsilon_g$, set $z^k = z^k_{\text{MILP}}$.

Step 5.3. If $f(z^k_{\text{MILP}}) > f_r + \varepsilon_g$, calculate z^f_{int} from (16.10). Find z^k such that $f(z^k) = f_r + \varepsilon_g$ with a line search between z^f_{int} and z^k_{MILP}.

Step 5.4. Add to (ESH-MILP$_{k+1}$) the linear constraint $f_r + \xi^T(z - z^k) \leq \mu$, where $\xi \in \partial f(z^k)$.

Step 6. Set $k = k + 1$ and go to Step 2.

Algorithm 16.4 will find an ε_g-feasible point after a finite number of iterations if $\varepsilon_g > 0$. Then we can add new constraint (16.8) and sometimes find a new upper bound. If $\varepsilon_f = 0$ the algorithm will converge to a global minimizer.

Theorem 16.4 *Algorithm 16.4 converges to an ε_g-feasible global minimizer of* (MINSO) *or finds one after a finite number of iterations.*

Proof Proof can be found in [51]. □

While the ESH may require more function evaluations than the αECP due to the line searches, sometimes it creates tighter approximation of the feasible set. This can result in finding a minimizer in less number of iterations. Furthermore, in theory the ESH handles f°-pseudoconvex functions better without needing the α coefficients. Due to the α coefficients the αECP may cut off small parts of the feasible set and, updating the coefficients, the number of MILP problems to be solved increases. In [20] there was a remarkable difference in solving times between the ESH and αECP in favour of the ESH in an instance of a facility layout problem with pseudoconvex constraints. This was a result of the αECP generating harder MILP problems with α-cutting-planes.

In [51] there was an example that showed the benefits of being able to solve MINSO problems with f°-pseudoconvex objective function rigorously. The objective function of the problem is maximum of four pseudoconvex functions

while the constraints were linear. This problem can be solved with Algorithm 16.4. Alternatively, we can also get rid of nonsmoothness by introducing four nonlinear constraints $f_i(z) - \mu \leq 0$, but then we need a nonconvex MINLP method to solve the problem to the global minimum. The BONMIN solver in GAMS [23] solved the problem fastest (30 s) followed by the ESH (70 s) while solvers like SCIP and LINDOGlobal could not find the minimum in 1000 s.

16.2.5 Extended Level Bundle Method

In [12] the *extended level bundle method* (ELBM) was introduced. It is based on the NSO algorithm level bundle method [32, 44]. Similarly to the ECP method no NSO problems are needed to solve while solving a sequence of MILP subproblems will lead to the global minimum of the problem (MINSO).

As in the ECP method, cutting-planes are created from nonlinear functions in MILP solution points. In the ELBM linearizations are generated from all nonlinear functions contrary to the ECP, where some linearizations are generated from some violated constraints. The objective function of the MILP problem is the stability function $\varphi(\cdot\,; \hat{z}^k)$, where $\hat{z}^k$ is the stability center at iteration k. The stability center can be e.g. a fixed point, the last iterate z^k or the incumbent iterate defined later in the algorithm pattern. In [12] stability functions

$$\varphi(z; \hat{z}^k) = \|z - \hat{z}^k\|_1 \text{ and } \varphi(z; \hat{z}^k) = \|z - \hat{z}^k\|_\infty$$

were considered in more detail. Both of these functions can be reformulated with auxiliary variables so that the objective becomes linear and the resulting problem is an MILP problem. An interesting case to notice is the stability function

$$\varphi(z; \hat{z}^k) = \hat{f}(z) + \frac{1}{2t_k}\|z - \hat{z}^k\|,$$

where $t_k > 0$ is a parameter and $\hat{f}$ is the linear approximation of f so far. In this case the ELBM algorithm is straightforward generalization of the level bundle method presented in [14] to the mixed integer problems. However, this stability function results in MIQP problems instead of MILP problems.

The purpose of the stability function is to keep the solutions near the point where approximation of the functions are good. This ability is lacking for example in the ECP method which may cause the consecutive solutions to be far away. Stability function also allows better utilization of good initial solution z^0.

At iteration k we solve subproblem

$$\begin{cases} \text{minimize} & \varphi(z; \hat{z}^k) \\ \text{subject to} & f(z^i) + \boldsymbol{\xi}^T(z^i)(z - z^i) \leq f^k_{\text{lev}}, i \in B^k, \\ & g_j(z^i) + \boldsymbol{\xi}_j(z^i)^T(z - z^i) \leq 0, j \in \mathcal{J}, i \in B^k, \\ & z \in L \cap Z, \end{cases} \qquad \text{(ELBM-MILP}_k\text{)}$$

where $\boldsymbol{\xi}(z^i) \in \partial f(z^i)$ and $\boldsymbol{\xi}_j(z^i) \in \partial g_j(z^i)$ are arbitrary. The set B^k, called bundle, defines at which solution points the functions are linearized. The constant f_{lev}^k is defined by

$$f_{\text{lev}}^k = f_{\text{low}}^k + \gamma h^k, \tag{16.12}$$

where f_{low}^k is the current lower bound, $\gamma \in (0, 1)$ is a chosen parameter and h^k is the residual of optimality defined by

$$h^k = \min_{i=1,2,\ldots,k} \max_{j \in \mathcal{J}} \left\{ f(z^i) - f_{\text{low}}^k, g_j(z^i) \right\}. \tag{16.13}$$

By solving (ELBM-MILP$_k$) we get another point where linearizations are created and possibly a new value for h^k. If the MILP problem is infeasible there are no feasible points where f attains value f_{lev}^k. Thus, we obtain a new lower bound $f_{\text{low}}^k = f_{\text{lev}}^k$, which may make the MILP problem feasible. This way the ELBM generates increasing sequence of lower bounds converging to the global minimum. The ELBM algorithm is presented in Algorithm 16.5.

The rule for bundle update is user specified. The smaller the bundle the easier the MILP problems are. However, then the linear approximations become less accurate which may result in more iterations. Bundle can be updated only if reasonable improvement of h^k has occurred and this prevents infinite loops due to too frequent bundle updates.

Theorem 16.5 *If $\varepsilon_g = 0$, Algorithm 16.5 converges to a global minimizer of* (MINSO) *or finds it after a finite number of iterations.*

Proof The convergence is proven in [12]. □

Algorithm 16.5: ELBM algorithm

Data:	Give the tolerance parameter $\varepsilon_g > 0$, $\gamma \in (0, 1)$ and initial point $\hat{z}^0 = z^0 \in L \cap Z$. Give f_{low}^0 or calculate it from an MILP problem based on linear approximations on z^0.
Step 1.	Set $k = 0$, $l = 0$, $k_l = 0$ and $B^0 = \{0\}$.
Step 2.	Define h^k as in (16.13) and denote z_{best}^k the point at which it is attained. If $h^k \leq \varepsilon_g$, **stop**: the optimum is z_{best}^k.
Step 3.	If $h^k \leq (1-\gamma)h^{k_l}$, choose bundle B^k with restriction that $\{k\} \subset B^k$. A new stability center $\hat{z}^k = z_{\text{best}}^k$ is set. Set $k_{l+1} = k$ and $l = l + 1$.
Step 4.	Calculate f_{lev}^k by Eq. (16.12) and solve the MILP problem (ELBM-MILP$_k$).
Step 5.	If the MILP problem is infeasible, update $f_{\text{low}}^{k+1} = f_{\text{lev}}^k$. Set $B^{k+1} = B^k$, $k_{l+1} = k$ and $l = l + 1$.
Step 6.	If the MILP problem has solution z^{k+1}, update the bundle $B^{k+1} = B^k \cup \{k+1\}$ and set $f_{\text{low}}^{k+1} = f_{\text{low}}^k$.
Step 7.	Set $k = k + 1$ and go to Step 2.

The convergence can be seen as follows. For a given f_{lev}^k the problem

$$\begin{cases} \text{minimize} & \varphi(z; \hat{z}^k) \\ \text{subject to} & f(z) \leq f_{\text{lev}}^k, \\ & g_j(z) \leq 0,\ j \in \mathcal{J}, \\ & z \in L \cap Z. \end{cases} \tag{ELBM-MINSO}$$

is solved with a sequence of MILP problems (ELBM-MILP$_k$) as in the ECP method. Suppose that index l remains the same, that is, no new f_{low}^k is found and h^k is greater than $(1-\gamma)h^{k_l}$. Similarly as in the convergence proof of the ECP method, the ELBM will generate a sequence that will converge to a feasible point of (ELBM-MINSO). At the feasible point $\overline{z}$ we have, according to the Eq. (16.12),

$$f(\overline{z}) - f_{\text{low}}^k \leq f_{\text{lev}}^k - f_{\text{low}}^k = \gamma h^k < h^k.$$

Thus, after a finite number of iterations h^k and f_{lev}^k will decrease. Also h^k would converge to 0 unless (ELBM-MINSO) becomes infeasible. In both cases index l will be increased after finite number of iterations.

The index l is increased if one of the conditions in Step 3 and 5 holds. In Step 5 the lower bound will be increased by amount γh^k. If h^k does not converge to 0, there can be only finitely many increases of the lower bound, otherwise, $f_{\text{low}}^k \to \infty$. In turn in Step 3 the condition $h^k \leq (1-\gamma)h^{k_l}$ holds. If this holds true infinitely many times $h^k \to 0$ since $(1-\gamma) < 1$. Thus, $h^k \to 0$ if the algorithm does not stop after a finite number of iterations.

The stability function does not play any role in the convergence proof and consequently MILP problems do not need to be solved to the optimality, feasibility being enough. the ELBM carries the same strength as the ECP, namely the lack of NSO problems which usually lead to fewer number of function evaluations. Another strength that also the ECP and OA share is that the functions are needed to evaluate only at point where integer variables attain integer values. In numerical comparison [12] the ELBM was found out to be more effective on average than a variant of the ECP method in terms of CPU and number of function evaluations when solving example problems. The test set included 39 problems from the MINLP Library [35], several instances of an academic chance constrained problem and several instances of stochastic programming problem. The latter two problem types have hard to evaluate functions making cutting-plane type methods preferable than the ones needing to solve NSO problems.

16.3 Illustrative Example

In this section we illustrate the presented algorithms by solving an easy example problem. The problem reads

$$\begin{cases} \text{minimize} & f(x, y) := |x - 4| + |y - 4| \\ \text{subject to} & g(x, y) := \max\left\{(y-2)^2 + x^2 - 9, x + 2y - 9\right\} \le 0, \\ & 0 \le x \le 5, y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

The objective function and the constraint function are convex but nonsmooth. Parameter value $\varepsilon = 0.1$ for stopping criteria has been used for all the algorithms.

NSO Branch and Bound Method The NSO branch and bound begins by solving the integer relaxed problem NSO(0,5) and finds $(x_1, y_1) = (1 + \frac{4}{\sqrt{5}}, 4 - \frac{2}{\sqrt{5}})$. It is not integer feasible, and thus, two problems NSO(0, 3) and NSO(4, 5) are created. Solving NSO(0, 3) results in the integer feasible solution $(x_1, y_1) = (2\sqrt{2}, 3)$ and no new problems need to be created. Also the integer feasible solution gives an upper bound $f(2\sqrt{2}, 3) \approx 2.17$. The solution to the problem NSO(4, 5) is $(x_2, y_2) = (1, 4)$ giving an upper bound $f(1, 4) \approx 3$. There are no problems left to solve and the best upper bound 2.17 is the global minimum found at $(2\sqrt{2}, 3)$. The solution process of NSO B&B as well as the example problem are illustrated in Fig. 16.1.

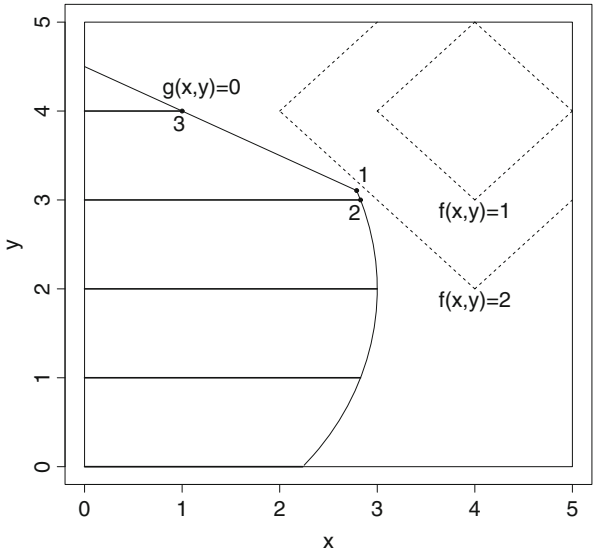

Fig. 16.1 The solution points found by NSO branch and bound algorithm. Thick lines correspond to feasible region of the example problem

OA Method For the OA algorithm suppose that $y^1 = 4$. The solution of OA-NSO(4) is $x^1 = 1$ giving an upper bound $U^1 = 3$. At this point g is differentiable and its linearization is $x + 2y - 9$. The objective function is not differentiable at this point and its subdifferential is $\partial f(1, 4) = \{(-1, \lambda) \mid \lambda \in [-1, 1]\}$. The component corresponding to x is unique, and thus, we may choose any subgradient. Let us choose $\boldsymbol{\xi} = (-1, 0)$ resulting in linearization $2 - x$. The problem OA-MILP$_1$ then reads

$$\begin{cases} \text{minimize} & \eta \\ \text{subject to} & \eta \leq 3 - 0.1, \\ & 2 - x \leq \eta, \\ & x + 2y - 9 \leq 0, \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

The solution to this problem is not unique with possibilities being $(5, 0)$, $(5, 1)$ and $(5, 2)$ giving η value -3. Suppose the MILP algorithm gives us $(5, 2)$. The solution point of OA-NSO(2) is $(3, 2)$ giving the same upper bound 3 as the previous integer solution. At this point both of the nonlinear functions are differentiable and OA-MILP$_2$ is

$$\begin{cases} \text{minimize} & \eta \\ \text{subject to} & \eta \leq 3 - 0.1 \\ & 2 - x \leq \eta \\ & 8 - x - y \leq \eta \\ & x + 2y - 9 \leq 0 \\ & 6x - 18 \leq 0 \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

The unique solution to this problem is $(3, 3)$ giving $\eta = 2$. Solving OA-NSO(2) gives $(2\sqrt{2}, 3)$ with a new upper bound $U^3 = 2.17$. The next MILP problem will be infeasible stopping the algorithm. The solution process is depicted in Fig. 16.2.

ECP Method The ECP method begins with solving MILP problem without any linearizations from the nonlinear functions. This, essentially feasibility problem, reads

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\},\ -10 \leq \mu \leq 10, \end{cases}$$

where bounds to the μ guarantee a finite solution. Suppose the obtained solution is $(x, y, \mu) = (5, 5, -10)$. Both nonlinear constraints $f - \mu$ and g are violated at this point. To speed up the solving process we make linearizations from all violated

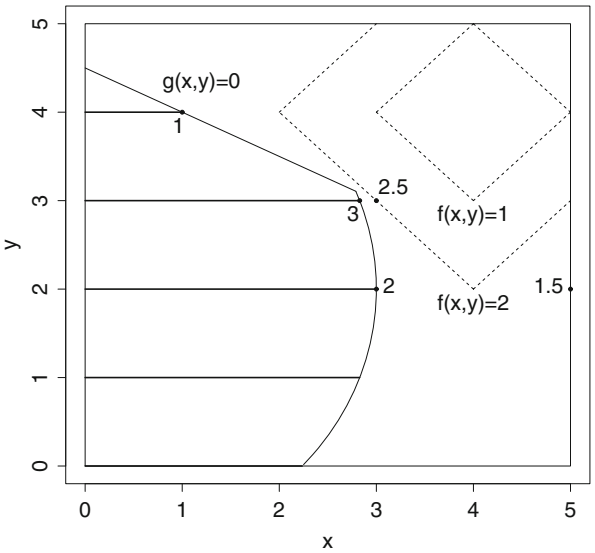

Fig. 16.2 The solution points found by outer approximation method. Points "1", "2" and "3" correspond to solution points of the NSO problems while "1.5" and "2.5" are solution points of the MILP problems

constraints. The problem ECP-MILP$_2$ is

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & x + y - 8 \leq \mu, \\ & 10x + 6y - 55 \leq 0, \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\},\ -10 \leq \mu \leq 10, \end{cases}$$

and its solution is $(0, 0, -8)$. Only $f - \mu$ is violated and a new linearization $-x - y + 8 \leq \mu$ is added to the MILP problem. The solution to this MILP problem is $(2.5, 5, 0.5)$. After adding the new linearizations from this point the problem ECP-MILP$_4$ is

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & x + y - 8 \leq \mu \\ & -x - y + 8 \leq \mu \\ & -x + y \leq \mu \\ & 5x + 6y - 36.25 \leq 0 \\ & 10x + 6y - 55 \leq 0 \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\},\ -10 \leq \mu \leq 10. \end{cases}$$

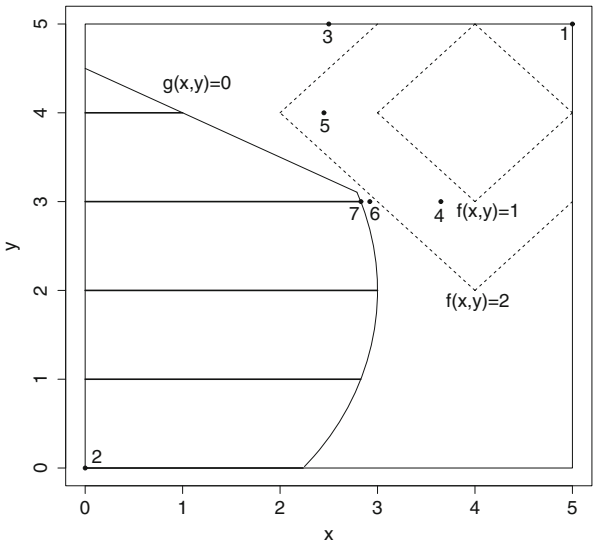

Fig. 16.3 The solution points found by extended cutting-plane method

The point $(3.65, 3, 1.35)$ will solve this problem and a new linearization $7.3x + 2y \leq 27.3$ is added to the problem. The algorithm continues similarly and stops at $(2.83, 3, 2.17)$ after 7th iteration. The solving process is depicted in Fig. 16.3.

ESH Method For the ESH method we will use readily found inner point $(0, 0, 10)$. The first solution point $(x, y, \mu) = (5, 5, -10)$ coincides with the solution point of the ECP method. Line search between these points finds $(1, 1, 6)$, where $f - \mu$ is active. A supporting hyperplane plane will be added to the MILP problem resulting in problem

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & -x - y + 8 \leq \mu, \\ & 0 \leq x \leq 5,\, y \in \{0, 1, 2, 3, 4, 5\}\,. \end{cases}$$

The solution point of this problem is $(5, 5, -2)$. The line search will find point $(2.87, 2.87, 3.11)$ where g is active. The constraint $5.74x + 1.74y - 21.5 \leq 0$ is added to the MILP problem. The next solution is $(2.22, 5, 0.77)$ and the line search will find point $(1.64, 3.68, 3.21)$. The linearization of g at this point is $x + 2y - 9$.

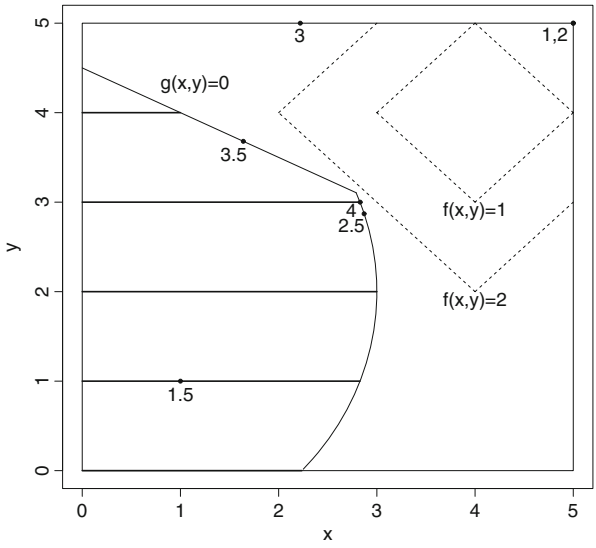

Fig. 16.4 The solution points found by extended supporting hyperplane method. Points "1", "2", "3", and "4" corresponds to MILP solution points. Points "1.5", "2.5" and "3.5" are found by the line search

The problem ESH-MILP$_4$ is

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & -x - y + 8 \leq \mu, \\ & 5.74x + 1.74y - 21.5 \leq 0, \\ & x + 2y - 9 \leq 0, \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

The solution point is $(2.83, 3, 2.17)$. At this point the stopping criterion is met and the algorithm stops. The solving process is depicted in Fig. 16.4.

ELBM For the ELBM let $x^0 = y^0 = 5$, $\gamma = 0.2$ and stability function $\varphi = \|\cdot\|_1$. To get the lower bound we first linearize nonlinear functions at $z^0 = (x^0, y^0) = (5, 5)$ and solve problem

$$\begin{cases} \text{minimize} & \mu \\ \text{subject to} & x + y - 8 \leq \mu, \\ & 10x + 6y - 55 \leq 0, \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

The solution is $(0, 0, -8)$ giving lower bound $f^0_{\text{low}} = -8$. Currently $B^0 = \{0\}$ and thus the stability center $\hat{z}^0 = z^0$,

$$h^0 = \max\left\{|5-4| + |5-4| - (-8), (5-2)^2 + 5^2 - 9\right\} = 25,$$
$$f^0_{\text{lev}} = f^0_{\text{low}} + \gamma h^0 = -8 + 0.2 \cdot 25 = -3.$$

The next point will be found by solving ELBM-MILP$_0$

$$\begin{cases} \text{minimize} & |x-5| + |y-5| \\ \text{subject to} & x + y - 8 \le -3, \\ & 10x + 6y - 55 \le 0, \\ & 0 \le x \le 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

Note that we could rewrite the objective by $\mu + \eta$ and add constraints

$$x - 5 \le \mu,\ -x + 5 \le \mu,\ y - 5 \le \eta,\ 5 - y \le \eta$$

to the problem to make it linear, and thus, an MILP problem. For simplicity, we do not do so in this presentation. The next iterate is not unique and we choose $z^1 = (0, 5)$. Since

$$\max\left\{|0-4| + |5-4| - (-8), (5-2)^2 + 0^2 - 9, 0 + 2 \cdot 5 - 9\right\} = 13 < h^0,$$

we set $h^1 = 13$. Inequality $h^1 \le (1 - 0.2)h^0$ holds and we set a new stability center $\hat{z}^1 = z^1$. We will keep every iterate in the bundle throughout this example and, thus, neglect updating the bundle B^k. Calculating $f^1_{\text{lev}} = -8 + 0.2 \cdot 13 = -5.4$ we obtain ELBM-MILP$_1$

$$\begin{cases} \text{minimize} & |x| + |y-5| \\ \text{subject to} & x + y - 8 \le -5.4, \\ & -x + y \le -5.4, \\ & 10x + 6y - 55 \le 0, \\ & x + 2y - 9 \le 0, \\ & 0 \le x \le 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

Table 16.1 Information on iterations when solving the example problem with the ELBM

Iteration	z^k	$\hat{z}^k$	z^k_{best}	h^k	f^k_{low}	f^k_{lev}
0	(5, 5)	(5, 5)	(5, 5)	25	− 8	− 3
1	(0, 5)	(0, 5)	(0, 5)	13	−8	−5.4
2	Infeasible	(0, 5)	(0, 5)	10.4	−5.4	−3.32
3	(3.32, 0)	(0, 5)	(3.32, 0)	10.08	−5.4	−3.38
4	Infeasible	(3.32, 0)	(3.32, 0)	8.06	−3.38	−1.77
5	Infeasible	(3.32, 0)	(3.32, 0)	6.45	−1.77	−0.48
6	Infeasible	(3.32, 0)	(3.32, 0)	5.48	−0.48	0.62
7	Infeasible	(3.32, 0)	(3.32, 0)	4.38	0.62	1.49
8	Infeasible	(3.32, 0)	(3.32, 0)	3.51	1.49	2.19
9	(3, 3)	(3, 3)	(3, 3)	1	1.49	1.69
10	Infeasible	(3, 3)	(3, 3)	1	1.69	1.89
11	Infeasible	(3, 3)	(3, 3)	1	1.89	2.09
12	Infeasible	(3, 3)	(3, 3)	1	2.09	2.29
13	(2.83, 3)	(2.83, 3)	(2.83, 3)	0.07	2.09	2.11

This problem is infeasible and we update $f^2_{\text{low}} = f^1_{\text{lev}} = -5.4$. Now $h^2 = 13 - 8 + 5.4 = 10.4$ and $f^2_{\text{lev}} = -5.4 + 0.2 \cdot 10.4 = -3.32$. The next MILP problem is

$$\begin{cases} \text{minimize} & |x| + |y - 5| \\ \text{subject to} & x + y - 8 \leq -3.32, \\ & -x + y \leq -3.32, \\ & 10x + 6y - 55 \leq 0, \\ & x + 2y - 9 \leq 0, \\ & 0 \leq x \leq 5,\ y \in \{0, 1, 2, 3, 4, 5\}. \end{cases}$$

Again, the solution is not unique and we pick $z^2 = (3.32, 0)$. The procedure continues and converges to the global optimum after 14th iteration. The solving process is reported in Table 16.1 and depicted in Fig. 16.5.

16.4 Concluding Remarks

In this chapter we considered deterministic algorithms for convex mixed integer NSO problems. Some of the algorithms can also solve problems with generalized convex functions. All of the algorithms require at least that we can calculate a subgradient from the Clarke subdifferential of the nonlinear functions at any point which algorithm visits. While most of the methods are based on MINLP algorithms for differentiable case, the ELBM is a generalization of an NSO algorithm to the

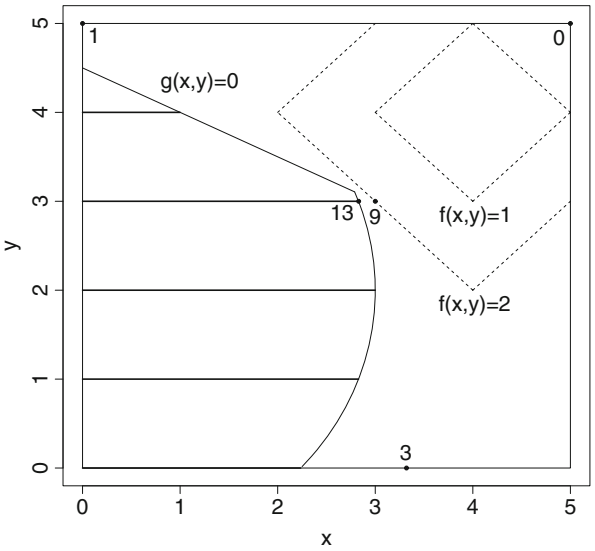

Fig. 16.5 The solution points found by the extended level bundle method

mixed integer case. By showing that this is possible, it also rises a question whether other bundle methods can be generalized to solve mixed integer problems as well.

References

1. Bagirov, A., Karmitsa, N., Mäkelä, M.M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, Berlin (2014)
2. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan A.: Mixed-integer nonlinear optimization. Acta Numer. **22**, 1–131 (2013)
3. Biegler, L.T., Grossmann, I.E., Westerberg, A.W.: Systematic Methods for Chemical Process Design. Prentice Hall, Upper Saddle River (1997)
4. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Program. **119**, 331–352 (2009)
5. Castillo, I., Westerlund, J., Emet, S., Westerlund, T.: Optimization of block layout design problems with unequal areas: a comparison of MILP and MINLP optimization methods. Comput. Chem. Eng. **30**, 54–69 (2005)
6. Chaib, A.E., Bouchekara, H.R.E.H., Mehasni, R., Abido, M.A.: Optimal power flow with emission and non-smooth cost functions using backtracking search optimization algorithm. Electr. Power Energy Syst. **81**, 64–77 (2016)
7. Clarke, F. H.: Optimization and Nonsmooth Analysis. Wiley-Interscience, New York (1983)
8. CPLEX. https://www.ibm.com/analytics/cplex-optimizer
9. Crouzeix, J.-P., Lindberg, P. O.: Additively decomposed quasiconvex functions. Math. Program. **35**, 42–57 (1986)
10. Dakin, R.J.: A tree-search algorithm for mixed integer programming problems. Comput. J. **8**, 250–255 (1965)

11. Delfino, A., de Oliveira, W.: Outer-approximation algorithms for nonsmooth convex MINLP problems. Optimization **67**(6), 797–819 (2018)
12. de Oliveira, W.: Regularized optimization methods for convex MINLP problems. TOP **24**, 665–692 (2016)
13. de Oliveira, W., Sagastizábal, C.: Bundle methods in the XXIst century: a bird's eye view. Pesquisa Operacional **34**, 647–670 (2014)
14. de Oliveira, W., Solodov, M.: A doubly stabilized bundle method for nonsmooth convex optimization. Math. Program. **156**(1), 125–159 (2016)
15. Drewes, S., Ulbrich, S.: Subgradient based outer approximation for mixed integer second order cone programming. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, pp. 41–59. Springer, New York (2012)
16. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**, 307–339 (1986)
17. Emet, S., Westerlund, T.: Comparisons of solving a chromatographic separation problem using MINLP methods. Comput. Chem. Eng. **28**, 673–682 (2004)
18. Eronen, V.-P., Mäkelä, M.M., Westerlund, T.: On the generalization of ECP and OA methods to nonsmooth MINLP problems. Optimization **63**(7), 1057–1073 (2014)
19. Eronen, V.-P., Mäkelä, M.M., Westerlund, T.: Extended cutting-plane method for a class of nonsmooth nonconvex MINLP problems. Optimization **64**(3), 641–661 (2015)
20. Eronen, V.-P., Kronqvist, J., Westerlund, T., Mäkelä, M.M., Karmitsa, N.: Method for solving generalized convex nonsmooth mixed-integer nonlinear programming problems. J. Glob. Optim. **69**(2), 443–459 (2017)
21. Fernandes, F.P., Costa, M.F.P., Fernandes, E.M.G.P.: Branch and bound based coordinate search filter algorithm for nonsmooth nonconvex mixed-integer nonlinear programming problems. In: Murgante, B., et al. (eds.) Computational Science and Its Applications—ICCSA 2014. ICCSA 2014. Lecture Notes in Computer Science, vol. 8580, pp. 140–153. Springer, Cham (2014)
22. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program. **66**, 327–349 (1994)
23. GAMS. https://www.gams.com/
24. Grossmann, I.E.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optim. Eng. **3**, 227–252 (2002)
25. Gurobi. http://www.gurobi.com/
26. Hamzeei, M., Luedtke, J.: Linearization-based algorithms for mixed-integer nonlinear programs with convex continuous relaxation. J. Glob. Optim. **59**, 343–365 (2014)
27. Kelley, J.E.: The cutting-plane method for solving convex programs. J. SIAM **8**, 703–712 (1960)
28. Khoury, G.A., Smadbeck, J., Kieslich, C.A., Floudas, C.A.: Protein folding and de novo protein design for biotechnological applications. Trends Biotechnol. **32**(2), 99–109 (2014)
29. Kravanja, S., Šilih, S., Kravanja, Z.: The multilevel MINLP optimization approach to structural synthesis: the simultaneous topology, material, standard and rounded dimension optimization. Adv. Eng. Softw. **36**(9), 568–583 (2005)
30. Kronqvist, J., Lundell, A., Westerlund, T.: The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. J. Glob. Optim. **64**, 249–272 (2016)
31. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica **28**(3), 497–520 (1960)
32. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**, 111–147 (1995)
33. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for mixed-integer constrained optimization problems. J. Optim. Theory Appl. **164**(3), 933–965 (2015)
34. Mäkelä, M.M.: Survey of bundle methods for nonsmooth optimization. Optim. Methods Softw. **17**(1), 1–29 (2002)
35. MINLP Library. http://www.minlplib.org/instances.html

36. Müller, J., Shoemaker, C.A., Piché, R.: SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. Comput. Oper. Res. **40**, 1383–1400 (2013)
37. Newby, E., Ali, M.M.: A trust-region-based derivative free algorithm for mixed integer programming. Comput. Optim. Appl. **60**, 199–229 (2015)
38. Pörn, R., Westerlund, T.: A cutting-plane method for minimizing pseudo-convex functions in the mixed integer case. Comput. Chem. Eng. **24**, 2655–2665 (2000)
39. Pörn, R., Nissfolk, O., Jansson, F., Westerlund, T.: The Coulomb glass—modeling and computational experience with a large scale 0–1 QP problem. Comput. Aided Chem. Eng. **29**, 658–662 (2011)
40. Quesada, I., Grossmann, I.E.: An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems. Comput. Chem. Eng. **16**, 937–947 (1999)
41. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**(3), 1247–1293 (2013)
42. Schmidt, M., Steinbach, M.C., Willert, B.M.: A primal heuristic for nonsmooth mixed integer nonlinear optimization. In: Jünger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 295–320. Springer, Berlin (2013)
43. Shor, N.Z.: Minimization Methods for Non-Differentiable Functions. Springer, Berlin (1985)
44. van Ackooij, W., de Oliveira, W.: Level bundle methods for constrained convex optimization with various oracles. Comput. Optim. Appl. **57**, 555–597 (2014)
45. Veinott Jr, A.F.: The supporting hyperplane method for unimodal programming. Oper. Res. **15**(1), 147–152 (1967)
46. Wei, Z., Ali, M.M.: Convex mixed integer nonlinear programming problems and an outer approximation algorithm. J. Glob. Optim. **63**(2), 213–227 (2015)
47. Wei, Z., Ali, M.M.: Outer approximation algorithm for one class of convex mixed-integer nonlinear programming problems with partial differentiability. J. Optim. Theory Appl. **167**(2), 644–652 (2015)
48. Westerlund, T., Pettersson, F.: An extended cutting-plane method for solving convex MINLP problems. Comput. Chem. Eng. **19**(Supplement 1), 131–136 (1995)
49. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting-plane techniques. Optim. Eng. **3**, 253–280 (2002)
50. Westerlund, T., Skrifvars, H., Harjunkoski, I., Pörn, R.: An extended cutting-plane method for solving a class of non-convex MINLP problems. Comput. Chem. Eng. **22**, 357–365 (1998)
51. Westerlund, T., Eronen, V.-P., Mäkelä, M.M.: On solving generalized convex MINLP problems using supporting hyperplane techniques. J. Glob. Optim. **71**(4), 987–1011 (2018). https://doi.org/10.1007/s10898-018-0644-z
52. Zioutas, G., Chatzinakos, C., Nguyen, T.D., Pitsoulis, L.: Optimization techniques for multivariate least trimmed absolute deviation estimation. J. Comb. Optim. **34**, 781–797 (2017)

Chapter 17
A View of Lagrangian Relaxation and Its Applications

Manlio Gaudioso

Abstract We provide an introduction to Lagrangian relaxation, a methodology which consists in moving into the objective function, by means of appropriate *multipliers*, certain *complicating* constraints of integer programming problems. We focus, in particular, on the solution of the *Lagrangian dual*, a nonsmooth optimization (NSO) problem aimed at finding the best multiplier configuration. The algorithm for solving the Lagrangian dual can be equipped with heuristic procedures for finding feasible solutions of the original integer programming problem. Such an approach is usually referred to as *Lagrangian heuristic*. The core of the chapter is the presentation of several examples of Lagrangian heuristic algorithms in areas such as assignment problems, network optimization, wireless sensor networks and machine learning.

17.1 Introduction

The relevance of any mathematical concept lies in its ability to generate many fruits in diverse areas and to produce long-lasting effects. This is definitely the case of the Lagrange multipliers [38], which have influenced the development of modern mathematics and are still fertile as far as mathematical programming theory and algorithms are concerned.

In this chapter we confine the discussion to the treatment of numerical optimization problems in a finite dimension setting, where the decision variables are the vectors of $\mathbb{R}^n$.

Looking at the cornerstones of the historical development of such an area, we observe that Lagrange multipliers have survived the crucial passages from equality- to inequality-constrained problems, as well as from continuous to discrete

M. Gaudioso (✉)
Universitá della Calabria, Rende, Italy
e-mail: manlio.gaudioso@unical.it

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_17

optimization. In addition they have shown their potential also when nonsmooth analysis has come into play [48, 49].

Originally the rationale of the introduction of Lagrange multipliers was to extend to the equality constrained case the possible reduction of an optimization problem to the solution of a system of nonlinear equations, based on the observation that an optimality condition must necessarily involve both objective function and constraints. The geometry of the constraints became more and more important as soon as inequality constrained problems were taken into consideration, and Lagrangian multipliers were crucial in the definition of the related optimality conditions.

The introduction of the duality theory highlighted the role of the Lagrangian multipliers in the game-theoretic setting, as well as in the study of value functions associated to optimization problems. More recently, the 1960s of last century, it was clear [18] that a Lagrangian multiplier-based approach was promising even in dealing with optimization problems of discrete nature, showing that in some sense it was possible to reduce the gap between continuous and integer optimization.

In this chapter we focus on Lagrangian relaxation, which was introduced in [29], for dealing with integer or mixed integer optimization problems and immediately conquered the attention of many scientist as a general-purpose tools for handling hard problems [21]. Pervasiveness of Lagrangian relaxation is well evidenced in [39].

The objective of the presentation is to demonstrate the usefulness of combined use of Lagrangian relaxation and heuristic algorithms to tackle hard integer programming problems. We will introduce several examples of application in fairly diverse areas of practical optimization.

The chapter is organized as follows. Basic notions on Lagrangian relaxation are in Sect. 17.2, while solution methods for tackling the Lagrangian dual are discussed in Sect. 17.3. A basic scheme of Lagrangian heuristics, together with an example of dual ascent for the classic set covering problem, is in Sect. 17.4. A number of applications in areas such as assignment, network optimization, sensor location, logistics and machine learning are discussed in Sect. 17.5. Some conclusions are drawn in Sect. 17.6.

17.2 Basic Concepts

We introduce first the following general definition.

Definition 17.1 (Relaxation) Given a minimization problem (P) in the following form:

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in X \subset \mathbb{R}^n, \end{cases} \tag{17.1}$$

with $f : \mathbb{R}^n \to R$, any problem (P_R)

$$\begin{cases} \text{minimize} & h(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in X_R \subset \mathbb{R}^n, \end{cases} \tag{17.2}$$

with $h : \mathbb{R}^n \to \mathbb{R}$, is a *relaxation* of (P), provided that the two following conditions hold:

$$X \subset X_R; \tag{17.3}$$

$$h(\boldsymbol{x}) \leq f(\boldsymbol{x}), \quad \boldsymbol{x} \in X. \tag{17.4}$$

Under the above definition, if $\boldsymbol{x}^*$ is any (global) minimum of (P_R), then $h(\boldsymbol{x}^*)$ is a lower bound on the optimal value of the problem (P). Of course it is advantageous to spend some effort to obtain such information whenever (P_R) is substantially easier to solve than (P).

There exist many different ways for constructing relaxations. For example, the usual continuous relaxation of any *integer linear programming* (ILP) problem satisfies the conditions (17.3) and (17.4).

We introduce Lagrangian relaxation starting from a *linear program* (LP) in standard form:

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq 0, \end{cases} \tag{17.5}$$

with $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. We assume that the problem is feasible and has optimal solution, so that the dual problem is feasible as well.

For any choice of the Lagrangian multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^m$ (or, simply, the multiplier vector), we define the Lagrangian relaxation $(LR(\boldsymbol{\lambda}))$

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \geq 0, \end{cases} \tag{17.6}$$

which can be rewritten as

$$\begin{aligned} \boldsymbol{b}^T \boldsymbol{\lambda} + \min \ & (\boldsymbol{c} - A^T \boldsymbol{\lambda})^T \boldsymbol{x} \\ & \text{subject to } \boldsymbol{x} \geq 0. \end{aligned} \tag{17.7}$$

Letting $z_{LR}(\boldsymbol{\lambda})$ be the optimal value of the problem (17.7), we obtain

$$z_{LR}(\boldsymbol{\lambda}) = \begin{cases} \boldsymbol{b}^T \boldsymbol{\lambda}, & \text{if} \quad \boldsymbol{c} - A^T \boldsymbol{\lambda} \geq 0, \\ -\infty, & \text{otherwise.} \end{cases} \tag{17.8}$$

From the definition of relaxation, $z_{LR}(\boldsymbol{\lambda})$ is a lower bound for the LP problem (17.5), possibly the trivial one for those values of $\boldsymbol{\lambda}$ which do not satisfy the condition $A^T\boldsymbol{\lambda} \leq \boldsymbol{c}$. It is quite natural to look for the *best* lower bound and this results in solving the problem

$$\text{maximize} \quad z_{LR}(\boldsymbol{\lambda}), \tag{17.9}$$

which, taking into account (17.8), is exactly the dual of (17.5),

$$\begin{cases} \text{maximize} & \boldsymbol{b}^T\boldsymbol{\lambda} \\ \text{subject to} & A^T\boldsymbol{\lambda} \leq \boldsymbol{c}. \end{cases} \tag{17.10}$$

The problem (17.9) will be referred to as the *Lagrangian dual* of the LP problem (17.5), and in fact the optimal solution $\boldsymbol{\lambda}^*$ of the dual is optimal for (17.9) as well.

The main motivation for the introduction of Lagrangian relaxation is the treatment of ILP problems. We will not consider in the sequel the *mixed integer linear programming* (MILP) case, where Lagrangian relaxation theory can be developed in a completely analogous way as in the pure case. The *binary linear programming* (BLP) problems can be considered as a special case of ILP. Thus we focus on the following problem:

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b}, \\ & B\boldsymbol{x} = \boldsymbol{d}, \\ & \boldsymbol{x} \geq 0, \ \text{integer}, \end{cases} \tag{17.11}$$

with $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{d} \in \mathbb{R}^p$. We assume that the problem is feasible and that the set

$$X = \{\boldsymbol{x} \in \mathbb{R}^n \mid B\boldsymbol{x} = \boldsymbol{d}, \ \ \boldsymbol{x} \geq 0, \ \text{integer}\}$$

is finite, that is $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K\}$ and we denote by $\mathcal{K} = \{1, 2, \ldots, K\}$ the corresponding index set. In writing the problem (17.11) two different families of constraints are highlighted, those defined through $A\boldsymbol{x} = \boldsymbol{b}$ being the *complicating* ones. By focusing exclusively on such set of constraints, we come out with the Lagrangian relaxation defined for $\boldsymbol{\lambda} \in \mathbb{R}^m$

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{b} - A\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in X, \end{cases} \tag{17.12}$$

which is supposed easier to solve than the original problem. By letting $\boldsymbol{x}(\boldsymbol{\lambda})$ and $z_{LR}(\boldsymbol{\lambda})$ be, respectively, the optimal solution and the optimal value of (17.12), it is

$$z_{LR}(\boldsymbol{\lambda}) = \boldsymbol{c}^T \boldsymbol{x}(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) = \min_{k \in \mathcal{K}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k). \tag{17.13}$$

Remark 17.1 Function $z_{LR}(\boldsymbol{\lambda})$ is often referred to as the *dual* function. Note that, in case $\boldsymbol{x}(\boldsymbol{\lambda})$ is feasible, that is $A\boldsymbol{x}(\boldsymbol{\lambda}) = \boldsymbol{b}$, then it is also optimal for the original problem (17.11)

In order to look for the best among the lower bounds $z_{LR}(\boldsymbol{\lambda})$, we define also in this case the Lagrangian dual

$$z_{LD} = \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad z_{LR}(\boldsymbol{\lambda}), \tag{17.14}$$

that is, from (17.13)

$$z_{LD} = \max_{\boldsymbol{\lambda}} \min_{k \in \mathcal{K}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k). \tag{17.15}$$

The optimal value z_{LD} is the best lower bound obtainable through Lagrangian relaxation.

It is worth noting that the problem (17.15), which we will discuss later in more details, consists in the maximization of a concave and piecewise affine function. It is in fact the NSO problems which can be tackled by means of any of the methods described in this book. On the other hand, by introducing the additional variable $v \in \mathbb{R}$, it can be rewritten in the following LP form:

$$\begin{cases} \underset{\boldsymbol{\lambda}, v}{\text{maximize}} & v \\ \text{subject to} & v \leq \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k), \quad k \in \mathcal{K}, \end{cases} \tag{17.16}$$

whose dual, in turn, is

$$\begin{cases} \underset{\boldsymbol{\mu}}{\text{minimize}} & \boldsymbol{c}^T (\sum_{k \in \mathcal{K}} \mu_k \boldsymbol{x}_k) \\ \text{subject to} & A(\sum_{k \in \mathcal{K}} \mu_k \boldsymbol{x}_k) = \boldsymbol{b}, \\ & \sum_{k \in \mathcal{K}} \mu_k = 1, \\ & \mu_k \geq 0, \ k \in \mathcal{K}, \end{cases} \tag{17.17}$$

or, equivalently,

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \in \text{conv}\, X, \end{cases} \tag{17.18}$$

where $X = \{\boldsymbol{x} \in \mathbb{R}^n \mid B\boldsymbol{x} = \boldsymbol{d},\ \boldsymbol{x} \geq 0 \text{ integer}\}$. The Lagrangian dual is thus a *partially convexified* version of the ILP (17.11). From the inclusion

$$\text{conv}\, X \subseteq \{\boldsymbol{x} \in \mathbb{R}^n \mid B\boldsymbol{x} = \boldsymbol{d},\ \boldsymbol{x} \geq 0\} = \bar{X},$$

it follows

$$z_{LD} \geq z_{LP}, \tag{17.19}$$

where z_{LP} is the optimal value of the continuous LP relaxation of (17.11):

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b}, \\ & B\boldsymbol{x} = \boldsymbol{d}, \\ & \boldsymbol{x} \geq 0. \end{cases} \tag{17.20}$$

The property (17.19) indicates that the lower bound provided by the *best* (not by any!) multiplier vector setting is not worse than the optimal value of the continuous relaxation. Moreover, it is important to observe that in case the so-called *integrality property* holds, that is in case the vertices of the polyhedron $\bar{X}$ are integer, it is $z_{LD} = z_{LP}$.

Whenever the integrality property is satisfied (and this is a rather common case in integer programming applications) Lagrangian relaxation may appear a quite weak approach compared to classic continuous relaxation: we need to solve a NSO problem of the *maxmin* type just to get he same bound obtainable by solving a LP! Nonetheless, Lagrangian relaxation may be a useful tool also in this case for the following reasons:

- in several applications the continuous relaxation is a huge LP and it may be a good idea not to tackle it;
- in Lagrangian relaxation the decision variables (think e.g. binary variables) keep the original physical meaning, which, instead, gets lost in continuous relaxation;
- the possibly infeasible solutions of the Lagrangian relaxation are often more suitable for *repairing* heuristics than those of the continuous relaxation.

Coming back to the two formulations (17.15) and (17.17), we observe that they offer two possible schemes for solving the Lagrangian dual.

Consider first (17.15), which is a finite *maxmin* problem where the function to be maximized is concave. In fact it is defined as the pointwise minimum of a finite (possibly very large) number of affine functions of variable $\boldsymbol{\lambda}$, one for each $\boldsymbol{x}_k \in X$. Application of the classic cutting plane model consists in generating an (upper) approximation of the min function, initially based on a relatively small number of affine pieces, which becomes more and more accurate as new affine pieces (*cuts*) are added. In practice, taking any subset $\mathcal{S} \subset \mathcal{K}$ (and the correspondent points $\boldsymbol{x}_k \in X$, $k \in \mathcal{S}$) the *cutting plane* approximation $z_{\mathcal{S}}(\boldsymbol{\lambda})$ of $z_{LR}(\boldsymbol{\lambda})$ is defined as

$$z_{\mathcal{S}}(\boldsymbol{\lambda}) = \min_{k \in \mathcal{S}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k),$$

and thus we come out with the *restricted primal* problem

$$z_{restr}(\mathcal{S}) = \max_{\boldsymbol{\lambda}} \min_{k \in \mathcal{S}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k), \tag{17.21}$$

which, in turn, can be put in the LP form of the type (17.16):

$$\begin{cases} \underset{\boldsymbol{\lambda}, v}{\text{maximize}} & v \\ \text{subject to} & v \le \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k), \quad k \in \mathcal{S}. \end{cases} \tag{17.22}$$

Assuming the problem (17.21) is not unbounded and letting $\boldsymbol{\lambda}_{\mathcal{S}}$ be any optimal solution, we calculate now $z_{LR}(\boldsymbol{\lambda}_{\mathcal{S}})$ (in fact we solve the Lagrangian relaxation for $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{\mathcal{S}}$):

$$z_{LR}(\boldsymbol{\lambda}_{\mathcal{S}}) = \min_{k \in \mathcal{K}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}_{\mathcal{S}}{}^T (\boldsymbol{b} - A\boldsymbol{x}_k) = \boldsymbol{b}^T \boldsymbol{\lambda}_{\mathcal{S}} + \min_{k \in \mathcal{K}} (\boldsymbol{c} - A^T \boldsymbol{\lambda}_{\mathcal{S}})^T \boldsymbol{x}_k. \tag{17.23}$$

Letting the optimal solution of the above problem be attained in correspondence to any index, say $k_{\mathcal{S}}$, and assuming $z_{LR}(\boldsymbol{\lambda}_{\mathcal{S}})$ be sufficiently smaller than $z_{restr}(\mathcal{S})$, the procedure is then iterated after augmenting the set of affine pieces in (17.21) by the one associated to the newly generated point $\boldsymbol{x}_{k_{\mathcal{S}}}$.

Consider now the formulation of the Lagrangian dual provided by (17.17). It is a (possibly very large) LP in a form particularly suitable for column generation [53]. We observe in fact that the columns are associated to points $\boldsymbol{x}_k$, $k \in \mathcal{K}$, in perfect analogy with the association affine pieces–points of X in the formulation (17.15). To apply the simplex method it is necessary to start from any subset of columns providing a basic feasible solution and then look for a column with negative reduced cost. Such operation is not necessarily accomplished by examining the reduced costs of *all* non basic columns, instead it can be implemented by solving a *pricing* problem. In fact the constraint matrix in (17.17) has size $(m + 1) \times K$ and has the form:

$$\begin{bmatrix} A\boldsymbol{x}_1 & \dots & A\boldsymbol{x}_K \\ 1 & \dots & 1 \end{bmatrix}. \tag{17.24}$$

Letting $\boldsymbol{\lambda}_B$ be the vector assembling the first m dual variables associated to the basis, the components of the reduced cost vector $\hat{\boldsymbol{c}}$ are:

$$\hat{\boldsymbol{c}}_k = (\boldsymbol{c} - A^T\boldsymbol{\lambda}_B)^T\boldsymbol{x}_k - \lambda_{m+1}, \quad k \in \mathcal{K},$$

where λ_{m+1} is the last dual variable and plays the role of a constant in the definition of $\hat{\boldsymbol{c}}_k$, $k \in \mathcal{K}$. Consequently, the pricing problem consists of solving

$$\min_{k\in\mathcal{K}} (\boldsymbol{c} - A^T\boldsymbol{\lambda}_B)^T\boldsymbol{x}_k, \tag{17.25}$$

which is a problem formally identical to (17.23).

Remark 17.2 It is worth noting that both (17.23) and (17.25) need not be solved at optimality, as for the former it is sufficient to generate a *cut*, that is to calculate any index k, $k \notin \mathcal{S}$ such that

$$\boldsymbol{c}^T\boldsymbol{x}_k + \boldsymbol{\lambda}_{\mathcal{S}}{}^T(\boldsymbol{b} - A\boldsymbol{x}_k) < z_{restr}(\mathcal{S}),$$

while for the latter the optimization process can be stopped as soon as a column with the negative reduced cost has been detected.

We observe, finally, that in the formulation (17.11) the complicating constraints $A\boldsymbol{x} = \boldsymbol{b}$ are in the equality form. Lagrangian relaxation is well defined also in case they are of the type $A\boldsymbol{x} \geq \boldsymbol{b}$, the only difference being in the need of setting $\boldsymbol{\lambda} \geq 0$. This fact implies that the Lagrangian dual is now

$$\max_{\boldsymbol{\lambda}\geq\boldsymbol{0}} \quad z_{LR}(\boldsymbol{\lambda}). \tag{17.26}$$

It is worth noting that feasibility of $\boldsymbol{x}(\boldsymbol{\lambda})$ no longer implies optimality. In fact it can be easily proved that now $\boldsymbol{x}(\boldsymbol{\lambda})$ is only ϵ-optimal, for

$$\epsilon = (A\boldsymbol{x}(\boldsymbol{\lambda}) - \boldsymbol{b})^T\boldsymbol{\lambda} \geq 0.$$

17.3 Tackling the Lagrangian Dual

Solving the Lagrangian dual problem (17.14) requires maximization of a concave and piecewise affine function (see (17.15)). Consequently, all the available machinery to deal with convex nondifferentiable optimization can be put in action. We have already sketched in previous section possible use of the cutting plane method [13, 35]. On the other hand the specific features of the Lagrangian dual can be fruitfully exploited.

The basic distinction is between algorithms which do or do not use the differential properties of function z_{LR}. Observe that a subgradient (concavity would suggest

the more appropriate term *supergradient*) is immediately available as soon as z_{LR} has been calculated. Letting in fact (see (17.13))

$$z_{LR}(\boldsymbol{\lambda}) = \boldsymbol{c}^T \boldsymbol{x}(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) = \min_{k \in \mathcal{K}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}_k)$$

we observe that, for any $\boldsymbol{\lambda}' \in \mathbb{R}^m$, it is

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda}') &= \min_{k \in \mathcal{K}} \boldsymbol{c}^T \boldsymbol{x}_k + \boldsymbol{\lambda}'^T (\boldsymbol{b} - A\boldsymbol{x}_k) \leq \boldsymbol{c}^T \boldsymbol{x}(\boldsymbol{\lambda}) + \boldsymbol{\lambda}'^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) \\ &= \boldsymbol{c}^T \boldsymbol{x}(\boldsymbol{\lambda}) + \boldsymbol{\lambda}'^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) + \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) - \boldsymbol{\lambda}^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) \\ &= z_{LR}(\boldsymbol{\lambda}) + (\boldsymbol{\lambda}' - \boldsymbol{\lambda})^T (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})), \end{aligned}$$

thus $(\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda}))$ is a subgradient of z_{LR} at $\boldsymbol{\lambda}$, that is $\boldsymbol{g}(\boldsymbol{\lambda}) = (\boldsymbol{b} - A\boldsymbol{x}(\boldsymbol{\lambda})) \in \partial z_{LR}(\boldsymbol{\lambda})$.

As for methods that use the subgradient for maximizing $z_{LR}(\boldsymbol{\lambda})$, we mention first the classic (normalized) subgradient method [46, 51], where the h-th iteration is

$$\boldsymbol{\lambda}_{h+1} = \boldsymbol{\lambda}_h + t_h \frac{\boldsymbol{g}(\boldsymbol{\lambda}_h)}{\|\boldsymbol{g}(\boldsymbol{\lambda}_h)\|},$$

t_h being the *step size* along the normalized subgradient direction. We remark that monotonicity of the sequence of values $z_{LR}(\boldsymbol{\lambda}_h)$ is not ensured, while convergence to a maximum is guaranteed under the well known conditions on the step size $t_h \to 0$ and $\sum_{h=1}^{\infty} t_h \to \infty$. Very popular formulae for setting t_h are

$$t_h = \frac{C}{h}, \tag{17.27}$$

and the Polyak formula

$$t_h = \frac{\hat{z}_{LD}^{(h)} - z_{LR}(\boldsymbol{\lambda}_h)}{\|\boldsymbol{g}(\boldsymbol{\lambda}_h)\|}, \tag{17.28}$$

where C is any positive constant and $\hat{z}_{LD}^{(h)}$ is an improving overestimate at the iteration h of z_{LD}, the optimal value of the Lagrangian dual.

Subgradient-type were the first and the most widely used methods for dealing with the Lagrangian dual, despite their slow convergence, mainly for their implementation simplicity. In more recent years, stemming from the approach introduced in [44], the so-called *fast gradient* methods have received considerable attention [26]. The approach consists in smoothing first the objective function and applying next gradient-type methods. Besides such stream, *incremental* subgradient methods, which are applicable whenever the objective function is expressed as a sum of several convex *component* functions, have been devised to deal with the Lagrangian dual [7, 36].

A careful analysis of the performance of subgradient methods for Lagrangian relaxation is in [25].

The cutting plane method previously summarized has been the building block for devising two families of algorithms, the bundle [34] and the analytic center [30] methods for convex minimization (or, equivalently, concave maximization). They can be considered as an evolution of the cutting plane, aimed at overcoming some inherent weakness in terms of stability, speed and well posedness. An update discussion on bundle methods is in Chap. 3 of this book.

The above methods have been successfully employed in several applications of Lagrangian relaxation in frameworks such as energy management [5, 8, 19, 31, 40], network design [25], train timetabling [20], telecommunication networks [41], logistics [43] etc.

As for methods which do not make explicit use of the subgradient of function z_{LR}, they are iterative algorithms of the coordinate or block-coordinate search type, according to the fact that at each iteration just one or a (small) subset of multipliers is modified. The update is aimed at increasing z_{LR}, thus such class of methods is, in general, referred to as the *dual ascent* one [33].

The choice of the component of the current multiplier vector $\boldsymbol{\lambda}$ to be updated is usually driven by the properties of the optimal solution $\boldsymbol{x}(\boldsymbol{\lambda})$. In general one picks up a violated constraint and modifies the corresponding multiplier trying to achieve a twofold objective:

- to increase z_{LR};
- to ensure feasibility of the previously violated constraint.

In several cases it is possible to calculate exactly (sometimes by solving some auxiliary problem) the multiplier update which guarantees satisfaction of both the above objectives. Most of the times, however, it is necessary to introduce a line search capable to handle possible *null step* exit, which cannot be excluded as consequence of nonsmoothness of function z_{LR}.

From the computational point of view, comparison of dual ascent and subgradient methods shows that the former produce in general more rapid growth of the dual function z_{LR}. On the other hand, dual ascent algorithms are often affected by premature stop at points fairly far from the maximum, whenever no coordinate direction is actually an ascent one. In such a case it is useful to accommodate for re-initialization, by adopting any subgradient as restart direction.

17.4 Lagrangian Heuristics

As pointed out in [39], Lagrangian relaxation is more than just a technique to calculate lower bounds. Instead, it is a general philosophy to approach problems which are difficult to tackle, because of their intrinsic complexity.

Lagrangian relaxation has been extensively used in the framework of *exact* methods for integer programming. We will not enter into the discussion on the best ways to embed Lagrangian relaxation into branch and bound, branch and cut and branch and price algorithms. We refer the interested reader to the surveys [24] and [32].

We will focus, instead, on the use of Lagrangian relaxation in the so-called *Lagrangian heuristic* framework for solving the problem (17.11). The computational scheme is fundamentally the following:

Algorithm 17.1: Lagrangian heuristic

Step 0.	(*Initialization*) Choose $\boldsymbol{\lambda}^{(0)}$. Set $k = 0$ and set $z_{UB} = \infty$.
Step 1.	(*Lagrangian relaxation*) Calculate $z_{LR}(\boldsymbol{\lambda}^{(k)})$ and the corresponding $\boldsymbol{x}(\boldsymbol{\lambda}^{(k)})$. If $\boldsymbol{x}(\boldsymbol{\lambda}^{(k)})$ is feasible, then **stop**.
Step 2.	(*Repairing heuristic*) Implement any heuristic algorithm to provide, starting from $\boldsymbol{x}(\boldsymbol{\lambda}^{(k)})$, a feasible solution $\boldsymbol{x}_{eur}^{(k)}$. Set $z_{eur}^{(k)} = \boldsymbol{c}^T \boldsymbol{x}_{eur}^{(k)}$ and possibly update z_{UB}.
Step 3.	(*Multiplier update*) Calculate $\boldsymbol{\lambda}^{(k+1)}$ by applying any algorithm for the Lagrangian dual. Set $k = k + 1$. Perform a termination test and possibly return to Step 1.

The above scheme is just an abstract description of how a Lagrangian heuristic works and several points need to be specified.

As for the initialization, a possible choice, whenever the LP continuous relaxation (17.20) is not too hard to solve, is to set $\boldsymbol{\lambda}^{(0)} = \boldsymbol{\lambda}_{LP}^*$, where $\boldsymbol{\lambda}_{LP}^*$ is the dual optimal vector associated to constraints $A\boldsymbol{x} = \boldsymbol{b}$.

At Step 2 the Lagrangian relaxation (17.12) is solved. As previously mentioned, it is expected to be substantially easier than the original problem (17.11), and in fact it is solvable, in many applications, in polynomial or pseudo-polynomial time. This is not always the case, as (17.12) may be still a hard combinatorial problem. In such cases it is possible to substitute an approximate calculation of z_{LR} to the exact one, which amounts to adopt a heuristic algorithm to tackle the Lagrangian relaxation (17.12) at Step 1.

Inexact calculation of z_{LR} has a strong impact on the way in which the Lagrangian dual (17.14) is tackled at Step 3. Here all machinery of convex optimization with inexact calculation of the objective function enters into play. For an extensive treatment of the subject see [16, 37] and Chap. 12 of this book. An example of the use of inexact calculation of z_{LR} in a Lagrangian heuristic framework is in [27].

The repairing heuristic at Step 2 is, of course, the problem dependent and it is very often of the greedy type.

The termination test depends on the type of algorithm used for solving the Lagrangian dual. Classic termination tests based on approximate satisfaction of the condition $\mathbf{0} \in \partial z_{LR}(\boldsymbol{\lambda}^{(k)})$ can be used whenever bundle type algorithms are adopted. In case subgradient or dual ascent algorithms are at work, stopping tests based on variation of z_{LR} can be adopted, together with a bound on the maximum number of iterations.

An important feature of Lagrangian heuristics is that they provide both a lower and an upper bound and, consequently, significant indications on the upper–lower bound gap are often at hand.

It is worth remarking, however, that, differently from the case where Lagrangian relaxation is used within an exact algorithm, here the main aim is to produce good quality feasible solutions rather than to obtain tight lower bounds. In fact it happens in several applications that poor lower bounds are accompanied by a fairly good upper bound. This is probably due to the fact that, as the algorithm proceeds, many feasible solutions are explored, through a search mechanism where multiplier update is finalized to get feasibility. From this point of view, adopting a not too fast NSO algorithm is not always a disadvantage!

Consequence of the above observations is that Lagrangian heuristics can be satisfactorily applied even to problems which exhibit the integrality property, that is also in case one knows in advance that Lagrangian dual is unable to produce anything better than the continuous LP relaxation lower bound.

17.4.1 An Example of Dual Ascent

The following example shows how to get, by tackling the Lagrangian dual, both a lower and an upper bound for a classic combinatorial problem. The Lagrangian dual is approached by means of a dual ascent algorithm, which works as a co-ordinate search method and modifies just one multiplier at a time.

Consider the standard *set covering problem* (SCP), which is known to be NP-hard. Suppose we are given a ground-set $\mathcal{I} = \{1, \ldots, m\}$, a family of n subsets $\mathcal{S}_j \subseteq \mathcal{I}$, $j \in \mathcal{J} = \{1, \ldots n\}$ and a real cost vector $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{c} > 0$. The problem is to select a minimum cost cover, that is an index set $\mathcal{J}^* \subseteq \mathcal{J}$ such that $\cup_{j \in \mathcal{J}^*} \mathcal{S}_j = \mathcal{I}$ with minimal associated cost $\sum_{j \in \mathcal{J}^*} \boldsymbol{c}_j$. By defining the $m \times n$ binary incidence matrix A of the subsets $\mathcal{S}_j$ and letting $\boldsymbol{e}$ be a vector of m ones, the SCP reads as follows

$$\begin{cases} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} \geq \boldsymbol{e}, \\ & \boldsymbol{x} \text{ binary}, \end{cases} \tag{17.29}$$

where $\boldsymbol{x}$ is a binary decision variable vector with $x_j = 1$ if j is taken into the cover and $x_j = 0$ otherwise, $j \in \mathcal{J}$.

By relaxing the covering constraints $A\boldsymbol{x} \geq \boldsymbol{e}$ by means of the multiplier vector $\boldsymbol{\lambda} \geq \boldsymbol{0}$, $\boldsymbol{\lambda} \in \mathbb{R}^m$, we obtain

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda}) \; = \; & \min \; \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{e} - A\boldsymbol{x}) \\ & \text{subject to} \; \; \boldsymbol{x} \text{ binary}, \end{aligned} \tag{17.30}$$

which can be in turn rewritten as

$$z_{LR}(\boldsymbol{\lambda}) = \boldsymbol{e}^T\boldsymbol{\lambda} + \min\ (\boldsymbol{c} - A^T\boldsymbol{\lambda})^T\boldsymbol{x} \quad \text{subject to } \boldsymbol{x} \text{ binary.} \tag{17.31}$$

An optimal solution $\boldsymbol{x}(\boldsymbol{\lambda})$ to the problem (17.31) can be obtained by simple inspection of the (reduced) cost vector $\boldsymbol{c}(\boldsymbol{\lambda}) = \boldsymbol{c} - A^T\boldsymbol{\lambda}$, by setting

$$x_j(\boldsymbol{\lambda}) = \begin{cases} 1, & \text{if } c_j(\boldsymbol{\lambda}) = c_j - \boldsymbol{a}_j{}^T\boldsymbol{\lambda} \le 0, \\ 0, & \text{otherwise,} \end{cases} \tag{17.32}$$

where $\boldsymbol{a}_j$ is the column j of matrix A. Observe that in this case the integrality property holds, thus $z_{LD} = z_{LP}$.

We introduce now a rule for updating the multiplier vector $\boldsymbol{\lambda}$, in case the corresponding $\boldsymbol{x}(\boldsymbol{\lambda})$ is infeasible, so that in the new multiplier setting:

- the number of satisfied constraints is increased;
- the function z_{LR} increases as well.

We proceed by updating just one component of $\boldsymbol{\lambda}$. Since we assume that $\boldsymbol{x}(\boldsymbol{\lambda})$ is infeasible, there exists at least one row index, say h, such that:

$$\sum_{j \in \mathcal{J}} a_{hj} x_j(\boldsymbol{\lambda}) = 0, \tag{17.33}$$

which implies, taking into account (17.32),

$$c_j(\boldsymbol{\lambda}) = c_j - \boldsymbol{a}_j{}^T\boldsymbol{\lambda} > 0, \quad \text{for all } j \in \mathcal{J}^{(h)} = \{j \mid a_{hj} = 1\}.$$

Now, defining a new multiplier setting in the form $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda} + \delta\boldsymbol{e}_h$ for some $\delta > 0$, where $\boldsymbol{e}_h$ is the h-th unit vector, the updated reduced cost is $\boldsymbol{c}(\boldsymbol{\lambda}^+) = \boldsymbol{c}(\boldsymbol{\lambda}) - \delta A^T\boldsymbol{e}_h$, that is

$$c_j(\boldsymbol{\lambda}^+) = \begin{cases} c_j(\boldsymbol{\lambda}) - \delta, & \text{if } j \in \mathcal{J}^{(h)}, \\ c_j(\boldsymbol{\lambda}), & \text{otherwise.} \end{cases} \tag{17.34}$$

In particular, by setting $\delta = \min_{j \in \mathcal{J}^{(h)}} c_j(\boldsymbol{\lambda}) = c_{j^*}(\boldsymbol{\lambda})$, it is $c_{j^*}(\boldsymbol{\lambda}^+) = 0$ and thus, from (17.32), $x_{j^*}(\boldsymbol{\lambda}^+) = 1$. Summing up it is

$$x_j(\boldsymbol{\lambda}^+) = \begin{cases} x_j(\boldsymbol{\lambda}), & \text{if } j \ne j^*, \\ 1, & \text{if } j = j^*. \end{cases} \tag{17.35}$$

Corresponding to the new solution $\boldsymbol{x}(\boldsymbol{\lambda}^+)$, the constraint h is no longer violated and it is $z_{LR}(\boldsymbol{\lambda}^+) = z_{LR}(\boldsymbol{\lambda}) + \delta$. Summing up, dual ascent has been achieved and at

least one of the constraints previously violated is satisfied, while the satisfied ones remain such. By iterating the multiplier update, we obtain, in at most m steps, both a feasible solution and a lower bound, produced by a sequence of ascent steps.

Algorithm 17.2: Dual ascent for set covering

Step 0. (*Initialization*) Set $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$, $\boldsymbol{c}(\boldsymbol{\lambda}^{(0)}) = \boldsymbol{c}$, $z_{LR}(\boldsymbol{\lambda}^{(0)}) = 0$, $\boldsymbol{x}(\boldsymbol{\lambda}^{(0)}) = \mathbf{0}$ and $k = 0$.

Step 1. (*Termination test*) If $A\boldsymbol{x}(\boldsymbol{\lambda}^{(k)}) \geq \boldsymbol{e}$, then calculate $z_{UB} = \boldsymbol{c}^T \boldsymbol{x}(\boldsymbol{\lambda}^{(k)})$ and **stop**. Otherwise, select h such that $\sum_{j=1}^{n} a_{hj} x_j(\boldsymbol{\lambda}) = 0$, calculate $0 < \delta = \min_{j \in \mathcal{J}^{(h)}} c_j(\boldsymbol{\lambda}^{(k)})$ and $j^* = \operatorname{argmin}_{j \in \mathcal{J}^{(h)}} \{c_j(\boldsymbol{\lambda}^{(k)})\}$, with $\mathcal{J}^{(h)} = \{j \mid a_{hj} = 1\}$.

Step 2. (*Multiplier and reduced cost update*) Put $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \delta \boldsymbol{e}_h$, $\boldsymbol{c}(\boldsymbol{\lambda}^{(k+1)}) = \boldsymbol{c}(\boldsymbol{\lambda}^{(k)}) - \delta A^T \boldsymbol{e}_h$,

$$x_j(\boldsymbol{\lambda}^{(k+1)}) = \begin{cases} x_j(\boldsymbol{\lambda}^{(k)}), & \text{if } j \neq j^*, \\ 1, & \text{if } j = j^*, \end{cases}$$

$z_{LR}(\boldsymbol{\lambda}^{(k+1)}) = z_{LR}(\boldsymbol{\lambda}^{(k)}) + \delta$. Set $k = k + 1$ and return to Step 1.

At stop in correspondence to any iteration index k, we obtain both a feasible solution $\boldsymbol{x}(\boldsymbol{\lambda}^{(k)})$, with associate objective function value z_{UB}, together with the lower bound $z_{LR}(\boldsymbol{\lambda}^{(k)})$.

Remark 17.3 For set covering problem the integrality property holds true, thus we cannot expect from the above procedure anything better than the LP lower bound. Moreover, since it is not at all guaranteed that the optimum of the Lagrangian is achieved when the algorithm stops, the lower bound provided might be definitely worse than the LP one. On the other hand the interplay between quest for feasibility and dual function improvement is a typical aspect of the applications we are going to describe in next section.

17.5 Applications

In this section we describe a number of applications of Lagrangian relaxation to integer programming problems coming from fairly diverse areas. In almost all cases a Lagrangian heuristic based on the abstract scheme sketched in Sect. 17.4 is designed. The Lagrangian dual is dealt with either via dual ascent or via subgradient algorithms. In particular the following problems will be treated:

- generalized assignment [22];
- spanning tree with minimum branch vertices [10];
- directional sensors location [3];

- cross docking scheduling [14];
- feature selection in support vector machines [28];
- multiple instance learning [4].

The presentation is necessarily synthetic and no numerical results are presented. The interested reader is referred to the original papers and to the references therein.

17.5.1 Generalized Assignment

A classic application of a dual ascent procedure based on updating one multiplier at a time is the method for solving the *generalized assignment problem* (GAP) described in [22]. GAP can be seen as the problem of assigning jobs to machines with limited amount of a resource (e.g. time or space), with the objective of maximizing the value of the assignment.

The sets $\mathcal{I}$ and $\mathcal{J}$ of machine and job indices, respectively, are given, together with the following data:

- a_{ij}, the resource required by job j when processed on machine i, $i \in \mathcal{I}$ and $j \in \mathcal{J}$;
- b_i, resource availability of machine i, $i \in \mathcal{I}$;
- c_{ij}, value of assigning job j to machine i, $i \in \mathcal{I}$ and $j \in \mathcal{J}$.

Defining, for $i \in \mathcal{I}$ and $j \in \mathcal{J}$, the decision variable $x_{ij} = 1$ if job j is assigned to machine i and $x_{ij} = 0$ otherwise, the GAP is then formulated:

$$\begin{cases} \text{maximize} & \displaystyle\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}x_{ij} \\ \text{subject to} & \displaystyle\sum_{i\in\mathcal{I}} x_{ij} = 1, \quad j \in \mathcal{J}, \\ & \displaystyle\sum_{j\in\mathcal{J}} a_{ij}x_{ij} \le b_i, \quad i \in \mathcal{I}, \\ & x_{ij} \text{ binary}, \quad i \in \mathcal{I}, \quad j \in \mathcal{J}. \end{cases} \tag{17.36}$$

A possible relaxation is obtained by acting on the semi-assignment constraints $\sum_{i\in\mathcal{I}} x_{ij} = 1$, $j \in \mathcal{J}$, thus obtaining, for each choice of the multipliers λ_j, $j \in \mathcal{J}$ (they are grouped into vector $\boldsymbol{\lambda}$), the following upper bound $z_{LR}(\boldsymbol{\lambda})$ (note that (17.36) is a maximization problem, hence the Lagrangian dual is a minimization problem):

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda}) = \sum_{j\in\mathcal{J}} \lambda_j + \max \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} (c_{ij} - \lambda_j)x_{ij} \\ \text{subject to} \quad & \sum_{j\in\mathcal{J}} a_{ij}x_{ij} \le b_i, \quad i \in \mathcal{I}, \\ & x_{ij} \text{ binary}, \quad i \in \mathcal{I}, \quad j \in \mathcal{J}. \end{aligned} \tag{17.37}$$

It is easy to verify that the problem (17.37) decomposes into $|\mathcal{I}|$ binary knapsack subproblems, that is

$$z_{LR}(\boldsymbol{\lambda}) = \sum_{j\in\mathcal{J}} \lambda_j + \sum_{i\in\mathcal{I}} z_{LR}^{(i)}(\boldsymbol{\lambda}),$$

where

$$\begin{aligned} z_{LR}^{(i)}(\boldsymbol{\lambda}) \;=\; \max \quad & \sum_{j\in\mathcal{J}} (c_{ij} - \lambda_j) x_{ij} \\ \text{subject to} \quad & \sum_{j\in\mathcal{J}} a_{ij} x_{ij} \le b_i, \\ & x_{ij} \text{ binary}, \quad j \in \mathcal{J}. \end{aligned} \tag{17.38}$$

If for any $\boldsymbol{\lambda}$ the solution $\boldsymbol{x}(\boldsymbol{\lambda})$ of the Lagrangian relaxation satisfies the relaxed constraints, then it is optimal for the GAP as well. In [22] a judicious selection of the initial values of the multipliers by setting $\lambda_j = \max_{i\in\mathcal{I}}^{(2)} c_{ij}$, where $\max^{(2)}$ indicates the second largest number in a set, makes $\boldsymbol{x}(\boldsymbol{\lambda})$ to satisfy the condition $\sum_{i\in\mathcal{I}} x_{ij}(\boldsymbol{\lambda}) \le 1$, $j \in \mathcal{J}$. Thus the only possible infeasibilities of such solution are of the type $\sum_{i\in\mathcal{I}} x_{ij}(\boldsymbol{\lambda}) = 0$, for one or more jobs.

The core of the algorithm is the multiplier vector update which is driven by such kind of infeasibility. In fact, consider any job h such that $\sum_{i\in\mathcal{I}} x_{ih}(\boldsymbol{\lambda}) = 0$. Any decrease in multiplier λ_h makes all reduced costs $(c_{ih} - \lambda_h)$ of such unassigned job increase. As consequence, job h becomes more competitive in view of possible assignment. The key point of the algorithm is the possibility of calculating exactly the minimum reduction Δ_h of multiplier λ_h which allows, under the new setting, assignment of the previously unassigned job to at least one machine.

This calculation can be performed as follows. It is first calculated Δ_{ih}, the minimum reduction in λ_h which allows assignment of job h to machine i, $i \in \mathcal{I}$. To this aim, the following auxiliary knapsack problem is solved:

$$\begin{aligned} z_{LR}^{(i,h)}(\boldsymbol{\lambda}) \;=\; \max \quad & \sum_{j\in\mathcal{J},\, j\neq h} (c_{ij} - \lambda_j) x_{ij} \\ \text{subject to} \quad & \sum_{j\in\mathcal{J},\, j\neq h} a_{ij} x_{ij} \le b_i - a_{ih}, \\ & x_{ij} \text{ binary}, \quad j \in \mathcal{J},\ j \neq h, \end{aligned} \tag{17.39}$$

and then it is

$$\Delta_{ih} = z_{LR}^{(i)}(\boldsymbol{\lambda}) - \left(c_{ih} - \lambda_h + z_{LR}^{(i,h)}(\boldsymbol{\lambda})\right) \ge 0.$$

Finally we have

$$\Delta_h = \min_{i \in \mathcal{I}} \Delta_{ih},$$

and it is easy to verify that, in case $\Delta_h > 0$ and letting $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda} - \Delta_h \boldsymbol{e}_h$, function z_{LR} reduces of exactly Δ_h.

We skip here some details of the algorithm, e.g., how to enforce condition $\sum_{i \in \mathcal{I}} x_{ij}(\boldsymbol{\lambda}) \leq 1$, $j \in J$ to hold throughout the execution. We wish to emphasize, instead, that at each iteration just one multiplier is updated (thus the algorithm is a co-ordinate descent one). Moreover, unlike the algorithm for set covering discussed in previous section, it is not ensured that the number of satisfied constraints increases monotonically. However, termination at a feasible (and hence optimal) solution is guaranteed.

An approach inspired by this method was introduced in [11] to deal with a classic location–allocation problem known as terminal location.

17.5.2 Spanning Tree with Minimum Branch Vertices

In previous subsection it has been presented a dual ascent procedure, based on modification of one multiplier at a time, with *exact* calculation of the step size along the corresponding coordinate axis. Here we discuss a dual ascent algorithm for the *spanning tree with minimum branch vertices* (ST-MBV) problem which still works modifying just one multiplier at a time, but it is equipped with a *line search* along the coordinate axis.

Another application of Lagrangian relaxation to a variant of the Steiner tree problem is in [17].

The ST-MBV problem arises in the design of optical networks, where it is necessary to guarantee connection to all nodes of a given network. Thus a spanning tree is to be found. Since at least one switch must be installed at each node of the tree whose degree is greater than two (branch vertices) the problem is to find a ST-MBV.

This problem admits several formulations. We focus on the *integer programming* (IP) one described in [10], where a Lagrangian heuristic is presented in details.

We consider an undirected network $G = (V, E)$, where V denotes the set of n vertices and E the set of m edges. The decision variables are the following:

- x_e, $e \in E$, binary; $x_e = 1$ if edge e is selected and $x_e = 0$ otherwise;
- y_v, $v \in V$, binary; $y_v = 1$ if vertex v is of the branch type (that is its degree, as vertex of the tree, is greater than two), and $y_v = 0$ otherwise.

Then, the IP formulation of the MBV is the following:

$$\begin{cases} \text{minimize} & \displaystyle\sum_{v\in V} y_v \\ \text{subject to} & \displaystyle\sum_{e\in E} x_e = n-1, \\ & \displaystyle\sum_{e\in E(S)} x_e \le |S|-1, \ \ S\subseteq V, \\ & \displaystyle\sum_{e\in A(v)} x_e - 2 \le \delta_v y_v, \ \ v \in V, \\ & y_v \text{ binary}, \quad v\in V \text{ and } x_e \text{ binary}, \quad e\in E, \end{cases} \tag{17.40}$$

where for any given subset of vertices S we denote by $E(S)$ the set of edges having both the endpoints in S. Moreover, we denote by $A(v)$ the set of incident edges to vertex v and by δ_v its size, i.e. $\delta_v = |A(v)|$. The objective function to be minimized is the total number of branch vertices. Constraints

$$\sum_{e\in E} x_e = n-1 \quad \text{and} \quad \sum_{e\in E(S)} x_e \le |S|-1, \ \ S\subseteq V,$$

ensure that a spanning tree is actually detected, while the complicating constraints are $\sum_{e\in A(v)} x_e - 2 \le \delta_v y_v$, $v \in V$. They guarantee that variable y_v is set to 1 whenever v has more than two incident edges in the selected tree.

By introducing the multipliers $\lambda_v \ge 0$, $v \in V$ (grouped, as usual, in vector $\boldsymbol{\lambda}$), we obtain the following Lagrangian relaxation:

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda}) \ = \ \min \quad & \sum_{v\in V} y_v + \sum_{v\in V} \lambda_v \Big(\sum_{e\in A(v)} x_e - 2 - \delta_v y_v \Big) \\ \text{subject to} \quad & \sum_{e\in E} x_e = n-1, \\ & \sum_{e\in E(S)} x_e \le |S|-1, \ \ S\subseteq V, \\ & y_v \text{ binary}, \quad v\in V \text{ and } x_e \text{ binary}, \quad e\in E. \end{aligned} \tag{17.41}$$

By simple manipulations, $z_{LR}(\boldsymbol{\lambda})$ may be rewritten as

$$z_{LR}(\boldsymbol{\lambda}) = -2\sum_{v\in V} \lambda_v + z_{LR}^{(1)}(\boldsymbol{\lambda}) + z_{LR}^{(2)}(\boldsymbol{\lambda}), \tag{17.42}$$

where $z_{LR}^{(1)}(\boldsymbol{\lambda})$ and $z_{LR}^{(2)}(\boldsymbol{\lambda})$ are defined as follows:

$$\begin{aligned} z_{LR}^{(1)}(\boldsymbol{\lambda}) \ = \ \min \quad & \sum_{v\in V} y_v(1-\delta_v\lambda_v) \\ \text{subject to} \quad & y_v \text{ binary}, \quad v\in V, \end{aligned} \tag{17.43}$$

and

$$
\begin{aligned}
z_{LR}^{(2)}(\boldsymbol{\lambda}) \;=\; \min \quad & \sum_{v \in V} \sum_{e \in A(v)} \lambda_v x_e \\
\text{subject to} \quad & \sum_{e \in E} x_e = n - 1, \\
& \sum_{e \in E(S)} x_e \leq |S| - 1, \;\; S \subseteq V, \\
& x_e \text{ binary}, \quad e \in E.
\end{aligned}
\tag{17.44}
$$

Note that $z_{LR}(\boldsymbol{\lambda})$ is rather easy to calculate. In fact the problem (17.43) is solved by inspection of the cost coefficients, by setting

$$
\begin{cases}
y_v = 1, & \text{if } \; 1 - \delta_v \lambda_v \leq 0, \\
y_v = 0, & \text{otherwise},
\end{cases}
\tag{17.45}
$$

while $z_{LR}^{(1)}(\boldsymbol{\lambda})$ is the optimal value of the minimum spanning tree problem where the weight of edge $e = (u, v)$ is $\lambda_u + \lambda_v$.

As for the Lagrangian dual, it is possible to prove [10] that the optimal multiplier vector $\boldsymbol{\lambda}^*$ satisfies the condition

$$
\lambda_v^* \leq \frac{1}{\delta_v}, \;\; v \in V. \tag{17.46}
$$

On the basis of such property it is possible to devise an ascent strategy which modifies one multiplier at a time. Suppose that $(\boldsymbol{x}(\boldsymbol{\lambda}), \boldsymbol{y}(\boldsymbol{\lambda}))$ is an optimal solution to the Lagrangian relaxation for a given $\boldsymbol{\lambda}$, then such strategy is at hand whenever one of the two following cases occurs for some $v \in V$:

1. $\sum_{e \in A(v)} x_e(\boldsymbol{\lambda}) > 2$ and $y_v(\boldsymbol{\lambda}) = 0$;
2. $\sum_{e \in A(v)} x_e(\boldsymbol{\lambda}) \leq 2$ and $y_v(\boldsymbol{\lambda}) = 1$.

Consider first the case 1 and observe (see the objective function of the problem (17.41)) that the v component $\sum_{e \in A(v)} x_e(\boldsymbol{\lambda}) - 2 - \delta_v y_v(\boldsymbol{\lambda}))$ of a subgradient of z_{LR} is strictly positive, thus one can expect an increase in the objective if λ_v is increased. Observe, in addition, that property (17.46) suggests to adopt an Armijo-type line search along the co-ordinate direction $\boldsymbol{e}_v$ with $\frac{1}{\delta_v}$ as initial step size. On the other hand, a consequence of nonsmoothness of z_{LR} is that direction $\boldsymbol{e}_v$ is not necessarily an ascent one, thus the backtracking line search must accommodate for possible failure (the so-called *null step*, to use the terminology of bundle methods).

As for the case 2, the v subgradient component $\sum_{e \in A(v)} x_e(\boldsymbol{\lambda}) - 2 - \delta_v y_v(\boldsymbol{\lambda}))$ is negative and, in addition (see (17.45)) it is $\lambda_v > 0$. Thus it is worth to consider

a possible reduction on λ_v, that is a move along the co-ordinate direction $-\boldsymbol{e}_v$, adopting, also in this case, an Armijo-type line search equipped with possible null step declaration.

The use of the co-ordinate search approach has both advantages and drawbacks. As pointed out in [10], co-ordinate search is definitely faster than standard subgradient but the lower bound is slightly worse due to possible premature stop (see Sect. 17.3).

We remark that for this application a feasible solution of the problem (17.40) is available with no additional computational cost at each iteration of the dual ascent procedure. After all, the Lagrangian relaxation provides a spanning tree, thus, to implement a Lagrangian heuristic, it is only needed to calculate the corresponding cost in terms of the objective function of (17.40).

17.5.3 Directional Sensors Location

In *wireless sensor networks* (WSN) [54] a sensor is a device capable to receive (and possibly store and forward) information coming from a sufficiently close area. In general, such an area is a circle of given radius, whose centre is the sensor location itself. Points inside the area are deemed *covered* by the sensor. In case the area, instead of being a circle, is an adjustable circular sector, the sensor is defined *directional* [12].

The *directional sensors continuous coverage problem* (DSCCP) is about covering several *targets*, distributed in a plane, by a set of directional sensors whose locations are known. Each sensor is adjustable, being characterized by a discrete set of possible radii and aperture angles. The sensor orientation is a decision variable too. The model accommodates for possible sensor switch off. Since different power consumption is associated to the sensor adjustments, the objective is to minimize the total power cost of coverage.

We report here the formulation given in [3] as a *mixed integer nonlinear program* (MINLP). Note that in most of the literature only a discrete number of sensor orientations are considered (see [50, Lemma 1 and Corollary 1]). The motivation for defining, instead, the orientation as a continuous variable is in the choice of Lagrangian relaxation as the attack strategy. It will be clear in the sequel, that solving the relaxed problem is easy once the sensor orientation is assumed to be a continuous variable.

Suppose that a given set $\mathcal{I}$ of sensors is located in a certain area and $\boldsymbol{s}_i \in \mathbb{R}^2$ is the known position of sensor i, $i \in \mathcal{I}$. The location $\boldsymbol{t}_j \in \mathbb{R}^2$, $j \in \mathcal{J}$, of the targets is also known, together with the sensor-target distance parameters $\boldsymbol{d}_{ij} = \boldsymbol{t}_j - \boldsymbol{s}_i$ and $c_{ij} = \|\boldsymbol{t}_j - \boldsymbol{s}_i\|$, $i \in \mathcal{I}$ and $j \in \mathcal{J}$. A set of $K + 1$ different power levels k, $k = 0, \ldots, K$ can be considered for sensors, each of them corresponding to a couple of values of the radius r_k and of the half-aperture angle α_k of the circular sector. In particular, the level $k = 0$ is associated to an inactive sensor ($r_0 = 0$ and $\alpha_0 = 0$). We also introduce the parameter $q_k = \cos\alpha_k$, $q_k \in [-1, 1]$.

The area covered by the sensor i, activated at level k, is then the revolution cone defined as

$$\left\{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^2,\ \|\boldsymbol{x}-\boldsymbol{s}_i\| \le r_k,\ \ q_k\|\boldsymbol{x}-\boldsymbol{s}_i\| \le (\boldsymbol{x}-\boldsymbol{s}_i)^T \boldsymbol{w}_i\right\}, \tag{17.47}$$

where $\boldsymbol{w}_i \in \mathbb{R}^2$, $\|\boldsymbol{w}_i\| = 1$, is the orientation direction assigned to sensor i, $i \in \mathcal{I}$ ($\|.\|$ is assumed to be the Euclidean norm).

As a consequence, target j is covered by the sensor i, activated at level k, with orientation direction $\boldsymbol{w}_i$ if and only if the following two conditions hold:

$$r_k \ge c_{ij} \qquad \text{and} \qquad q_k c_{ij} \le \boldsymbol{d}_{ij}{}^T \boldsymbol{w}_i. \tag{17.48}$$

The decision variables are:

- $\boldsymbol{w}_i \in \mathbb{R}^2$, the orientation direction assigned to sensor i, $i \in \mathcal{I}$;
- x_{ik}, $i \in \mathcal{I}$, $k = 0, \ldots, K$, binary: $x_{ik} = 1$ if sensor i is activated at power level k and $x_{ik} = 0$ otherwise;
- σ_{ij}, $i \in \mathcal{I}$, $j \in \mathcal{J}$, binary: $\sigma_{ij} = 0$ implies that both conditions (17.48) are satisfied;
- u_j, $j \in \mathcal{J}$, binary: $u_j = 0$ implies that the target j is covered by at least one sensor.

The model considers two types of costs:

- p_k, the activation cost for turning on any sensor at power level k, $k = 0, \ldots, K$ (with $p_0 = 0$);
- H, the penalty cost associated to an uncovered target.

Finally the DSCCP can be stated as follows:

$$\begin{cases} \text{minimize} & \displaystyle\sum_{i\in\mathcal{I}}\sum_{k=0}^{K} p_k x_{ik} + H \sum_{j\in\mathcal{J}} u_j \\ \text{subject to} & \displaystyle\sum_{k=0}^{K} x_{ik} = 1, \quad i \in \mathcal{I}, \\ & \displaystyle\sum_{k=0}^{K} x_{ik} r_k \ge c_{ij} - M\sigma_{ij},\ i \in \mathcal{I},\ \ j \in \mathcal{J}, \\ & \displaystyle c_{ij} \sum_{k=0}^{K} q_k x_{ik} - \boldsymbol{d}_{ij}{}^T \boldsymbol{w}_i \le M\sigma_{ij},\ i \in \mathcal{I},\ \ j \in \mathcal{J}, \\ & \displaystyle u_j \ge \sum_{i\in\mathcal{I}} \sigma_{ij} - (|I| - 1), \quad j \in \mathcal{J}, \\ & \|\boldsymbol{w}_i\| = 1, \quad i \in \mathcal{I}, \\ & \boldsymbol{x}, \boldsymbol{\sigma}, \boldsymbol{u} \ \text{ binary and } \boldsymbol{w}_i \in \mathbb{R}^2,\ i \in \mathcal{I}. \end{cases} \tag{17.49}$$

The model contains the "big M" positive input parameter, which is common in formulation of location problems. Variables x_{ik}, σ_{ij} and u_j are grouped into the vectors $\boldsymbol{x}$, $\boldsymbol{\sigma}$ and $\boldsymbol{u}$, respectively.

The objective function is the sum of activation cost and penalty for possibly uncovered targets. The first set of constraints are classic semi-assignment, ensuring that exactly one power level (possibly the 0-level) is assigned to each sensor. Constraints

$$\begin{cases} \displaystyle\sum_{k=0}^{K} x_{ik} r_k \geq c_{ij} - M\sigma_{ij}, \ i \in \mathcal{I}, \quad j \in \mathcal{J}, \\ \displaystyle c_{ij} \sum_{k=0}^{K} q_k x_{ik} - \boldsymbol{d}_{ij}{}^T \boldsymbol{w}_i \leq M\sigma_{ij}, \ i \in \mathcal{I}, \ j \in \mathcal{J}, \end{cases} \tag{17.50}$$

are related to target coverage. They are trivially satisfied if $\sigma_{ij} = 1$ but, whenever they are satisfied with $\sigma_{ij} = 0$, then target j is covered by sensor i (see (17.48)). Constraints

$$u_j \geq \sum_{i \in \mathcal{I}} \sigma_{ij} - (|\mathcal{I}| - 1), \quad j \in \mathcal{J} \tag{17.51}$$

impose that, for any target j, $u_j = 1$ in case $\sigma_{ij} = 1$ for all $i \in \mathcal{I}$, that is if the target j remains uncovered. At the optimum $u_j = 1$ if and only if $\sum_{i \in \mathcal{I}} \sigma_{ij} = |\mathcal{I}|$. Finally the (nonconvex) constraints $\|\boldsymbol{w}_i\| = 1$, $i \in \mathcal{I}$, aim at normalizing the orientation direction assigned to each sensor.

By introducing the nonnegative multiplier vectors $\boldsymbol{\lambda}$, $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$, the following Lagrangian relaxation is defined

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda}, \boldsymbol{\theta}, \boldsymbol{\gamma}) \ = \ \min & \sum_{i \in \mathcal{I}} \sum_{k=0}^{K} p_k x_{ik} + H \sum_{j=1}^{n} u_j \\ & + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \lambda_{ij} (c_{ij} - M\sigma_{ij} - \sum_{k=0}^{K} x_{ik} r_k) \\ & + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \theta_{ij} (c_{ij} \sum_{k=0}^{K} x_{ik} q_k - \boldsymbol{d}_{ij}{}^T \boldsymbol{w}_i - M\sigma_{ij}) \\ & + \sum_{j \in \mathcal{J}} \gamma_j (\sum_{i \in \mathcal{I}} \sigma_{ij} - (|\mathcal{I}| - 1) - u_j) \\ \text{subject to} & \sum_{k=0}^{K} x_{ik} = 1, \quad i \in \mathcal{I}, \\ & \|\boldsymbol{w}_i\| = 1, \quad i \in \mathcal{I}, \\ & \boldsymbol{x}, \boldsymbol{\sigma}, \boldsymbol{u} \ \text{ binary}, \ \boldsymbol{w}_i \in \mathbb{R}^2, \ i \in \mathcal{I}, \end{aligned} \tag{17.52}$$

which, rearranging the objective function, may be rewritten as

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\gamma}) = & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\lambda_{ij}c_{ij} + (1-|\mathcal{I}|)\sum_{j\in\mathcal{J}}\gamma_j \\ & + \min \sum_{i\in\mathcal{I}}\sum_{k=0}^{K}[p_k + \sum_{j\in\mathcal{J}}(\theta_{ij}c_{ij}q_k - \lambda_{ij}r_k)]x_{ik} \\ & + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}[\gamma_j - M(\lambda_{ij}+\theta_{ij})]\sigma_{ij} \\ & - \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\theta_{ij}\boldsymbol{d}_{ij}{}^T\boldsymbol{w}_i + \sum_{j\in\mathcal{J}}(H-\gamma_j)u_j \\ & \text{subject to } \sum_{k=0}^{K}x_{ik} = 1, \quad i\in\mathcal{I}, \\ & \|\boldsymbol{w}_i\| = 1, \quad i\in\mathcal{I}, \\ & \boldsymbol{x},\boldsymbol{\sigma},\boldsymbol{u} \text{ binary}, \ \boldsymbol{w}_i\in\mathbb{R}^2, \ i\in\mathcal{I}, \end{aligned} \tag{17.53}$$

The above formulation leads to the following decomposition:

$$z_{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\gamma}) = C(\boldsymbol{\lambda},\boldsymbol{\gamma}) + z_{LR}^{(1)}(\boldsymbol{\lambda},\boldsymbol{\theta}) + z_{LR}^{(2)}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\gamma}) + z_{LR}^{(3)}(\boldsymbol{\theta}) + z_{LR}^{(4)}(\boldsymbol{\gamma}), \tag{17.54}$$

where

$$C(\boldsymbol{\lambda},\boldsymbol{\gamma}) = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\lambda_{ij}c_{ij} + (1-|\mathcal{I}|)\sum_{j\in J}\gamma_j, \tag{17.55}$$

$$\begin{aligned} z_{LR}^{(1)}(\boldsymbol{\lambda},\boldsymbol{\theta}) = & \min \sum_{i\in\mathcal{I}}\sum_{k=0}^{K}[p_k + \sum_{j\in\mathcal{J}}(\theta_{ij}c_{ij}q_k - \lambda_{ij}r_k)]x_{ik} \\ & \text{subject to } \sum_{k=0}^{K}x_{ik} = 1, \quad i\in\mathcal{I}, \\ & \boldsymbol{x} \text{ binary}, \end{aligned} \tag{17.56}$$

$$\begin{aligned} z_{LR}^{(2)}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\gamma}) = & \min \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}[\gamma_j - M(\lambda_{ij}+\theta_{ij})]\sigma_{ij} \\ & \text{subject to } \sum_{k=0}^{K}x_{ik} = 1, \quad i\in\mathcal{I}, \\ & \boldsymbol{\sigma} \text{ binary}, \end{aligned} \tag{17.57}$$

$$z_{LR}^{(3)}(\boldsymbol{\theta}) = - \max \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \theta_{ij} \boldsymbol{d}_{ij}{}^T \boldsymbol{w}_i$$
$$\text{subject to } \|\boldsymbol{w}_i\| = 1, \quad i \in \mathcal{I}, \tag{17.58}$$
$$\boldsymbol{w}_i \in \mathbb{R}^2, \ i \in \mathcal{I},$$

and

$$z_{LR}^{(4)}(\boldsymbol{\gamma}) = \min \sum_{j\in\mathcal{J}} (H - \gamma_j) u_j$$
$$\text{subject to } \boldsymbol{u} \text{ binary.} \tag{17.59}$$

Calculation of the constant $C(\boldsymbol{\lambda}, \boldsymbol{\gamma})$ is immediate. The solution of problems (17.56), (17.57), and (17.59) can be obtained by simple inspection of the corresponding cost coefficients, while the problem (17.58) has optimal solution $\boldsymbol{w}_i(\boldsymbol{\theta})$, $i \in \mathcal{I}$, in the following closed form:

$$\boldsymbol{w}_i(\boldsymbol{\theta}) = \begin{cases} \dfrac{\sum_{j\in\mathcal{J}} \theta_{ij} \boldsymbol{d}_{ij}}{\| \sum_{j\in\mathcal{J}} \theta_{ij} \boldsymbol{d}_{ij} \|}, & \text{if } \sum\limits_{j\in\mathcal{J}} \theta_{ij} \boldsymbol{d}_{ij} \neq \boldsymbol{0}, \\ \text{any vector } \boldsymbol{w} \in \mathbb{R}^2, \ \|\boldsymbol{w}\| = 1, & \text{otherwise.} \end{cases} \tag{17.60}$$

In [3] the Lagrangian dual is tackled by applying a coordinate search method which modifies one multiplier at a time, in view of possible increase of function $z_{LR}(\boldsymbol{\lambda}, \boldsymbol{\theta}, \boldsymbol{\gamma})$. The choice of such multiplier is driven, as usual, by the type of infeasibility occurring at the current solution of the Lagrangian relaxation. Thus different rules are designed, according to the choice of the component either of $\boldsymbol{\lambda}$ or $\boldsymbol{\theta}$ or $\boldsymbol{\gamma}$ to be updated. For the details of such rules we refer the reader to [3].

The dual ascent procedure is particularly suitable for embedding a Lagrangian heuristic. In fact the solution of the Lagrangian relaxation can be made feasible at a quite low computational cost, hence providing an upper bound.

More specifically, whenever the solution to (17.52) is not feasible for (17.49), the set $\bar{\mathcal{J}}$ of possibly uncovered targets is considered. For each of them at least one of the following conditions holds:

$$\sum_{k=0}^{K} x_{ik} r_k < c_{ij} \quad \text{for all } i \in \mathcal{I}, \tag{17.61}$$

$$c_{ij} \sum_{k=0}^{K} q_k x_{ik} > \boldsymbol{d}_{ij}{}^T \boldsymbol{w}_i \quad \text{for all } i \in \mathcal{I} \tag{17.62}$$

at the optimum of the Lagrangian relaxation. Thus, for each target $j \in \bar{\mathcal{J}}$, possible coverage is sought by acting on the power levels (variables $\boldsymbol{x}$) assigned to those sensors which are eligible for covering it. In such search, sensor orientations (variables $\boldsymbol{w}_i$) returned by the Lagrangian relaxation are not modified.

Summing up, at each iteration of the dual ascent procedure both a lower and an upper bounds are calculated. In [3] it is highlighted that while the lower bound provided by the algorithm is extremely poor, the best feasible solution found is, in general, of good quality also for relatively large scale problems. It is worth noting that the algorithm is equipped with possible restart along a subgradient direction in case the Armijo line search fails along all coordinate directions.

17.5.4 Cross Docking Scheduling

A *cross docking distribution centre* (CD centre, in the following) is a modern logistic node where goods are unloaded from inbound trucks, consolidated with respect to the customer orders and then immediately loaded on outbound trucks, cancelling in practice the traditional and costly storing and retrieval phases. Management of a CD centre is a serious challenge since the processes of arrival and departure of goods are strongly coupled and sophisticated synchronization schemes are to be fulfilled.

Intensive research efforts have been made to devise effective models and algorithms for optimizing the combined scheduling of the inbound and outbound trucks; more recently Lagrangian relaxation has been applied in this area (see [14] and the references therein). The problem addressed is about finding the optimal inbound and outbound sequences at a CD centre characterized by only two gates (or doors), one for the inbound and the other for the outbound trucks. The objective is to minimize the total completion time (the *makespan* in scheduling theory parlance).

The rules of the game are the following:

- only one truck at a time can be handled at a door and no interruption (*preemption*) is allowed;
- the loading of an outbound truck cannot start until all goods it is expected to deliver have been unloaded;
- all trucks require the same processing time (one *slot*) and are ready at the time 0.

As an input data we consider a set of n inbound trucks ($\mathcal{I} = \{1, \ldots, n\}$) to be unloaded, together with a set of m outbound trucks ($\mathcal{J} = \{1, \ldots, m\}$) to be loaded. The following sets are also given:

- $\mathcal{J}_i$, the set of outbound trucks receiving goods from the inbound truck i, $i \in \mathcal{I}$;
- $\mathcal{I}_j$, the set of inbound trucks providing goods to the outbound truck j, $j \in \mathcal{J}$.

The planning horizon is discretized into time-slots, each of them being capable to accommodate for processing of one truck. Let $\mathcal{K} = \{1, \ldots, n\}$ and $\mathcal{L} = \{1, \ldots, H\}$, $H \geq m+n$, be the time horizon for the inbound and outbound services,

respectively (note that, under the assumptions made, $n + m$ is an upper bound on the makespan).

Introducing the following binary decision variables:

- $x_{ik} = 1$, if the inbound truck i is assigned to the time-slot k and $x_{ik} = 0$ otherwise, $i \in \mathcal{I}, k \in \mathcal{K}$;
- $y_{jh} = 1$, if the outbound truck j is assigned to the time-slot h, $y_{jh} = 0$ otherwise, $j \in \mathcal{J}, h \in \mathcal{L}$;

and the integer variable C_{Max}, the makespan, then the *one door cross docking* (ODCD) problem is modelled as follows:

$$
\begin{aligned}
Z \;=\; \min \quad & C_{Max} \\
\text{subject to } & \sum_{k\in\mathcal{K}} x_{ik} = 1, \quad i \in \mathcal{I}, \\
& \sum_{i\in\mathcal{I}} x_{ik} = 1, \quad k \in \mathcal{K}, \\
& \sum_{h\in\mathcal{L}} y_{jh} = 1, \quad j \in \mathcal{J}, \\
& \sum_{j\in\mathcal{J}} y_{jh} \le 1, \quad h \in \mathcal{L}, \\
& C_{Max} \ge \sum_{h\in\mathcal{L}} h y_{jh}, \quad j \in \mathcal{J}, \\
& \sum_{h\in\mathcal{L}} h y_{jh} \ge \sum_{k\in\mathcal{K}} k x_{ik} + 1, \quad j \in \mathcal{J},\ i \in \mathcal{I}_j, \\
& x_{ik} \text{ binary}, \ i \in \mathcal{I},\ k \in \mathcal{K}, \\
& y_{jh} \text{ binary}, \ j \in \mathcal{J},\ h \in \mathcal{L}.
\end{aligned}
\tag{17.63}
$$

The first four constraint sets regulate the time slot-truck assignment at both the inbound and outbound door. The constraints $C_{Max} \ge \sum_{h\in\mathcal{L}} h y_{jh}$, $j \in \mathcal{J}$ define the makespan C_{Max} as the maximum truck completion time, of course on the outbound side. The constraints

$$
\sum_{h\in\mathcal{L}} h y_{jh} \ge \sum_{k\in\mathcal{K}} k x_{ik} + 1, \quad j \in \mathcal{J},\ i \in \mathcal{I}_j
$$

ensure that loading of each outbound truck cannot start until the unloading of all corresponding inbound trucks has been completed. They are the complicating ones and lead to the following Lagrangian relaxation obtained via the multiplier

vector $\boldsymbol{\lambda} \geq 0$.

$$
\begin{aligned}
z_{LR}(\boldsymbol{\lambda}) \;=\; \min \quad & C_{Max} + \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}_j} \lambda_{ij} \Big(\sum_{k\in\mathcal{K}} k x_{ik} - \sum_{h\in\mathcal{L}} h y_{jh} + 1 \Big) \\
\text{subject to } & \sum_{k\in\mathcal{K}} x_{ik} = 1, \quad i \in \mathcal{I}, \\
& \sum_{i\in\mathcal{I}} x_{ik} = 1, \quad k \in \mathcal{K}, \\
& \sum_{h\in\mathcal{L}} y_{jh} = 1, \quad j \in \mathcal{J}, \\
& \sum_{j\in\mathcal{J}} y_{jh} \leq 1, \quad h \in \mathcal{L}, \\
& C_{Max} \geq \sum_{h\in\mathcal{L}} h y_{jh}, \quad j \in \mathcal{J}, \\
& x_{ik} \text{ binary}, \; i \in \mathcal{I}, \; k \in \mathcal{K}, \\
& y_{jh} \text{ binary}, \; j \in \mathcal{J}, \; h \in \mathcal{L}.
\end{aligned}
\tag{17.64}
$$

The relaxation decomposes into two independent matching problems, related, respectively, to the inbound and outbound door. In fact, by simple manipulations based on the observation $i \in \mathcal{I}_j \Leftrightarrow j \in \mathcal{J}_i$, and setting

$$
s = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}_j} \lambda_{ij} = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}_i} \lambda_{ij}, \tag{17.65}
$$

and

$$
\rho_i = \sum_{j\in\mathcal{J}_i} \lambda_{ij}, \; i \in \mathcal{I}, \qquad \sigma_j = \sum_{i\in\mathcal{I}_j} \lambda_{ij}, \; j \in \mathcal{J}, \tag{17.66}
$$

we rewrite z_{LR} as a function of vectors $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ (grouping the ρ_is and the σ_js, respectively) as follows

$$
z_{LR}(\boldsymbol{\rho}, \boldsymbol{\sigma}) = s + \min \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}} k\rho_i x_{ik} + C_{Max} - \sum_{j\in\mathcal{J}} \sum_{h\in\mathcal{L}} h\sigma_j y_{jh}. \tag{17.67}
$$

Thus we define

$$
z_{LR}(\boldsymbol{\rho}, \boldsymbol{\sigma}) = s + z_{LR}^{(1)}(\boldsymbol{\rho}) + z_{LR}^{(2)}(\boldsymbol{\sigma}),
$$

with

$$
\begin{aligned}
z_{LR}^{(1)}(\boldsymbol{\rho}) \;=\; \min \quad & \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}} k\rho_i x_{ik} \\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} x_{ik} = 1, \quad i \in \mathcal{I}, \\
& \sum_{i\in\mathcal{I}} x_{ik} = 1, \quad k \in \mathcal{K} \\
& x_{ik} \text{ binary}, \; i \in \mathcal{I}, \; k \in \mathcal{K},
\end{aligned}
\tag{17.68}
$$

and

$$
\begin{aligned}
z_{LR}^{(2)}(\boldsymbol{\sigma}) \;=\; \min \quad & C_{Max} - \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{L}} h\sigma_j y_{jh} \\
\text{subject to} \quad & \sum_{j\in\mathcal{J}} y_{jh} \le 1, \quad h \in \mathcal{L}, \\
& C_{Max} \ge \sum_{h\in\mathcal{L}} h y_{jh}, \quad j \in \mathcal{J}, \\
& y_{jh} \text{ binary}, \; j \in \mathcal{J}, \; h \in \mathcal{L}.
\end{aligned}
\tag{17.69}
$$

Problems (17.68) and (17.69) [14] are two simple single machine scheduling problems which can be solved by appropriate sorting of the vectors $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$.

The following holds for the Lagrangian dual.

Theorem 17.1 *There exists an optimal solution to the Lagrangian dual problem such that*

$$
s = \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij} = \sum_{i\in I} \rho_i = \sum_{j\in J}\sum_{i\in I_j} \lambda_{ij} = \sum_{j\in J} \sigma_j \le 1. \tag{17.70}
$$

The above property is helpful in designing algorithms to solve the Lagrangian dual, mainly as it enables appropriate sizing of the step size in implementing a line search.

It is worth noting that the ODCD does not enjoy the integrality property (see, in particular, the problem (17.69)), thus the lower bound obtained from the Lagrangian dual is more accurate than that provided by the LP relaxation.

In addition, any solution of a Lagrangian relaxation which is not feasible for the ODCD can be *repaired*, at low computational cost, by implementing some heuristic method based on simple forward-shifting of those outbounds trucks j for which the

relaxed constraints

$$\sum_{h\in\mathcal{L}} hy_{jh} \geq \sum_{k\in\mathcal{K}} kx_{ik} + 1$$

have been violated for at least one $i \in \mathcal{I}_j$ at the optimum of (17.64).

In [14] the Lagrangian dual has been tackled by resorting to a standard (projected) subgradient method, which has been equipped, at each iteration, with a repairing procedure, thus allowing to possibly update both the lower and upper bounds.

The Lagrangian heuristic has provided satisfactory results on test problem characterized by a number of inbound trucks n up to 20 and a number of outbound ones m in the range [10, 40].

17.5.5 Feature Selection in Support Vector Machines

Feature selection (FS) is a relevant issue in machine learning and, in particular, in pattern classification. In fact classification is a kind of diagnostic process which consists in attaching a *label* (that is certifying exactly one class membership) to an individual (a *sample* or a *pattern*), on the basis of a given number of known parameters (the *features*). Most of the research work has been focussed on binary classification, where the classes are only two. A binary *classifier* then is a tool which is able to attach the appropriate label to a sample whose class membership is unknown.

The classification models we are dealing with are of the *supervised* type, since the classifier is constructed on the basis of the information provided by a certain number of samples (the *training set*) whose class membership is known.

A fundamental paradigm to construct such classifier is the *support vector machines* (SVM) [15, 45] which consists of separating the samples belonging to the training set by means of a hyperplane, either in the feature space or in a higher dimension one, upon an appropriate kernel transformation. Once the hyperplane has been calculated, it is used to classify newly incoming patterns whose class membership is unknown.

In the standard SVM approach, the training set is formed by two given point-sets $A = \{\boldsymbol{a}_i, i \in \mathcal{I}\}$ and $B = \{\boldsymbol{b}_j, j \in \mathcal{J}\}$ in $\mathbb{R}^n$ (the feature space). The problem is about finding a hyperplane defined by a couple $(\boldsymbol{w} \in \mathbb{R}^n, \gamma \in \mathbb{R})$ that strictly separates A and B. Thus one would require

$$\boldsymbol{w}^T \boldsymbol{a}_i + \gamma \leq -1, \quad i \in \mathcal{I} \tag{17.71}$$

and

$$\boldsymbol{w}^T \boldsymbol{b}_j + \gamma \geq 1, \quad j \in \mathcal{J}. \tag{17.72}$$

Since the necessary and sufficient condition for existence of such strictly separating hyperplane

$$\text{conv}\{A\} \cap \text{conv}\{B\} = \emptyset$$

is, in general, not known to hold in advance, we define the *point classification error functions* $\xi_i(\boldsymbol{w}, \gamma)$, $i \in \mathcal{I}$ and $\zeta_j(\boldsymbol{w}, \gamma)$, $j \in \mathcal{J}$ as follows:

$$\xi_i(\boldsymbol{w}, \gamma) = \max\left\{0, 1 + (\boldsymbol{w}^T \boldsymbol{a}_i + \gamma)\right\}, \quad i \in \mathcal{I}, \tag{17.73}$$

and

$$\zeta_j(\boldsymbol{w}, \gamma) = \max\left\{0, 1 - (\boldsymbol{w}^T \boldsymbol{b}_j + \gamma)\right\}, \quad j \in \mathcal{J}. \tag{17.74}$$

Note that ξ_i and ζ_j are positive if and only if (17.71) and (17.72) are violated, respectively. Consequently, they can be considered as a measure of the classification error related to point $\boldsymbol{a}_i \in A$ and $\boldsymbol{b}_j \in B$.

The convex and nonsmooth error function $E(\boldsymbol{w}, \gamma)$ is then defined as

$$\begin{aligned} E(\boldsymbol{w}, \gamma) &= \sum_{i \in \mathcal{I}} \max\left\{0, 1 + (\boldsymbol{w}^T \boldsymbol{a}_i + \gamma)\right\} + \sum_{j \in \mathcal{J}} \max\left\{0, 1 - (\boldsymbol{w}^T \boldsymbol{b}_j + \gamma)\right\} \\ &= \sum_{i \in \mathcal{I}} \xi_i(\boldsymbol{w}, \gamma) + \sum_{j \in \mathcal{J}} \zeta_j(\boldsymbol{w}, \gamma). \end{aligned}$$

Finally we come out with the standard SVM model:

$$\begin{cases} \underset{\boldsymbol{w}, \gamma, \boldsymbol{\xi}, \boldsymbol{\zeta}}{\text{minimize}} & \|\boldsymbol{w}\|^2 + C\left(\sum_{i \in \mathcal{I}} \xi_i + \sum_{j \in \mathcal{J}} \zeta_j\right) \\ \text{subject to} & \boldsymbol{a}_i^T \boldsymbol{w} + \gamma \leq \xi_i - 1, \quad i \in \mathcal{I}, \\ & -\boldsymbol{b}_l^T \boldsymbol{w} - \gamma \leq \zeta_j - 1, \quad j \in \mathcal{J}, \\ & \xi_i \geq 0, \quad i \in \mathcal{I}, \\ & \zeta_j \geq 0, \quad j \in J. \end{cases} \tag{17.75}$$

The objective function is the sum of the error function E weighted by the parameter $C > 0$ and the square of the norm of $\boldsymbol{w}$. The latter term is aimed at maximizing the *separation margin* between the two sets A and B. In fact (see [15, Chapter 6]) $\frac{2}{\|\boldsymbol{w}\|}$

is the distance (the separation margin) between the hyperplanes

$$H^- = \{\boldsymbol{x} \mid \boldsymbol{w}^T\boldsymbol{x} + b = -1\} \text{ and } H^+ = \{\boldsymbol{x} \mid \boldsymbol{w}^T\boldsymbol{x} + b = 1\},$$

thus minimization of $\|\boldsymbol{w}\|$ leads to maximization of the margin. In practical applications the squared norm $\|\boldsymbol{w}\|^2$ replaces $\|w\|$ in the objective function.

In the SVM framework the role of the FS is to detect those features that are really relevant for classification purposes. In other words a classifier embedding a FS mechanism is expected to guarantee classification correctness (effective separation of A and B) and, also, to be *parsimonious*, that is to provide a vector $\boldsymbol{w}$ (the normal to the separating hyperplane) with as few as possible non-zero components.

Several different optimization-based approaches for the FS are available in literature; we cite here, among the others, [9] and [47]. We focus on treatment of the FS problem via mixed integer programming (see [6, 42]). In particular, we refer to the model described in [28], where a Lagrangian relaxation approach has been implemented.

Aiming at enforcing a feature selection mechanism, that is at reducing the number of the non-zero components of $\boldsymbol{w}$, the binary *feature variable* vector $\boldsymbol{y} \in \mathbb{R}^n$ is embedded into the model (17.75), with y_k indicating whether or not feature k is active. The following mixed binary formulation of the SVM-FS problem is stated:

$$\begin{cases} \underset{\boldsymbol{w},\gamma,\boldsymbol{\xi},\boldsymbol{\zeta}}{\text{minimize}} & \|\boldsymbol{w}\|^2 + C(\sum_{i\in\mathcal{I}} \xi_i + \sum_{j\in\mathcal{J}} \zeta_j) + D\sum_{k=1}^{n} y_k \\ \text{subject to} & \boldsymbol{a}_i^T\boldsymbol{w} + \gamma \le \xi_i - 1, \quad i \in \mathcal{I}, \\ & -\boldsymbol{b}_l^T\boldsymbol{w} - \gamma \le \zeta_j - 1, \quad j \in \mathcal{J}, \\ & -u_k y_k \le w_k \le u_k y_k, \quad k = 1,\dots,n, \\ & -u_k \le w_k \le u_k, \qquad k = 1,\dots,n, \\ & \xi_i \ge 0, \quad i \in \mathcal{I}, \\ & \zeta_j \ge 0, \quad j \in J, \\ & y_k \text{ binary}, \quad k = 1,\dots,n. \end{cases} \tag{17.76}$$

The objective function of the problem (17.76) is the sum of three terms. The norm $\|\boldsymbol{w}\|$, as usual in SVM-type models, is intended to maximize the separation margin (we note in passing that in [28] the L_1-norm, instead of the more commonly used L_2, is adopted). The second term is the error function $E(\boldsymbol{w}, \gamma)$ and, finally, the third one represents the number of nonzero components of $\boldsymbol{w}$. Note also the presence of the positive weights C and D in the objective function and of the upper bounds $u_k > 0$ on the modulus of each component w_k of $\boldsymbol{w}$.

In [28] the problem above has been tackled by resorting to Lagrangian relaxation of the constraints linking the variables $\boldsymbol{w}$ and $\boldsymbol{y}$, by means of the multiplier vectors

$\boldsymbol{\lambda} \geq 0$ and $\boldsymbol{\mu} \geq 0$. Thus we obtain

$$\begin{aligned} z_{LR}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \;=\; \min_{\boldsymbol{w},\gamma,\xi,\zeta,\boldsymbol{y}} \quad & \|\boldsymbol{w}\| + C\left(\sum_{i=1}^{m_1}\xi_i + \sum_{l=1}^{m_2}\zeta_l\right) + D\sum_{k=1}^{n} y_k \\ & + \sum_{k=1}^{n}\lambda_k(w_k - u_k y_k) - \sum_{k=1}^{n}\mu_k(w_k + u_k y_k) \\ \text{subject to} \quad & \boldsymbol{a}_i^T\boldsymbol{w} + \gamma \leq \xi_i - 1, \quad i \in \mathcal{I}, \\ & -\boldsymbol{b}_l^T\boldsymbol{w} - \gamma \leq \zeta_j - 1, \quad j \in \mathcal{J}, \\ & -u_k \leq w_k \leq u_k, \qquad k = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i \in \mathcal{I}, \\ & \zeta_j \geq 0, \quad j \in J, \\ & y_k \text{ binary}, \quad k = 1, \ldots, n. \end{aligned} \tag{17.77}$$

By rearranging the objective function, we get to the following decomposed formulation:

$$z_{LR}(\boldsymbol{\lambda}, \boldsymbol{\mu}) = z_{LR}^{(1)}(\boldsymbol{\lambda}, \boldsymbol{\mu}) + z_{LR}^{(2)}(\boldsymbol{\lambda}, \boldsymbol{\mu}),$$

with

$$\begin{aligned} z_{LR}^{(1)}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \;=\; \min_{\boldsymbol{w},\gamma,\xi,\zeta} \quad & \|\boldsymbol{w}\|^2 + C\left(\sum_{i=1}^{m_1}\xi_i + \sum_{l=1}^{m_2}\zeta_l\right) + \sum_{k=1}^{n}(\lambda_k - \mu_k)w_k \\ \text{subject to} \quad & \boldsymbol{a}_i^T\boldsymbol{w} + \gamma \leq \xi_i - 1, \quad i \in \mathcal{I}, \\ & -\boldsymbol{b}_l^T\boldsymbol{w} - \gamma \leq \zeta_j - 1, \quad j \in \mathcal{J}, \\ & -u_k \leq w_k \leq u_k, \qquad k = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i \in \mathcal{I}, \\ & \zeta_j \geq 0, \quad j \in J, \end{aligned} \tag{17.78}$$

and

$$\begin{aligned} z_{LR}^{(2)}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \;=\; \min_{\boldsymbol{y}} \quad & \sum_{k=1}^{n}(D - u_k(\lambda_k + \mu_k))y_k \\ \text{subject to} \quad & y_k \text{ binary}, \quad k = 1, \ldots, n. \end{aligned} \tag{17.79}$$

Note that calculation of $z_{LR}^{(1)}(\boldsymbol{\lambda}, \boldsymbol{\mu})$ requires solution of a problem SVM-like, the only difference being the presence of the linear term $\sum_{k=1}^{n}(\lambda_k - \mu_k)w_k$ into the objective function. As for the second problem, $z_{LR}^{(2)}(\boldsymbol{\lambda}, \boldsymbol{\mu})$ can be simply calculated by sign inspection of the cost coefficients $(D - u_k(\lambda_k + \mu_k))$, $k = 1, \ldots, n$.

As for the Lagrangian dual

$$z_{LD} = \max_{(\boldsymbol{\lambda}, \boldsymbol{\mu}) \geq 0} \quad z_{LR}(\boldsymbol{\lambda}, \boldsymbol{\mu}),$$

the following proposition [28] holds:

Proposition 17.1 *There exists an optimal solution to the Lagrangian dual satisfying the condition*

$$u_k(\lambda_k + \mu_k) = D, \quad k = 1, \ldots, n. \tag{17.80}$$

In [28] the above property is utilized to eliminate the variables μ_k, $k = 1, \ldots, n$ in the Lagrangian dual, which is then tackled by resorting to the general purpose C++ bundle code developed in [23].

Solution of the Lagrangian dual is embedded into a Lagrangian heuristic algorithm. Note in fact that a feasible solution (and consequently an upper bound) for the problem (17.76) can be easily obtained starting from the optimal $\boldsymbol{w}(\boldsymbol{\lambda})$ obtained by solving the Lagrangian relaxation for any feasible choice of the multiplier vector $\boldsymbol{\lambda}$. It is in fact sufficient to set $y_k = 1$ whenever $|w_k(\boldsymbol{\lambda})| > 0$ and $y_k = 0$ otherwise.

Also in this application the Lagrangian heuristic approach has proved to work well. The numerical results [28] are quite satisfactory, particularly in terms of trade-off between the classification quality and the number of active features.

17.5.6 Multiple Instance Learning

Multiple instance learning (MIL) [1] is a classification paradigm, connected to SVM [15], which is capable to handle complex problems, mainly in medical image analysis and in text categorization. While the objective of SVM-like methods is to classify samples in a given feature space, MIL deals with classification of *sets* of samples, bags in Machine Learning parlance. We discuss here a specific MIL problem where binary classification of bags is considered. The approach has been introduced in [2] and Lagrangian relaxation has been applied in [4], giving rise to a Lagrangian heuristic algorithm.

To provide an intuitive explanation of the problem, we describe an example driven from image classification where the objects to be classified are images, each of them representing a certain mix of geometric figures (e.g. triangles, squares, circles and stars) of different size. Each image is a bag which is segmented according to some appropriate rule [55] and in turn a feature vector in $\mathbb{R}^n$ is associated

to each image segment, where aggregate information about segment luminosity, texture, geometry etc. are reported. Thus each image is represented by a set of points (instances), each of them being a vector in the feature spaces representing one of its segments. In our example the problem at hand is to discriminate between images containing at least one star (positive images) and those which contain no star (negative ones). In the supervised classification framework, the class membership of the images is known, thus each of them is labelled either as positive or negative.

A classic SVM scheme, aimed at separating by means of a hyperplane the instances of positive bags from those of the negative ones, does not seem profitable in this case. In fact the similarity degree between positive and negative images is high (after all triangles, square and circles may definitely appear in images from both classes) and, consequently, the expected separation quality is poor.

The MIL introduces a different paradigm. It is assumed in fact that a hyperplane in the feature space correctly classifies the images if *all* instances of each negative bag are in the same halfspace, while *at least* one instance of each positive bag is on the other side.

The MINLP formulation of the problem proposed in [2] follows: Assume m positive bags are given and let $J^+ = \{\mathcal{J}_1^+, \ldots, \mathcal{J}_m^+\}$ be the family of the index sets of the instances of each bag. Analogously let $J^- = \{\mathcal{J}_1^-, \ldots, \mathcal{J}_k^-\}$ be the family of the index sets for k given negative bags. We indicate by $\boldsymbol{x}_j \in \mathbb{R}^n$ the j-th instance belonging to a positive or negative bag.

In the classic *instance based* SVM, we would look for a hyperplane defined by a couple $(\boldsymbol{w} \in \mathbb{R}^n, \gamma \in \mathbb{R})$ separating the instances belonging to the negative bags from those belonging to the positive ones.

Instead, according to [2], one looks for a hyperplane

$$H(\boldsymbol{w}, \gamma) = \{\boldsymbol{x} \mid \boldsymbol{w}^T \boldsymbol{x} + \gamma = 0\},$$

such that

1. all negative bags are contained in the set $S^- = \{\boldsymbol{x} \mid \boldsymbol{w}^T \boldsymbol{x} + \gamma \leq -1\}$;
2. at least one instance of each positive bag belongs to the set $S^+ = \{\boldsymbol{x} \mid \boldsymbol{w}^T \boldsymbol{x} + \gamma \geq 1\}$.

The following optimization model, aimed at finding such a hyperplane, is then introduced. The decision variables, apart the couple $(\boldsymbol{w} \in \mathbb{R}^n, \gamma \in R)$, are the labels $y_j \in \{-1, 1\}$ to be assigned to all instances of the positive bags. The twofold objective consists of minimizing the classification error (which is equal to zero in case a separating hyperplane is actually found) and of maximizing the separation margin, defined as the distance between the shifted hyperplanes (see Sect. 17.5.5)

$$H^- = \{\boldsymbol{x} \mid \boldsymbol{w}^T \boldsymbol{x} + \gamma = -1\} \quad \text{and} \quad H^+ = \{\boldsymbol{x} \mid \boldsymbol{w}^T \boldsymbol{x} + \gamma = 1\}.$$

That is,

$$\begin{aligned} z^* = \min_{\boldsymbol{w},\gamma,\boldsymbol{y}} \quad & f(\boldsymbol{w},\gamma,\boldsymbol{y}) \\ \text{subject to} \quad & \sum_{j\in\mathcal{J}_i^+} \frac{y_j+1}{2} \geq 1, \quad i=1,\ldots,m, \\ & y_j \in \{-1,1\}, \quad j\in\mathcal{J}_i^+, \quad i=1,\ldots,m, \end{aligned} \tag{17.81}$$

where

$$\begin{aligned} f(\boldsymbol{w},\gamma,\boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{w}\|^2 &+ C\sum_{i=1}^{k}\sum_{j\in\mathcal{J}_i^-} \max\{0, 1+(\boldsymbol{w}^T\boldsymbol{x}_j+\gamma)\} \\ &+ C\sum_{i=1}^{m}\sum_{j\in\mathcal{J}_i^+} \max\{0, 1-y_j(\boldsymbol{w}^T\boldsymbol{x}_j+\gamma)\}, \end{aligned}$$

with $\|\cdot\|$ being the Euclidean norm in $\mathbb{R}^n$ and $C>0$ the trade-off parameter. Note that constraints

$$\sum_{j\in\mathcal{J}_i^+} \frac{y_j+1}{2} \geq 1, \quad i=1,\ldots m$$

impose that, for each positive bag, at least one of its samples must be labelled as a positive one. Function f is the sum of three terms:

1. $\frac{1}{2}\|\boldsymbol{w}\|^2$. As previously mentioned, minimization of $\|\boldsymbol{w}\|$ leads to maximization of the margin;
2. $\sum_{i=1}^{k}\sum_{j\in\mathcal{J}_i^-} \max\{0, 1+(\boldsymbol{w}^T\boldsymbol{x}_j+\gamma)\}$. This term is the total classification error relatively to the negative bags;
3. $\sum_{i=1}^{m}\sum_{j\in\mathcal{J}_i^+} \max\{0, 1-y_j(\boldsymbol{w}^T\boldsymbol{x}_j+\gamma)\}$. This term represents the total classification error of the instances belonging to positive bags. Notice that such an error is zero if and only if for each positive bag $\mathcal{J}_i^+$, $i=1,\ldots,m$, there exists at least one instance $j\in\mathcal{J}_i^+$ such that $\boldsymbol{w}^T\boldsymbol{x}_j+\gamma \geq 1$. Note that, by letting the corresponding label $y_j=1$, feasibility with respect to constraint $\sum_{j\in\mathcal{J}_i^+}\frac{y_j+1}{2}\geq 1$ is achieved and, in addition, the classification error associated to such a bag is driven to zero, provided no instance of such bag falls into the "no man land", that is in the area where $|\boldsymbol{w}^T\boldsymbol{x}+\gamma|<1$.

Summing up the classification error is equal to zero if and only if all negative bags are contained in the set S^-, at least one instance of each positive bag belongs to the set S^+ and no instance $\boldsymbol{x}_j$ of any positive bag satisfies the condition $|\boldsymbol{w}^T\boldsymbol{x}_j+\gamma|<1$.

The Lagrangian heuristic introduced in [4] is based on relaxation of the linear constraints $\sum_{j\in\mathcal{J}_i^+} \frac{y_j+1}{2} \geq 1$. The following problem is then obtained:

$$z_{LR}(\boldsymbol{\lambda}) = \min_{\boldsymbol{w},\gamma,\boldsymbol{y}} \quad f(\boldsymbol{w},\gamma,\boldsymbol{y}) + \sum_{i=1}^{m} \lambda_i \left(1 - \sum_{j\in\mathcal{J}_i^+} \frac{y_j+1}{2}\right) \tag{17.82}$$

$$\text{subject to } \; y_j \in \{-1,1\}, \quad j \in \mathcal{J}_i^+, \quad i = 1,\ldots,m,$$

where $\boldsymbol{\lambda} \geq 0$ is a vector of multipliers in $\mathbb{R}^m$.

The Lagrangian dual problem is, as usual,

$$z_{LD} = \max_{\lambda\geq 0} \quad z_{LR}(\boldsymbol{\lambda}). \tag{17.83}$$

It is of course $z_{LR}(\boldsymbol{\lambda}) \leq z^*$, for any choice of the multiplier $\boldsymbol{\lambda} \geq 0$, and $z_{LD} \leq z^*$. The tackling problem (17.83) within a Lagrangian heuristic scheme requires at each iteration calculation of function $z_{LR}(\boldsymbol{\lambda})$ by solving (17.82), a MINLP which is particularly suitable for application of a block coordinate descent method (see [52]). In fact the algorithm adopted in [4] works by alternately fixing, at each iteration, the values of $\boldsymbol{y}$ and of the couple $(\boldsymbol{w},\gamma)$, according to the following scheme.

Algorithm 17.3: Calculation of $z_{LR}(\boldsymbol{\lambda})$

Step 0. Choose a feasible point $\boldsymbol{y}^{(0)}$. Set $l = 0$.

Step 1. For the current $\boldsymbol{y}^{(l)}$ solve the convex problem

$$\min_{\boldsymbol{w},\gamma} f(\boldsymbol{w},\gamma,\boldsymbol{y}^{(l)}) + \sum_{i=1}^{m} \lambda_i \left(1 - \sum_{j\in J_i^+} \frac{y_j^{(l)}+1}{2}\right)$$

and obtain the couple $(\boldsymbol{w}^{(l+1)}, \gamma^{(l+1)})$.

Step 2. For the current couple $(\boldsymbol{w}^{(l+1)}, \gamma^{(l+1)})$ solve the problem

$$\min_{y_j\in\{-1,1\}} f(\boldsymbol{w}^{(l+1)}, \gamma^{(l+1)}, \boldsymbol{y}) + \sum_{i=1}^{m} \lambda_i \left(1 - \sum_{j\in J_i^+} \frac{y_j+1}{2}\right)$$

and obtain $\boldsymbol{y}^{(l+1)}$. Set $l = l+1$ and go to Step 1.

We remark that the minimization problem at Step 1 is a standard SVM-like problem, while solution of the problem at Step 2 can be easily obtained by inspection of the values $h_j^{(l+1)} = \boldsymbol{w}^{(l+1)T}\boldsymbol{x}_j + \gamma^{(l+1)}$.

In [4] update of the multiplier vector $\boldsymbol{\lambda}$ is performed by standard subgradient method adopting the Polyak step size (17.28). A projection mechanism to take into account the nonnegativity of $\boldsymbol{\lambda}$ is also embedded.

To complete the bird's-eye view of the Lagrangian heuristic, we observe that any solution $(\boldsymbol{w}(\boldsymbol{\lambda}), \gamma(\boldsymbol{\lambda}), \boldsymbol{y}(\boldsymbol{\lambda}))$ of the Lagrangian relaxation which violates the relaxed constraints can be easily "repaired" to get feasibility by means of greedy modification of one or more label variables y_j.

It is worth noting that the Lagrangian dual (17.83) enjoys the relevant property that the duality gap is equal to zero, that is $z_{LD} = z^*$. Moreover, by solving the Lagrangian dual, one gets, in fact, also an optimal solution for the problem (17.81). These results are provided by the following theorem [4].

Theorem 17.2 *Let $\boldsymbol{\lambda}^*$ be any optimal solution to the Lagrangian dual (17.83) and let $(\boldsymbol{w}^*, \gamma^*, \boldsymbol{y}^*)$ be any optimal solution to the Lagrangian relaxation (17.82) for $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$. Then $(\boldsymbol{w}^*, \gamma^*, \boldsymbol{y}^*)$ is optimal for the original problem (17.81) and $z_{LD} = z^*$.*

The implementation of the Lagrangian heuristic has provided satisfactory results on a number of benchmark test problems. In particular the zero duality gap has been fully confirmed.

17.6 Conclusions

We have provided some basic notions on Lagrangian relaxation, focusing on its application in designing heuristic algorithms of the so-called Lagrangian heuristic type. Although the number of applications available in the literature is practically uncountable, we are convinced that the potential for future application is still enormous.

References

1. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. Artif. Intell. **201**, 81–105 (2013)
2. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, pp. 561–568. MIT, Cambridge (2003)
3. Astorino, A., Gaudioso, M., Miglionico, G.: Lagrangian relaxation for the directional sensor coverage problem with continuous orientation. Omega **75**, 1339–1351 (2018)
4. Astorino, A., Fuduli, A., Gaudioso, M.: A Lagrangian relaxation approach for binary multiple instance classification. IEEE Trans. Neural Netw. Learn. Syst. (2019). https://doi.org/10.1109/TNNLS.2018.2885852
5. Bacaud, L., Lemaréchal, C., Renaud, A., Sagastizábal, C.: Bundle methods in stochastic optimal power management: a disaggregated approach using preconditioners. Comput. Optim. Appl. **20**, 227–244 (2001)

6. Bertolazzi, P., Felici, G., Festa, P., Fiscon, G., Weitschek, E.: Integer programming models for feature selection: new extensions and a randomized solution algorithm. Eur. J. Oper. Res. **250**, 389–399 (2016)
7. Bertsekas, D.P., Nedic, A.: Incremental subgradient methods for nondifferentiable optimization. SIAM J. Optim. **12**, 109–138 (2001)
8. Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C.A.: Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment. IEEE Trans. Power Syst. **18**, 313–323 (2003)
9. Bradley, P.S., Mangasarian, O.L., Street, W.N.: Feature selection via mathematical programming. INFORMS J. Comput. **10**, 209–217 (1998)
10. Carrabs, F., Cerulli, R., Gaudioso, M., Gentili, M.: Lower and upper bounds for the spanning tree with minimum branch vertices. Comput. Optim. Appl. **56**, 405–438 (2013)
11. Celani, M., Cerulli, R., Gaudioso, M., Sergeyev, Ya.D.: A multiplier adjustment technique for the capacitated concentrator location problem. Optim. Methods Softw. **10**, 87–102 (1998)
12. Cerulli, R., De Donato, R., Raiconi, A.: Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges. Eur. J. Oper. Res. **220**, 58–66 (2012)
13. Cheney, E.W., Goldstein, A.A.: Newton's method for convex programming and Tchebycheff approximation. Numer. Math. **1**, 253–268 (1959)
14. Chiarello, A., Gaudioso, M., Sammarra, M.: Truck synchronization at single door cross-docking terminals. OR Spectr. **40**, 395–447 (2018)
15. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University. Cambridge (2000)
16. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Program. **148**, 241–277 (2014)
17. Di Puglia Pugliese, L., Gaudioso, M., Guerriero, F., Miglionico, G.: A Lagrangean-based decomposition approach for the link constrained Steiner tree problem. Optim. Methods Softw. **33**, 650–670 (2018)
18. Everett, H. III: Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper. Res. **11**, 399–417 (1963)
19. Finardi, E., da Silva, E., Sagastizábal, C.: Solving the unit commitment problem of hydropower plants via Lagrangian relaxation and sequential quadratic programming. Comput. Appl. Math. **24**, 317–342 (2005)
20. Fischer, F., Helmberg C., Janßen, J., Krostitz, B.: Towards solving very large scale train timetabling problems by Lagrangian relaxation. In: Fischetti, M., Widmayer, P. (eds.) 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl (2008)
21. Fisher, M.L.: The Lagrangian relaxation method for solving integer programming problems. Manag. Sci. **27**, 1–18 (1981)
22. Fisher, M.L., Jaikumar, R., Van Wassenhove, L.N.: A multiplier adjustment method for the generalized assignment problem. Manag. Sci. **32**, 1095–1103 (1986)
23. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **13**, 117–156 (2002)
24. Frangioni, A.: About Lagrangian methods in integer optimization. Ann. Oper. Res. **139**, 163–193 (2005)
25. Frangioni, A., Gorgone, E., Gendron, B.: On the computational efficiency of subgradient methods: a case study with Lagrangian bounds. Math. Program. Comput. **9**, 573–604 (2017)
26. Frangioni, A., Gorgone, E., Gendron, B.: Dynamic smoothness parameter for fast gradient methods. Optim. Lett. **12**, 43–53 (2018)
27. Gaudioso, M., Giallombardo, G., Miglionico, G.: On solving the Lagrangian dual of integer programs via an incremental approach. Comput. Optim. Appl. **44**, 117–138 (2009)
28. Gaudioso, M., Gorgone, E., Labbé M., Rodríguez-Chía, A.: Lagrangian relaxation for SVM feature selection. Comput. Oper. Res. **87**, 137–145 (2017)

29. Geoffrion A.M.: Lagrangean relaxation for integer programming. Math. Program. Study **2**, 82–114 (1974)
30. Goffin J.-L., Haurie, A., Vial, J.-P.: On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. Math. Program. **60**, 81–92 (1993)
31. Goffin J.-L., Gondzio, J., Sarkissian, R., Vial, J.-P.: Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. Math. Program. **76**, 131–154 (1996)
32. Guignard, M.: Lagrangean relaxation. Top **11**, 151–228 (2003)
33. Guignard, M., Rosenwein, B.M.: An application-oriented guide for designing Lagrangean dual ascent algorithms. Eur. J. Oper. Res. **43**, 197–205 (1989)
34. Hiriart-Urruty, J.-B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms, vols. I–II. Springer, Heidelberg (1993)
35. Kelley, J.E.: The cutting-plane method for solving convex programs. J. SIAM **8**, 703–712 (1960)
36. Kiwiel, K.: Convergence of approximate and incremental subgradient methods for convex optimization. SIAM J. Optim. **14**, 807–840 (2003)
37. Kiwiel, K.C., Lemaréchal, C.: An inexact bundle variant suited to column generation. Math. Program. **118**, 177–206 (2009)
38. Lagrange, J.L.: Mécanique Analytique. Courcier, Paris (1811)
39. Lemaréchal, C.: The omnipresence of Lagrange. Ann. Oper. Res. **153**, 9–27 (2007)
40. Lemaréchal, C., Pellegrino, F., Renaud, A., Sagastizábal, C.: Bundle methods applied to the unit-commitment problem. In: System Modelling and Optimization, pp. 395–402. Chapman and Hall, London (1996)
41. Lemaréchal, C., Ouorou, A., Petrou, G.: A bundle-type algorithm for routing in telecommunication data networks. Comput. Optim. Appl. **44**, 385–409 (2009)
42. Maldonado, S., Pérez, J., Weber, R., Labbé, M.: Feature selection for support vector machines via mixed integer linear programming. Inf. Sci. **279**, 163–175 (2014)
43. Monaco, M.F., Sammarra, M.: The berth allocation problem: a strong formulation solved by a Lagrangean approach. Transp. Sci. **41**, 265–280 (2007)
44. Nesterov, Yu.: Smooth minimization of non-smooth functions. Math. Program. **103**, 127–152 (2005)
45. Piccialli, V., Sciandrone, M.: Nonlinear optimization and support vector machines. 4OR **16**, 111–149 (2018)
46. Polyak, B.T.: Introduction to Optimization. Optimization Software, New York (1987)
47. Rinaldi, F., Sciandrone, M.: Feature selection combining linear support vector machines and concave optimization. Optim. Methods Softw. **10**, 117–128 (2010)
48. Rockafellar, R.T.: Convex Analysis. Princeton University, Princeton (1970)
49. Rockafellar, R.T.: Lagrange multipliers and optimality. SIAM Rev. **35**, 183–238 (1993)
50. Rossi, A., Singh, A., Sevaux, M.: Lifetime maximization in wireless directional sensor network. Eur. J. Oper. Res. **231**, 229–241 (2013)
51. Shor, N.Z., Minimization Methods for Non-differentiable Functions. Springer, New York (1985)
52. Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. **109**, 475–494 (2001)
53. Wolsey, L.A.: Integer Programming. Wiley-Interscience, New York (1998)
54. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Netw. **52**, 2292–2330 (2008)
55. Zhang, H., Fritts, J.E., Goldman, S.A.: Image segmentation evaluation: a survey of unsupervised methods. Comput. Vis. Image Underst. **110**, 260–280 (2008)

Part IV
Derivative-Free Methods

Chapter 18
Discrete Gradient Methods

Adil M. Bagirov, Sona Taheri, and Napsu Karmitsa

Abstract In this chapter, the notion of a discrete gradient is introduced and it is shown that the discrete gradients can be used to approximate subdifferentials of a broad class of nonsmooth functions. Two methods based on such approximations, more specifically, the discrete gradient method (DGM) and its limited memory version (LDGB), are described. These methods are semi derivative-free methods for solving nonsmooth and, in general, nonconvex optimization problems. The performance of the methods is demonstrated using some academic test problems.

18.1 Introduction

In this chapter, we describe the notion of the discrete gradient for approximating subdifferentials of nonsmooth functions and discuss two minimization algorithms based on such approximation. Consider the unconstrained nonsmooth optimization (NSO)

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{18.1}$$

where the objective function f is, in general, locally Lipschitz continuous (LLC) and not necessarily convex.

The problem (18.1) has been a subject of research for almost 60 years. Different methods have been developed to solve this problem including the subgradient

A. M. Bagirov (✉) · S. Taheri
School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, VIC, Australia
e-mail: a.bagirov@federation.edu.au; s.taheri@federation.edu.au

N. Karmitsa
Department of Mathematics and Statistics, University of Turku, Turku, Finland
e-mail: napsu@karmitsa.fi

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_18

method [26], the bundle-type methods [9, 11, 12, 16, 20–23, 29, 31], algorithms based on smoothing techniques [8, 24, 25, 30], and the gradient sampling algorithm [10] (see also Chaps. 2, 3, and 6).

In most NSO methods it is assumed that the value of the objective function and one of its subgradients can be computed at any given point. However, in some practical applications computing subgradients of involved functions is not possible or it is time consuming. This may happen when the chain rule does not exist or analytic expressions of functions are not available. In such situations, derivative free methods can be applied to solve the problems. In principle, most of these methods (for example, the generalized pattern search methods [1, 2, 28]) can be applied to solve NSO problems, although, the convergence of these methods can be proved under rather restrictive differentiability or strict differentiability conditions. For example, in [1] (see, also [2]), the convergence of the generalized pattern search method is proved under the assumption that the function f is strictly differentiable near the limit point. However, in many important practical problems the objective functions are not strictly differentiable at their local minimizers. Moreover, the conventional finite difference approximations are not applicable to nonsmooth functions since they may produce an approximation that does not belong to the subdifferential.

In this chapter, we describe the discrete gradients that can be considered as a generalization of finite difference approximations for nonsmooth functions. Different approaches have been used to develop finite difference estimates to subgradients of nonconvex nonsmooth functions in [26, 27]. However, these estimates are able to approximate only one subgradient at a given point while the discrete gradients can be used to approximate the whole subdifferential. The notion of the discrete gradient was introduced in [3], and then modified and applied to design numerical algorithms in [4–7, 17, 18]. Here we present a further modification of the discrete gradient by removing infinitesimal functions in its definition.

Using the discrete gradients, we design two bundle-type semi-derivative-free methods for solving nonsmooth and, in general, nonconvex optimization problems: the discrete gradient method (DGM) [7] and the limited memory discrete gradient bundle method (LDGB) [18]. Both these methods use the discrete gradients that are computed using only function values to find the descent direction of a nonsmooth objective function. Furthermore, neither of them uses approximations of subgradients in most iterations and such approximations are utilized only in the final phase of the methods. Therefore, they can be considered as semi derivative-free methods.

The DGM has been developed for solving NSO problems with relatively small dimensions while the LDGB is capable of solving large scale NSO problems. On the other hand, in small problems the accuracy of the DGM is usually better than that of the LDGB. Although these two methods are different, they have some interesting similarities in their structures, that is, in addition to using the discrete gradients and bundling the information, both methods wipe out the old information whenever the serious step occurs. This feature is different from standard bundle methods, where the old information is collected near the current iteration point and stored to be

used in the next iterations. In practice, storing all information collected in previous iterations may have several disadvantages: first, it needs a storage space, although unbounded storage requirement may be addressed by the so-called aggregation procedure introduced in [19]; second, it adds computational cost; and, furthermore, it may store and use information that is no longer relevant due to the fact that it might have been collected far away from the current iteration point. The last one may be especially problematic in the nonconvex case.

The rest of this chapter is organized as follows. Section 18.2 provides some theoretical background which will be used to define the discrete gradients. In Sect. 18.3 we describe the discrete gradients, and in Sect. 18.4 we discuss approximation of subdifferentials using the discrete gradients. The DGM is presented in Sect. 18.5 and the LDGB is described in Sect. 18.6. The performance of the methods using several academic test problems are given in Sect. 18.7. Section 18.8 concludes the chapter.

18.2 Theoretical Background

In this section we provide theoretical background and preliminaries that will be used throughout this chapter. Let $\bar{A} \subset \mathbb{R}^n$ be a polytope represented as a convex hull of a set $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\} \subset \mathbb{R}^n$ of a finite number of points. In addition, let G be a set of all vertices of the unit hypercube in $\mathbb{R}^n$, that is

$$G = \{\boldsymbol{e} \in \mathbb{R}^n \ : \ \boldsymbol{e} = (e_1, \ldots, e_n), \ |e_j| = 1, \ j = 1, \ldots, n\}.$$

Given $\boldsymbol{e} \in G$ and $\alpha \in (0, 1]$, define the sequence of n vectors

$$\boldsymbol{e}_j = \boldsymbol{e}_j(\alpha) = (\alpha e_1, \alpha^2 e_2, \ldots, \alpha^j e_j, 0, \ldots, 0), \ j = 1, \ldots, n. \tag{18.2}$$

Consider the sets $R_0 \equiv R_0(\boldsymbol{e}) = A$ and

$$R_j \equiv R_j(\boldsymbol{e}) = \left\{\boldsymbol{v} \in R_{j-1}(\boldsymbol{e}) : \ v_j e_j = \max\{w_j e_j : \ \boldsymbol{w} \in R_{j-1}(\boldsymbol{e})\}\right\}$$

for $j = 1, \ldots, n$. The sets $R_j(\boldsymbol{e})$ are called R-sets of the polytope $\bar{A}$ for the given $\boldsymbol{e} \in G$. Note that

$$\begin{aligned} R_j &\neq \emptyset \qquad \text{for all } \ j \in \{0, \ldots, n\}, \\ R_j &\subseteq R_{j-1} \ \text{ for all } \ j \in \{1, \ldots, n\}, \end{aligned}$$

and

$$v_r = u_r \ \text{ for all } \ \boldsymbol{v}, \boldsymbol{u} \in R_j, \ r = 1, \ldots, j. \tag{18.3}$$

Proposition 18.1 *The set R_n is a singleton.*

Proof The proof follows from (18.3). □

Take any $\boldsymbol{a} \in A$. If $\boldsymbol{a} \notin R_n$, then there exists $r_{\boldsymbol{a}} \in \{1, \ldots, n\}$ such that $\boldsymbol{a} \in R_t$, $t = 0, \ldots, r_{\boldsymbol{a}} - 1$ and $\boldsymbol{a} \notin R_{r_{\boldsymbol{a}}}$. For each $\boldsymbol{a} \in A \setminus R_n$ we determine such $r_{\boldsymbol{a}} \in \{1, \ldots, n\}$ and define a number $z(\boldsymbol{a}) = v_{r_{\boldsymbol{a}}} e_{r_{\boldsymbol{a}}} - a_{r_{\boldsymbol{a}}} e_{r_{\boldsymbol{a}}} > 0$. Compute

$$z_1 = \min\{z(\boldsymbol{a}) : \boldsymbol{a} \in A \setminus R_n\}.$$

Since the set A is finite and $z(\boldsymbol{a}) > 0$ for all $\boldsymbol{a} \in A \setminus R_n$ it follows that $z_1 > 0$. Let

$$z_2 = \max\{\|\boldsymbol{a}\| : \boldsymbol{a} \in A\} < +\infty.$$

Take any $r, j \in \{0, \ldots, n\}$ such that $r < j$. Then for all $\boldsymbol{v}, \boldsymbol{w} \in \bar{A}$ and $\alpha \in (0, 1]$ we have

$$\left| \sum_{t=r+1}^{j} (v_t - w_t)\alpha^{t-r} e_t \right| < 2z_2 \alpha n.$$

Define

$$\alpha_0 = \min\{1, z_1/(4z_2 n)\}. \tag{18.4}$$

It is clear that $\alpha_0 \in (0, 1]$, and for any $\alpha \in (0, \alpha_0]$ we get

$$\left| \sum_{t=r+1}^{j} (v_t - w_t)\alpha^{t-r} e_t \right| < \frac{z_1}{2}. \tag{18.5}$$

For given $\alpha \in (0, 1]$, $j \in \{1, \ldots, n\}$ and the vector $\boldsymbol{e}_j$ define the set

$$\bar{R}(\boldsymbol{e}_j(\alpha)) = \left\{ \boldsymbol{v} \in A : \langle \boldsymbol{v}, \boldsymbol{e}_j \rangle = \max \{\langle \boldsymbol{u}, \boldsymbol{e}_j \rangle : \boldsymbol{u} \in A\} \right\}. \tag{18.6}$$

Proposition 18.2 *For all $j \in \{1, \ldots, n\}$ and $\alpha \in (0, \alpha_0]$, where α_0 is defined in (18.4), we have $\bar{R}(\boldsymbol{e}_j(\alpha)) \subseteq R_j(\boldsymbol{e})$.*

Proof Assume the contrary. Then for some $j \in \{1, \ldots, n\}$ and $\alpha \in (0, \alpha_0]$ there exists $\boldsymbol{y} \in \bar{R}(\boldsymbol{e}_j(\alpha))$ such that $\boldsymbol{y} \notin R_j(\boldsymbol{e})$. The latter implies that there exists $r \in \{1, \ldots, n\}$, $r \leq j$, such that $\boldsymbol{y} \notin R_r(\boldsymbol{e})$ and $\boldsymbol{y} \in R_t(\boldsymbol{e})$ for any $t = 0, \ldots, r - 1$. Take any $\boldsymbol{v} \in R_j(\boldsymbol{e})$, then $\boldsymbol{v} \in R_{r-1}(\boldsymbol{e})$ and from (18.3) we have

$$v_t = y_t \quad \text{for all } \boldsymbol{v}, \boldsymbol{y} \in R_{r-1}, \quad t = 1, \ldots, r - 1.$$

Therefore,

$$v_t e_t = y_t e_t, \quad t = 1, \ldots, r-1, \quad \text{and} \quad v_r e_r - y_r e_r \geq z_1.$$

This together with (18.5) imply that for $\boldsymbol{v}, \boldsymbol{y} \in \bar{A}$ we get

$$\begin{aligned}\langle \boldsymbol{v}, \boldsymbol{e}_j \rangle - \langle \boldsymbol{y}, \boldsymbol{e}_j \rangle &= \sum_{t=1}^{j} (v_t - y_t)\alpha^t e_t \\ &= \alpha^r \left(v_r e_r - y_r e_r + \sum_{t=r+1}^{j} (v_t - y_t)\alpha^{t-r} e_t \right) > \alpha^r z_1/2 > 0.\end{aligned}$$

Since $\boldsymbol{y} \in \bar{R}(\boldsymbol{e}_j(\alpha))$ we have $\langle \boldsymbol{y}, \boldsymbol{e}_j \rangle = \max\{\langle \boldsymbol{u}, \boldsymbol{e}_j \rangle : \boldsymbol{u} \in \bar{A}\}$ and thus,

$$\langle \boldsymbol{y}, \boldsymbol{e}_j \rangle \geq \langle \boldsymbol{v}, \boldsymbol{e}_j \rangle > \langle \boldsymbol{y}, \boldsymbol{e}_j \rangle + \alpha^r z_1/2$$

which is a contradiction. □

Next we consider a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and assume that its subdifferential $\partial f(\boldsymbol{x})$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is a polytope. It is well known [9] that the function f is directionally differentiable on $\mathbb{R}^n$ and

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \max\{\langle \boldsymbol{\xi}, \boldsymbol{d} \rangle : \boldsymbol{\xi} \in \partial f(\boldsymbol{x})\} \quad \text{for all } \boldsymbol{x}, \ \boldsymbol{d} \in \mathbb{R}^n. \tag{18.7}$$

Now let A be a set of extreme points of the subdifferential $\partial f(\boldsymbol{x})$. Then (18.7) can be written as

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \max\{\langle \boldsymbol{v}, \boldsymbol{d} \rangle : \boldsymbol{v} \in A\}. \tag{18.8}$$

Proposition 18.3 *For a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$, there exists $\alpha_0 > 0$ such that*

$$f'(\boldsymbol{x}; \boldsymbol{e}_j(\alpha)) = f'(\boldsymbol{x}; \boldsymbol{e}_{j-1}(\alpha)) + v_j \alpha^j e_j,$$

for all $\alpha \in (0, \alpha_0]$, $\boldsymbol{v} \in R_j(\boldsymbol{e})$, $\boldsymbol{e} \in G$ and $j \in \{1, \ldots, n\}$.

Proof Take any $\boldsymbol{e} \in G$ and $j \in \{1, \ldots, n\}$. It follows from (18.6) and (18.8) that

$$f'(\boldsymbol{x}; \boldsymbol{e}_j(\alpha)) = \langle \boldsymbol{v}, \boldsymbol{e}_j(\alpha) \rangle \quad \text{for all } \boldsymbol{v} \in \bar{R}(\boldsymbol{e}_j(\alpha)).$$

Proposition 18.2 implies that $\boldsymbol{v} \in R_j(\boldsymbol{e})$ for all $\alpha \in (0, \alpha_0]$ and $j \in \{1, \ldots, n\}$. Therefore,

$$f'(\boldsymbol{x}; \boldsymbol{e}_j(\alpha)) - f'(\boldsymbol{x}; \boldsymbol{e}_{j-1}(\alpha)) = \langle \boldsymbol{v}, \boldsymbol{e}_j \rangle - \langle \boldsymbol{v}, \boldsymbol{e}_{j-1} \rangle.$$

Since $\boldsymbol{v} \in R_j(\boldsymbol{e})$ we have $\boldsymbol{v} \in R_{j-1}(\boldsymbol{e})$. Then it follows from (18.3) that $v_t = u_t,\ t = 1, \ldots, j-1$, and therefore, we get

$$f'(\boldsymbol{x}; \boldsymbol{e}_j(\alpha)) - f'(\boldsymbol{x}; \boldsymbol{e}_{j-1}(\alpha)) = v_j \alpha^j e_j.$$

This completes the proof. □

Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ represented as a difference of two convex (DC) functions

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}). \tag{18.9}$$

Assume that subdifferentials $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$ of the functions f_1 and f_2 at a point $\boldsymbol{x} \in \mathbb{R}^n$ are polytopes. For a given $\boldsymbol{e} \in G$ define the R-sets $R_{1j}(\boldsymbol{e})$ and $R_{2j}(\boldsymbol{e})$ for polytopes $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$, respectively.

Corollary 18.1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a DC function, defined in (18.9). Then at a point $\boldsymbol{x} \in \mathbb{R}^n$ there exists $\alpha_0 > 0$ such that*

$$f'(\boldsymbol{x}; \boldsymbol{d}) = f_1'(\boldsymbol{x}; \boldsymbol{d}) - f_2'(\boldsymbol{x}; \boldsymbol{d}) = f'(\boldsymbol{x}; \boldsymbol{e}_{j-1}(\alpha)) + (v_j - w_j)\alpha^j e_j$$

for all $\alpha \in (0, \alpha_0],\ \boldsymbol{v} \in R_{1j}(\boldsymbol{e}),\ \boldsymbol{w} \in R_{2j}(\boldsymbol{e}),\ \boldsymbol{e} \in G$ and $j \in \{1, \ldots, n\}$.

Proof The proof is similar to that of Proposition 18.3. □

Let $\boldsymbol{e} \in G$, and $\lambda > 0,\ \alpha > 0$ be given numbers. Given $\boldsymbol{x} \in \mathbb{R}^n$, consider the following points:

$$\boldsymbol{x}^0 = \boldsymbol{x},\ \ \boldsymbol{x}^j = \boldsymbol{x}^0 + \lambda \boldsymbol{e}_j(\alpha),\ \ j = 1, \ldots, n.$$

It is clear that $\boldsymbol{x}^j = \boldsymbol{x}^{j-1} + (0, \ldots, 0, \lambda\alpha^j e_j, 0, \ldots, 0),\ j = 1, \ldots, n$. Introduce a vector $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{e}, \alpha, \lambda) \in \mathbb{R}^n$ with coordinates

$$v_j = v_j(\boldsymbol{e}, \alpha, \lambda) = (\lambda\alpha^j e_j)^{-1}\left[f(\boldsymbol{x}^j) - f(\boldsymbol{x}^{j-1})\right],\ \ j = 1, \ldots, n. \tag{18.10}$$

For any fixed $\boldsymbol{e} \in G$ and $\alpha > 0$ define the set

$$W(\boldsymbol{x}, \boldsymbol{e}, \alpha) = \left\{\boldsymbol{w} \in \mathbb{R}^n :\ \text{there exist } (\lambda_k \downarrow 0,\ k \to \infty),\ \boldsymbol{w} = \lim_{k\to\infty} \boldsymbol{v}(\boldsymbol{e}, \alpha, \lambda_k)\right\}.$$

Proposition 18.4 *Assume that a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, its subdifferential at a point $\boldsymbol{x} \in \mathbb{R}^n$ is a polytope and the number $\alpha_0 > 0$ is defined by (18.4). Then we have*

$$W(\boldsymbol{x}, \boldsymbol{e}, \alpha) \subseteq \partial f(\boldsymbol{x}) \quad \text{for all } \alpha \in (0, \alpha_0].$$

Proof It follows from (18.10) that

$$\begin{aligned} v_j(\boldsymbol{e},\alpha,\lambda) &= (\lambda\alpha^j e_j)^{-1}\left(f(\boldsymbol{x}^j)-f(\boldsymbol{x}^{j-1})\right)\\ &= (\lambda\alpha^j e_j)^{-1}\left(f(\boldsymbol{x}^j)-f(\boldsymbol{x})-(f(\boldsymbol{x}^{j-1})-f(\boldsymbol{x}))\right)\\ &= (\lambda\alpha^j e_j)^{-1}\left(\lambda f'(\boldsymbol{x},\boldsymbol{e}_j)-\lambda f'(\boldsymbol{x};\boldsymbol{e}_{j-1})+o(\lambda,\boldsymbol{e}_j)-o(\lambda,\boldsymbol{e}_{j-1})\right), \end{aligned}$$

where $\lambda^{-1}o(\lambda,\boldsymbol{e}_i)\to 0$ as $\lambda\downarrow 0$, $i=j-1,j$. Take $\boldsymbol{w}\in R_n(\boldsymbol{e})$. According to Proposition 18.1 the vector $\boldsymbol{w}$ is unique. Since $\boldsymbol{w}\in R_n(\boldsymbol{e})$ we get $\boldsymbol{w}\in R_j(\boldsymbol{e})$ for all $j\in\{1,\ldots,n\}$. Then it follows from Proposition 18.3 that for any fixed $\alpha\in(0,\alpha_0]$ we have

$$\begin{aligned} v_j(\boldsymbol{e},\alpha,\lambda) &= (\lambda\alpha^j e_j)^{-1}\left(\lambda\alpha^j e_j w_j+o(\lambda,\boldsymbol{e}_j)-o(\lambda,\boldsymbol{e}_{j-1})\right)\\ &= w_j+(\lambda\alpha^j e_j)^{-1}\left(o(\lambda,\boldsymbol{e}_j)-o(\lambda,\boldsymbol{e}_{j-1})\right), \end{aligned}$$

and therefore, we get

$$\lim_{\lambda\downarrow 0}|v_j(\boldsymbol{e},\alpha,\lambda)-w_j|=0.$$

This means that $\lim_{\lambda\downarrow 0}\boldsymbol{v}(\boldsymbol{e},\alpha,\lambda)=\boldsymbol{w}$, where $\boldsymbol{w}\in\partial f(\boldsymbol{x})$. □

Corollary 18.2 *Assume that a function $f:\mathbb{R}^n\to\mathbb{R}$ is DC, subdifferentials $\partial f_1(\boldsymbol{x})$ and $\partial f_2(\boldsymbol{x})$ of its DC components f_1 and f_2, respectively, at a point $\boldsymbol{x}\in\mathbb{R}^n$ are polytopes and the number $\alpha_0>0$ is defined in (18.4). Then*

$$W(\boldsymbol{x},\boldsymbol{e},\alpha)\subseteq\partial f_1(\boldsymbol{x})-\partial f_2(\boldsymbol{x})\quad \text{for all } \alpha\in(0,\alpha_0].$$

Corollary 18.3 *Let $f:\mathbb{R}^n\to\mathbb{R}$ be a DC function. For any fixed $\alpha\in(0,\alpha_0]$ consider the set*

$$\overline{W}(\boldsymbol{x},\alpha)=\operatorname{conv}\bigcup_{\boldsymbol{e}\in G}W(\boldsymbol{x},\boldsymbol{e},\alpha).$$

Then $\overline{W}(\boldsymbol{x},\alpha)\subseteq\partial f_1(\boldsymbol{x})-\partial f_2(\boldsymbol{x})$.

Results presented in Proposition 18.4, Corollaries 18.2 and 18.3 demonstrate how subgradients of a DC function and, in particular, a convex function f can be approximated using a vector $\boldsymbol{v}(\boldsymbol{e},\alpha,\lambda)$ with its coordinates defined in (18.10). In order to get such an approximation it is sufficient to compute the vector $\boldsymbol{v}(\boldsymbol{e},\alpha,\lambda)$ for any $\boldsymbol{e}\in G$ and sufficiently small $\alpha,\lambda>0$.

18.3 Discrete Gradients

We define the discrete gradients as finite difference approximations of subgradients. Note that the conventional finite difference estimations of gradients cannot be used to approximate subgradients as is it is shown in Example 18.1.

Example 18.1 Let the function $f : \mathbb{R}^2 \to \mathbb{R}$ be defined as

$$f(x_1, x_2) = \max\{2x_1 - x_2, -x_1 + x_2\}.$$

The subdifferential of this function at the point $\boldsymbol{x} = (0, 0)$ is

$$\partial f(0, 0) = \text{conv}\{(2, -1), (-1, 1)\}.$$

There exist various finite difference estimations of the gradients. We consider the following simplest one, that is, the vector $\boldsymbol{u} = (u_1, u_2) \in \mathbb{R}^2$, whose coordinates are

$$u_j = \lambda^{-1}\left(f(\boldsymbol{x} + \lambda \hat{\boldsymbol{e}}_j) - f(\boldsymbol{x})\right), \ j = 1, 2.$$

Here $\hat{\boldsymbol{e}}_j$ stands for the j-th unit vector. Then for the function f at the point $\boldsymbol{x} = (0, 0)$ we have $\boldsymbol{u} = (2, 1)$. It is obvious that $\boldsymbol{u} \notin \partial f(0, 0)$. The similar results can be obtained applying more sophisticated finite difference estimations of the gradients.

Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be LLC. Denote by S_1, the unit sphere in $\mathbb{R}^n$. Take any vectors $\boldsymbol{g} \in S_1$, $\boldsymbol{e} \in G$ and positive numbers $\lambda > 0$, $\alpha > 0$. Define vectors $\boldsymbol{e}_j(\alpha)$, $j = 1, \ldots, n$ as in (18.2). Consider the points

$$\boldsymbol{x}^0 = \boldsymbol{x} + \lambda \boldsymbol{g}, \ \boldsymbol{x}^j = \boldsymbol{x}^0 + \lambda \boldsymbol{e}_j(\alpha), \ j = 1, \ldots, n. \tag{18.11}$$

The illustration of these points in $\mathbb{R}^2$ is given in Fig. 18.1.

Take any $\boldsymbol{g} \in S_1$, compute $\bar{c} = \max\{|g_k|, \ k = 1, \ldots, n\}$ and define the set

$$\mathcal{I}(\boldsymbol{g}) = \{i \in \{1, \ldots, n\} : \ |g_i| = \bar{c}\}.$$

It is clear that $|g_i| \geq 1/n$ for all $i \in \mathcal{I}(\boldsymbol{g})$. Then we have the following definition for the discrete gradient of the function $f : \mathbb{R}^n \to \mathbb{R}$ at the point $\boldsymbol{x} \in \mathbb{R}^n$.

Definition 18.1 The discrete gradient of the function $f : \mathbb{R}^n \to \mathbb{R}$ at the point $\boldsymbol{x} \in \mathbb{R}^n$ is the vector

$$\boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \lambda, \alpha) = (\Gamma_1^i, \ldots, \Gamma_n^i) \in \mathbb{R}^n, \ \boldsymbol{g} \in S_1, \ i \in \mathcal{I}(\boldsymbol{g})$$

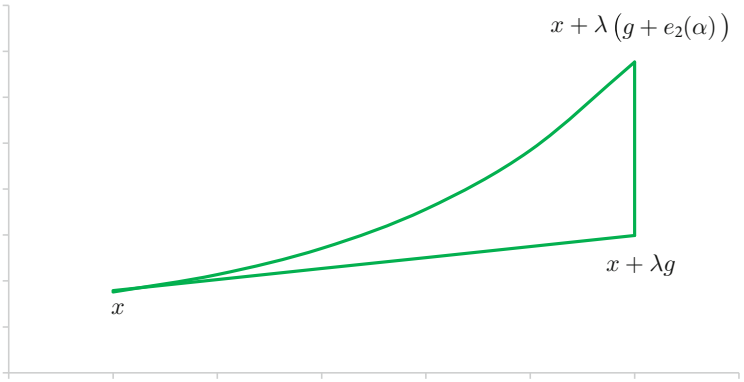

Fig. 18.1 Points $\boldsymbol{x}$, $\boldsymbol{x}^0$ and $\boldsymbol{x}^1$ in $\mathbb{R}^2$

with the following coordinates

$$\boldsymbol{\Gamma}_j^i = (\lambda\alpha^j e_j)^{-1}\left(f(\boldsymbol{x}^j) - f(\boldsymbol{x}^{j-1})\right), \quad j = 1, \ldots, n, \quad j \neq i \ \text{ and}$$

$$\boldsymbol{\Gamma}_i^i = (\lambda g_i)^{-1}\left(f(\boldsymbol{x} + \lambda\boldsymbol{g}) - f(\boldsymbol{x}) - \lambda \sum_{j=1, j\neq i}^{n} \Gamma_j^i g_j\right).$$

The discrete gradient $\boldsymbol{\Gamma}^i$ of the function f at the point $\boldsymbol{x} \in \mathbb{R}^n$ is defined in a direction $\boldsymbol{g} \in S_1$. More precisely, using a sequence of points $\boldsymbol{x}^0, \ldots, \boldsymbol{x}^n$, defined in (18.11), we calculate the values of the function f at these points and also at the point $\boldsymbol{x}$, that is we compute $n + 2$ values of this function. To compute $\boldsymbol{\Gamma}^i$, we calculate $n - 1$ coordinates Γ_j^i similar to those of the vector $\boldsymbol{v}(\boldsymbol{e}, \alpha, \lambda)$, given in (18.10). The i-th coordinate Γ_i^i is computed so that the equality

$$\begin{aligned} &f(\boldsymbol{x} + \lambda\boldsymbol{g}) - f(\boldsymbol{x}) = \lambda\langle\boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \lambda, \alpha), \boldsymbol{g}\rangle, \\ &\boldsymbol{g} \in S_1, \ \boldsymbol{e} \in G, \ \lambda > 0, \ \alpha > 0 \end{aligned} \tag{18.12}$$

is satisfied. This equality can be considered as a version of the mean value theorem.

Proposition 18.5 *Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be LLC and $L > 0$ be its Lipschitz constant on an open bounded set $X \subset \mathbb{R}^n$. Then for any $\boldsymbol{x} \in X$, $\boldsymbol{g} \in S_1$, $i \in \mathcal{I}(\boldsymbol{g})$, $\boldsymbol{e} \in G$, $\lambda > 0$, $\alpha > 0$ such that $\boldsymbol{x}^j \in X$, $j = 0, 1, \ldots, n$ we have*

$$\|\boldsymbol{\Gamma}^i\| \leq C(n)L, \quad C(n) = (4n^2 - 3n)^{1/2}.$$

Proof It follows from the definition of the discrete gradient that $|\Gamma_j^i| \leq L$ for all $j = 1, \ldots, n,\ j \neq i$. For $j = i$ we get

$$|\Gamma_i^i| \leq L \left(|g_i|^{-1} \|\boldsymbol{g}\| + \sum_{j=1, j\neq i}^{n} |g_i|^{-1} |g_j| \right).$$

Since $|g_i| = \max\{|g_j|,\ j = 1, \ldots, n\}$ we have $|g_i|^{-1}|g_j| \leq 1,\ j = 1, \ldots, n$. Moreover, since $\boldsymbol{g} \in S_1$ and $|g_i| \geq 1/n$ it follows that $|g_i|^{-1}\|\boldsymbol{g}\| \leq n$ and therefore, $|\Gamma_i^i| \leq L(2n-1)$. This together with $|\Gamma_j^i| \leq L$ complete the proof. □

18.4 Approximation of Subdifferentials

In this section we study the discrete gradients to approximate subdifferentials of convex and DC functions. For given $\alpha,\ \lambda > 0$ define the set

$$V(\boldsymbol{x}, \alpha, \lambda) = \{\boldsymbol{v} \in \mathbb{R}^n :\ \text{there exist } (\boldsymbol{g} \in S_1,\ i \in \mathcal{I}(\boldsymbol{g}),\ \boldsymbol{e} \in G),$$
$$\boldsymbol{v} = \boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \alpha, \lambda)\},$$

and for a given $\alpha > 0$, the set

$$\overline{V}(\boldsymbol{x}, \alpha) = \{\boldsymbol{v} \in \mathbb{R}^n :\ \text{there exist } (\boldsymbol{g} \in S_1,\ i \in \mathcal{I}(\boldsymbol{g}),\ \boldsymbol{e} \in G,$$
$$\{\lambda_k\},\ \lambda_k \downarrow 0 \text{ as } k \to \infty),\ \boldsymbol{v} = \lim_{k\to\infty} \boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \alpha, \lambda_k)\}.$$

Proposition 18.6 *Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be LLC. Then*

- *the set $V(\boldsymbol{x}, \alpha, \lambda)$ is compact for any $\boldsymbol{x} \in \mathbb{R}^n,\ \alpha,\ \lambda > 0$;*
- *the set $\overline{V}(\boldsymbol{x}, \alpha)$ is bounded for any $\boldsymbol{x} \in \mathbb{R}^n$ and $\alpha > 0$.*

Proof It follows from Proposition 18.5 that for any $\boldsymbol{x} \in \mathbb{R}^n$, both sets $V(\boldsymbol{x}, \alpha, \lambda)$ and $\overline{V}(\boldsymbol{x}, \alpha)$ are bounded for any $\alpha,\ \lambda > 0$ and for any $\alpha > 0$, respectively.

To prove the closedness of the set $V(\boldsymbol{x}, \alpha, \lambda)$, we take any sequence $\boldsymbol{w}_k \in V(\boldsymbol{x}, \alpha, \lambda)$ and suppose that $\boldsymbol{w}_k \to \boldsymbol{w}$ as $k \to \infty$. Since $\boldsymbol{w}_k \in V(\boldsymbol{x}, \alpha, \lambda)$ we have $\boldsymbol{w}_k = \Gamma^{i_k}(\boldsymbol{x}, \boldsymbol{g}_k, \boldsymbol{e}, \alpha, \lambda)$ for some $\boldsymbol{g}_k \in S_1$ and $i_k \in \mathcal{I}(\boldsymbol{g}_k)$. The set S_1 is compact and therefore, the sequence $\{\boldsymbol{g}_k\}$ has at least one limit point. Assume without loss of generality that $\boldsymbol{g}_k \to \bar{\boldsymbol{g}}$ as $k \to \infty$. It is clear that $\bar{\boldsymbol{g}} \in S_1$. Since the set G is finite we can assume that the vector $\boldsymbol{e}$ is the same for all k. In addition, there exists $k_0 > 0$ such that $\mathcal{I}(\boldsymbol{g}_k) \subseteq \mathcal{I}(\boldsymbol{g})$ for all $k > k_0$. This together with the continuity of the function f imply that $\boldsymbol{w}_k = \boldsymbol{\Gamma}^{i_k}(\boldsymbol{x}, \boldsymbol{g}_k, \boldsymbol{e}, \alpha, \lambda) \to \boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \alpha, \lambda) = \boldsymbol{w}$ for some $i \in \mathcal{I}(\boldsymbol{g})$ and therefore, $\boldsymbol{w} \in V(\boldsymbol{x}, \alpha, \lambda)$. □

Next we show the relationships between the subdifferential of the function f and the sets $V(\boldsymbol{x}, \alpha, \lambda)$ and $\overline{V}(\boldsymbol{x}, \alpha)$.

Continuously Differentiable Functions We start by considering continuously differentiable, possibly nonconvex, functions.

Proposition 18.7 *Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable in some neighborhood of a point $\boldsymbol{x} \in \mathbb{R}^n$. Then*

$$\overline{V}(\boldsymbol{x}, \alpha) = \{\nabla f(\boldsymbol{x})\}$$

for any $\alpha > 0$.

Proof The proof follows from the definition of the discrete gradients, continuous differentiability of the function f and the mean value theorem. □

For any convex function, not necessarily differentiable, we have the following results.

Proposition 18.8 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a finite valued convex function. Then*

$$\partial f(\boldsymbol{x}) \subseteq \operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda)$$

for any $\alpha \in (0, 1]$ and $\lambda > 0$.

Proof The convexity of f implies that this function is directionally differentiable on $\mathbb{R}^n$ and we have

$$f'(\boldsymbol{x}; \boldsymbol{g}) \leq \frac{f(\boldsymbol{x} + \lambda \boldsymbol{g}) - f(\boldsymbol{x})}{\lambda}$$

for all $\lambda > 0$. From (18.7) and (18.12), for any $\boldsymbol{g} \in S_1$ and $i \in \mathcal{I}(\boldsymbol{g})$ we get

$$\begin{aligned}
\max_{\xi \in \partial f(\boldsymbol{x})} \langle \boldsymbol{\xi}, \boldsymbol{g} \rangle &= f'(\boldsymbol{x}; \boldsymbol{g}) \\
&\leq \frac{f(\boldsymbol{x} + \lambda \boldsymbol{g}) - f(\boldsymbol{x})}{\lambda} \\
&= \langle \boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \lambda, \alpha), \boldsymbol{g} \rangle \\
&\leq \max\{\langle \boldsymbol{w}, \boldsymbol{g} \rangle : \ \boldsymbol{w} \in \operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda)\}.
\end{aligned}$$

This means that

$$\max\{\langle \boldsymbol{\xi}, \boldsymbol{g} \rangle : \ \boldsymbol{\xi} \in \partial f(\boldsymbol{x})\} \leq \max\{\langle \boldsymbol{w}, \boldsymbol{g} \rangle : \ \boldsymbol{w} \in \operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda)\}$$

for all $\boldsymbol{g} \in \mathbb{R}^n$. Then the proof follows from the compactness and convexity of the sets $\partial f(\boldsymbol{x})$ and $\operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda)$. □

Corollary 18.4 *Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a finite valued convex function. Then for any $\alpha \in (0, 1]$ we have*

$$\partial f(\boldsymbol{x}) \subseteq \operatorname{cl} \operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha).$$

Proof The proof follows from Proposition 18.8 and the definition of the set $\overline{V}(\boldsymbol{x}, \alpha)$. □

Convex Polyhedral Functions Next we consider convex polyhedral functions defined as

$$f(\boldsymbol{x}) = \max_{j \in \mathcal{J}} \{\langle \boldsymbol{a}_j, \boldsymbol{x} \rangle + b_j\}. \tag{18.13}$$

Here $\mathcal{J}$ is a finite index set and $\boldsymbol{a}_j \in \mathbb{R}^n,\ b_j \in \mathbb{R},\ j \in \mathcal{J}$. The subdifferential of this function at a point $\boldsymbol{x} \in \mathbb{R}^n$ is

$$\partial f(\boldsymbol{x}) = \operatorname{conv}\{\boldsymbol{a}_j,\ j \in \mathcal{J}(\boldsymbol{x})\},$$

where

$$\mathcal{J}(\boldsymbol{x}) = \left\{ j \in \mathcal{J} :\ \langle \boldsymbol{a}_j, \boldsymbol{x} \rangle + b_j = f(\boldsymbol{x}) \right\}.$$

Upper semicontinuity of the subdifferential mapping and the finiteness of the set $\mathcal{J}$ imply that for the function f there exists $\delta > 0$ such that

$$\partial f(\boldsymbol{y}) \subseteq \partial f(\boldsymbol{x}) \tag{18.14}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in B(\boldsymbol{x}; \delta)$.

Proposition 18.9 *For the convex polyhedral function f there exist $\alpha_0 > 0$ and $\lambda_0 > 0$ such that*

$$\partial f(\boldsymbol{x}) = \operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda)$$

for all $\boldsymbol{x} \in \mathbb{R}^n,\ \alpha \in (0, \alpha_0]$ and $\lambda \in (0, \lambda_0)$.

Proof The inclusion $\partial f(\boldsymbol{x}) \subseteq \operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda)$ follows from Proposition 18.8. Therefore, we only prove the opposite inclusion. Take any $\boldsymbol{g} \in S_1$. Then according to (18.14) we have that $\partial f(\boldsymbol{x} + \lambda \boldsymbol{g}) \subseteq \partial f(\boldsymbol{x})$ for all $\lambda \in (0, \lambda_0)$, where $\lambda_0 = \delta$. For the given direction $\boldsymbol{g} \in S_1$ consider the set

$$\mathcal{J}(\boldsymbol{x}, \boldsymbol{g}) = \left\{ j \in \mathcal{J}(\boldsymbol{x}) :\ \langle \boldsymbol{a}_j, \boldsymbol{g} \rangle = f'(\boldsymbol{x}; \boldsymbol{g}) \right\}.$$

Then $\partial f(\boldsymbol{x} + \lambda \boldsymbol{g}) \subseteq \operatorname{conv}\{\boldsymbol{a}_j :\ j \in \mathcal{J}(\boldsymbol{x}, \boldsymbol{g})\}$ for all $\lambda \in (0, \lambda_0)$.

Construct directions $\boldsymbol{e}_j(\alpha),\ j = 1, \ldots, n$ by applying (18.2). Then compute a vector $\boldsymbol{v}(\alpha) = (v_1, \ldots, v_n)$ at the point $\boldsymbol{x} + \lambda \boldsymbol{g}$ using (18.10). According to

Proposition 18.4 for $\alpha_0 > 0$, defined in (18.4), we have $\boldsymbol{v}(\alpha) \in \partial f(\boldsymbol{x} + \lambda \boldsymbol{g})$ for all $\alpha \in (0, \alpha_0]$. Take any $i \in \mathcal{I}(\boldsymbol{g})$ and apply Definition 18.1 to calculate the discrete gradient $\boldsymbol{\Gamma}^i$ at the point $\boldsymbol{x} \in \mathbb{R}^n$ with respect to the direction $\boldsymbol{g} \in S_1$. It is clear that $\Gamma_j^i = v_j(\alpha),\ j = 1, \ldots, n,\ j \neq i$. To compute the i-th coordinate Γ_i^i, note that for the function f, defined in (18.13), we have

$$f(\boldsymbol{x} + \lambda \boldsymbol{g}) - f(\boldsymbol{x}) = \lambda \langle \boldsymbol{\xi}, \boldsymbol{g} \rangle$$

for all $\boldsymbol{\xi} \in \partial f(\boldsymbol{x} + \lambda \boldsymbol{g})$ and $\lambda \in (0, \lambda_0)$, and therefore, using the subgradient $\boldsymbol{v}(\alpha)$ we get

$$\Gamma_i^i = (\lambda g_i)^{-1} \left(\lambda \langle \boldsymbol{v}(\alpha), \boldsymbol{g} \rangle - \lambda \sum_{j=1, j \neq i}^{n} v_j(\alpha) g_j \right) = v_i(\alpha).$$

Then it follows from (18.14) that $\boldsymbol{v}(\alpha) \in \partial f(\boldsymbol{x})$. This completes the proof. □

Corollary 18.5 *For the convex polyhedral function f there exists $\alpha_0 > 0$ such that*

$$\partial f(\boldsymbol{x}) = \operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha)$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\alpha \in (0, \alpha_0]$.

Convex Functions We say that a convex function $f : \mathbb{R}^n \to \mathbb{R}$ belongs to the class $\mathcal{F}_0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ if for some $\varepsilon > 0$ there exists a polyhedral function $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + \varphi(\boldsymbol{y} - \boldsymbol{x}) + o(\|\boldsymbol{y} - \boldsymbol{x}\|), \quad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \varepsilon)$$

and $\partial \varphi(\boldsymbol{x}) = \partial f(\boldsymbol{x})$. Here

$$\varphi(\boldsymbol{x}) = \max_{j \in \mathcal{J}} \langle \boldsymbol{a}_j, \boldsymbol{x} \rangle,$$

and $\mathcal{J}$ is a finite set of indices. If the function $f \in \mathcal{F}_0$ at $\boldsymbol{x} \in \mathbb{R}^n$, then for any $\boldsymbol{g} \in S_1$ we have

$$f(\boldsymbol{x} + \lambda \boldsymbol{g}) = f(\boldsymbol{x}) + \lambda \varphi(\boldsymbol{g}) + o(\lambda) \tag{18.15}$$

where $\lambda^{-1} o(\lambda) \to 0$ as $\lambda \downarrow 0$.

Remark 18.1 One class of nonsmooth functions that belongs to $\mathcal{F}_0$ is the function $f : \mathbb{R}^n \to \mathbb{R}$ given as

$$f(\boldsymbol{x}) = \max_{j \in \mathcal{J}} f_j(\boldsymbol{x}),$$

where the functions $f_j,\ j \in \mathcal{J}$ are convex and smooth.

Proposition 18.10 *Let $f \in \mathcal{F}_0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$. Then there exists $\alpha_0 > 0$ such that*

$$\partial f(\boldsymbol{x}) = \operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha)$$

for all $\alpha \in (0, \alpha_0)$.

Proof The inclusion $\partial f(\boldsymbol{x}) \subseteq \operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha)$ follows from Corollary 18.5 and therefore, we only show that $\operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha) \subseteq \partial f(\boldsymbol{x})$. Take any $\boldsymbol{w} \in \overline{V}(\boldsymbol{x}, \alpha)$. Then for some $\boldsymbol{g} \in S_1,\ i \in \mathcal{I}(\boldsymbol{g})$ and $\boldsymbol{e} \in G$ we have

$$\boldsymbol{w} = \lim_{k \to \infty} \boldsymbol{\Gamma}^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, \alpha, \lambda_k)$$

where $\lambda_k \downarrow 0$ as $k \to \infty$. Applying Definition 18.1 and considering (18.15), the j-th coordinate Γ^i_{jf} for the function $f,\ j \in \{1, \ldots, n\},\ j \neq i$ is

$$\begin{aligned}
\Gamma^i_{jf} &= \frac{f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{j-1})}{\lambda \alpha^j \boldsymbol{e}_j} \\
&= \frac{f(\boldsymbol{x}) + \varphi(\boldsymbol{x}_j - \boldsymbol{x}) + o(\lambda) - [f(\boldsymbol{x}) + \varphi(\boldsymbol{x}_{j-1} - \boldsymbol{x}) + o(\lambda)]}{\lambda \alpha^j \boldsymbol{e}_j} \\
&= \frac{\varphi(\boldsymbol{x}_j - \boldsymbol{x}) - \varphi(\boldsymbol{x}_{j-1} - \boldsymbol{x}) + o(\lambda)}{\lambda \alpha^j \boldsymbol{e}_j} \\
&= \frac{\varphi(\boldsymbol{x}_j) - \varphi(\boldsymbol{x}_{j-1}) + o(\lambda)}{\lambda \alpha^j \boldsymbol{e}_j} \\
&= \Gamma^i_{j\varphi} + \frac{o(\lambda)}{\lambda \alpha^j \boldsymbol{e}_j}.
\end{aligned}$$

This means that $\lim_{\lambda \downarrow 0} \Gamma^i_{jf} = \lim_{\lambda \downarrow 0} \Gamma^i_{j\varphi},\ j = 1, \ldots, n,\ j \neq i$. In addition, according to Proposition 18.11 there exist $\alpha_0 > 0$ and $\lambda_0 > 0$ such that for some $\boldsymbol{v} \in \partial \varphi(\boldsymbol{x})$ we have $\boldsymbol{\Gamma}^i_{\varphi} = \boldsymbol{v}$ for all $\alpha \in (0, \alpha_0)$ and $\lambda \in (0, \lambda_0)$. Since $f \in \mathcal{F}_0$ at $\boldsymbol{x}$ we have $\boldsymbol{v} \in \partial f(\boldsymbol{x})$, and therefore, we get

$$\lim_{\lambda \downarrow 0} \Gamma^i_{jf} = v_j,\ j = 1, \ldots, n,\ j \neq i.$$

Since the function f is convex we have $f'(\boldsymbol{x}; \boldsymbol{g}) = \langle \boldsymbol{v}, \boldsymbol{g} \rangle$. Then it follows from Definition 18.1 and (18.12) that $\lim_{\lambda \downarrow 0} \Gamma^i_{if} = v_i$, and we have $\boldsymbol{w} = \boldsymbol{v} \in \partial f(\boldsymbol{x})$. This completes the proof. □

Corollary 18.6 *Let the function $f \in \mathcal{F}_0$ at $\boldsymbol{x} \in \mathbb{R}^n$. Then for any $\varepsilon > 0$ there exist $\lambda_0 > 0$ and $\alpha_0 > 0$ such that*

$$\operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda) \subset \partial f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)$$

for all $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$.

Proof According to the definition of the set $\overline{V}(\boldsymbol{x}, \alpha)$, for any $\varepsilon > 0$ there exists $\lambda_0 > 0$ such that

$$V(\boldsymbol{x}, \alpha, \lambda) \subseteq \overline{V}(\boldsymbol{x}, \alpha) \text{ for all } \lambda \in (0, \lambda_0).$$

Then the proof follows from Proposition 18.10. □

Consider the set

$$\tilde{V}(\boldsymbol{x}) = \bigcap_{\alpha > 0} \operatorname{cl} \operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha).$$

It is obvious that the set $\tilde{V}(\boldsymbol{x})$ is convex and compact at any $\boldsymbol{x} \in \mathbb{R}^n$.

Corollary 18.7 *Assume that $f \in \mathcal{F}_0$ at $\boldsymbol{x} \in \mathbb{R}^n$. Then $\tilde{V}(\boldsymbol{x}) = \partial f(\boldsymbol{x})$.*

Proof The proof follows from Corollary 18.4 and Proposition 18.10. □

Difference of Convex Polyhedral Functions Consider the function

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}), \ \boldsymbol{x} \in \mathbb{R}^n, \tag{18.16}$$

where

$$f_1(\boldsymbol{x}) = \max_{i \in I} \{\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle + b_i\}, \quad f_2(\boldsymbol{x}) = \max_{k \in K} \{\langle \boldsymbol{c}_k, \boldsymbol{x} \rangle + d_k\}.$$

Here I and K are finite index sets, $\boldsymbol{a}_i, \boldsymbol{c}_k \in \mathbb{R}^n$, $b_i, d_k \in \mathbb{R}$, $i \in I$, $k \in K$. The subdifferentials of the functions f_1 and f_2 at a point $\boldsymbol{x} \in \mathbb{R}^n$ are

$$\partial f_1(\boldsymbol{x}) = \operatorname{conv} \{\boldsymbol{a}_i, \ i \in I(\boldsymbol{x})\}, \quad \partial f_2(\boldsymbol{x}) = \operatorname{conv} \{\boldsymbol{c}_k, \ k \in K(\boldsymbol{x})\},$$

where

$$I(\boldsymbol{x}) = \{i \in I : \ \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle + b_i = f_1(\boldsymbol{x})\} \text{ and}$$
$$K(\boldsymbol{x}) = \{k \in K : \ \langle \boldsymbol{c}_k, \boldsymbol{x} \rangle + d_k = f_2(\boldsymbol{x})\}.$$

Applying upper semicontinuity of the subdifferential mappings and taking into account the finiteness of the index sets I and K we get that there exists $\delta > 0$

such that

$$\partial f_p(\boldsymbol{y}) \subseteq \partial f_p(\boldsymbol{x}),\ \ p = 1, 2 \tag{18.17}$$

for all $\boldsymbol{y} \in B(\boldsymbol{x}; \delta)$.

Proposition 18.11 *For the difference of polyhedral functions f there exist $\alpha_0 > 0$ and $\lambda_0 > 0$ such that*

$$\operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda) \subseteq \partial f(\boldsymbol{x})$$

for all $\boldsymbol{x} \in \mathbb{R}^n,\ \alpha \in (0, \alpha_0]$ and $\lambda \in (0, \lambda_0)$.

Proof For a given $\boldsymbol{g} \in S_1$ introduce the following sets:

$$\bar{I}(\boldsymbol{x}, \boldsymbol{g}) = \left\{i \in I(\boldsymbol{x}) :\ f_1'(\boldsymbol{x}; \boldsymbol{g}) = \langle \boldsymbol{a}_i, \boldsymbol{g}\rangle\right\} \text{ and}$$
$$\bar{K}(\boldsymbol{x}, \boldsymbol{g}) = \left\{k \in K(\boldsymbol{x}) :\ f_2'(\boldsymbol{x}; \boldsymbol{g}) = \langle \boldsymbol{c}_k, \boldsymbol{g}\rangle\right\}.$$

Since both functions f_1 and f_2 are convex polyhedral it follows that at the point $\boldsymbol{x} \in \mathbb{R}^n$ there exists $\lambda_0 > 0$ such that $I(\boldsymbol{x} + \lambda \boldsymbol{g}) = \bar{I}(\boldsymbol{x}, \boldsymbol{g})$ and $K(\boldsymbol{x} + \lambda \boldsymbol{g}) = \bar{K}(\boldsymbol{x}, \boldsymbol{g})$ for all $\lambda \in (0, \lambda_0)$. This means that

$$\partial f_1(\boldsymbol{x} + \lambda \boldsymbol{g}) = \operatorname{conv}\left\{\boldsymbol{a}_i :\ i \in \bar{I}(\boldsymbol{x}, \boldsymbol{g})\right\} \text{ and}$$
$$\partial f_2(\boldsymbol{x} + \lambda \boldsymbol{g}) = \operatorname{conv}\left\{\boldsymbol{c}_k :\ k \in \bar{K}(\boldsymbol{x}, \boldsymbol{g})\right\}$$

for all $\lambda \in (0, \lambda_0)$.

Let $\lambda \in (0, \lambda_0)$. Take any $\alpha > 0$, construct the vectors $\boldsymbol{e}^j(\alpha)$ by applying (18.2) and then using (18.10) compute vectors $\boldsymbol{v}_1(\alpha) = (v_{11}, \ldots, v_{1n})$ and $\boldsymbol{v}_2(\alpha) = (v_{21}, \ldots, v_{2n})$ at the point $\boldsymbol{x} + \lambda \boldsymbol{g}$ for functions f_1 and f_2, respectively. Define

$$\boldsymbol{v}(\alpha) = \boldsymbol{v}_1(\alpha) - \boldsymbol{v}_2(\alpha).$$

Proposition 18.4 and the fact that functions f_1 and f_2 are convex polyhedral imply that there exists $\alpha_0 > 0$ such that $\boldsymbol{v}_1(\alpha) \in \partial f_1(\boldsymbol{x} + \lambda \boldsymbol{g})$ and $\boldsymbol{v}_2(\alpha) \in \partial f_2(\boldsymbol{x} + \lambda \boldsymbol{g})$ for any $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$.

Polyhedral representations of the functions f_1 and f_2 and the construction of the R-sets for sets $\{\boldsymbol{a}_i,\ i \in \bar{I}(\boldsymbol{x}, \boldsymbol{g})\}$ and $\{\boldsymbol{c}_k,\ k \in \bar{K}(\boldsymbol{x}, \boldsymbol{g})\}$ imply that subdifferentials $\partial f_1(\boldsymbol{x}^n),\ \partial f_2(\boldsymbol{x}^n)$ are singletons for any $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$. Let $\boldsymbol{u}_1 \in \partial f_1(\boldsymbol{x}^n)$ and $\boldsymbol{u}_2 \in \partial f_2(\boldsymbol{x}^n)$. From the definition of R-sets we have $\boldsymbol{v}_1(\alpha) = \boldsymbol{u}_1$ and $\boldsymbol{v}_2(\alpha) = \boldsymbol{u}_2$. Furthermore, the subdifferential calculus implies that $\partial f(\boldsymbol{x}^n) = \{\boldsymbol{u}_1 - \boldsymbol{u}_2\}$. Then it follows from (18.17) that $\boldsymbol{u}_1 - \boldsymbol{u}_2 \in \partial f(\boldsymbol{x})$.

Take any $i \in \mathcal{I}(\boldsymbol{g})$ and apply Definition 18.1 to calculate the discrete gradient $\boldsymbol{\Gamma}^i$ at the point $\boldsymbol{x} \in \mathbb{R}^n$ with respect to the direction $\boldsymbol{g} \in S_1$. It is clear that $\Gamma_j^i = v_j(\alpha),\ j = 1, \ldots, n,\ j \neq i$. To compute the i-th coordinate Γ_i^i, note that for the

function f, defined in (18.16), we have

$$f(\boldsymbol{x}+\lambda \boldsymbol{g})-f(\boldsymbol{x})=\lambda\langle \boldsymbol{\xi}, \boldsymbol{g}\rangle$$

for all $\boldsymbol{\xi} \in \partial f(\boldsymbol{x}+\lambda \boldsymbol{g})$ and $\lambda \in (0, \lambda_0)$, and therefore, using the subgradient $\boldsymbol{v}(\alpha)$ we get

$$\Gamma_i^i = (\lambda g_i)^{-1}\left(\lambda\langle \boldsymbol{v}(\alpha), \boldsymbol{g}\rangle - \lambda \sum_{j=1, j\neq i}^{n} v_j(\alpha) g_j\right) = v_i(\alpha).$$

Then it follows from (18.14) that $\boldsymbol{v}(\alpha) \in \partial f(\boldsymbol{x})$. This completes the proof. □

Corollary 18.8 *For the function f, defined in (18.16), there exists $\alpha_0 > 0$ such that*

$$\operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha) \subseteq \partial f(\boldsymbol{x})$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\alpha \in (0, \alpha_0)$.

Difference of Convex Functions Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a DC function, defined in (18.9) with f_1 and f_2 being finite valued convex functions.

Proposition 18.12 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a DC function, defined in (18.9). Assume that $f_1, f_2 \in \mathcal{F}_0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$. Then there exists $\alpha_0 > 0$ such that*

$$\operatorname{conv} \overline{V}(\boldsymbol{x}, \alpha) \subseteq \partial f(\boldsymbol{x})$$

for all $\alpha \in (0, \alpha_0)$.

Proof The proof repeats that of Proposition 18.10 and therefore is omitted. □

Corollary 18.9 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a DC function, defined in (18.9), and $f_1, f_2 \in \mathcal{F}_0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$. Then for any $\varepsilon > 0$ there exist $\lambda_0 > 0$ and $\alpha_0 > 0$ such that*

$$V(\boldsymbol{x}, \lambda, \alpha) \subset \partial f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)$$

for all $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$.

Corollary 18.10 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a DC function, defined in (18.9), and $f_1, f_2 \in \mathcal{F}_0$ at a point $\boldsymbol{x} \in \mathbb{R}^n$. Then*

$$\tilde{V}(\boldsymbol{x}) \subseteq \partial f(\boldsymbol{x}). \tag{18.18}$$

In this section we showed that the discrete gradients can be used to approximate subdifferentials of convex and DC functions. We say that a function f belongs to the class $\mathcal{F}$ if it satisfies the inclusion (18.18) for any $\boldsymbol{x} \in \mathbb{R}^n$. This class contains, in particular, differentiable, nonsmooth convex and nonsmooth DC functions.

Proposition 18.13 *Let the function $f \in \mathcal{F}$. Then at a point $\boldsymbol{x} \in \mathbb{R}^n$ for any $\varepsilon > 0$ there exist $\lambda_0 > 0$ and $\alpha_0 > 0$ such that*

$$\operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda) \subset \partial f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)$$

for all $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$.

Proof According to the definition of the set $\tilde{V}(\boldsymbol{x})$, at the point $\boldsymbol{x} \in \mathbb{R}^n$ for any $\varepsilon > 0$ there exist $\lambda_0 > 0$ and $\alpha_0 > 0$ such that

$$\operatorname{conv} V(\boldsymbol{x}, \alpha, \lambda) \subset \tilde{V}(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)$$

for all $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$. Then the rest of the proof follows from the fact that for the function $f \in \mathcal{F}$ the inclusion (18.18) is satisfied. □

18.5 Discrete Gradient Method

The idea of the original DGM introduced in [7] is to hybridize derivative free methods with bundle methods. In contrast with bundle methods, which require the computation of a subgradient of the objective function at each trial point, the DGM approximates subgradients by the discrete gradients using only values of the objective function. Similar to bundle methods, the previous values of the discrete gradients are gathered into a bundle and a null step is used if the current search direction is not good enough.

The DGM consists of inner and outer iterations. The inner iteration has serious and null steps. We select sequences $\{\delta_k\}$ and $\{\lambda_k\}$ such that $\delta_k,\ \lambda_k \downarrow 0$ as $k \to \infty$. The outer iteration depends on the index k and in this iteration parameters δ_k and λ_k are updated. The inner iteration depends on the index s. In the inner iteration we compute the search direction and either update the solution or add an element into the set of the discrete gradients $\overline{V}(\boldsymbol{x}_{k_s})$. In other words, we either take a serious step or a null step occurs. At the beginning of each inner iteration (i.e. $s = 1$) we first compute the discrete gradient $\boldsymbol{v}_{k_1} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_1}, \boldsymbol{g}, \boldsymbol{e}, \lambda_k, \alpha)$ with respect to any initial direction $\boldsymbol{g} \in S_1$ and any fixed $\boldsymbol{e} \in G$. We set the initial bundle of the discrete gradients $\overline{V}(\boldsymbol{x}_{k_1}) = \{\boldsymbol{v}_{k_1}\}$, $\bar{\boldsymbol{v}}_{k_1} = \boldsymbol{v}_{k_1}$ and find the search direction $\boldsymbol{d}_{k_1}$

$$\boldsymbol{d}_{k_1} = -\frac{\bar{\boldsymbol{v}}_{k_1}}{\|\bar{\boldsymbol{v}}_{k_1}\|}.$$

In the next inner iterations (i.e. $s > 1$) we compute the vector

$$\bar{\boldsymbol{v}}_{k_s} = \operatorname*{argmin}_{\boldsymbol{v} \in \overline{V}(\boldsymbol{x}_{k_s})} \|\boldsymbol{v}\|^2,$$

that is the distance between the convex hull of all computed the discrete gradients and the origin. Then we find the search direction

$$\boldsymbol{d}_{k_s} = -\frac{\bar{\boldsymbol{v}}_{k_s}}{\|\bar{\boldsymbol{v}}_{k_s}\|}.$$

Next, we check whether this direction $\boldsymbol{d}_{k_s}$ $(s \geq 1)$ is descent or not. If it is, then we have

$$f(\boldsymbol{x}_{k_s} + \lambda_k \boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) \leq -\epsilon_L \lambda_k \|\bar{\boldsymbol{v}}_{k_s}\|,$$

with the given numbers $\epsilon_L \in (0, 1)$ and $\lambda_k > 0$. In this case we compute the next (inner) iteration point

$$\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s} + t_{k_s} \boldsymbol{d}_{k_s},$$

where the step size t_{k_s} is defined as

$$t_{k_s} = \operatorname{argmax} \left\{ t \geq 0 : f(\boldsymbol{x}_{k_s} + t\boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) \leq -\epsilon_R t \|\bar{\boldsymbol{v}}_{k_s}\| \right\},$$

with the given $\epsilon_R \in (0, \epsilon_L]$.

In the case of a serious step, we compute the new discrete gradient $\boldsymbol{v}_{k_{s+1}} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_{s+1}}, \boldsymbol{g}, \boldsymbol{e}, \lambda_k, \alpha)$ with respect to any direction $\boldsymbol{g} \in S_1$, set $\overline{V}(\boldsymbol{x}_{k_{s+1}}) = \{\boldsymbol{v}_{k_{s+1}}\}$, and continue to the next inner iteration with $s = s + 1$. Otherwise, a null step occurs and we compute another discrete gradient $\boldsymbol{v}_{k_{s+1}} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_s}, \boldsymbol{d}_{k_s}, \boldsymbol{e}, \lambda_k, \alpha)$ in the direction $\boldsymbol{d}_{k_s}$, update the bundle of the discrete gradients

$$\overline{V}(\boldsymbol{x}_{k_{s+1}}) = \operatorname{conv}\{\overline{V}(\boldsymbol{x}_{k_s}) \cup \{\boldsymbol{v}_{k_{s+1}}\}\},$$

set $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s}$, and continue the inner iterations with $s = s + 1$. Note that, at each null step the approximation of the subdifferential $\partial f(\boldsymbol{x})$ is improved.

The inner iteration stops if $\|\bar{\boldsymbol{v}}_{k_s}\| \leq \delta_k$, that is for given values of δ_k and λ_k the last iteration $\boldsymbol{x}_{k_s}$ can be considered as an approximate solution to the problem (18.1) and this solution cannot be significantly improved using the same values of parameters. Therefore, this point is accepted as a new iteration $\boldsymbol{x}_{k+1}$ and the algorithm returns to the outer iteration to update the values of parameters δ_k and λ_k. In its turn, the outer iteration has one stopping criterion. It stops if $\delta_k < \varepsilon$ and $\lambda_k < \varepsilon$ with a given tolerance $\varepsilon > 0$. This means that the further decrease of values of δ_k and λ_k will not improve the approximation of the subdifferential and the solution $\boldsymbol{x}_k$.

Algorithm 18.1: DGM

Data: $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\varepsilon > 0$, $\{\lambda_k\}$, $\{\delta_k\}$, $\lambda_k, \delta_k > 0$, $\lambda_k, \delta_k \downarrow 0$ as $k \to \infty$, $\alpha \in (0, 1]$, $\epsilon_L \in (0, 1]$ and $\epsilon_R \in (0, \epsilon_L]$.

Step 1. *(Outer iteration initialization)* Set $k = 1$.

Step 2. *(Inner iteration initialization)* Set $s = 1$ and $\boldsymbol{x}_{k_s} = \boldsymbol{x}_k$. Choose any $\boldsymbol{g} \in S_1$, $\boldsymbol{e} \in G$. Compute the discrete gradient

$$\boldsymbol{v}_{k_s} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_s}, \boldsymbol{g}, \boldsymbol{e}, \lambda_k, \alpha).$$

Set $\overline{V}(\boldsymbol{x}_{k_s}) = \{\boldsymbol{v}_{k_s}\}$.

Step 3. *(Stopping criterion)* If $\lambda_k < \varepsilon$ and $\delta_k < \varepsilon$, then **stop** with $\boldsymbol{x}_k$ as a final solution.

Step 4. *(Minimum norm)* Compute the vector

$$\bar{\boldsymbol{v}}_{k_s} = \operatorname*{argmin}_{\boldsymbol{v} \in \overline{V}(\boldsymbol{x}_{k_s})} \|\boldsymbol{v}\|^2.$$

Step 5. *(Inner iteration termination)* If $\|\bar{\boldsymbol{v}}_{k_s}\| \le \delta_k$, then update λ_{k+1} and δ_{k+1}. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k_s}$, $k = k + 1$ and go to Step 2.

Step 6. *(Search direction)* Compute the search direction

$$\boldsymbol{d}_{k_s} = -\frac{\bar{\boldsymbol{v}}_{k_s}}{\|\bar{\boldsymbol{v}}_{k_s}\|}.$$

Step 7. If $f(\boldsymbol{x}_{k_s} + \lambda_k \boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) > -\epsilon_L \lambda_k \|\bar{\boldsymbol{v}}_{k_s}\|$, then go to Step 9.

Step 8. *(Serious step)* Construct $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s} + t_{k_s} \boldsymbol{d}_{k_s}$, where the step size t_{k_s} is computed as

$$t_{k_s} = \operatorname{argmax}\left\{t \ge 0 : f(\boldsymbol{x}_{k_s} + t\boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) \le -t\epsilon_R \|\bar{\boldsymbol{v}}_{k_s}\|\right\}.$$

Compute a new discrete gradient $\boldsymbol{v}_{k_{s+1}}$ using $\boldsymbol{x}_{k_{s+1}}$ and any $\boldsymbol{g} \in S_1$:

$$\boldsymbol{v}_{k_{s+1}} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_{s+1}}, \boldsymbol{g}, \boldsymbol{e}, \lambda_k, \alpha).$$

Set $\overline{V}(\boldsymbol{x}_{k_{s+1}}) = \{\boldsymbol{v}_{k_{s+1}}\}$, $s = s + 1$, and go to Step 4.

Step 9. *(Null step)* Compute a new discrete gradient $\boldsymbol{v}_{k_{s+1}}$ using $\boldsymbol{x}_{k_s}$ and $\boldsymbol{d}_{k_s}$:

$$\boldsymbol{v}_{k_{s+1}} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_s}, \boldsymbol{d}_{k_s}, \boldsymbol{e}, \lambda_k, \alpha).$$

Update the set

$$\overline{V}(\boldsymbol{x}_{k_{s+1}}) = \operatorname{conv}\{\overline{V}(\boldsymbol{x}_{k_s}) \cup \{\boldsymbol{v}_{k_{s+1}}\}\}. \tag{18.19}$$

Set $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s}$, $s = s + 1$, and go to Step 4.

Convergence of the DGM Next we prove that for given δ_k and λ_k the number of serious and null steps in the inner iterations are finite. We start with the null step.

Proposition 18.14 *Let the objective function $f : \mathbb{R}^n \to \mathbb{R}$ be LLC and $L > 0$ be its Lipschitz constant. For any $\delta_k \in (0, \bar{C})$ the number s_0 of null steps in an inner loop of the* DGM *is finite, where*

$$s_0 \leq 1 + 2\left\lceil \left(\frac{\log_2(\delta_k/\bar{C})}{\log_2 r} \right) \right\rceil, \tag{18.20}$$

and $r = 1 - \left((1-\epsilon_L)(2\bar{C})^{-1}\delta_k\right)^2$, $\bar{C} = C(n)L$ with $C(n)$ given in Proposition 18.5. Here $\lceil \cdot \rceil$ is ceiling of a number.

Proof It is clear that the null step is called when $\|\bar{\boldsymbol{v}}_{k_s}\| > \delta_k$ and $f(\boldsymbol{x}_{k_s} + \lambda_k \boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) > -\epsilon_L \lambda_k \|\bar{\boldsymbol{v}}_{k_s}\|$. Then a new discrete gradient $\boldsymbol{v}_{k_{s+1}}$ computed in the null step does not belong to the set $\overline{V}_s(\boldsymbol{x}_{k_s})$. Indeed, in this case it follows from (18.12) that

$$\begin{aligned} f(\boldsymbol{x}_{k_s} + \lambda_k \boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) &= \lambda_k \langle \boldsymbol{\Gamma}^i(\boldsymbol{x}_{k_s}, \boldsymbol{d}_{k_s}, \boldsymbol{e}, \lambda_k, \alpha), \boldsymbol{d}_{k_s} \rangle \\ &= \lambda_k \langle \boldsymbol{v}_{k_{s+1}}, \boldsymbol{d}_{k_s} \rangle, \end{aligned}$$

and therefore, $\lambda_k \langle \boldsymbol{v}_{k_{s+1}}, \boldsymbol{d}_{k_s} \rangle > -\epsilon_L \lambda_k \|\bar{\boldsymbol{v}}_{k_s}\|$. Then we get

$$\langle \boldsymbol{v}_{k_{s+1}}, \bar{\boldsymbol{v}}_{k_s} \rangle < \epsilon_L \|\bar{\boldsymbol{v}}_{k_s}\|^2. \tag{18.21}$$

On the other hand, since $\bar{\boldsymbol{v}}_{k_s} = \operatorname{argmin}_{\boldsymbol{v} \in \overline{V}(\boldsymbol{x}_{k_s})} \|\boldsymbol{v}\|^2$ it follows from the necessary condition for a minimum that $\langle \bar{\boldsymbol{v}}_{k_s}, \boldsymbol{v} - \bar{\boldsymbol{v}}_{k_s} \rangle \geq 0$ for any $\boldsymbol{v} \in \overline{V}_s(\boldsymbol{x}_{k_s})$, or $\langle \bar{\boldsymbol{v}}_{k_s}, \boldsymbol{v} \rangle \geq \|\bar{\boldsymbol{v}}_{k_s}\|^2$. This together with (18.21) imply that $\boldsymbol{v}_{k_{s+1}} \notin \overline{V}_s(\boldsymbol{x}_{k_s})$.

In order to prove that the number s of null steps is finite for a given δ_k, we find an upper estimation s_0 for the number of the discrete gradients to achieve $\|\bar{\boldsymbol{v}}_{k_{s_0}}\| \leq \delta_k$. According to (18.19), we have $\|\bar{\boldsymbol{v}}_{k_{s+1}}\|^2 \leq \|t\boldsymbol{v}_{k_{s+1}} + (1-t)\bar{\boldsymbol{v}}_{k_s}\|^2$ for all $t \in [0, 1]$, and therefore, we get

$$\|\bar{\boldsymbol{v}}_{k_{s+1}}\|^2 \leq \|\bar{\boldsymbol{v}}_{k_s}\|^2 + 2t\langle \bar{\boldsymbol{v}}_{k_s}, \boldsymbol{v}_{k_{s+1}} - \bar{\boldsymbol{v}}_{k_s} \rangle + t^2 \|\boldsymbol{v}_{k_{s+1}} - \bar{\boldsymbol{v}}_{k_s}\|^2.$$

In addition, it follows from Proposition 18.5 that $\|\boldsymbol{v}_{k_{s+1}} - \bar{\boldsymbol{v}}_{k_s}\| \leq 2\bar{C}$. This together with (18.21) imply that

$$\|\bar{\boldsymbol{v}}_{k_{s+1}}\|^2 < \|\bar{\boldsymbol{v}}_{k_s}\|^2 - 2t(1-\epsilon_L)\|\bar{\boldsymbol{v}}_{k_s}\|^2 + 4t^2\bar{C}^2.$$

For $t = (1-\epsilon_L)(2\bar{C})^{-2}\|\bar{\boldsymbol{v}}_{k_s}\|^2 \in (0, 1)$ we get

$$\|\bar{\boldsymbol{v}}_{k_{s+1}}\|^2 < [1 - \left((1-\epsilon_L)(2\bar{C})^{-1}\|\bar{\boldsymbol{v}}_{k_s}\|\right)^2]\|\bar{\boldsymbol{v}}_{k_s}\|^2. \tag{18.22}$$

Let $\delta_k \in (0, \bar{C})$. It follows from (18.22) and the condition $\|\bar{\boldsymbol{v}}_{k_s}\| > \delta_k$, $s = 1, \ldots, s_0 - 1$ that $\|\bar{\boldsymbol{v}}_{k_{s+1}}\|^2 < [1 - \left((1-\epsilon_L)(2\bar{C})^{-1}\delta_k\right)^2]\|\bar{\boldsymbol{v}}_{k_s}\|^2$. Denote by $r = 1 - \left((1-\epsilon_L)(2\bar{C})^{-1}\delta_k\right)^2$. It is clear that $r \in (0, 1)$. Then we have

$$\|\bar{\boldsymbol{v}}_{k_{s_0}}\|^2 < r\|\bar{\boldsymbol{v}}_{k_{s_0-1}}\|^2 < \ldots < r^{s_0-1}\|\bar{\boldsymbol{v}}_{k_1}\|^2 < r^{s_0-1}\bar{C}^2,$$

where the last inequality comes from Proposition 18.5. If $r^{s_0-1}\bar{C}^2 \leq \delta_k^2$, then $\|\bar{\boldsymbol{v}}_{k_{s_0}}\| \leq \delta_k$ and we get the estimation (18.20) for the number of steps s_0. This completes the proof. □

In the next proposition we prove that for given δ_k and λ_k the number of serious steps in the inner iterations is finite.

Proposition 18.15 *Assume that the objective function f satisfies the condition $f^* > -\infty$, where $f^* = \min\{f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^n\}$. Then, for any $\delta_k, \lambda_k > 0$ the number $s_{\max}$ of serious steps in an inner loop of the* DGM *is finite, where*

$$s_{\max} \leq \left\lceil \frac{f(\boldsymbol{x}_k) - f^*}{\epsilon_L \lambda_k \delta_k} \right\rceil.$$

Proof At the beginning of the k-th inner iteration in the DGM, the value of the objective function f is $f(\boldsymbol{x}_k)$. Furthermore, at each serious step the condition

$$f(\boldsymbol{x}_{k_s} + \lambda_k \boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) \leq -\epsilon_L \lambda_k \|\bar{\boldsymbol{v}}_{k_s}\|,$$

is satisfied and $\|\bar{\boldsymbol{v}}_{k_s}\| > \delta_k$. Therefore, we have

$$\begin{aligned} f(\boldsymbol{x}_{k_{s+1}}) - f(\boldsymbol{x}_{k_s}) &\leq f(\boldsymbol{x}_{k_s} + \lambda_k \boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) \\ &\leq -\epsilon_L \lambda_k \|\bar{\boldsymbol{v}}_{k_s}\| \\ &< -\epsilon_L \lambda_k \delta_k. \end{aligned}$$

This means that at each iteration the value of the objective function is reduced by at least $u_k = \epsilon_L \lambda_k \delta_k$ and this number does not depend on the serious steps. Since $f^* > -\infty$ the number of the serious steps cannot be more than $\lceil (f(\boldsymbol{x}_k) - f^*)/u_k \rceil$. This completes the proof. □

Corollary 18.11 *Assume that conditions of Propositions 18.14 and 18.15 are satisfied. Then for any k the number of steps M_k in the k-th inner iteration of the* DGM *is finite and $M_k \leq s_0 s_{\max}$.*

In Proposition 18.13 we showed that for the function $f \in \mathcal{F}$ the closed convex set of the discrete gradients $V(\boldsymbol{x}, \alpha, \lambda)$ is an approximation of the subdifferential $\partial f(\boldsymbol{x})$ for $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$. However, this is true only at a given point $\boldsymbol{x} \in \mathbb{R}^n$. In order to get convergence results for the DGM we need a relationship between the set $V(\boldsymbol{x}, \alpha, \lambda)$ and $\partial f(\boldsymbol{x})$ also in some neighborhood of $\boldsymbol{x}$.

Therefore, we give an additional assumption that the set of the discrete gradients in the neighborhood of $\boldsymbol{x}$ approximates the Goldstein ε-subdifferential of the function f (see Definition 1.10).

Assumption 18.1 *Let $\boldsymbol{x} \in \mathbb{R}^n$ be a given point. For any $\varepsilon > 0$ there exist $\delta > 0$, $\alpha_0 > 0$ and $\lambda_0 > 0$ such that*

$$V(\boldsymbol{y}, \alpha, \lambda) \subset \partial_\varepsilon^G f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)$$

for all $\boldsymbol{y} \in B(\boldsymbol{x}; \delta)$, $\alpha \in (0, \alpha_0)$ and $\lambda \in (0, \lambda_0)$.

Proposition 18.16 *Assume that the function $f \in \mathcal{F}$, Assumption 18.1 is satisfied on $\mathbb{R}^n$ and the set $\mathcal{L}(\boldsymbol{x}_1) = \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \le f(\boldsymbol{x}_1)\}$ is bounded for any $\boldsymbol{x}_1 \in \mathbb{R}^n$. Then, every accumulation point of the sequence $\{\boldsymbol{x}_k\}$ generated by the* DGM *belongs to the set $X^0 = \{\boldsymbol{x} \in \mathbb{R}^n : \mathbf{0} \in \partial f(\boldsymbol{x})\}$.*

Proof Since the function f is continuous and the set $\mathcal{L}(\boldsymbol{x}_1)$ is compact we get $f^* > -\infty$. Therefore, conditions of Proposition 18.15 are satisfied and the inner loop of the DGM terminates after a finite number of steps with a point $\boldsymbol{x}_{k+1}$ such that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k_s}$ for some $s > 0$ and

$$\min_{\boldsymbol{v} \in \overline{V}(\boldsymbol{x}_{k+1})} \|\boldsymbol{v}\| \le \delta_k. \tag{18.23}$$

In addition, it is clear that

$$\overline{V}(\boldsymbol{x}_{k+1}) \subset V(\boldsymbol{x}_{k+1}, \alpha, \lambda_k).$$

Since $\{f(\boldsymbol{x}_k)\}$ is a decreasing sequence the point $\boldsymbol{x}_k \in \mathcal{L}(\boldsymbol{x}_1)$ for all $k \ge 0$. Then the sequence $\{\boldsymbol{x}_k\}$ is bounded and has at least one accumulation point. Assume that $\bar{\boldsymbol{x}}$ is an accumulation point of the sequence $\{\boldsymbol{x}_k\}$ and there exists a sequence $\{\boldsymbol{x}_{k_i}\}$ such that $\boldsymbol{x}_{k_i} \to \bar{\boldsymbol{x}}$ as $i \to \infty$. It follows from (18.23) and (18.5) that

$$\min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in V(\boldsymbol{x}_{k_i}, \alpha, \lambda_{k_i - 1})\} \le \delta_{k_i - 1}. \tag{18.24}$$

According to Assumption 18.1 at the point $\bar{\boldsymbol{x}}$ for any $\varepsilon > 0$ there exist $\beta > 0$, $\alpha_0 > 0$ and $\lambda_0 > 0$ such that

$$V(\boldsymbol{y}, \alpha, \lambda) \subset \partial_\varepsilon^G f(\bar{\boldsymbol{x}}) + B(\mathbf{0}; \varepsilon) \tag{18.25}$$

for all $\boldsymbol{y} \in B(\bar{\boldsymbol{x}}; \beta)$, $\alpha \in (0, \alpha_0)$ and $\lambda \in (0, \lambda_0)$. Since the sequence $\{\boldsymbol{x}_{k_i}\}$ converges to $\bar{\boldsymbol{x}}$ there exists $i_0 > 0$ such that $\boldsymbol{x}_{k_i} \in B(\bar{\boldsymbol{x}}; \beta)$ for all $i \ge i_0$. On the other hand, we have $\delta_k, \lambda_k \to 0$ as $k \to +\infty$ and therefore, there exists $k_0 > 0$ such that $\delta_k < \varepsilon$ and $\lambda_k < \lambda_0$ for all $k > k_0$. Then there exists $i_1 \ge i_0$ such that

$k_i \geq k_0 + 1$ for all $i \geq i_1$. Thus, it follows from (18.24) and (18.25) that

$$\min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in \partial_{\varepsilon}^{G} f(\bar{\boldsymbol{x}}\} \leq 2\varepsilon\}.$$

Since $\varepsilon > 0$ is arbitrary and the mapping $\partial f(\boldsymbol{x})$ is upper semicontinuous we get $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$. This completes the proof. □

Remark 18.2 Since in the DGM descent directions can be computed for any values of $\lambda > 0$ one can take $\lambda_1 \in (0, 1)$, some $\beta \in (0, 1)$ and update λ_k, $k \geq 1$ for instance by the formula $\lambda_k = \beta^k \lambda_1$, $k \geq 1$. Thus, approximations to subgradients are used only at the final stage which guarantees the convergence of the DGM. In most of iterations such approximations are not used and therefore, the DGM is a semi-derivative-free method.

Remark 18.3 It is obvious that in the DGM always the step size $t_{k_s} \geq \lambda_k$. In order to compute t_{k_s} one can define a sequence $\theta_m = m\lambda_k$, $m \geq 1$ and take t_{k_s} as the largest θ_m satisfying the inequality in the line search step.

Remark 18.4 There are similarities between the DGM and bundle methods. More specifically, the method presented here can be considered as a semi-derivative-free version of the bundle method introduced in [29]. Algorithms for the computation of descent directions in both methods are similar. Nevertheless, the DGM uses the discrete gradients instead of subgradients.

18.6 Limited Memory Discrete Gradient Bundle Method

In this section we recall the limited memory discrete gradient bundle method [18] for large scale derivative free nonconvex NSO. The idea of the method is to combine the discrete gradient method DGM described in the previous section with the limited memory bundle method LMBM [13–15] described in Chap. 5.

The DGM is a semi derivative-free method for (small or medium size) nonconvex NSO while the LMBM utilizes the subgradient information to solve large scale NSO problems. In the LMBM we bundle the subgradients that are computed in a small neighborhood of the current iteration point. On the other hand, in the DGM we gather into a bundle the discrete gradients that are calculated only at the current iteration point but with respect to different directions (see Definition 18.1 and Sect. 18.5). In the LDGB we combine these ideas and compute the discrete gradients in a small neighborhood of the current iteration point with respect to different directions.

In the DGM a quadratic subproblem (see Step 4 of Algorithm 18.1) needs to be solved to find the discrete gradient with the least norm and, as a consequence, to calculate the search direction. The convex combination of at most three discrete gradients is computed and the search direction is calculated using the limited memory approach. Thus, a time consuming direction finding prob-

lem needs not to be solved in this method. In addition, the difficulty with the unbounded storage space that may be required in the DGM is dealt with in the LDGB.

The obvious difference between the LDGB and the LMBM is that we now use the discrete gradients instead of subgradients of the objective function. In both methods, inner and outer iterations are used in order to avoid too tight approximations to the subgradients at the beginning of computation (thus, we have a derivative free method similarly to the DGM). The inner iteration of the LDGB is essentially same as the LMBM. That is, the search direction is computed by the formula

$$\boldsymbol{d}_{k_s} = -D_{k_s}\tilde{\boldsymbol{v}}_{k_s},$$

where s and k are the indices of inner and outer iterations, $\tilde{\boldsymbol{v}}_{k_s}$ is an aggregate discrete gradient and D_{k_s} is a limited memory variable metric update. In addition, the line search procedure (see Chap. 5, Eqs. (5.3)–(5.5)) is used to determine a new iteration and auxiliary points $\boldsymbol{x}_{k_{s+1}}$ and $\boldsymbol{y}_{k_{s+1}}$, and the aggregation procedure (see Chap. 5, Eqs. (5.7) and (5.8)) is used to compute a new aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_{s+1}}$ and a new aggregate subgradient locality measure $\tilde{\beta}_{k_{s+1}}$.

The first discrete gradient $\boldsymbol{v}_{1_1} = \Gamma^i(\boldsymbol{x}, \boldsymbol{g}_{1_1}, \boldsymbol{e}, \alpha, \lambda)$, where $i = \operatorname{argmax}\{|g_j| : j = 1, \ldots, n\}$, is computed using an arbitrary initial direction $\boldsymbol{g}_{1_1} \in S_1$. After that we always use the previous normalized search direction $\boldsymbol{g}_{k_{s+1}} = \boldsymbol{d}_{k_s}/\|\boldsymbol{d}_{k_s}\|$ to compute the next discrete gradient $\boldsymbol{v}_{k_{s+1}}$. Parameters $\lambda > 0$ and $\alpha > 0$ are selected similarly to the DGM.

The inner iteration is terminated if we have

$$\frac{1}{2}\|\tilde{\boldsymbol{v}}_{k_s}\|^2 + \tilde{\beta}_{k_s} \leq \delta_k$$

for some outer iteration parameter $\delta_k > 0$.

The LDGB uses an adaptive updating strategy for the selection of outer iteration parameter δ_k. At the beginning, the outer iteration parameter δ_1 is set to a large number. Each time the inner iteration is terminated we set

$$\delta_{k+1} = \min\{\sigma\delta_k, w_{k_s}\},$$

where $\sigma \in (0, 1)$ and $w_{k_s} = -\langle\tilde{\boldsymbol{v}}_{k_s}, \boldsymbol{d}_{k_s}\rangle + 2\tilde{\beta}_{k_s}$. Similarly to the LMBM, the parameter w_{k_s} is used also during the line search procedure to represent the desirable amount of descent (see Chap. 5, Eqs. (5.4) and (5.5)).

Let us assume that the sequence $\lambda_k > 0$, $\lambda_k \downarrow 0$, $k \to \infty$, a sufficiently small number $\alpha > 0$ and the line search parameter $\varepsilon_L^{k_s} \in (0, 1/2)$ are given. The LDGB proceeds as follows.

Algorithm 18.2: LDGB

Data: $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\varepsilon > 0$, $\delta_1 > \varepsilon$, $\lambda_1 > \varepsilon$, $\alpha \in (0, 1]$, $\epsilon_L \in (0, 1/2)$.

Step 1. (*Outer iteration initialization*) Set $k = 1$ and choose $\boldsymbol{g}_{k_1} \in S_1$.

Step 2. (*Inner iteration initialization*) Set $s = 1$ and $\boldsymbol{x}_{k_s} = \boldsymbol{x}_k$.

Step 3. (*Stopping criterion*) If $\lambda_k < \varepsilon$ and $\delta_k < \varepsilon$, then **stop** with $\boldsymbol{x}_k$ as a final solution.

Step 4. (*Serious step 1*) Compute a discrete gradient

$$\boldsymbol{v}_{k_s} = \Gamma^i(\boldsymbol{x}_{k_s}, \boldsymbol{g}_{k_s}, \boldsymbol{e}, \lambda_k, \alpha).$$

Set $m = s$, $\tilde{\boldsymbol{v}}_{k_s} = \boldsymbol{v}_{k_s}$, and $\tilde{\beta}_{k_s} = 0$.

Step 5. (*Direction finding for serious steps*) Compute $\boldsymbol{d}_{k_s}$ using $\tilde{\boldsymbol{v}}_{k_s}$ and the L-BFGS update.

Step 6. (*Inner iteration termination*) If $1/2\|\tilde{\boldsymbol{v}}_{k_s}\|^2 + \tilde{\beta}_{k_s} \leq \delta_k$, then set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k_s}$, $\boldsymbol{g}_{k+1_1} = \boldsymbol{d}_{k_s}/\|\boldsymbol{d}_{k_s}\|$, and update λ_{k+1} and δ_{k+1}. Set $k = k + 1$ and go to Step 2.

Step 7. (*Line search and updating*) Find step sizes $t_L^{k_s}$ and $t_R^{k_s}$, and the subgradient locality measure $\beta_{k_{s+1}}$. Update the limited memory matrices.

Step 8. If $f(\boldsymbol{x}_{k_s} + t_R^{k_s}\boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) > -\epsilon_L t_R^{k_s}(-\langle \boldsymbol{d}_{k_s}, \tilde{\boldsymbol{v}}_{k_s}\rangle + 2\tilde{\beta}_{k_s})$, then go to Step 10.

Step 9. (*Serious step 2*) Construct the iteration $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s} + t_L^{k_s}\boldsymbol{d}_{k_s}$. Set $\boldsymbol{g}_{k_{s+1}} = \boldsymbol{d}_{k_s}/\|\boldsymbol{d}_{k_s}\|$, $s = s + 1$ and go to Step 4.

Step 10. (*Null step*) Construct a trial point $\boldsymbol{y}_{k_{s+1}} = \boldsymbol{x}_{k_s} + t_R^{k_s}\boldsymbol{d}_{k_s}$ and set $\boldsymbol{g}_{k_{s+1}} = \boldsymbol{d}_{k_s}/\|\boldsymbol{d}_{k_s}\|$. Compute a new discrete gradient

$$\boldsymbol{v}_{k_{s+1}} = \Gamma^i(\boldsymbol{y}_{k_{s+1}}, \boldsymbol{g}_{k_{s+1}}, \boldsymbol{e}, \lambda_k, \alpha)$$

and the aggregate values

$$\tilde{\boldsymbol{v}}_{k_{s+1}} = \zeta_1^{k_s}\boldsymbol{v}_{k_m} + \zeta_2^{k_s}\boldsymbol{v}_{k_{s+1}} + \zeta_3^{k_s}\tilde{\boldsymbol{v}}_{k_s} \quad \text{and}$$
$$\tilde{\beta}_{k_{s+1}} = \zeta_2^{k_s}\beta_{k_{s+1}} + \zeta_3^{k_s}\tilde{\beta}_{k_s}.$$

Step 11. (*Direction finding for null steps*) Compute $\boldsymbol{d}_{k_{s+1}}$ using $\tilde{\boldsymbol{v}}_{k_{s+1}}$ and the L-SR1 update. Set $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s}$, $s = s + 1$, and go to Step 6.

As in the DGM the discrete gradient is computed according to Definition 18.1. Similarly to the LMBM the search direction and the aggregate values are computed by using the L-BFGS update after serious steps and L-SR1 update otherwise (see Chap. 5 for more details of the limited memory matrix updating and their usage in the LMBM). The step sizes $t_R^{k_s} \in (0, t_{max}]$ and $t_L^{k_s} \in [0, t_R^{k_s}]$ with $t_{max} > 1$ are computed such that either the serious step (5.4) or the null step condition (5.5) in Chap. 5 is satisfied. In addition, the subgradient locality measure $\beta_{k_{s+1}}$ as well as the

multipliers $\zeta_i^{k_s}$ satisfying $\zeta_i^{k_s} \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^{3} \zeta_i^{k_s} = 1$ utilized in the aggregation procedure are computed similarly to the LMBM (see Chap. 5, Eqs. (5.6) and (5.7)).

Convergence of the LDGB We now prove the global convergence of the LDGB under the assumptions that the function $f \in \mathcal{F}$ and the set $\mathcal{L}(\boldsymbol{x}_1) = \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\}$ is bounded for any $\boldsymbol{x}_1 \in \mathbb{R}^n$. In [18] the convergence of the LDGB is proved for general semismooth functions.

Definition 18.2 Let $\varepsilon > 0$, $\lambda > 0$, and $\alpha > 0$. We define *the ε-set of discrete gradients* by

$$V_\varepsilon(\boldsymbol{x}, \alpha, \lambda) = \operatorname{conv}\{V(\boldsymbol{y}, \alpha, \lambda) : \boldsymbol{y} \in \bar{B}(\boldsymbol{x}; \varepsilon)\}.$$

We first show that the ε-set of discrete gradients can be used to approximate the Goldstein subdifferential of a function. Note the analogue of this Proposition with Assumption 18.1.

Proposition 18.17 *Suppose that the assumptions of Proposition 18.13 are satisfied at every $\boldsymbol{y} \in \bar{B}(\boldsymbol{x}; \varepsilon)$ with some $\varepsilon > 0$. Then, there exist $\lambda_0 > 0$ and $\alpha_0 > 0$ such that the ε-set of discrete gradients approximates the Goldstein subdifferential by*

$$V_\varepsilon(\boldsymbol{x}, \alpha, \lambda) \subset \partial_\varepsilon^G f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)$$

with all $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$.

Proof Let $\boldsymbol{v} \in V_\varepsilon(\boldsymbol{x}, \alpha, \lambda)$. Then $\boldsymbol{v} = \sum_i \sigma_i \boldsymbol{v}_i^{\boldsymbol{y}}$, where $\boldsymbol{v}_i^{\boldsymbol{y}} \in V(\boldsymbol{y}^i, \alpha, \lambda)$, $\boldsymbol{y}^i \in \bar{B}(\boldsymbol{x}; \varepsilon)$, $\sum_i \sigma_i = 1$ and $\sigma_i \geq 0$ for all i. Now, by Proposition 18.13 there exists λ_0^i and α_0^i for any $\varepsilon > 0$ such that $V(\boldsymbol{y}^i, \alpha, \lambda) \subset \partial f(\boldsymbol{y}^i) + B(\mathbf{0}; \varepsilon)$ for all $\lambda \in (0, \lambda_0^i)$ and $\alpha \in (0, \alpha_0^i)$. Thus, we have $\boldsymbol{v}_i^{\boldsymbol{y}} = \boldsymbol{\xi}_i^{\boldsymbol{y}} + \boldsymbol{s}_i^\varepsilon$, where $\boldsymbol{\xi}_i^{\boldsymbol{y}} \in \partial f(\boldsymbol{y}^i)$ and $\boldsymbol{s}_i^\varepsilon \in B(\mathbf{0}; \varepsilon)$. Hence,

$$\begin{aligned}
\boldsymbol{v} &= \sum_i \sigma_i (\boldsymbol{\xi}_i^{\boldsymbol{y}} + \boldsymbol{s}_i^\varepsilon) \\
&= \sum_i \sigma_i \boldsymbol{\xi}_i^{\boldsymbol{y}} + \sum_i \sigma_i \boldsymbol{s}_i^\varepsilon \\
&\subset \operatorname{conv}\{\partial f(\boldsymbol{y}^i) : \boldsymbol{y}^i \in \bar{B}(\boldsymbol{x}; \varepsilon)\} + B(\mathbf{0}; \varepsilon) \\
&= \partial_\varepsilon^G f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)
\end{aligned}$$

with all $\lambda \in (0, \min_i\{\lambda_0^i\})$ and $\alpha \in (0, \min_i\{\alpha_0^i\})$. □

Next we show that all the aggregate discrete gradients generated by the LDGB belong to the ε-set of discrete gradients.

Lemma 18.1 *For all inner iterations k_s, there exists $\varepsilon_{k_s} \geq 0$ such that*

$$\tilde{\boldsymbol{v}}_{k_s} \in V_{\varepsilon_{k_s}}(\boldsymbol{x}_{k_s}, \alpha, \lambda_k).$$

Proof After a serious step we have $\tilde{\boldsymbol{v}}_{k_{s+1}} = \boldsymbol{v}_{k_{s+1}} \in V(\boldsymbol{x}_{k_{s+1}}, \alpha, \lambda_k)$ and the result is true with any $\varepsilon_{ks} \geq 0$. After a null step the aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_{s+1}}$ is computed as a convex combination of three discrete gradients: $\boldsymbol{v}_{k_s} \in V(\boldsymbol{x}_{k_s}, \alpha, \lambda_k)$, $\boldsymbol{v}_{k_{s+1}} \in V(\boldsymbol{y}_{k_{s+1}}, \alpha, \lambda_k)$ and $\tilde{\boldsymbol{v}}_{k_s}$. For null steps we use the scaled direction vector $\theta_k \boldsymbol{d}_k$ with $\theta_k = \min\{1, C/\|\boldsymbol{d}_k\|\}$ and predefined $C > 0$ in the line search (for more details, see [15]). Thus, we have $\|\boldsymbol{y}_{k_{s+1}} - \boldsymbol{x}_{k_s}\| \leq t_{max} C$, and we have $\boldsymbol{v}_{k_{s+1}} \in V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \alpha, \lambda_k)$ with some $0 < \varepsilon_{ks} \leq t_{max} C$. Since the first aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_s}$ after a serious step is equal to $\boldsymbol{v}_{k_s}$, they both belong to the set $V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \alpha, \lambda_k)$ with any $\varepsilon_{ks} \geq 0$.

As a convex combination of these three discrete gradients, the second aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_{s+1}}$ after the serious step belongs to the set $V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \alpha, \lambda_k)$ with some $0 < \varepsilon_{ks} \leq t_{max} C$. The rest of the proof follows by induction. □

Proposition 18.18 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a LLC semismooth function. Then the number of inner iterations in the* LDGB *is finite and, at the termination, we have*

$$\min_{\boldsymbol{v} \in V(\boldsymbol{x}_{k+1}, \alpha, \lambda_k)} \|\boldsymbol{v}\|^2 \leq 2\delta_k. \tag{18.26}$$

Proof The inner iteration loop in the LDGB is essentially the same as the LMBM with subgradients replaced by the discrete gradients. It has been shown that the LMBM terminates after finite number of iterations, if the stopping parameter (cf. δ_k here) is greater than zero (see, [15] and Chap. 5). The use of the discrete gradients does not alter this result. Thus, the inner iteration loop in the LDGB is terminating after finite number of steps.

At the termination we have

$$\delta_k \geq \frac{1}{2}\|\tilde{\boldsymbol{v}}_{k_s}\|^2 + \tilde{\beta}_{k_s} \geq \frac{1}{2}\|\tilde{\boldsymbol{v}}_{k_s}\|^2.$$

By Lemma 18.1 we have $\tilde{\boldsymbol{v}}_{k_s} \in V_\mu(\boldsymbol{x}_{k_s}, \alpha, \lambda_k)$ with some $\mu \geq 0$ and we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k_s}$ at the termination. Thus, we have

$$\min_{\boldsymbol{v} \in V(\boldsymbol{x}_{k+1}, \alpha, \lambda_k)} \|\boldsymbol{v}\|^2 \leq \|\tilde{\boldsymbol{v}}_{k_s}\|^2 \leq 2\delta_k.$$

□

Proposition 18.19 *Assume that the function $f \in \mathcal{F}$ and for any $\boldsymbol{x}_1 \in \mathbb{R}^n$, the set $\mathcal{L}(\boldsymbol{x}_1) = \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\}$ is bounded. Then, every accumulation point of the sequence $\{\boldsymbol{x}_k\}$ generated by the* LDGB *belongs to the set $X^0 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{0} \in \partial f(\boldsymbol{x})\}$.*

Proof Since f is continuous and the set $\mathcal{L}(\boldsymbol{x}_1)$ is bounded, we have $f^* > -\infty$. By Proposition 18.18 the inner iterations terminates after a finite number of steps with a point $\boldsymbol{x}_{k+1}$ such that (18.26) holds. Thus,

$$\min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in V(\boldsymbol{x}_{k+1}, \alpha, \lambda_k)\} \leq \sqrt{2\delta_k} \tag{18.27}$$

for any $k \geq 1$. Since $\{f(\boldsymbol{x}_k)\}$ is a nonincreasing sequence, the point $\boldsymbol{x}_k \in \mathcal{L}(\boldsymbol{x}_1)$ with all $k \geq 1$. Moreover, the boundedness of the level set implies that the sequence $\{\boldsymbol{x}_k\}$ has at least one accumulation point. Let $\bar{\boldsymbol{x}}$ be an accumulation point of $\{\boldsymbol{x}_k\}$ and let $\boldsymbol{x}_{k_i} \to \bar{\boldsymbol{x}}$ when $i \to \infty$. Then we have from (18.27) that

$$\min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in V(\boldsymbol{x}_{k_i}, \alpha, \lambda_{k_i-1})\} \leq \sqrt{2\delta_{k_i-1}}. \tag{18.28}$$

Now, by Proposition 18.17 for any $\boldsymbol{y} \in \bar{B}(\bar{\boldsymbol{x}}; \mu)$ with $\mu > 0$ there exists $\lambda_0 > 0$ and $\alpha_0 > 0$ such that

$$V(\boldsymbol{y}, \alpha, \lambda) \subseteq V_\mu(\bar{\boldsymbol{x}}, \alpha, \lambda) \subseteq \partial_\mu^G f(\bar{\boldsymbol{x}}) + B(\boldsymbol{0}; \mu) \tag{18.29}$$

for all $\lambda \in (0, \lambda_0)$ and $\alpha \in (0, \alpha_0)$. Since $\boldsymbol{x}_{k_i}$ converges to $\bar{\boldsymbol{x}}$ there exists $i_0 \geq 1$ such that $\boldsymbol{x}_{k_i} \in B(\bar{\boldsymbol{x}}; \mu)$ for all $i \geq i_0$. In addition, since $\delta_k \to 0$ and $\lambda_k \to 0$ as $k \to \infty$ there exists $k_0 > 1$ such that $\delta_k \leq \varepsilon$ with some $\varepsilon > 0$ and $\lambda_k < \lambda_0$ for all $k > k_0$. Then there exists $l_0 \geq i_0$ such that $k_l \geq k_0 + 1$ for all $l \geq l_0$. It follows from (18.28) and (18.29) that

$$\min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in \partial_\mu^G f(\bar{\boldsymbol{x}})\} \leq \mu + \sqrt{2\varepsilon}.$$

Now, since $\mu, \varepsilon > 0$ are arbitrary and the mapping $\partial f(\boldsymbol{x})$ is upper semicontinuous we get $\boldsymbol{0} \in \partial f(\bar{\boldsymbol{x}})$. This completes the proof. □

18.7 Illustrative Examples

In this section we present illustrative examples to demonstrate the performance of both the discrete and the limited memory discrete gradient methods. We include four small size convex, four small size nonconvex and four large scale (both convex and nonconvex) problems. Test problems are listed in Table 18.1. Their details can be found in [9].

Results are presented in Tables 18.2, 18.3, 18.4, 18.5, 18.6, and 18.7. The following notations are adopted to report results:

- f_{best}—the best known estimate for the global minimum value;
- f—the best objective function value obtained by an algorithm;
- N_f—the number of function evaluations;
- CPU—computational time, in seconds, used by an algorithm.

Table 18.1 Test problems

Small convex problems		Small nonconvex problems		Large scale problems	
P1	Shor	P5	Crescent	P9	Chained LQ
P2	Maxq	P6	El-Attar	P10	Chained CB3
P3	Maxl	P7	L_1-Rosenbrock	P11	Chained Mifflin 2
P4	Goffin	P8	L_1-Wood	P12	Chained Crescent I

Table 18.2 Results for small convex problems

		DGM			LDGB		
Prob	f_{best}	f	N_f	CPU	f	N_f	CPU
P1	22.60	22.6002	4697	0.00	37.7919	371	0.00
P2	0	0.0000	9808	0.00	0.0000	5841	0.00
P3	0	0.0000	10935	0.00	0.0003	29313	0.00
P4	0	0.0000	403440	3.30	225.1001	35173	0.01

Table 18.3 Results for small nonconvex problems

		DGM			LDGB		
Prob	f_{best}	f	N_f	CPU	f	N_f	CPU
P5	0	0.0000	795	0.00	0.0000	353	0.00
P6	0.56	0.5598	9235	0.02	0.5619	8848	0.03
P7	0	1.0000	583	0.00	0.0000	789	0.00
P8	0	0.0000	1988	0.00	0.0000	8609	0.00

Table 18.4 Large scale convex problem-P9

		DGM			LDGB		
Prob	f_{best}	f	N_f	CPU	f	N_f	CPU
10	-12.73	-12.7279	7601	0.00	-12.7279	8723	0.00
20	-26.87	-26.8700	28070	0.01	-26.8694	11890	0.00
50	-69.30	-69.2954	123201	0.10	-69.2927	23143	0.01
100	-140.01	-140.0071	461270	1.33	-140.0052	43989	0.01
200	-281.43	-281.4283	794650	1.92	-281.4148	106316	0.06
500	-705.69	-705.6914	2217624	3.48	-705.6507	242397	0.28
1000	-1412.77	-1412.7688	4475147	7.99	-1412.7646	619571	1.40

Table 18.5 Large scale convex problem-P10

		DGM			LDGB		
Prob	f_{best}	f	N_f	CPU	f	N_f	CPU
10	18	18.0000	7363	0.00	18.0000	13041	0.00
20	38	38.0000	25141	0.01	38.0000	25893	0.01
50	98	98.0003	120470	0.14	98.0032	28520	0.02
100	198	198.0000	398897	1.78	198.0056	48061	0.04
200	398	398.0000	807894	3.06	398.0049	102334	0.17
500	998	998.0000	1994797	9.79	998.0332	360158	1.42
1000	1998	1998.0005	4361977	36.32	1998.1608	1054782	8.39

Table 18.6 Large scale nonconvex problem-P11

		DGM			LDGB		
Prob	f_{best}	f	N_f	CPU	f	N_f	CPU
10	-6.52	-6.5146	8368	0.00	-6.5146	16681	0.00
20	-13.58	-13.5823	27196	0.01	-13.5819	42397	0.01
50	-34.79	-34.7811	114786	0.07	-34.7871	47722	0.01
100	-70.15	-70.1468	481212	1.00	-70.1365	70008	0.03
200	-140.85	-140.8473	1036091	1.78	-140.8387	166332	0.11
500	-352.95	-352.8119	2677212	5.50	-352.9542	212768	0.33
1000	-706.42	-706.0061	5601267	18.83	-706.4151	852903	2.53

Results for small size convex test problems are reported in Table 18.2. We can see that the DGM is able to find solutions with high accuracy, however the LDGB failed in two cases. On the other side the latter method requires significantly less computational effort than the former algorithm with the exception of the test problem Maxl.

Table 18.3 contains results for small size nonconvex test problems. The DGM failed to find the global minimizer of the function P7 and in all other cases it finds global minimizers with high accuracy. The LDGB finds global solutions in all cases except for the problem P6 where the obtained solution is not an accurate solution. Overall, the LDGB requires more computational effort than the DGM for solving small scale nonconvex NSO problems.

Tables 18.4 and 18.5 present results on large scale convex problems. The DGM produces more accurate results than the LDGB for the first problem P9 whereas for the second problem P10 we can observe the opposite outcome. In both cases the LDGB requires significantly less computational effort than the DGM as the number of variables increase. Although the CPU time used by both methods is reasonable the number of function evaluations is large.

Results for large scale nonconvex NSO problems are reported in Tables 18.6 and 18.7. Here, both methods demonstrate the similar performance in finding accurate solutions. If for solving the problem P11 the LDGB method requires less number of function evaluations and CPU time we can observe the opposite outcome for the problem P12. Again computational time used by methods is reasonable but the number of function evaluations increases significantly as the number of variables increase.

Results reported in this section demonstrate that both the DGM and LDGB are efficient and in general accurate methods for solving both convex and nonconvex NSO problems when the evaluation of the objective function is not expensive. These methods may become inefficient when the evaluation of the objective function is time consuming.

Table 18.7 Large scale nonconvex problem-P12

		DGM			LDGB		
Prob	f_{best}	f	N_f	CPU	f	N_f	CPU
10	0	0.0000	5147	0.00	0.0000	3945	0.00
20	0	0.0000	13712	0.00	0.0000	9700	0.00
50	0	0.0000	61086	0.02	0.0000	12402	0.00
100	0	0.0000	197368	0.45	0.0000	16444	0.01
200	0	0.0000	422367	0.90	0.0000	2000220	1.81
500	0	0.0000	1069370	2.27	0.0000	5000000	4.41
1000	0	0.0000	2044425	7.00	0.0000	10000620	41.27

18.8 Conclusions

In this chapter the discrete gradients are considered to approximate subdifferentials of nonsmooth functions. The discrete gradients are computed using only function values. They can be considered as finite difference estimates of subdifferentials. It is shown that the discrete gradients can be used to approximate subdifferentials of a broad class of nonsmooth functions including the nonsmooth convex and nonsmooth difference of convex functions. Two methods—the discrete gradient and the limited memory discrete gradient bundle methods—are designed using these approximations. Both methods are some type of semi derivative-free methods because they use approximation of subdifferentails only at the final stages of methods. Convergence of methods is studied and their performance is demonstrated using several academic NSO test problems with both convex and nonconvex objective functions. Numerical results show that the methods considered in this chapter are able to find accurate solutions in most cases, however the number of function evaluations may increase significantly as the number of variables increases.

Acknowledgements This research by Dr. Adil Bagirov and Dr. Sona Taheri was supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project No. DP190100580). The research by Dr. Napsu Karmitsa and Dr. Sona Taheri was supported by the Academy of Finland (Project No. 289500 and 319274).

References

1. Audet, C., Dennis, J.E.: Analysis of generalized pattern searches. SIAM J. Optim. **13**, 889–903 (2003)
2. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer, Cham (1992)
3. Bagirov, A.M.: A method of approximating a subdifferential. Comput. Math. Math. Phys. **32** (4), 561–566 (1992)

4. Bagirov, A.M.: A method for minimizing convex functions based on continuous approximations to the subdifferential. Optim. Methods Softw. **9**(1–3), 1–17 (1998)
5. Bagirov, A.M.: Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In: Eberhard, A., et al. (eds.) Progress in Optimization: Contributions from Australasia, pp. 147–175. Kluwer, Dordrecht, Springer, Berlin (1999)
6. Bagirov, A.M.: Continuous subdifferential approximations and their applications. J. Math. Sci. **115**, 2567–2609 (2003)
7. Bagirov, A.M., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**, 317–334 (2008)
8. Bagirov, A.M., Al Nuaimat, A., Sultanova, N.: Hyperbolic smoothing function method for minimax problems. Optimization **62**(6), 759–782 (2013)
9. Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: Introduction to Nonsmooth Optimization: Theory, Practice and Software. Springer, Berlin (2014)
10. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**, 751–779 (2005)
11. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **113**, 117–156 (2002)
12. Gaudioso, M., Monaco, M.F.: A bundle type approach to the unconstrained minimization of convex nonsmooth functions. Math. Program. **23**, 216–226 (1982)
13. Haarala, M.: Large-scale nonsmooth optimization: variable metric bundle method with limited memory. Ph.D. thesis, University of Jyväskylä, Department of Mathematical Information Technology (2004)
14. Haarala, M., Miettinen, K., Mäkelä, M.M.: New limited memory bundle method for large-scale nonsmooth optimization. Optim. Methods Softw. **19**(6), 673–692 (2004)
15. Haarala, N., Miettinen, K., Mäkelä, M.M.: Globally convergent limited memory bundle method for large-scale nonsmooth optimization. Math. Program. **109**(1), 181–205 (2007)
16. Hiriart-Urruty, J.B., Lemarechal, C.: Convex Analysis and Minimization Algorithms. Springer, Heidelberg (1993)
17. Karmitsa, N.: Diagonal discrete gradient bundle method for derivative free nonsmooth optimization. Optimization **85**(8), 1599–1614 (2016)
18. Karmitsa, N., Bagirov, A.M.: Limited memory discrete gradient bundle method for nonsmooth derivative-free optimization. Optimization **61**(12), 1491–1509 (2012)
19. Kiwiel, K.: An aggregate subgradient method for nonsmooth convex minimization. Math. Program. **27**(3), 320–341 (1983)
20. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics, vol. 1133. Springer, Berlin (1985)
21. Lemarechal, C.: An extension of Davidon methods to non differentiable problems. In: Balinski, M. L., Wolfe, Ph. (eds.) Nondifferentiable Optimization, pp. 95–109. Springer, Berlin (1975)
22. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific, Singapore (1992)
23. Mifflin, R.: An algorithm for constrained optimization with semismooth functions. Math. Oper. Res. **2**, 191–207 (1977)
24. Nesterov, Yu.: Smooth minimization of nonsmooth functions. Math. Program. **103**(1), 127–152 (2005)
25. Polak, E., Royset, J.O.: Algorithms for finite and semi-infinite min–max–min problems using adaptive smoothing techniques. J. Optim. Theory Appl. **119**(3), 421–457 (2003)
26. Shor, N.Z.: Minimization Methods for Non-differentiable Functions. Springer, Berlin (1985)
27. Studniarski, M., Rahmo, El D.: Approximating Clarke subgradients of semismooth functions by divided differences. Numer. Algorithms **43**, 385–392 (2006)
28. Torzcon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **1**, 1–25 (1997)
29. Wolfe, Ph.: A method of conjugate subgradients for minimizing nondifferentiable functions. In: Balinski, M.L., Wolfe, Ph. (eds.) Nondifferentiable Optimization, pp. 145–173. Springer, Berlin (1975)

30. Xu, S.: Smoothing method for minimax problems. Comput. Optim. Appl. **20**(3), 267–279 (2001)
31. Zowe, J.: Nondifferentiable optimization: a motivation and a short introduction into the subgradient and the bundle concept. In: Schittkowski, K. (ed.) Computational Mathematical Programming, pp. 323–356. Springer, Berlin (1985)

Chapter 19
Model-Based Methods in Derivative-Free Nonsmooth Optimization

Charles Audet and Warren Hare

Abstract Derivative-free optimization (DFO) is the mathematical study of the optimization algorithms that do not use derivatives. One branch of DFO focuses on model-based DFO methods, where an approximation of the objective function is used to guide the optimization algorithm. Historically, model-based DFO has often assumed that the objective function is smooth, but unavailable analytically. However, recent progress has brought model-based DFO into the realm of nonsmooth optimization (NSO). In this chapter, we survey some of the progress of model-based DFO for nonsmooth functions. We begin with some historical context on model-based DFO. From there, we discuss methods for constructing models of smooth functions and their accuracy. This leads to modelling techniques for nonsmooth functions and a discussion on several frameworks for model-based DFO for NSO. We conclude the chapter with some of our opinions on profitable research directions in model-based DFO for NSO.

19.1 Introduction

In 1969, Winfield defended a Ph.D. thesis titled "Function and functional optimization by interpolation in data tables" [92]. The associated paper appeared 4 years later in the *Journal of the Institute of Mathematics and its Applications* [93]. In it, Winfield presented a method to optimize a function using only function values. Winfield was not the first to consider the challenge of optimization using only function values (consider, for example, the influential papers by Hooke and Jeeves [52] or Nelder and Mead [67]). However, Winfield was the first to use the function

C. Audet
GERAD and Département de Mathématiques et Génie Industriel, École Polytechnique de Montréal, Montréal, QC, Canada
e-mail: Charles.Audet@gerad.ca

W. Hare (✉)
Department of Mathematics, University of British Columbia, Kelowna, BC, Canada
e-mail: warren.hare@ubc.ca

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3_19

values to build a model of the objective function and then use the model to guide the search for a new incumbent solution. It is now 50 years later, and this framework is the cornerstone of *model-based derivative-free optimization*.

Let us define *derivative-free optimization* (DFO) as the mathematical study of algorithms for continuous optimization that do not use first-order information (derivatives, gradients, directional derivatives, subgradients, etc.). Note that, we said that the algorithms do not use first-order information, which is quite different from saying that the objective function is nonsmooth. Also note, we use the term *mathematical study* to emphasize that research examines concepts such as proof of convergence, accuracy of stopping conditions, and rigorous analysis of numerical tests. This allows us to remove heuristic methods from our definition. In our opinion, when the mathematical analysis of an algorithm is provided, it is no longer a heuristic.

DFO methods have been successfully applied in a wide collection of areas. Some examples include oil production optimization problems [38, 43], molecular geometry [2, 60], helicopter rotor blade design [18, 19, 85], research on water resources [1, 40, 63, 65], alloy and process design [41, 42, 80]. Many other engineering applications and extensions are given in [4, 36].

Using our definition, DFO can be broadly split into two algorithmic styles: direct search methods, and model-based methods.

Direct search DFO methods work from an incumbent solution and examine a collection of trial points to seek improvement. If improvement is found, then the incumbent solution is updated; while if no improvement is found, then a step size parameter is decreased and a new collection of trial points is examined. The aforementioned works of Hooke and Jeeves [52] and Nelder and Mead [67] are classic examples of direct search methods.[1] Direct search methods have lead to many successful algorithms and been applied in many applications. However, the focus of this chapter is model-based DFO, so we simply refer the curious reader to a few detailed surveys on the subject [4, 58, 94] and the books [7, Part II] and [30, Chapters 7 and 8].

Model-based DFO methods begin by approximating the objective function with a model function, and then use the model function to help guide optimization. In [93], the model is built through quadratic interpolation, and the information is used by minimizing the model over a trust-region. This has remained one of the most popular techniques in model-based DFO (see Table 19.1). However, as we shall explore in this chapter, other ideas have recently emerged.

To understand model-based DFO methods, a first step is to study modelling techniques. Intuitively, we require techniques that produce "good" models. This can be interpreted in many ways. Winfield's quadratic interpolation approach interprets

[1]The Nelder-Mead method as presented in [67] is a heuristic. However, subsequent variants, including Tseng's fortified Nelder-Mead method [89] or Price, Coope, and Byatt's Nelder-Mead reset method [77], have proven convergence. This places the Nelder-Mead method into the realm of DFO.

"good" in the form of a second-order Taylor-like error term: (under some conditions) there exists constants κ_f, κ_g, and κ_h such that

$$\begin{aligned} |f(\boldsymbol{y}) - Q(\boldsymbol{y})| &\leq \kappa_f \Delta^3 && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta), \\ \|\nabla f(\boldsymbol{y}) - \nabla Q(\boldsymbol{y})\| &\leq \kappa_g \Delta^2 && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta), \\ \text{and} \quad \|\nabla^2 f(\boldsymbol{y}) - \nabla^2 Q(\boldsymbol{y})\| &\leq \kappa_h \Delta && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta), \end{aligned}$$

where f is the true objective function, Q is a model based on quadratic interpolation centered at $\boldsymbol{x}$, and $\Delta > 0$ is a scalar parameter which the optimizer controls (we shall explore details of this in Sect. 19.3). Obviously, one condition required for the above equations to hold is that $\nabla^2 f(\boldsymbol{y})$ is well-defined. This would apparently separate model-based DFO from the field of *nonsmooth optimization* (NSO).

Let us define NSO as the study of algorithms for continuous optimization problems that are not everywhere differentiable. In particular, NSO problems are typically not differentiable at their minimizers (or maximizers), which makes convergence analysis based on Taylor-like approximations problematic. Derivatives and gradients are generalized through variational analysis to construct directional derivatives, subgradients, or other similar objects. Precise definitions of variational analysis object relevant to this chapter will appear as required.

In order to merge the fields of model-based DFO and NSO, it is necessary to break away from the traditional Taylor-like error bounds, and pursue novel modelling techniques and analysis. In this chapter, we demonstrate that the merging of model-based DFO and NSO is not just possible, but underway. We begin with a more complete background on model-based DFO (Sect. 19.2) and then move on to some basic methods for constructing and using models of smooth functions (Sect. 19.3). We also discuss ways to analyze their accuracy. The methods for constructing models of smooth functions will allow us to discuss modelling methods and their accuracy for nonsmooth functions, which we present in Sect. 19.4. In addition, we provide further details on several frameworks that employ such models in model-based DFO for NSO. Section 19.5 concludes the chapter with some of our opinions on profitable research directions in model-based DFO for NSO.

19.2 Historical Overview

As mentioned in Sect. 19.1, model-based DFO dates back to at least 1969 [92]. However, we did not mention that Winfield's paper made very little impact on the field. In fact, the paper was largely ignored by the DFO community until Han and Liu [44] cited it in 2004 as a first contribution to the field.[2]

[2]Interestingly, Winfield's paper also appears in the references of DFO papers from 2002 to 2003 [58, 72, 73], but is not actually mentioned in the text of those articles.

Despite the early beginning of model-based DFO, it was not until 1994 that model-based DFO received the spark it required to take on life. That spark came in the form of a fully analyzed and implemented model-based DFO method called COBYLA (constrained optimization by linear approximation) [71]. Powell's COBYLA method uses linear interpolation to build a linear model of the objective function, then minimizes the linear model over a trust-region. Over the course of the algorithm, model accuracy is improved until asymptotically it is equivalent to a first-order Taylor expansion. Thus, asymptotically the method becomes equivalent to steepest descent, and its convergence is ensured [71] under some standard conditions.

Shortly after Powell's COBYLA, Conn and Toint [24] developed a similar method based on quadratic interpolation and a quadratic model function.[3] The use of a quadratic model means that the trust-region minimization can now take curvature into account and is no longer a reframing of steepest descent. However, the quadratic model comes at the price of making the model construction and the trust-region minimization more difficult. Like Powell, Conn and Toint's research included a proof of convergence along with a fully implemented algorithm that was numerically tested. Combined, the papers of Powell and Conn and Toint launched model-based DFO as a field full of possibilities.

Motivated by these ideas, more researchers started exploring the ability to minimize an objective function using model-based DFO. In many cases, the development of a model-based DFO algorithm began from a classical algorithm for smooth optimization. Researchers began by showing that they could build a model of the true objective function that enjoys a property similar to a first- or second-order Taylor-expansion (see Sect. 19.3). These models can then be used in place of the first- or second-order Taylor-expansion. In order to show convergence, some mechanism is added to the algorithm to ensure that the model converges to the truth as the algorithm approaches a critical point. Some examples of algorithms that fit this description are provided in Table 19.1. The first two algorithms in Table 19.1 are separated out, as they are algorithmic frameworks, not complete methods. The remaining algorithms are listed in order of first appearance in literature.

In examining Table 19.1, notice that the majority of methods follow the second-order trust-region framework described in [7, Chapter 10] and [30, Chapter 10]. We include details on this framework in Sect. 19.3.5.

[3] Conn and Toint's paper argues that there are 5 classes of DFO algorithms: finite-difference methods, pattern search methods, random sampling methods, successive one-dimension minimization methods, and model-based methods. Researchers have now unified pattern search methods, random sampling methods, and successive one-dimension minimization methods, as direct search methods. While finite-difference methods fall under our broad description of model-based methods.

Table 19.1 Model-based DFO algorithms that are closely related to classical smooth optimization algorithms

Framework	Reference	Smooth equivalent
MBD	[7, Chap. 10] and [30, Chap. 9]	Gradient-based line-search methods
MBTR	[7, Chap. 11] and [30, Chap. 10]	second-order Trust-region method

Algorithm	Reference	Year	Smooth equivalent
SQM	[92, 93]	1969	Second-order trust-region
unnamed	[61]	1975	Newton
COBYLA	[71]	1994	First-order trust-region
unnamed	[24]	1996	Second-order trust-region
DFO	[27]	2001	Second-order trust-region
UOBYQA	[72]	2002	Second-order trust-region
CONDOR	[14, 15]	2004	Second-order trust-region (for parallel computing)
BOOSTERS	[68, 69]	2005	Second-order trust-region (using radial basis functions)
NEWUOA	[75]	2006	Second-order trust-region (advances UOBYQA)
ORBIT	[90, 91]	2008	Second-order trust-region (using radial basis functions)
BOBYQA	[76]	2009	Second-order trust-region (advances UOBYQA to allow bound constraints)
DFTR	[29]	2009	2trust-region (first- and second-order convergence analysis)
CSV2	[17]	2013	Second-order trust-region
DFPP	[49, 50]	2014	Proximal point method
DEFT-FUNNEL	[82]	2015	SQP trust-region
CONORBIT	[79]	2017	Second-order trust-region for constrained optimization (adaptation of ORBIT)

19.2.1 Model-Based DFO for Nonsmooth Functions

The proof of convergence of each algorithm in Table 19.1 requires a smoothness assumption on the objective function. First-order methods, such as [49, 50, 71], use approximate gradients in the algorithm, and the proof of convergence requires that the approximate gradients asymptotically converge to the true gradients. This

immediately means the objective function must be continuously differentiable (i.e., $f \in \mathcal{C}^1$). Furthermore, assumptions that imply the approximate gradients asymptotically converge to the true gradients typically include that the gradients are locally Lipschitz continuous (i.e., $f \in \mathcal{C}^{1+}$). Second-order methods (the rest of Table 19.1) use approximate Hessians, and require assumptions that ensure these approximations are suitably well-behaved. In some cases, it is sufficient to assume the approximate Hessians are bounded in norm. In these cases, $f \in \mathcal{C}^{1+}$ may be enough to ensure convergence. In other cases, convergence requires the approximate Hessians to asymptotically converge to the true Hessian, which typically requires $f \in \mathcal{C}^{2+}$.

More recently, some research has begun exploring the notion of solving NSO via model-based DFO. The first published research on this (to the best of the authors' knowledge) appeared in 2008 [11]. In this work, Bagirov, Karasözen and Sezer presented a discrete gradient derivative-free method for unconstrained NSO problems. To understand further, it is necessary to remind the reader of the definition of the subdifferential for locally Lipschitz functions. To do so, we recall that if a function is locally Lipschitz, then it is differentiable almost everywhere.

Definition 19.1 (Subdifferential) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function. Let $D(f)$ be the set of points where f is differentiable. The *subdifferential* of f at the point $\boldsymbol{x} \in \mathbb{R}^n$ is defined

$$\partial f(\boldsymbol{x}) = \text{conv}\{\boldsymbol{v} \in \mathbb{R}^n : \text{there exists } \boldsymbol{y}^i \in D(f), \text{ such that } \boldsymbol{y}^i \to \boldsymbol{x}, \ \nabla f(\boldsymbol{y}^i) \to \boldsymbol{v}\},$$

where conv denotes the convex hull.

At a given incumbent solution $\boldsymbol{x}^k$, Bagirov, Karasözen and Sezer's approach constructs a large number of approximate gradients at points $\boldsymbol{y}^{k,i}$ near $\boldsymbol{x}^k$. The convex hull of the approximate gradients can then be used as an approximation of the subdifferential. The resulting approximation was used within a conjugate subgradient style algorithm. A similar approach [57] was published in 2010, where Kiwiel used a large number of approximate gradients to construct an approximate subdifferential and used the approximate subdifferential in a gradient sampling style algorithm.

Bagirov, Karasözen and Sezer's algorithm was implemented and tested under the name DGM. While the numerical tests show very high accuracy, they do not discuss the number of function evaluations required in each iteration. Kiwiel's work contains no numerical testing the result.

In DFO, the speed of convergence is typically measured in function evaluations, not time [13, 64]. This is done because, while academic test problems are extremely fast to compute, in many real-world applications each function evaluation takes a notable amount of time. For example, in [95], the authors consider the planning of bioequivalence studies in the pharmaceutical industry. In this problem, each function evaluation launches a Monte-Carlo simulation and requires approximately 3 min to

complete on a cluster of 6 computers. Thus, an algorithm that requires one million of function evaluations, would require almost 6 years to complete. In [95], the authors limit the number of function evaluations to 100 in order to get a solution within a 5 h time period.

With this in mind, let us reconsider the approaches of Bagirov, Karasözen and Sezer and of Kiwiel. Each approximate gradient requires $n + 1$ or $2n$ function evaluations (the first being from [11], the second from [57]). So, the natural question is, how many approximate gradients are needed to provide a reasonable approximation to the subdifferential?

A heuristic answer to this was given by Burke et al. [22], who suggested $2n$ gradients be sampled at each iteration. Another answer was given in [8], where the authors examined the question of how many exact gradients are needed to reconstruct a polyhedral subdifferential? In $\mathbb{R}^2$, the answer is three times the number of edges of the polytope. In $\mathbb{R}^n$, a precise answer is still unknown, but does depend on the number of faces.

The planning of bioequivalence studies problem from [95] has only $n = 4$ optimization variables. Therefore, each iteration of the approach of [11] or [57] would require $2n^2 = 32$ function evaluations, which corresponds to more than 1.5 h. So, only three complete iterations could be completed in the 5 h time period. A similar problem with $n = 8$ and 6 min function evaluations would not even complete a single iteration in 5 h. Clearly, these methods are intractable, even for small dimension problems, unless the time per function evaluation is negligible.

Happily, this is not the end of the story for model-based DFO of nonsmooth problems. Recognizing the difficulty of approximating the subdifferential of an arbitrary function, researchers have turned their attention to approximating the subdifferential of a structured function. In [47, 48], the authors considered finite-max functions, i.e., functions of the form $f(\boldsymbol{x}) = \max\{F_i(\boldsymbol{x}) : i = 1, 2, \ldots, N_f\}$ and each $F_i \in \mathcal{C}^2$. Finite-max functions do arise naturally in DFO problems; for example, an application in seismic retrofitting design is examined in [16].

The finite-max structure allows for a simple formula for the subdifferential:

$$f(\boldsymbol{x}) = \max\{F_i(\boldsymbol{x}) : i = 1, 2, \ldots, N_f\},\ F_i \in \mathcal{C}^2,\ \text{implies that}$$
$$\partial f(\boldsymbol{x}) = \operatorname{conv}\{\nabla F_i(\boldsymbol{x}) : F_i(\boldsymbol{x}) = f(\boldsymbol{x})\}.$$

Using this formula it is possible to approximate the subdifferential of a finite-max function using only $n + 1$ function evaluations, which allows for the application of these methods in practice. The details of this construction are presented in Sect. 19.4.

Another structured function was studied in [59]. In that paper, Larson, Menickelly, and Wild, considered problems where the objective function takes the form of an L_1-norm: $f(\boldsymbol{x}) = \sum_{i=1}^{N_f} |F_i(\boldsymbol{x})|$ and each $F_i \in \mathcal{C}^2$. In this case we have,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_f} |F_i(\boldsymbol{x})|,\ F_i \in \mathcal{C}^2,\ \text{implies that}\ \partial f(\boldsymbol{x}) = \sum_{i=1}^{N_f} \alpha_i \nabla F_i(\boldsymbol{x}),$$

where

$$\alpha_i = \begin{cases} 1, & F_i(\boldsymbol{x}) > 0, \\ [-1, 1], & F_i(\boldsymbol{x}) = 0, \\ -1, & F_i(\boldsymbol{x}) < 0. \end{cases}$$

Once again, using the simpler formula allows the ability to approximate the subdifferential using just $n + 1$ function evaluations. The technique has recently been extended to piecewise-linear functions [56].

In [47, 48, 56, 59], the authors focused on line-search and trust-region methods. However, the subdifferential approximation techniques therein can be employed with other algorithms. For example, in [12], the comirror algorithm for NSO is reexamined in light of the subdifferential approximation ideas in [48]. Another example is the recent work [51], where the authors examine the $\mathcal{VU}$-algorithm from [62] in the setting of DFO.

Table 19.2 summarizes the current model-based DFO algorithms for NSO.

In all of these cases, the trick to a successful model-based DFO algorithm for NSO is to focus on structured functions where the subdifferential can be approximated using a reasonable number of function evaluations. Section 19.4 returns to this statement, and discusses model building techniques for nonsmooth functions.

19.2.2 Model-Based DFO Used Within Direct Search

Before moving on to more technical details of model-based DFO, it is worth a small detour into the realm of direct search methods. In particular, we discuss some of the work that has considered applying model-based methods as subroutines in direct search methods.

Some direct search methods, such as generalized pattern search (GPS [88]) and mesh adaptive direct search (MADS [5]), are of a search-poll paradigm [20]. The poll step is a local exploration intended for theoretical and practical local

Table 19.2 Model-based DFO algorithms that are designed for NSO

Algorithm	Reference	Year	Smooth equivalent
DGM	[11]	2008	Conjugate subgradient
WASG	[47]	2012	Gradient sampling
RAGS	[48]	2013	Subgradient descent with line search
$\text{DFO}_{\text{comirror}}$	[12]	2015	Comirror
CMS, GDMS, DMS, SMS	[59]	2016	Manifold sampling
MS_4PL	[56]	2018	Manifold sampling
DFO_$\mathcal{VU}$	[51]	preprint	$\mathcal{VU}$-algorithm

Table 19.3 Uses of model-based DFO techniques with a direct search framework

Algorithm	Reference	Year	Smooth algorithm with similarities to model-based portion
Unconstrained or bound-constrained optimization			
IMFIL	[21]	1998	Gradient-based search
GPS	[33]	2007	Gradient-based line-search methods
SNOBFIT	[53]	2008	Sequential quadratic programming
GPS & MADS	[34]	2008	Gradient-based line-search methods Newton with diagonal Hessian approximation
SID-PSM	[35]	2010	Trust-region
MADS	[25]	2013	Trust-region
Constrained optimization			
DFN	[39]	2014	Projected linesearch with penalty function
MADS	[9]	2014	Line search
QPMADS	[23]	2015	Directional derivative-based Hessian update, linear models of the constraints
PBTR	[10]	2016	Trust-region
GOSAC	[66]	2017	Cubic radial basis function
LOWESS	[86]	2018	Locally weighted regression
MADS	[3]	2018	L_∞-norm trust-region

convergence. The search step is a global exploration in the space of variables, whose purpose is to identify promising regions. Model-based strategies have been incorporated into the search step for a better global exploration of the space of variables, as well as into the poll step to improve the efficiency of the local exploration. Table 19.3 lists some direct search methods that exploit simplex gradients or even simplex Hessians. This information is used in the search step with line search strategies, or with trust-region methods, and is also used in the poll step for opportunistic termination.

The algorithms listed in the top half of the table are for unconstrained or bound-constrained optimization. The algorithms on the bottom half are for problems with nonsmooth constraints, and they construct models of the objective function and of each constraint, or penalize constraints into the objective function. As with the previous tables, the last column indicates a connection with more classical smooth algorithms. But here, we are talking about similarities rather than equivalences.

19.3 Building Models of Smooth Functions

In order to study the mathematical details of model-based DFO for smooth optimization it is valuable to provide some background on model-based DFO for smooth optimization. As mentioned, the first step in model-based DFO is the construction of a model.

19.3.1 Linear Interpolation and the Simplex Gradient

For a smooth function, the simplest model is that of linear interpolation. The following definition presents conditions on the points used to build such model.

Definition 19.2 (Poised for Linear Interpolation) The set $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^n\} \subset \mathbb{R}^n$, is *poised*[4] *for linear interpolation* if the $(n+1) \times (n+1)$ matrix $\begin{bmatrix}\mathbf{1} & Y^T\end{bmatrix}$ is invertible, where $Y = \begin{bmatrix}\boldsymbol{y}^0 & \boldsymbol{y}^1 & \ldots & \boldsymbol{y}^n\end{bmatrix}$ and $\mathbf{1} \in \mathbb{R}^{n+1}$ is the vector of all ones.

The next definition defines the linear interpolation function.

Definition 19.3 (Linear Interpolation Function) Let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^n\} \subset \mathbb{R}^n$ be poised for linear interpolation and $f : \mathbb{R}^n \mapsto \mathbb{R}$. Then the *linear interpolation function of* f *over* $\mathbb{Y}$ is

$$L_Y(\boldsymbol{x}) := \alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{x},$$

where $(\alpha_0, \boldsymbol{\alpha})$ is the unique solution to $\begin{bmatrix}\mathbf{1} & Y^T\end{bmatrix} \begin{bmatrix}\alpha_0 \\ \boldsymbol{\alpha}\end{bmatrix} = f(\mathbb{Y})$.

The linear interpolation function provides a first-order approximation of the function f. As such, the gradient of the linear interpolation function can be used to check (approximately) descent directions and first-order stopping conditions.

Recall that a simplex in $\mathbb{R}^n$ is a bounded polytope with a non-empty interior and exactly $n+1$ vertices.[5] It is fairly easy to show that $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^n\} \subset \mathbb{R}^n$, is poised for linear interpolation if and only if conv($\mathbb{Y}$) is a simplex [7, Proposition 9.1]. As such, the gradient of a linear interpolation model has been named the *simplex gradient*.[6]

[4]The term *poised* was introduced by Sauer and Xu [83] in 1995 and imported into the context of DFO by Conn and Toint [24] in the following year.

[5]The term *simplex* was first used by Schoute [84] back in 1902. Since then, the simplices have arisen in numerous areas of optimization, largely because they are the simplest full dimensional polytope.

[6]The *simplex gradient* was introduced by Kelley [54] to monitor the performance of a modified Nelder-Mead algorithm.

Definition 19.4 (Simplex Gradient) Let $f : \mathbb{R}^n \mapsto \mathbb{R}$, and let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^n\} \subset \mathbb{R}^n$ be poised for linear interpolation. The *simplex gradient of f over $\mathbb{Y}$*, denoted $\nabla_S f(\mathbb{Y})$, is the gradient of the linear interpolation function L_Y of f over $\mathbb{Y}$ and defined by

$$\nabla_S f(\mathbb{Y}) := \nabla L_Y(\boldsymbol{x}) = \boldsymbol{\alpha}.$$

The quality of the linear interpolation model, and the simplex gradient, are analyzed in the following theorem.

Theorem 19.1 (Linear Interpolation Error Bounds) *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, and $f \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$ with the Lipschitz constant (of the gradient) given by K. Let $\mathbb{Y}$ be poised for linear interpolation with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\Delta = \Delta(\mathbb{Y}) \leq \bar{\Delta}$. Define*

$$\widehat{L} := \frac{1}{\Delta}\left[\boldsymbol{y}^1 - \boldsymbol{y}^0 \ \boldsymbol{y}^2 - \boldsymbol{y}^0 \ \ldots \ \boldsymbol{y}^n - \boldsymbol{y}^0\right].$$

Then the linear interpolation function $L(\boldsymbol{y}) = \alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}$ satisfies

$$\|f(\boldsymbol{y}) - L(\boldsymbol{y})\| \leq \left(\frac{1}{2}K(1 + \sqrt{n}\|\widehat{L}^{-1}\|)\right)\Delta^2 \quad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta). \tag{19.1}$$

Moreover, the simplex gradient $\boldsymbol{\alpha} = \nabla_S f(\mathbb{Y})$ satisfies

$$\|\nabla f(\boldsymbol{y}) - \boldsymbol{\alpha}\| \leq \left(\frac{1}{2}K\sqrt{n}\|\widehat{L}^{-1}\|\right)\Delta. \tag{19.2}$$

Furthermore,

$$\|\nabla f(\boldsymbol{y}) - \boldsymbol{\alpha}\| \leq \left(\frac{1}{2}K(2 + \sqrt{n}\|\widehat{L}^{-1}\|)\right)\Delta \quad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta). \tag{19.3}$$

Proof Proofs can be found in [30, Chapter 2], [55, Chapter 5], and [7, Chapter 9], for example. □

The complexity of computing simplex gradients has recently been studied in [31].

19.3.2 Fully Linear and Fully Quadratic Models

Theorem 19.1 shows that linear interpolation provides an accuracy similar to that of a 1st order Taylor expansion. In [91], Wild and Shoemaker formalize this property under the name *fully linear models*, although they give credit to Conn et al. [28] for inspiring the terminology. Indeed, the latter authors use the terms *fully linear, quadratic* and even *fully cubic* but do not provide a formal definition. They further

introduce the name *fully quadratic models* for models that provide an accuracy similar to that of a 2nd order Taylor expansion. These terms provide a convenient language for model-based DFO, so we now take a brief interlude from constructing models to formalize these definitions. We begin with fully linear.

Definition 19.5 (Class of Fully Linear Models) Given $f \in \mathcal{C}^1$, $\boldsymbol{x} \in \mathbb{R}^n$, and $\bar{\Delta} > 0$ we say that $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a *class of fully linear models* of f at $\boldsymbol{x}$ parameterized by Δ if there exists a pair of scalars $\kappa_f(\boldsymbol{x}) > 0$ and $\kappa_g(\boldsymbol{x}) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies

$$\begin{aligned} & |f(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{y})| \le \kappa_f(\boldsymbol{x})\Delta^2 && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x};\Delta), \\ \text{and} \quad & \|\nabla f(\boldsymbol{y}) - \tilde{g}_\Delta(\boldsymbol{y})\| \le \kappa_g(\boldsymbol{x})\Delta && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x};\Delta), \end{aligned}$$

where $\tilde{g}_\Delta = \nabla \tilde{f}_\Delta$.

The previous definition involves classes of models at a given $\boldsymbol{x}$. Let us now generalize the definition to classes that adapt to any $\boldsymbol{x} \in \mathbb{R}^n$.

Definition 19.6 (Fully Linear Models) Let $f \in \mathcal{C}^1$. We say that $\tilde{f}_\Delta$ *are fully linear models of* f to mean that, given any $\boldsymbol{x} \in \mathbb{R}^n$ and $\bar{\Delta} > 0$, there exists a class $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ of fully linear models of f at $\boldsymbol{x}$ parameterized by Δ.

We say that $\tilde{f}_\Delta$ *are fully linear models of* f *with constants* κ_f *and* κ_g to mean that, given any $\boldsymbol{x} \in \mathbb{R}^n$ and $\bar{\Delta} > 0$, there exists a class $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ of fully linear models of f at $\boldsymbol{x}$ parameterized by Δ and the scalars $\kappa_f(\boldsymbol{x})$ and $\kappa_g(\boldsymbol{x})$ can be taken as the constants: $\kappa_f(\boldsymbol{x}) = \kappa_f$ and $\kappa_g(\boldsymbol{x}) = \kappa_g$ (independent of $\boldsymbol{x}$).

Fully linear models provide a linear level of accuracy for approximating the function. If a higher accuracy is desired, then we demand *fully quadratic models*.

Definition 19.7 (Class of Fully Quadratic Models) Given $f \in \mathcal{C}^2$, $\boldsymbol{x} \in \mathbb{R}^n$, and $\bar{\Delta} > 0$ we say that $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a *class of fully quadratic models* of f at $\boldsymbol{x}$ parameterized by Δ if there exists scalars $\kappa_f(\boldsymbol{x}) > 0$, $\kappa_g(\boldsymbol{x}) > 0$, and $\kappa_h(\boldsymbol{x}) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies

$$\begin{aligned} & |f(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{y})| \le \kappa_f(\boldsymbol{x})\Delta^3 && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x};\Delta), \\ & \|\nabla f(\boldsymbol{y}) - \tilde{g}_\Delta(\boldsymbol{y})\| \le \kappa_g(\boldsymbol{x})\Delta^2 && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x};\Delta), \\ \text{and} \quad & \|\nabla^2 f(\boldsymbol{y}) - \tilde{h}_\Delta(\boldsymbol{y})\| \le \kappa_h(\boldsymbol{x})\Delta && \text{for all } \boldsymbol{y} \in B(\boldsymbol{x};\Delta), \end{aligned}$$

where $\tilde{g}_\Delta = \nabla \tilde{f}_\Delta$ and $\tilde{H}_\Delta = \nabla^2 \tilde{f}_\Delta$.

Again, we generalize to any $\boldsymbol{x} \in \mathbb{R}^n$.

Definition 19.8 (Fully Quadratic Models) Let $f \in \mathcal{C}^2$. We say that $\tilde{f}_\Delta$ *are fully quadratic models of* f to mean that, given any $\boldsymbol{x} \in \mathbb{R}^n$ and $\bar{\Delta} > 0$, there exists a class $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ of fully quadratic models of f at $\boldsymbol{x}$ parameterized by Δ.

We say that $\tilde{f}_\Delta$ *are fully quadratic models of* f *with constants* κ_f, κ_g, *and* κ_h to mean that, given any $\boldsymbol{x} \in \mathbb{R}^n$ and $\bar{\Delta} > 0$, there exists a class $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ of fully quadratic models of f at $\boldsymbol{x}$ parameterized by Δ and the scalars $\kappa_f(\boldsymbol{x})$, $\kappa_g(\boldsymbol{x})$, and $\kappa_h(\boldsymbol{x})$ can be taken as the constants: $\kappa_f(\boldsymbol{x}) = \kappa_f$, $\kappa_g(\boldsymbol{x}) = \kappa_g$, and $\kappa_h(\boldsymbol{x}) = \kappa_h$ (independent of $\boldsymbol{x}$).

Basically, fully linear models enjoy a 1st order Taylor-like behaviour and fully quadratic models enjoy a 2nd order Taylor-like behaviour.

The value of this terminology is that it removes the need to define the model building process within a model-based DFO algorithm. Instead, algorithms can begin with the assumption that a class of fully linear or quadratic models is available, and then prove convergence using Definition 19.5 or 19.7 as appropriate. We will return to this in Sect. 19.3.5, but for now we return attention to the construction of models.

19.3.3 The Generalized Simplex Gradient

One weakness of the simplex gradient is its dependance on the simplex. In particular, $\mathbb{Y}$ must contain exactly well-poised $n+1$ points. Custódio et al. [34] and Regis [78] present a more general framework that may be applied when the number of points is not $n+1$. Suppose $\mathbb{Y}$ contains $m+1$ elements and consider the linear system

$$L^T \boldsymbol{\alpha} = \boldsymbol{\delta}^{f(\mathbb{Y})} \tag{19.4}$$

where

$$L = \left[\boldsymbol{y}^1 - \boldsymbol{y}^0 \;\; \boldsymbol{y}^2 - \boldsymbol{y}^0 \;\; \ldots \;\; \boldsymbol{y}^m - \boldsymbol{y}^0\right] \in \mathbb{R}^{n \times m} \quad \text{and}$$

$$\boldsymbol{\delta}^{f(\mathbb{Y})} = \left[f(\boldsymbol{y}^1) - f(\boldsymbol{y}^0) \;\; f(\boldsymbol{y}^2) - f(\boldsymbol{y}^0) \;\; \ldots \;\; f(\boldsymbol{y}^m) - f(\boldsymbol{y}^0)\right]^T \in \mathbb{R}^n.$$

If $m = n$ and if $\mathbb{Y}$ is a simplex, then the system is *determined* and the solution $\boldsymbol{\alpha}$ to Eq. (19.4) is unique and is the simplex gradient from Definition 19.4. If $m < n$, then the system is *underdetermined*, so has an infinite set of solutions. This can be resolved by seeking the solution with minimal norm

$$\text{if } m < n, \text{ then solve } \boldsymbol{\alpha} \in \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \{\|\boldsymbol{\alpha}\|^2 : L^T \boldsymbol{\alpha} = \boldsymbol{\delta}^{f(\mathbb{Y})}\}.$$

If $m > n$, then the system is *overdetermined*, so may not have a solution. This can be resolved by seeking the least square solution

$$\text{if } m > n, \text{ then solve } \boldsymbol{\alpha} \in \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \{\|L^T \boldsymbol{\alpha} - \boldsymbol{\delta}^{f(\mathbb{Y})}\|^2\}.$$

In all three cases, the solution can be written as $\boldsymbol{\alpha} = (L^\dagger)^T \boldsymbol{\delta}^{f(\mathbb{Y})}$, where $L^\dagger$ is the Moore–Penrose pseudoinverse of the matrix L [34, 78].

Since a the resulting generalization no longer requires $\mathbb{Y}$ to form a simplex, the term simplex gradient no longer seems appropriate.

Definition 19.9 (Generalized Simplex Gradient) Let $f : \mathbb{R}^n \mapsto \mathbb{R}$, and let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$ be an ordered set with $m \geq 1$. The *generalized simplex gradient of f over $\mathbb{Y}$*, denoted $\nabla_S f(\mathbb{Y})$, is given by

$$\nabla_S f(\mathbb{Y}) := (L^\dagger)^T \boldsymbol{\delta}^{f(\mathbb{Y})},$$

where

$$L = \left[\boldsymbol{y}^1 - \boldsymbol{y}^0 \;\; \boldsymbol{y}^2 - \boldsymbol{y}^0 \;\; \ldots \;\; \boldsymbol{y}^m - \boldsymbol{y}^0\right] \quad \text{and} \quad \boldsymbol{\delta}^{f(\mathbb{Y})} = \begin{bmatrix} f(\boldsymbol{y}^1) - f(\boldsymbol{y}^0) \\ f(\boldsymbol{y}^2) - f(\boldsymbol{y}^0) \\ \vdots \\ f(\boldsymbol{y}^m) - f(\boldsymbol{y}^0) \end{bmatrix}.$$

The above definition leads to a generalization of the inequality (19.2).

Theorem 19.2 (Generalized Simplex Gradient Error Bounds) *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, and $f \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$ with Lipschitz constant (of the gradient) given by K. Let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\}$ with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\Delta = \Delta(\mathbb{Y}) \leq \bar{\Delta}$. Suppose $\mathbb{Y}$ is* poised *in the sense that $Y = [\boldsymbol{y}^0 \;\; \boldsymbol{y}^1 \;\; \ldots \;\; \boldsymbol{y}^m]$ is full-rank. Let $\widehat{U}(\mathbb{Y})\widehat{\Sigma}(\mathbb{Y})\widehat{V}(\mathbb{Y})^T$ be the* reduced *singular-value decomposition*[7] *of $\widehat{L}^T$ and define*

$$\tilde{V}(\mathbb{Y}) = \begin{cases} I_n, & \text{if } m \geq n, \\ \widehat{V}(\mathbb{Y}), & \text{if } m < n. \end{cases}$$

Then, the generalized simplex gradient $\nabla_S f(\mathbb{Y})$ satisfies

$$\|\tilde{V}(\mathbb{Y})^T \left[\nabla f(\boldsymbol{y}^0) - \nabla_S f(\mathbb{Y})\right]\| \leq \left(\frac{1}{2} K \sqrt{m} \|\widehat{\Sigma}(\mathbb{Y})^{-1}\|\right) \Delta. \tag{19.5}$$

Consequently,

$$\|\tilde{V}(\mathbb{Y})^T \left[\nabla f(\boldsymbol{y}) - \nabla_S f(\mathbb{Y})\right]\| \leq \left(\frac{1}{2} K (2 + \sqrt{m} \|\widehat{\Sigma}(\mathbb{Y})^{-1}\|)\right) \Delta \tag{19.6}$$

for all $\boldsymbol{y} \in B(\boldsymbol{x}; \Delta)$

[7] The reduced SVD of the matrix $\widehat{L}^T \in \mathbb{R}^{m \times n}$ of rank r is such that $\widehat{U}(\mathbb{Y}) \in \mathbb{R}^{m \times r}$ is an orthonormal basis for the column space, $\widehat{\Sigma}(\mathbb{Y}) \in \mathbb{R}^{r \times r}$ is a nonsingular diagonal matrix with positive entries, and $\widehat{V}(\mathbb{Y})^T \in \mathbb{R}^{n \times r}$ is an orthonormal basis for the row space.

Proof Proof of Eq. (19.5) can be found in [78, Corollary 1].[8] Proof of Eq. (19.6) is a trivial adaptation of the proof of Eq. (9.6) in [7, Theorem 9.5]. □

Comparing Theorem 19.2 to Theorem 19.1 we consider three cases.

(i) $(m = n)$ If $m = n$, then we are in the determined case. The statement, "$Y = [\boldsymbol{y}^0\ \boldsymbol{y}^1\ \ldots\ \boldsymbol{y}^m]$ is full-rank" is exactly equivalent to $\mathbb{Y}$ is poised for linear interpolation [7, Proposition 9.1]. The matrix $\tilde{V}(\mathbb{Y})^T$ is the identity, and $\|\widehat{L}(\mathbb{Y})^{-1}\| = \|\widehat{\Sigma}(\mathbb{Y})^{-1}\|$. Thus, the gradient error in Theorem 19.1 is reproduced by Theorem 19.2.

(ii) $(m < n)$ If $m < n$, then we are in the underdetermined case. The statement "Y is full-rank" now means that $\text{rank}(Y) = m + 1$. In this case $\tilde{V}(\mathbb{Y})^T = \widehat{V}(\mathbb{Y})^T$ and the gradient error in Theorem 19.1 is not reproduced.

Indeed, consider an example with $n = 2$ and $m = 1$ on $f(\boldsymbol{x}) = \beta(x_1 + x_2)$. Define the sample set $\mathbb{Y} = \{\Delta\boldsymbol{e}_1, \Delta\boldsymbol{e}_2\}$, where $\boldsymbol{e}_i$ are the coordinate vectors and $\Delta > 0$. In this case, $L = [\Delta(\boldsymbol{e}_1 - \boldsymbol{e}_2)] = [\Delta\ \ -\Delta]^T$ and $\boldsymbol{\delta}^{f(\mathbb{Y})} = [0]$. Therefore, the generalized simplex gradient is $\nabla_S f(\mathbb{Y}) = \mathbf{0}$, regardless of Δ. As $\nabla f(\boldsymbol{x}) = [\beta\ \ \beta]^T$, $\|\nabla f(\boldsymbol{y}^0) - \nabla_S f(\mathbb{Y})\| = \sqrt{2}\beta$, regardless of Δ. So the gradient approximation will never become accurate. To understand the error bound, note that the reduced singular-value decomposition of $\widehat{L}^T = [1\ \ -1]$ is

$$\widehat{U} = [1], \quad \widehat{\Sigma} = [\sqrt{2}], \quad \widehat{V}^T = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \end{bmatrix}.$$

Now examine

$$\left\| \tilde{V}(\mathbb{Y})^T \left[\nabla f(\boldsymbol{y}^0) - \nabla_S f(\mathbb{Y}) \right] \right\| = \left\| \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} \beta \\ \beta \end{bmatrix} \right\| = 0.$$

The effect of $\tilde{V}(\mathbb{Y})^T$ in the error bound is to project the vectors onto the subspace spanned by the matrix $\widehat{L}^T$. Thus, the error bound is saying, the approximate gradient is sufficient to provide approximate directional derivatives in directions parallel to the span$(\widehat{L}^T)$. In directions not parallel to span$(\widehat{L}^T)$, the directional derivatives created using $\nabla_S f(\mathbb{Y})$ are uncontrolled.

(iii) $(m > n)$ If $m > n$, then we are in the overdetermined case. The matrix $\tilde{V}(\mathbb{Y})^T$ is the identity, so we have a direct bound on the error in the gradient approximation, similar but not the same as Theorem 19.1. In particular, notice the error bound contains $\sqrt{m}$, instead of $\sqrt{n}$ as in Theorem 19.1. This implies that as the number of sample points increases the error may actually get worse.

Note, in cases (i) and (iii) above, we carefully state that we achieve a bound on the error in the gradient approximation. This is not quite equivalent to creating a fully

[8] Note that this result is presented in [34, Theorem 2.1] and in [32, Theorem 4.1.1] without proof.

linear model. Fully linear models also require the models to approximate function values and an error bound on the quality of those approximations. Conveniently however, a good gradient approximation along with the exact function value is sufficient to create a fully linear model.

Proposition 19.1 *Suppose $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, $f \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$ with Lipschitiz constant (of the gradient) given by K. Suppose that the model functions $\tilde{f}_\Delta \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$ with Lipschitz constant (of the gradient) given by $\tilde{K}$. Suppose there exists $\kappa_g > 0$ such that the gradients $\{\nabla \tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ satisfy*

$$\|\nabla f(\boldsymbol{y}) - \nabla \tilde{f}_\Delta(\boldsymbol{y})\| \leq \kappa_g(\boldsymbol{x})\Delta \qquad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta).$$

Define $\hat{f}_\Delta = \tilde{f}_\Delta - \tilde{f}_\Delta(\boldsymbol{x}) + f(\boldsymbol{x})$. Then $\{\hat{f}_\Delta\}$ is a class of fully linear models of f at $\boldsymbol{x}$. In particular, if $\tilde{f}_\Delta(\boldsymbol{x}) = f(\boldsymbol{x})$ for every $\Delta \in (0, \bar{\Delta}]$, then $\{\tilde{f}_\Delta\}$ is a class of fully linear models.

Proof We need to show the existence of $\kappa_f(\boldsymbol{x})$ such that

$$|f(\boldsymbol{y}) - \hat{f}_\Delta(\boldsymbol{y})| \leq \kappa_f(\boldsymbol{x})\Delta^2 \qquad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta).$$

Since that $\hat{f}_\Delta(\boldsymbol{x}) = f(\boldsymbol{x})$, we have that

$$\begin{aligned} |f(\boldsymbol{y}) - \hat{f}_\Delta(\boldsymbol{y})| \leq\ & |f(\boldsymbol{y}) - f(\boldsymbol{x}) - \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})| \\ & + |\nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) - \nabla \hat{f}_\Delta(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})| \\ & + |\hat{f}_\Delta(\boldsymbol{x}) + \nabla \hat{f}_\Delta(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) - \hat{f}_\Delta(\boldsymbol{y})|. \end{aligned} \tag{19.7}$$

Applying $f \in \mathcal{C}^{1+}$ and $\tilde{f}_\Delta \in \mathcal{C}^{1+}$, we note that $\hat{f}_\Delta \in \mathcal{C}^{1+}$, and therefore the 1st order Taylor approximation error bound holds [7, Lemma 9.4]. In particular,

$$\begin{aligned} |f(\boldsymbol{y}) - f(\boldsymbol{x}) - \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})| &\leq \tfrac{1}{2}K\|\boldsymbol{y} - \boldsymbol{x}\|^2 \\ &\leq \tfrac{1}{2}K\Delta^2 \qquad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta), \text{ and} \\ |\hat{f}_\Delta(\boldsymbol{x}) + \nabla \hat{f}_\Delta(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) - \hat{f}_\Delta(\boldsymbol{y})| &\leq \tfrac{1}{2}\tilde{K}\|\boldsymbol{y} - \boldsymbol{x}\|^2 \\ &\leq \tfrac{1}{2}\tilde{K}\Delta^2 \qquad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta). \end{aligned}$$

Examining the middle term of inequality (19.7) and using our gradient error bound, we find that

$$\begin{aligned} |\nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) - \nabla \hat{f}_\Delta(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})| &\leq \|\nabla f(\boldsymbol{x}) - \nabla \hat{f}_\Delta(\boldsymbol{x})\|\|\boldsymbol{y} - \boldsymbol{x}\| \\ &\leq \kappa_g(\boldsymbol{x})\Delta\|\boldsymbol{y} - \boldsymbol{x}\| \quad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta) \\ &\leq \kappa_g(\boldsymbol{x})\Delta^2 \qquad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta). \end{aligned}$$

Summing these approximations, we see that $\kappa_f(\boldsymbol{x}) = (K + \tilde{K})/2 + \kappa_g(\boldsymbol{x})$ satisfies the desired inequality. □

19.3.4 Quadratic Models

The framework of generalized simplex gradients, combined with Proposition 19.1, provides a straight-forward way to generate fully linear models. However, the error bound in Theorem 19.2 includes a "$\sqrt{m}$" term, which suggests that using more points will not necessarily improve the model quality. Moreover, when the number of sample points becomes sufficiently large, it may become possible to build quadratic models.

For quadratic models, we consider model functions of the form

$$Q(\boldsymbol{x}) = \alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T H \boldsymbol{x}, \quad \text{with } H = H^T.$$

As such, we seek $\alpha_0 \in \mathbb{R}$, $\boldsymbol{\alpha} \in \mathbb{R}^n$, and $H \in \mathbb{R}^{n\times n}$. Enforcing $H = H^T$ implies we need $1 + n + n(n+1)/2 = \frac{1}{2}(n+1)(n+2)$ well-poised sample points. We next formalize the definition of being poised for quadratic interpolation.

Definition 19.10 (Poised for Quadratic Interpolation) The set $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$ with $m = \frac{1}{2}(n+1)(n+2) - 1$, is *poised for quadratic interpolation* if the system

$$\alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}^i + \frac{1}{2}(\boldsymbol{y}^i)^T H \boldsymbol{y}^i = 0, \quad i = 0, 1, 2, \ldots m$$

has a unique (trivial) solution for α_0, $\boldsymbol{\alpha}$, and $H = H^T$.

Being poised for linear or quadratic interpolation has a nice geometrical interpretation. A $(n+1) \times (n+1)$ linear system of equations $\alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}^i = f(\boldsymbol{y}^i)$, $i \in \{0, 1, \ldots, n\}$ has a unique solution unless the points are collinear. That is, the set $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^n\} \subset \mathbb{R}^n$ is **not** poised for linear interpolation if and only if the points all lie on the level surface of a single nontrivial linear function. Similarly, a linear system of equations $\alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}^i + \frac{1}{2}(\boldsymbol{y}^i)^T H \boldsymbol{y}^i = f(\boldsymbol{y}^i)$, $i \in \{0, 1, \ldots, \frac{1}{2}(n+1)(n+2) - 1\}$ has a unique solution unless the points all lie on the level surface of a single nontrivial quadratic function. That is, the set $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n, m = \frac{1}{2}(n+1)(n+2) - 1$, is **not** poised for quadratic interpolation if and only if the points all lie on the level surface of a single nontrivial quadratic function. Figure 19.1 illustrates the notion of being poised for linear or for quadratic interpolation in the case where $n = 2$.

The figure illustrates that it can be difficult to visually confirm whether or not a set is poised for quadratic interpolation. Fortunately, the algebraic test is still straight-forward.

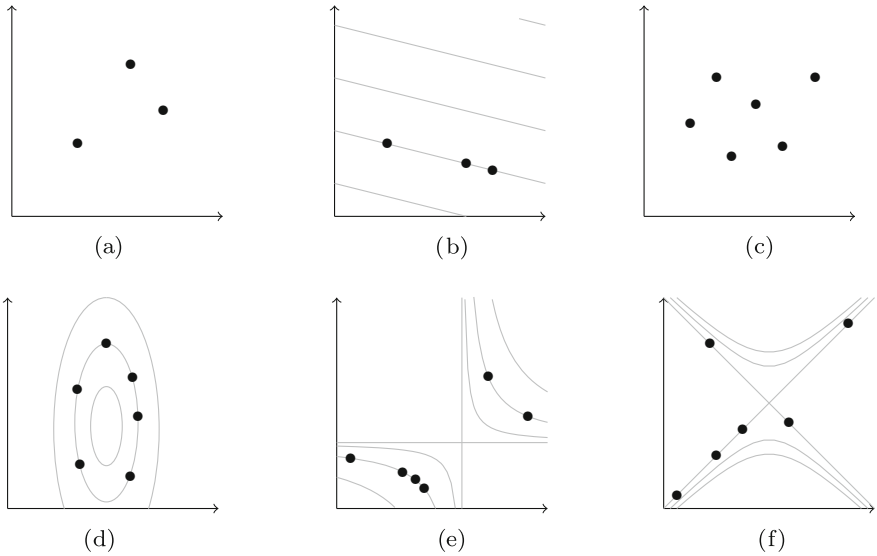

Fig. 19.1 Poised and non-poised sets for interpolation (figure inspired from [7]). (**a**) Poised for LI. (**b**) Not poised for LI. (**c**) Poised for QI. (**d**) Not poised for QI. (**e**) Not poised for QI. (**f**) Not poised for QI

Given a set that is poised for quadratic interpolation, we define the quadratic interpolation function.

Definition 19.11 (Quadratic Interpolation Function) Let $f : \mathbb{R}^n \mapsto \mathbb{R}$, and let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$ with $m = \frac{1}{2}(n+1)(n+2) - 1$, be poised for quadratic interpolation. Then the *quadratic interpolation function* of f over $\mathbb{Y}$ is

$$Q_Y(\boldsymbol{x}) := \alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T H \boldsymbol{x},$$

where $(\alpha_0, \boldsymbol{\alpha}, H = H^T)$ is the unique solution to

$$\alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}^i + \frac{1}{2}(\boldsymbol{y}^i)^T H \boldsymbol{y}^i = f(\boldsymbol{y}^i), \quad i = 0, 1, 2, \ldots, m.$$

In [28], Conn, Scheinberg and Vicente show that quadratic interpolation resulted in a class of fully quadratic models.

Theorem 19.3 (Quadratic Interpolation is Fully Quadratic) *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, and $f \in \mathcal{C}^{2+}$ on $B(\boldsymbol{x}; \bar{\Delta})$. Let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$ be poised for quadratic interpolation with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\Delta = \Delta(\mathbb{Y}) \leq \bar{\Delta}$.*

For $0 < \delta \leq 1$ define $\mathbb{Y}_\delta = \{\boldsymbol{y}^0, \boldsymbol{y}^0 + \delta(\boldsymbol{y}^1 - \boldsymbol{y}^0), \ldots, \boldsymbol{y}^0 + \delta(\boldsymbol{y}^m - \boldsymbol{y}^0)\}$. Then $\mathbb{Y}_\delta$ is poised for quadratic interpolation with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\Delta(\mathbb{Y}_\delta) = \delta\Delta < \bar{\Delta}$.

Let Q_δ be the quadratic interpolation function of f over $\mathbb{Y}_\delta$. Then $\{Q_\delta\}_{\delta\in(0,1]}$ defines a class of fully quadratic models at $\boldsymbol{x}$ parametrized by δ.

Proof Details can be found in [28]. □

In Sect. 19.3.3 we saw one method for dealing with sample sets with greater than $n+1$ points. Comparing Theorem 19.2 to Theorem 19.3 makes it clear that if the number of sample points reaches $\frac{1}{2}(n+1)(n+2)$, then quadratic interpolation should become the preferred option. However, if $n+1 < m+1 < \frac{1}{2}(n+1)(n+2)$ has $m+1$ closer to $\frac{1}{2}(n+1)(n+2)$ than $n+1$, then it still might be reasonable to try and squeeze some curvature information out of the data. To do this we turn to minimum Frobenius norm models.

Consider the situation where $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\}$ and $n+1 < m+1 < (n+1)(n+2)/2$. There are not enough sample points to build a unique quadratic model of f. Indeed, there can be infinitely many quadratic functions parameterized by $\alpha_0, \boldsymbol{\alpha}, H = H^T$ that satisfy

$$\alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}^i + \frac{1}{2}(\boldsymbol{y}^i)^T H \boldsymbol{y}^i = f(\boldsymbol{y}^i) \quad \text{for } i = 0, 1, 2, \ldots, m.$$

Among all these quadratic functions, we select the one whose quadratic coefficients $h_{i,j}$ of H are as small as possible in the Frobenius norm sense.[9] As such, we solve the following quadratic optimization problem

$$\begin{cases} \text{minimize} & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} h_{i,j}^2 \\ \text{subject to} & \alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{y}^i + \frac{1}{2}(\boldsymbol{y}^i)^T H \boldsymbol{y}^i = f(\boldsymbol{y}^i) \quad \text{for } i = 0, 1, 2, \ldots, m, \\ & H = H^T \in \mathbb{R}^{n\times n},\ \alpha_0 \in \mathbb{R},\ \boldsymbol{\alpha} \in \mathbb{R}^n. \end{cases} \tag{19.8}$$

Thus, we have the idea of *minimum Frobenius norm* models. The following definition extends the notion of a set of points being poised for Frobenius norm modelling.

Definition 19.12 (Poised for Minimum Frobenius Norm Modelling) The set $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$ with $n < m < \frac{1}{2}(n+1)(n+2) - 1$, is *poised for minimum Frobenius norm modelling* if the problem (19.8) has a unique solution $(\alpha_0, \boldsymbol{\alpha}, H)$.

Given a set of poised points, we next introduce the minimum Frobenius norm model.

[9] This Frobenius norm approach was suggested by Conn et al. [26] in 1998, by Powell [74] for the UOBYQA algorithm, and by Custódio et al. [35] for SID-PSM.

Definition 19.13 (Minimum Frobenius Norm Model Function) Let $f : \mathbb{R}^n \mapsto \mathbb{R}$, and $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$, $n < m < \frac{1}{2}(n+1)(n+2) - 1$, be poised for minimum Frobenius norm modelling. Then the *minimum Frobenius norm model function of f over $\mathbb{Y}$* is

$$M_Y(\boldsymbol{x}) := \alpha_0 + \boldsymbol{\alpha}^T \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T H \boldsymbol{x},$$

where $(\alpha_0, \boldsymbol{\alpha}, H)$ is the unique solution to the problem (19.8).

In these models, the number of points exceeds the required number of points for linear interpolation, but is not sufficient for quadratic interpolation. Therefore, the error bounds for minimum Frobenius norm models are not as strong as for quadratic interpolation, but at least as strong as those for linear interpolation models.

Theorem 19.4 (Minimum Frobenius Norm Modelling is Fully Linear) *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, and $f \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$. Let $\mathbb{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subset \mathbb{R}^n$ be poised for minimum Frobenius norm modelling with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\Delta = \Delta(\mathbb{Y}) \leq \bar{\Delta}$.*

For $0 < \delta \leq 1$ define $\mathbb{Y}_\delta = \{\boldsymbol{y}^0, \boldsymbol{y}^0 + \delta(\boldsymbol{y}^1 - \boldsymbol{y}^0), \ldots, \boldsymbol{y}^0 + \delta(\boldsymbol{y}^m - \boldsymbol{y}^0)\}$. Then $\mathbb{Y}_\delta$ is poised for minimum Frobenius norm modelling with $\boldsymbol{y}^0 = \boldsymbol{x}$ and $\Delta(\mathbb{Y}_\delta) = \delta\Delta < \bar{\Delta}$.

Let M_δ be the minimum Frobenius norm model function of f over $\mathbb{Y}_\delta$. Then $\{M_\delta\}_{\delta \in (0,1]}$ defines a class of fully linear models at $\boldsymbol{x}$ parametrized by δ.

Proof Proof of this result is found in [35]. □

On a final note, the particular case in which $m = n$ is such that $\mathbb{Y}$ forms a simplex. Quadratic interpolation with minimum Frobenius norm produces the same solution as linear interpolation because the optimal solution will set H to the zero matrix.

19.3.5 Using Fully Linear or Quadratic Models in a DFO Algorithm

Now that we have established the ability to construct fully linear and fully quadratic models, let us examine how they can be used within DFO algorithms. As mentioned, the value of the fully linear and fully quadratic terminology is that it decouples the algorithm from the model building process. This is perhaps best illustrated via an example. Therefore, we present the MODEL-BASED TRUST-REGION framework from [7, Chapter 11] (which is minor adaption of the framework in [30, Chapter 10]). As mentioned, many of the algorithms in Table 19.1 fit into this framework.

In examining the MODEL-BASED TRUST-REGION framework note that there is no mention of how the fully linear models are constructed. Convergence is established using the error bounds in Definition 19.5. This in done through two procedures imbedded within the algorithm.

Algorithm 19.1: Model-based trust-region

Data: Given $f \in \mathcal{C}^1$, starting point $\boldsymbol{x}^0$ and fully linear models $\tilde{f}_\Delta$ of f.

Step 0. (*Initialization*) Initialize $\Delta^0 \in (0, \infty)$ initial trust-region radius, $\tilde{f}^0$ initial model (a fully linear model of f on $B(\boldsymbol{x}^0; \Delta^0)$), μ model accuracy parameter, $\eta \in (0, 1)$ sufficient decrease test parameter, $\gamma \in (0, 1)$ trust-region update parameter, $\epsilon_{\texttt{stop}} \in [0, \infty)$ stopping tolerance, and $k \leftarrow 0$ iteration counter.

Step 1. (*Model building and accuracy check*) If $\Delta^k > \mu \|\nabla \tilde{f}^k(\boldsymbol{x}^k)\|$ or $\tilde{f}^k$ is not a fully linear model of f on $B(\boldsymbol{x}^k; \Delta^k)$, then

- decrease $\Delta^{k+1} = \gamma \Delta^k$, set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$;
- use $\tilde{f}_\Delta$ to create a fully linear model $\tilde{f}^{k+1}$ of f on $B(\boldsymbol{x}^k; \Delta^{k+1})$;
- increment $k \leftarrow k + 1$ and go to Step 1.

Step 2. (*Stopping criterion*)
If $\Delta^k < \epsilon_{\texttt{stop}}$ and $\|\nabla \tilde{f}^k(\boldsymbol{x}^k)\| < \epsilon_{\texttt{stop}}$, then

- declare algorithm success and **stop**.

Step 3. (*Trust-region subproblem*) Solve (or approximate)

$$\boldsymbol{x}^k_{\texttt{tmp}} \in \operatorname*{argmin}_{\boldsymbol{x}} \{\tilde{f}^k(\boldsymbol{x}) \ : \ \boldsymbol{x} \in B(\boldsymbol{x}^k; \Delta^k)\}.$$

Step 4. (*Candidate test and trust-region update*) Evaluate $f(\boldsymbol{x}^k_{\texttt{tmp}})$ and compute the ratio

$$\rho^k = \frac{f(\boldsymbol{x}^k) - f(\boldsymbol{x}^k_{\texttt{tmp}})}{\tilde{f}^k(\boldsymbol{x}^k) - \tilde{f}^k(\boldsymbol{x}^k_{\texttt{tmp}})}.$$

If $\rho^k > \eta$ (sufficient decrease exists), declare iterate success and

- let $\boldsymbol{x}^{k+1}$ be any point such that $f(\boldsymbol{x}^{k+1}) \leq f(\boldsymbol{x}^k_{\texttt{tmp}})$;
- increase next trust-region $\Delta^{k+1} = \gamma^{-1} \Delta^k$;
- create $\tilde{f}^{k+1}$ using $\tilde{f}^k$ and $\boldsymbol{x}^{k+1}$.

Otherwise $\rho^k \leq \eta$, declare iterate failure and

- set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$, decrease trust-region $\Delta^{k+1} = \gamma \Delta^k$ and keep $\tilde{f}^{k+1} = \tilde{f}^k$.

Increment $k \leftarrow k + 1$ and go to Step 1.

In Step 1, the algorithm checks if $\Delta^k \leq \mu \|\nabla \tilde{f}^k(\boldsymbol{x}^k)\|$. If this verification fails, then the trust-region radius Δ^k is decreased and a new model is constructed. As a result, the algorithm links the model accuracy with the first-order critical conditions: higher accuracy is demanded as a potential critical point is approached.

In Step 2, the algorithm checks if $\Delta^k < \epsilon_{\texttt{stop}}$ and $\|\nabla \tilde{f}^k(\boldsymbol{x}^k)\| < \epsilon_{\texttt{stop}}$. Thus, the algorithm should only stop when both model accuracy and the approximated gradient are sufficiently small. If both conditions are true, then applying the fully linear models ensures that

$$\|\nabla f(\boldsymbol{x}^k)\| \leq \|\nabla f(\boldsymbol{x}^k) - \nabla \tilde{f}^k(\boldsymbol{x}^k)\| + \|\nabla \tilde{f}^k(\boldsymbol{x}^k)\| \leq \kappa_g \Delta^k + \epsilon_{\texttt{stop}} \leq (\kappa_g + 1)\epsilon_{\texttt{stop}},$$

and an upper bound on the true gradient is obtained.

Using these two precautions, other smooth optimization algorithms can also be adapted to work with fully linear models. Gradient-based line-search methods [7, Chapter 10], [30, Chapter 9], and [71], Newton's method [61], the proximal point method [49], and SQP [82] have been adapted to work for DFO. Of these, [7, Chapter 10], [30, Chapter 9], and [49] designed the algorithm to be independent of the modelling technique. As such, these algorithms are better termed frameworks, as changing the modelling technique may drastically alter the behaviour of the algorithm [50].

As future researchers explore model-based algorithms for DFO, we recommend using language that decouples the algorithm and the model building technique.

19.4 Accuracy and Models of Nonsmooth Functions

19.4.1 Accuracy of Nonsmooth Functions

While Definitions 19.6 and 19.8 (fully linear and fully quadratic) provide a useful terminology discussing models of smooth functions, they depend on the true gradients and true Hessians to provide the marker for comparison. As such, if the objective function f is nonsmooth, then it is impossible to construct a class of fully linear approximations. We therefore introduce a new terminology that allows us to appraise the quality of a model. We will begin with *order N function accuracy*.

Definition 19.14 (Order *N* Function Accuracy) Given $f, \boldsymbol{x} \in \text{dom}(f)$, and $\bar{\Delta} > 0$ we say that $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of models of f at $\boldsymbol{x}$ parameterized by Δ that *provides order N function accuracy at* $\boldsymbol{x}$ if there exists a scalar $\kappa_f(\boldsymbol{x}) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies

$$|f(\boldsymbol{x}) - \tilde{f}_\Delta(\boldsymbol{x})| \leq \kappa_f(\boldsymbol{x})\Delta^N.$$

We say that $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of models of f at $\boldsymbol{x}$ parameterized by Δ that *provides order N function accuracy near* $\boldsymbol{x}$ if there exists a scalar $\kappa_f(\boldsymbol{x}) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies

$$|f(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{y})| \leq \kappa_f(\boldsymbol{x})\Delta^N \qquad \text{for all } \boldsymbol{y} \in B(\boldsymbol{x}; \Delta).$$

Clearly, if $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of fully linear models, then it is also a class of models that provides order 2 function accuracy near $\boldsymbol{x}$. Similarly, if $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a

class of fully quadratic models, then it is also a class of models that provides order 3 function accuracy near $\boldsymbol{x}$.

Let us now introduce the definition of subgradient accuracy of a Lipschitz function. The definition relies on the notion of the Clarke subdifferential given in Definition 19.1. The elements of the subdifferential are called subgradients.

Definition 19.15 (Order N Subgradient Accuracy) Given $f \in \mathcal{C}^{0+}$, $\boldsymbol{x} \in \mathbb{R}^n$, and $\bar{\Delta} > 0$ we say that $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of models of f at $\boldsymbol{x}$ parameterized by Δ that *provides order N subgradient accuracy at $\boldsymbol{x}$* if there exists a scalar $\kappa_g(\boldsymbol{x}) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies

(i) given any $\boldsymbol{v} \in \partial f(\boldsymbol{x})$ there exists $\tilde{\boldsymbol{v}} \in \partial \tilde{f}_\Delta(\boldsymbol{x})$ with $\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \leq \kappa_g(\boldsymbol{x})\Delta^N$; and
(ii) given any $\tilde{\boldsymbol{v}} \in \partial \tilde{f}_\Delta(\boldsymbol{x})$ there exists $\boldsymbol{v} \in \partial f(\boldsymbol{x})$ with $\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \leq \kappa_g(\boldsymbol{x})\Delta^N$.

We say that $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of models of f at $\boldsymbol{x}$ parameterized by Δ that *provides order N subgradient accuracy near $\boldsymbol{x}$* if there exists a scalar $\kappa_g(\boldsymbol{x}) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies

(i) for all $\boldsymbol{y} \in B(\boldsymbol{x}; \Delta)$ given any $\boldsymbol{v} \in \partial f(\boldsymbol{y})$ there exists $\tilde{\boldsymbol{v}} \in \partial \tilde{f}_\Delta(\boldsymbol{y})$ with $\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \leq \kappa_g(\boldsymbol{x})\Delta^N$; and
(ii) for all $\boldsymbol{y} \in B(\boldsymbol{x}; \Delta)$ given any $\tilde{\boldsymbol{v}} \in \partial \tilde{f}_\Delta(\boldsymbol{y})$ there exists $\boldsymbol{v} \in \partial f(\boldsymbol{y})$ with $\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \leq \kappa_g(\boldsymbol{x})\Delta^N$.

If $f \in \mathcal{C}^{1+}$ and $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of fully linear models, then it is also a class of models that provides order 1 subgradient accuracy near $\boldsymbol{x}$. Similarly, if $f \in \mathcal{C}^{1+}$ and $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of fully quadratic models, then it is also a class of models that provides order 2 subgradient accuracy near $\boldsymbol{x}$.

Surprisingly, if $f \in \mathcal{C}^{1+}$ and $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$, $\tilde{f}_\Delta \in \mathcal{C}^{1+}$, is a class of models that provides order 1 subgradient accuracy and order 2 function accuracy *at* $\boldsymbol{x}$, then $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ is a class of fully linear models.

Theorem 19.5 (Fully Linear Versus Order N Accuracy) *Suppose $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, and $f \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$ with Lipschitz constant (of the gradient) given by K. Suppose that the model functions $\tilde{f}_\Delta \in \mathcal{C}^{1+}$ on $B(\boldsymbol{x}; \bar{\Delta})$ with Lipschitz constant (of the gradient) given by $\tilde{K}$. Then the following are equivalent:*

(i) $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ *is a class of fully linear models;*
(ii) $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ *provides order 1 subgradient accuracy and order 2 function accuracy near $\boldsymbol{x}$;*
(iii) $\{\tilde{f}_\Delta\}_{\Delta \in (0,\bar{\Delta}]}$ *provides order 1 subgradient accuracy and order 2 function accuracy at $\boldsymbol{x}$.*

Proof Clearly (i) and (ii) are equivalent and imply (iii).

Suppose (iii) is true. Note that $f \in \mathcal{C}^{1+}$ and $\tilde{f}_\Delta \in \mathcal{C}^{1+}$ imply that $\partial f = \{\nabla f\}$ and $\partial \tilde{f}_\Delta = \{\nabla \tilde{f}_\Delta\}$. Thus, order 1 subgradient accuracy at $\boldsymbol{x}$ is equivalent to

$$\|\nabla f(\boldsymbol{x}) - \nabla \tilde{f}_\Delta(\boldsymbol{x})\| \leq \kappa_g \Delta,$$

where $\kappa_g = \kappa_g(\boldsymbol{x})$ is the parameter of subgradient accuracy. Considering any $\boldsymbol{y} \in B(\boldsymbol{x}; \bar{\Delta})$, we find that

$$\begin{aligned}\|\nabla f(\boldsymbol{y}) - \nabla \tilde{f}_\Delta(\boldsymbol{y})\| &\leq \|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\| + \|\nabla f(\boldsymbol{x}) - \nabla \tilde{f}_\Delta(\boldsymbol{x})\| \\ &\quad + \|\nabla \tilde{f}_\Delta(\boldsymbol{x}) - \nabla \tilde{f}_\Delta(\boldsymbol{y})\| \\ &\leq K\|\boldsymbol{y} - \boldsymbol{x}\| + \kappa_g \Delta + \tilde{K}\|\boldsymbol{y} - \boldsymbol{x}\| \\ &\leq (\kappa_g + K + \tilde{K})\Delta.\end{aligned}$$

Thus, the models provide order 1 subgradient accuracy *near* $\boldsymbol{x}$ using parameter $\kappa_g^* = \kappa_g + K + \tilde{K}$.

Next define $\hat{f}_\Delta = \tilde{f}_\Delta - \tilde{f}_\Delta(\boldsymbol{x}) + f(\boldsymbol{x})$. By Proposition 19.1 $\{\hat{f}_\Delta\}$ is a class of fully linear models of f at $\boldsymbol{x}$, so (ii) holds for $\hat{f}_\Delta$. Let κ_f be the parameter of function accuracy $\tilde{f}_\Delta$ *at* $\boldsymbol{x}$ and $\hat{\kappa}_f$ be the parameter of function accuracy $\hat{f}_\Delta$ *near* $\boldsymbol{x}$. Considering any $\boldsymbol{y} \in B(\boldsymbol{x}; \bar{\Delta})$, we find that

$$\begin{aligned}|f(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{y})| &\leq |f(\boldsymbol{y}) - \hat{f}_\Delta(\boldsymbol{y})| + |\hat{f}_\Delta(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{y})| \\ &\leq \hat{\kappa}_f \Delta^2 + |\tilde{f}_\Delta(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{x}) + f(\boldsymbol{x}) - \tilde{f}_\Delta(\boldsymbol{y})| \\ &= \hat{\kappa}_f \Delta^2 + |\tilde{f}_\Delta(\boldsymbol{x}) - f(\boldsymbol{x})| \\ &\leq \hat{\kappa}_f \Delta^2 + \kappa_f \Delta^2.\end{aligned}$$

Thus, the models provide order 2 function accuracy *near* $\boldsymbol{x}$ using parameter $\kappa_f^* = \hat{\kappa}_f + \kappa_f$. □

The notion of order N subgradient accuracy can be extended to higher (generalized) derivatives in the obvious manner. This suggests the following *conjecture* (since we leave this for future researchers, we do not formally define the notion of order N Hessian accuracy).

Conjecture 19.1 (Fully Quadratic Versus N Order Accuracy) Suppose $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\Delta} > 0$, and $f \in \mathcal{C}^{2+}$ on $B(\boldsymbol{x}; \bar{\Delta})$. Suppose that the model functions $\tilde{f}_\Delta \in \mathcal{C}^{2+}$ on $B(\boldsymbol{x}; \bar{\Delta})$. Then the following are equivalent:

(i) $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$ is a class of fully quadratic models;
(ii) $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$ provides order 1 Hessian accuracy, order 2 gradient accuracy, and order 3 function accuracy near $\boldsymbol{x}$;
(iii) $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$ provides order 1 Hessian accuracy, order 2 gradient accuracy, and order 3 function accuracy at $\boldsymbol{x}$.

In light of Theorem 19.5, it would appear that the minimum standard for a model-based DFO method to be expected to work might be that the class of models $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$ provides order 1 subgradient accuracy and order 2 function accuracy near $\boldsymbol{x}^k$. In fact, if the only target is first-order optimality, then it suffices to have access to a class of models that provides order 1 subgradient accuracy at $\boldsymbol{x}^k$.

Algorithm 19.2: Model-based steepest descent

Data: Given $f \in \mathcal{C}^1$, starting point $\boldsymbol{x}^0 \in \mathbb{R}^n$ and a class of models $\{\tilde{f}_\Delta\}_{\Delta\in(0,\bar{\Delta}]}$ that provides order 1 subgradient accuracy at given points.

Step 0. (*Initialization*) Initialize $\Delta^0 \in (0,\infty)$ the initial model accuracy parameter, $\mu^0 \in (0,\infty)$ the initial target accuracy parameter, $\eta \in (0,1)$ an Armijo parameter, $\varepsilon_d \in (0,1)$ minimum decrease angle parameter, $\epsilon_{\texttt{stop}} \in [0,\infty)$ stopping tolerance, and $k \leftarrow 0$ iteration counter.

Step 1. (*Model and descent direction*) Access the class of models to select $\tilde{f}_\Delta^k$ that provides order 1 subgradient accuracy at $\boldsymbol{x}^k$. Select $\tilde{\boldsymbol{g}}^k \in \partial\tilde{f}_\Delta^k(\boldsymbol{x}^k)$ to (approximately) solve $\operatorname{argmin}\{\|\boldsymbol{g}\| : \boldsymbol{g} \in \partial\tilde{f}_\Delta^k(\boldsymbol{x}^k)\}$.

Step 2. (*Model accuracy checks*)

(a) If $\Delta^k < \epsilon_{\texttt{stop}}$ and $\|\tilde{\boldsymbol{g}}^k\| < \epsilon_{\texttt{stop}}$,

- declare algorithm success and **stop**.

(b) If $\Delta^k > \mu^k\|\tilde{\boldsymbol{g}}^k\|$,

- declare the model inaccurate;
- decrease $\Delta^{k+1} \leq \frac{1}{2}\Delta^k$, set $\mu^{k+1} = \mu^k$, $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$, $k \leftarrow k+1$, and go to Step 1.

(c) If $\Delta^k \leq \mu^k\|\tilde{\boldsymbol{g}}^k\|$,

- declare the model accurate and proceed to Step 3.

Step 3. (*Line search*) Set

$$\boldsymbol{d}^k = -\frac{\tilde{\boldsymbol{g}}^k}{\|\tilde{\boldsymbol{g}}^k\|}. \qquad \text{(descent)}$$

Perform a line-search in the direction $\boldsymbol{d}^k$ to seek t^k with

$$f(\boldsymbol{x}^k + t^k\boldsymbol{d}^k) < f(\boldsymbol{x}^k) + \eta t^k(\boldsymbol{d}^k)^T\tilde{\boldsymbol{g}}^k \qquad \text{(Armijo)}$$

If t^k is found, declare `line-search success`.
Otherwise, declare `line-search failure`.

Step 4. (*Update*)
If `line-search success`,

- let $\boldsymbol{x}^{k+1}$ be any point such that $f(\boldsymbol{x}^{k+1}) \leq f(\boldsymbol{x}^k + t^k\boldsymbol{d}^k)$;
- set $\Delta^{k+1} = \Delta^k$, $\mu^{k+1} = \mu^k$.

Otherwise (`line-search failure`),

- set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$, $\Delta^{k+1} = \Delta^k$, and decrease $\mu^{k+1} \leq \frac{1}{2}\mu^k$.

Increment $k \leftarrow k+1$ and go to Step 1.

As the name suggests, the MODEL-BASED STEEPEST-DESCENT framework is essentially the nonsmooth steepest-descent algorithm with the true subdifferential replaced with a model-based approximate subdifferential. Convergence of the MODEL-BASED STEEPEST-DESCENT framework can be established using fairly standard DFO techniques. To begin, consider the stopping test (Step 2a).

Lemma 19.1 *Suppose the stopping test in Step (2a) is triggered. Then* $\boldsymbol{x}^k$ *satisfies*

$$\operatorname{dist}(\mathbf{0}, \partial f(\boldsymbol{x}^k)) \leq (1 + \kappa_g)\epsilon_{stop},$$

where κ_g *is the constant for subgradient accuracy and* $\operatorname{dist}(\boldsymbol{x}, S)$ *is the distance of the point* $\boldsymbol{x}$ *to the set* S.

Conversely, if

$$\operatorname{dist}(\mathbf{0}, \partial f(\boldsymbol{x}^k)) < \frac{1}{2}\min\{\epsilon_{stop}/\kappa_g, \epsilon_{stop}\} \text{ and } \Delta^k < \frac{1}{2}\min\{\epsilon_{stop}/\kappa_g, \epsilon_{stop}\},$$

then the algorithm will stop.

Proof First suppose the stopping test is triggered. Order 1 subgradient accuracy implies that for any $\tilde{\boldsymbol{g}}^k \in \partial \tilde{f}_{\Delta^k}$ there exists $\boldsymbol{g}^k \in \partial f(\boldsymbol{x}^k)$ with $\|\tilde{\boldsymbol{g}}^k - \boldsymbol{g}^k\| \leq \kappa_g \Delta^k$. So, if $\Delta^k < \epsilon_{\texttt{stop}}$ and $\|\tilde{\boldsymbol{g}}^k\| < \epsilon_{\texttt{stop}}$, using this $\boldsymbol{g}^k$, we note that

$$\begin{aligned}\operatorname{dist}(\mathbf{0}, \partial f(\boldsymbol{x}^k)) &\leq \|\mathbf{0} - \boldsymbol{g}^k\| \\ &\leq \|\mathbf{0} - \tilde{\boldsymbol{g}}^k\| + \|\tilde{\boldsymbol{g}}^k - \boldsymbol{g}^k\| \\ &\leq \epsilon_{\texttt{stop}} + \kappa_g \Delta^k \leq (1 + \kappa_g)\epsilon_{\texttt{stop}}.\end{aligned}$$

Now suppose

$$\operatorname{dist}(\mathbf{0}, \partial f(\boldsymbol{x}^k)) < \frac{1}{2}\min\{\epsilon_{\texttt{stop}}/\kappa_g, \epsilon_{\texttt{stop}}\} \text{ and } \Delta^k < \frac{1}{2}\min\{\epsilon_{\texttt{stop}}/\kappa_g, \epsilon_{\texttt{stop}}\}.$$

Let $\boldsymbol{g}^k \in \partial f(\boldsymbol{x}^k)$ with $\|\boldsymbol{g}^k\| < \frac{1}{2}\min\{\epsilon_{\texttt{stop}}/\kappa_g, \epsilon_{\texttt{stop}}\}$. Applying order 1 subgradient accuracy, we know that there exists $\tilde{\boldsymbol{g}}^k \in \partial \tilde{f}_{\Delta^k}$ with $\|\tilde{\boldsymbol{g}}^k - \boldsymbol{g}^k\| < \kappa_g \Delta^k$. Which yields

$$\begin{aligned}\|\tilde{\boldsymbol{g}}^k\| &< \kappa_g \Delta^k + \|\boldsymbol{g}^k\| \\ &< \kappa_g \frac{1}{2}\min\{\epsilon_{\texttt{stop}}/\kappa_g, \epsilon_{\texttt{stop}}\} + \frac{1}{2}\min\{\epsilon_{\texttt{stop}}/\kappa_g, \epsilon_{\texttt{stop}}\} \\ &\leq \epsilon_{\texttt{stop}}.\end{aligned}$$

Since $\Delta^k < \frac{1}{2}\min\{\epsilon_{\texttt{stop}}/\kappa_g, \epsilon_{\texttt{stop}}\} \leq \epsilon_{\texttt{stop}}$, the stopping conditions are triggered. □

Lemma 19.1 demonstrates the importance of the two parts of defining order 1 subgradient accuracy. Part (ii) of Definition 19.15 is used to show that if the algorithm stops, then an approximate critical point is found. Part (i) of Definition 19.15 is used to show that if the algorithm reaches an approximate critical point, then the stopping conditions will be triggered (once Δ^k is sufficiently small).

Next, we consider the situation where Algorithm 19.2 is run without stopping conditions (i.e., Step 2a is omitted).

Theorem 19.6 *Suppose f is bounded below and $f \in \mathcal{C}^{0+}$ with Lipschitz constant (of the function values) given by K. Suppose Algorithm 19.2 is run without stopping conditions and let $\boldsymbol{x}^k$ be the resulting infinite sequence. Suppose the step sizes t^k are bounded below by a constant $\bar{t} > 0$. Then either*

(i) $f(\boldsymbol{x}^k) \downarrow -\infty$*; or*
(ii) $\|\boldsymbol{d}^k\| \to 0$ *and every accumulation point $\hat{\boldsymbol{x}}$ of $\{\boldsymbol{x}^k\}$ satisfies $\mathbf{0} \in \partial f(\hat{\boldsymbol{x}})$.*

Proof A comparison of Algorithm 19.2 to [48, Algorithm AGS] shows that Algorithm 19.2 generalizes [48, Algorithm RAGS] to include any modelling technique that provides order 1 subgradient accuracy (whereas [48, Algorithm RAGS] employed specific modelling technique in the algorithm). All the proofs in [48] except [48, Lemma 1] use only order 1 subgradient accuracy assumption, while [48, Lemma 1] proves the applied modelling technique provides order 1 subgradient accuracy. As such, this result is an easy adaptation of [48]. □

In presenting the MODEL-BASED STEEPEST-DESCENT framework, we note that, like steepest descent, it is unlikely to be competitive with a more advanced algorithm.[10] Instead, our goal in presenting Algorithm 19.2 is to demonstrate how a model-based DFO algorithm can be decoupled from the models. Any modelling technique that satisfies order 1 subgradient accuracy at $\boldsymbol{x}^k$ can be used within the MODEL-BASED STEEPEST-DESCENT framework to create a convergent method.

Let us return now to Table 19.2. As we have just demonstrated, the RAGS algorithm in [48] is easily adapted to allow for any model that provides order 1 subgradient accuracy at $\boldsymbol{x}^k$. As the RAGS algorithm is an advancement to the WASG algorithm in [47], it is reasonable to say that WASG is also adaptable to any model that provides order 1 subgradient accuracy at $\boldsymbol{x}^k$. Considering the rest of Table 19.2:

- The DGM algorithm of [11] applies a model that converges in a limiting sense only[11];

[10] This said, in model-based DFO the relative effectiveness of algorithms is often less clear. For smooth optimization, Newton's method will almost always dramatically outperform steepest descent. In model-based DFO, the inaccuracies in the models and the difficulty in constructing models with good quality Hessian approximations may make steepest descent a competitive alternative [50].

[11] In [11], the model accuracy is forced to converge to 0 by pre-selecting the model accuracy level δ_k such that $\delta_k \to 0$.

- The $\text{DFO}_{\text{comirror}}$ of [12] applies a model that provides order 1 subgradient accuracy and order 2 function accuracy at $\boldsymbol{x}^k$ [12, Corollary 3.3] and [45, Theorems 2.1 and 3.1];
- The CMS, GDMS, DMS, SMS algorithms of [59] apply models that provide order 1 subgradient accuracy and order 2 function accuracy at $\boldsymbol{x}^k$ [59, Lemma 2] and [45, Theorems 2.1 and 3.1];
- The MS_4PL of [56] applies a model that provides order 1 subgradient accuracy ([56, Lemma 1] shows the first required inequality, the second required inequality can be obtained using similar techniques);
- The DFO_$\mathcal{VU}$-algorithm of [51] applies a model that provides order 1 subgradient accuracy and order 2 function accuracy at $\boldsymbol{x}^k$ [45, Theorems 2.1 and 3.1].

In each case, it is reasonable to conjecture that there is nothing special about the models used, except the types and quality of accuracy they provide. It would be valuable to reexamine the methods above and see if a model-flexible method of each method could be created.

19.4.2 Constructing Models of Nonsmooth Functions

Constructing models that provide order N function accuracy at $\boldsymbol{x}$ can be accomplished easily. Indeed, one only needs to ensure that the model value is correct at the point of interest. Constructing models that provide order N subgradient accuracy at $\boldsymbol{x}$ is more difficult.

As mentioned in Sect. 19.2, some researchers have explored constructing models that provide some subgradient accuracy for broad classes of functions [11, 57]. While theoretically interesting, the number of function calls required to create the models is too high for practical use.[12] As such, researchers have turned to studying constructing models for more structured functions [45, 47, 48, 56, 59].

Let us consider the situation where the objective function f is the composition of two functions $F : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}$, specifically $f(\boldsymbol{x}) = g(F(\boldsymbol{x}))$. Suppose that function values (for f) are computed in two steps: first the vector-valued function $F(\boldsymbol{x}) = [F_1(\boldsymbol{x}), F_2(\boldsymbol{x}), \ldots, F_m(\boldsymbol{x})]$ is evaluated through a blackbox, second the single-valued analytically available function g is evaluated at $F(\boldsymbol{x})$. Using this framework, if we can construct model functions for each F_i, then the analytic structure of g allows an obvious method to construct a model function of f. In particular, if $\tilde{F}_{i,\Delta}$ are model functions of F_i, then we can use $\tilde{f}_\Delta(\boldsymbol{x}) = g([\tilde{F}_{i,\Delta}(\boldsymbol{x}), \tilde{F}_{i,\Delta}(\boldsymbol{x}), \ldots, \tilde{F}_{i,\Delta}(\boldsymbol{x})])$ as our model function of f.

We next define a model accuracy for vector valued functions.

[12] The techniques of [11, 57] do not provide error bounds, but instead showed asymptotic convergence. Thus, independent of the number of function calls, the models do not satisfy order N accuracy conditions.

Definition 19.16 (Order N Accuracy for Vector-Valued Functions) Given

$$F : \mathbb{R}^n \to \mathbb{R}^m \ : \ \boldsymbol{x} \mapsto [F_1(\boldsymbol{x}), F_2(\boldsymbol{x}), \ldots, F_m(\boldsymbol{x})],$$

$\boldsymbol{x} \in \text{dom}(F)$, and $\bar{\Delta} > 0$, we say that $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ is a class of models of F parameterized by Δ that *provides order N function (subgradient) accuracy* at $\boldsymbol{x}$ (near $\boldsymbol{x}$) if

$$\tilde{F}_\Delta : \mathbb{R}^n \to \mathbb{R}^m \ : \ \boldsymbol{x} \mapsto [\tilde{F}_{1,\Delta}(\boldsymbol{x}), \tilde{F}_{2,\Delta}(\boldsymbol{x}), \ldots \tilde{F}_{m,\Delta}(\boldsymbol{x})],$$

and each $\tilde{F}_{i,\Delta}(\boldsymbol{x})$ is a class of models of F_i that provides order N function (subgradient) accuracy at $\boldsymbol{x}$ (near $\boldsymbol{x}$).

In this case, we use $\kappa_{F_i}(\boldsymbol{x})$ ($\kappa_{G_i}(\boldsymbol{x})$) as the accuracy parameter for subfunction F_i, and $\kappa_F(\boldsymbol{x}) = \max\{\kappa_{F_i}(\boldsymbol{x})\}$ ($\kappa_G(\boldsymbol{x}) = \max\{\kappa_{G_i}(\boldsymbol{x})\}$) in order to provide a single accuracy parameter suitable for all subfunctions.

The next result provides bounds on the function accuracy of a model constructed by composing g with models of the vector-valued function F. A similar result can be found in [45, Theorem 2.1].

Theorem 19.7 (Compositions of Order N Function Accuracy) *Suppose*

$$f : \mathbb{R}^n \to \mathbb{R} \ : \ \boldsymbol{x} \mapsto g(F(\boldsymbol{x})),$$

where $g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ and $F : \mathbb{R}^n \to \mathbb{R}^m$. Suppose $\boldsymbol{x} \in \mathbb{R}^n$ with $F(\boldsymbol{x}) \in \text{int}(\text{dom}(g))$ and g is locally Lipschitz on its domain. Let $\bar{\Delta} > 0$ and $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ be a class of models of F parameterized by Δ. Define

$$\tilde{f}_\Delta : \mathbb{R}^n \to \mathbb{R} \ : \ \boldsymbol{x} \mapsto g(\tilde{F}_\Delta(\boldsymbol{x})).$$

(i) *If $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N, $N \geq 1$, function accuracy at $\boldsymbol{x}$, then there exists $\bar{\Delta}_f > 0$ such that $\{\tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}_f]\}$ provides order N function accuracy at $\boldsymbol{x}$.*

(ii) *If $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N, $N \geq 1$, function accuracy near $\boldsymbol{x}$, then there exists $\bar{\Delta}_f > 0$ such that $\{\tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}_f]\}$ provides order N function accuracy near $\boldsymbol{x}$.*

Proof

(i) Suppose $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N function accuracy at $\boldsymbol{x}$. Let L be a local Lipschitz constant of g at $\boldsymbol{x}$ and $\kappa_F(\boldsymbol{x})$ be the parameter of order N function accuracy for F. Since $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N function accuracy at $\boldsymbol{x}$, there exists $\bar{\Delta}_f \in (0, \bar{\Delta}]$ such that $\tilde{F}_\Delta(\boldsymbol{x}) \in \text{int}(\text{dom}(g))$ for all $\Delta \in (0, \bar{\Delta}_f]$.

Given any $\Delta \in (0, \bar{\Delta}_f]$ we find that

$$\begin{aligned} |f(\boldsymbol{x}) - \tilde{f}_\Delta(\boldsymbol{x})|^2 &= |g(F(\boldsymbol{x})) - g(\tilde{F}_\Delta(\boldsymbol{x}))|^2 \\ &\le L^2 \|F(\boldsymbol{x}) - \tilde{F}_\Delta(\boldsymbol{x})\|^2 \\ &= L^2 \textstyle\sum_{i=1}^m |F_i(\boldsymbol{x}) - \tilde{F}_{i,\Delta}(\boldsymbol{x})|^2 \\ &\le L^2 \textstyle\sum_{i=1}^m (\kappa_F(\boldsymbol{x}))^2 \Delta^{2N} \\ &\le L^2 m (\kappa_F(\boldsymbol{x}))^2 \Delta^{2N}. \end{aligned}$$

Which shows that $|f(\boldsymbol{x}) - \tilde{f}_\Delta(\boldsymbol{x})| \le L\sqrt{m}\kappa_F(\boldsymbol{x})\Delta^N$, so $\{\tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}_f]\}$ provides order N function accuracy at $\boldsymbol{x}$.

(ii) Suppose $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N function accuracy near $\boldsymbol{x}$. As $F(\boldsymbol{x}) \in \text{int}(\text{dom}(g))$ and F is continuous, there exists $\delta > 0$ such that $F(\boldsymbol{y}) \in \text{int}(\text{dom}(g))$ for all $\boldsymbol{y} \in \text{cl}(B(\boldsymbol{x}; \delta))$. Since $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta})\}$ provides order N function accuracy near $\boldsymbol{x}$, this implies the existence of $\bar{\Delta}_f \in (0, \min\{\delta, \bar{\Delta}\}]$ such that $\tilde{F}_\Delta(\boldsymbol{y}) \in \text{int}(\text{dom}(g))$ for all $\Delta \in (0, \bar{\Delta}_f]$, $\boldsymbol{y} \in \text{cl}(B(\boldsymbol{x}; \bar{\Delta}_f))$.

By the continuity of F, $\{\boldsymbol{z} \in \mathbb{R}^m : \boldsymbol{z} = F(\boldsymbol{y}),\ \boldsymbol{y} \in \text{cl}(B(\boldsymbol{x}; \bar{\Delta}_f))\} \subseteq \text{int}(\text{dom}(g))$ is compact. As such, there exists a Lipschitz constant for g relative to this set [81, Theorem 9.2]. Let L be this Lipschitz constant and $\kappa_F(\boldsymbol{x})$ be the parameter of order N function accuracy for F.

Given any $\Delta \in (0, \bar{\Delta}_f]$ and $\boldsymbol{y} \in B(\bar{\boldsymbol{x}}; \Delta)$, following an analogous series of inequalities to the above, we find that

$$|f(\boldsymbol{y}) - \tilde{f}_\Delta(\boldsymbol{y})|^2 \le L^2 m (\kappa_F(\boldsymbol{x}))^2 \Delta^{2N}.$$

Which shows that $\{\tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}_f]\}$ provides order N function accuracy near $\boldsymbol{x}$. □

It is not difficult to grasp the importance of $F(\boldsymbol{x}) \in \text{int}(\text{dom}(g))$ in Theorem 19.7, as without it we can select models that result in $g(\tilde{F}_\Delta(\boldsymbol{x})) = \infty$ [45, Example 2.3].

We next examine subgradient accuracy of a model constructed by composing g with models of the vector-valued function F. Theorem 19.8 shows that given three key pieces the composition models retain subgradient accuracy properties. These key pieces are: the chain rule $\partial f(\boldsymbol{x}) = \nabla F(\boldsymbol{x})^T \partial g(F(\boldsymbol{x}))$ holds, the subdifferential is bounded, and the models give exact function values at the point of interest.

Theorem 19.8 (Compositions of Order *N* Subgradient Accuracy) *Suppose*

$$f : \mathbb{R}^n \to \mathbb{R} : \boldsymbol{x} \mapsto g(F(\boldsymbol{x})),$$

where $g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ *and* $F : \mathbb{R}^n \to \mathbb{R}^m$. *Suppose a nondegeneracy condition holds for* g *and* F *so that*

$$\partial f(\boldsymbol{x}) = \nabla F(\boldsymbol{x})^T \partial g(F(\boldsymbol{x}))$$

and $\partial g(F(\boldsymbol{x}))$ *is bounded.*

Let $\bar{\Delta} > 0$ and $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ be a class of models of F parameterized by Δ satisfying $F(\boldsymbol{x}) = \tilde{F}_\Delta(\boldsymbol{x})$. Define

$$\tilde{f}_\Delta : \mathbb{R}^n \to \mathbb{R} \ : \ \boldsymbol{x} \mapsto g(\tilde{F}_\Delta(\boldsymbol{x}))$$

and suppose a nondegeneracy condition holds for g and $\tilde{F}_\Delta$ so that

$$\partial \tilde{f}_\Delta(\boldsymbol{x}) = \nabla \tilde{F}_\Delta(\boldsymbol{x})^T \partial g(\tilde{F}_\Delta(\boldsymbol{x})).$$

If $\{\tilde{F}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N, $N \geq 1$, subgradient accuracy at $\boldsymbol{x}$, then $\{\tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}]\}$ provides order N subgradient accuracy at $\boldsymbol{x}$.

Proof Apply $F(\boldsymbol{x}) = \tilde{F}_\Delta(\boldsymbol{x})$ and the chain rule to see that

$$\partial \tilde{f}_\Delta(\boldsymbol{x}) = \nabla \tilde{F}_\Delta(\boldsymbol{x})^T \partial g(F(\boldsymbol{x})).$$

Note $\partial g(F(\boldsymbol{x}))$ is bounded and define $M = \sup\{\|\boldsymbol{w}\| : \boldsymbol{w} \in \partial g(F(\boldsymbol{x}))\}$. Let $\kappa_G(\boldsymbol{x})$ be the parameter of order N subgradient accuracy for F.

Selecting any $\boldsymbol{v} \in \partial f(\boldsymbol{x})$, there exists $\boldsymbol{w} \in \partial g(F(\boldsymbol{x}))$ such that $\boldsymbol{v} = \nabla F(\boldsymbol{x})^T \boldsymbol{w}$. Define $\tilde{\boldsymbol{v}} = \nabla \tilde{F}_\Delta(\boldsymbol{x})^T \boldsymbol{w} \in \partial \tilde{f}_\Delta(\boldsymbol{x})$, and notice that

$$\begin{aligned}
\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|^2 &= \|\nabla F(\boldsymbol{x})^T \boldsymbol{w} - \nabla \tilde{F}_\Delta(\boldsymbol{x})^T \boldsymbol{w}\|^2 \\
&= \left\| \begin{bmatrix} \nabla F_1(\boldsymbol{x})^T \boldsymbol{w} - \nabla \tilde{F}_{1,\Delta}(\boldsymbol{x})^T \boldsymbol{w} \\ \nabla F_2(\boldsymbol{x})^T \boldsymbol{w} - \nabla \tilde{F}_{2,\Delta}(\boldsymbol{x})^T \boldsymbol{w} \\ \vdots \\ \nabla F_m(\boldsymbol{x})^T \boldsymbol{w} - \nabla \tilde{F}_{m,\Delta}(\boldsymbol{x})^T \boldsymbol{w} \end{bmatrix} \right\|^2 \\
&= \sum_{i=1}^{m} \nabla (F_m(\boldsymbol{x})^T \boldsymbol{w} - \nabla \tilde{F}_{m,\Delta}(\boldsymbol{x})^T \boldsymbol{w})^2 \\
&\leq \sum_{i=1}^{m} \|\nabla F_m(\boldsymbol{x}) - \nabla \tilde{F}_{m,\Delta}(\boldsymbol{x})\|^2 \|\boldsymbol{w}\|^2 \\
&\leq \sum_{i=1}^{m} (\kappa_G(\boldsymbol{x}))^2 \Delta^{2N} M^2 \leq m(\kappa_G(\boldsymbol{x}))^2 M^2 \Delta^{2N}.
\end{aligned}$$

This yields $\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \leq \sqrt{m} \kappa_G(\boldsymbol{x}) M \Delta^N$, which gives the first inequality required for order N subgradient accuracy.

Conversely, selecting any $\tilde{\boldsymbol{v}} \in \partial \tilde{f}_\Delta(\boldsymbol{x})$, there exists $\boldsymbol{w} \in \partial g(F(\boldsymbol{x}))$ such that $\boldsymbol{v} = \nabla \tilde{F}_\Delta(\boldsymbol{x})^T \boldsymbol{w}$. Define $\boldsymbol{v} = \nabla F(\boldsymbol{x})^T \boldsymbol{w} \in \partial f(\boldsymbol{x})$. Repeating the sequence of inequalities above, yields the second inequality required for order N subgradient accuracy. □

The important of the chain rule in Theorem 19.8 is obvious. Examples of what can go wrong if the subdifferential is not bounded or the models do not give exact function values at the point of interest can be found in [45, Examples 3.2 and 3.3].

19.5 Conclusions and Open Directions

DFO studies algorithms that do not use derivative information, because explicit derivative information is unavailable. This might be because the derivatives are not explicitly known, or simply because they do not exist. Model-based methods build approximations and use them in an algorithm that explores the space of variables. The ways that the models are constructed and the way that the algorithm exploits them define various model-based methods.

Many model-based methods target unconstrained optimization problems, and assume that the objective function is once or twice differentiable. These assumptions motivate the use of models and allow the methods to be supported by a convergence analysis relating the accuracy of the model, its gradient, or possibly its Hessian. Models for such problems include interpolation, (generalized) simplex-gradient models, and other fully linear and fully quadratic models.

More recently, model-based DFO methods have targeted problems for which there is no reason to believe that the functions are differentiable. Different tools need to be used to construct models, and different analytical nonsmooth tools need to be applied to study their behaviour. We have presented some of this research.

We believe that research in model-based DFO for NSO is only at its beginning. Open research directions in model-based DFO for NSO include the following:

(1) Expanding the list of algorithms, and identifying which work best for specific classes of problems. We believe that an important strategy when developing new algorithms is to present it in a flexible way so that it is generic on which model is used. To this end we introduce the term "order N accuracy" (Definitions 19.14 and 19.15) to describe model quality without providing details on model construction.

(2) Expanding modelling techniques, in particular for nonsmooth functions. Herein we present some results for when the objective function is the composition of a nonsmooth function and a smooth function. However, other possible nonsmooth structures should be considered. For example, Poissant [70] recently studied a situation in which only incomplete monotonic information is known. Another obvious structure is splitting problems $f = F + G$, where F is smooth and G is some form of nonsmooth regularizer (e.g., as found in the LASSO problem [87]). While this could be rephrased as a composition of a nonsmooth function and a smooth function, it may be valuable to consider such popular structures directly.

(3) Determine how the knowledge of structure within a problem (as mentioned in (2)) can best be exploited by an optimization method. A step towards this might

be achieved by a better understanding of calculus rules for models, similar to the rules suggested for smooth functions in [78].

(4) Determine how model-based DFO methods can be applied to problems with nonlinear constraints. There has been some progress in the development of direct search algorithms for inequality constrained blackbox optimization [6], but model-based DFO method have largely remained in the realm of unconstrained or box-constrained problems. One step in model-based optimization has been made by Amaioua et al. [3]. Another step could be the development of a definition for "order N accuracy" of the normal cone (a starting place might be [37, 46]). Such a definition would allow for modelling techniques and model-based DFO algorithms for constrained optimization to be researched independently.

(5) Further research on interweaving model-based and direct search DFO methods. More research should be conducted in combining such methods and in comparing strengths and weaknesses of each.

Acknowledgements Audet's research was partially funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada, Discover Grant #2015-05311.

Hare's research was partially funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada, Discover Grant #355571-2013.

References

1. Alarie, S., Audet, C., Garnier, V., Le Digabel, S., Leclaire, L.-A.: Snow water equivalent estimation using blackbox optimization. Pac. J. Optim. **9**(1), 1–21 (2013)
2. Alberto, P., Nogueira, F., Rocha, H., Vicente, L.N.: Pattern search methods for user-provided points: application to molecular geometry problems. SIAM J. Optim. **14**(4), 1216–1236 (2004)
3. Amaioua, N., Audet, C., Conn, A.R., Le Digabel, S.: Efficient solution of quadratically constrained quadratic subproblems within the mesh adaptive direct search algorithm. Eur. J. Oper. Res. **268**(1), 13–24 (2018)
4. Audet, C.: A survey on direct search methods for blackbox optimization and their applications. In: Pardalos, P.M., Rassias, T.M. (eds.) Mathematics Without Boundaries: Surveys in Interdisciplinary Research, Chapter 2, pp. 31–56. Springer, Berlin (2014)
5. Audet, C., Dennis, Jr., J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
6. Audet, C., Dennis, Jr., J.E.: A progressive barrier for derivative-free nonlinear programming. SIAM J. Optim. **20**(1), 445–472 (2009)
7. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham (2017)
8. Audet, C., Hare, W.: Algorithmic construction of the subdifferential from directional derivatives. Set-Valued Var. Anal. **26**(3), 431–447 (2018)
9. Audet, C. Ianni, A., Le Digabel, S., Tribes, C.: Reducing the number of function evaluations in mesh adaptive direct search algorithms. SIAM J. Optim. **24**(2), 621–642 (2014)
10. Audet, C., Conn, A.R., Le Digabel, S., Peyrega, M.: A progressive barrier derivative-free trust-region algorithm for constrained optimization. Comput. Optim. Appl. **71**(2), 307–329 (2018)
11. Bagirov, A.M., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**(2), 317–334 (2008)

12. Bauschke, H., Hare, W., Moursi, W.: A derivative-free comirror algorithm for convex optimization. Optim. Methods Softw. **30**(4), 706–726 (2015)
13. Beiranvand, V., Hare, W., Lucet, Y.: Best practices for comparing optimization algorithms. Optim. Eng. **18**(4), 815–848 (2017)
14. Berghen, F.V.: CONDOR: A Constrained, Non-Linear, Derivative-Free Parallel Optimizer for Continuous, High Computing Load, Noisy Objective Functions. PhD thesis, Université Libre de Bruxelles, Belgium (2004)
15. Berghen, F.V., Bersini, H.: CONDOR, a new parallel, constrained extension of Powell's UOBYQA algorithm: experimental results and comparison with the DFO algorithm. J. Comput. Appl. Math. **181**, 157–175 (2005)
16. Bigdeli, K., Hare, W., Nutini, J., Tesfamariam, S.: Optimizing damper connectors for adjacent buildings. Optim. Eng. **17**(1), 47–75 (2016)
17. Billups, S.C., Larson, J., Graf, P.: Derivative-free optimization of expensive functions with computational error using weighted regression. SIAM J. Optim. **23**(1), 27–53 (2013)
18. Booker, A.J., Dennis, Jr., J.E., Frank, P.D., Moore, D.W., Serafini, D.B.: Managing surrogate objectives to optimize a helicopter rotor design further experiments. AIAA Paper 1998–4717, Presented at the 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis (1998)
19. Booker, A.J., Dennis, Jr., J.E., Frank, P.D., Serafini, D.B., Torczon, V.: Optimization using surrogate objectives on a helicopter test example. In: Borggaard, J., Burns, J., Cliff, E., Schreck, S. (eds.) Optimal Design and Control, Progress in Systems and Control Theory, pp. 49–58. Birkhäuser, Cambridge (1998)
20. Booker, A.J., Dennis, Jr., J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Struct. Multidiscip. Optim. **17**(1), 1–13 (1999)
21. Bortz, D.M., Kelley, C.T.: The simplex gradient and noisy optimization problems. In: Borggaard, J., Burns, J., Cliff, E., Schreck, S. (eds.) Optimal Design and Control, Progress in Systems and Control Theory, pp. 77–90. Birkhäuser, Cambridge (1998)
22. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**(3), 751–779 (2005)
23. Bűrmen, A., Olenšek, J., Tuma, T.: Mesh adaptive direct search with second directional derivative-based Hessian update. Comput. Optim. Appl. **62**(3), 693–715 (2015)
24. Conn, A.R., Toint, Ph.L.: An algorithm using quadratic interpolation for unconstrained derivative free optimization. In: Di Pillo, G., Gianessi, F. (eds.) Nonlinear Optimization and Applications, pp. 27–47. Plenum Publishing, New York (1996)
25. Conn, A.R., Le Digabel, S.: Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. Optim. Methods Softw. **28**(1), 139–158 (2013)
26. Conn, A.R., Scheinberg, K., Toint, Ph.L.: A derivative free optimization algorithm in practice. In: Proceedings the of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri (1998)
27. Conn, A.R., Scheinberg, K., Toint, Ph.L.: DFO (Derivative Free Optimization) (2001). https://projects.coin-or.org/Dfo
28. Conn, A.R., Scheinberg, K., Vicente, L.N.: Geometry of interpolation sets in derivative free optimization. Math. Program. **111**(1–2), 141–172 (2008)
29. Conn, A.R., Scheinberg, K., Vicente, L.N.: Global convergence of general derivative-free trust-region algorithms to first and second order critical points. SIAM J. Optim. **20**(1), 387–415 (2009)
30. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MOS-SIAM Series on Optimization. SIAM, Philadelphia (2009)
31. Coope, I., Tappenden, R.: Efficient calculation of regular simplex gradients. Technical Report https://arxiv.org/abs/1710.01427v1, Department of Mathematics and Statistics, University of Canterbury (2018)
32. Custódio, A.L.: Aplicações de Derivadas Simplécticas em Métodos de Procura Directa. Ph.D. thesis, Universidade Nova de Lisboa, Portugal (2008)

33. Custódio, A.L., Vicente, L.N.: Using sampling and simplex derivatives in pattern search methods. SIAM J. Optim. **18**(2), 537–555 (2007)
34. Custódio, A.L., Dennis, Jr., J.E., Vicente, L.N.: Using simplex gradients of nonsmooth functions in direct search methods. IMA J. Numer. Anal. **28**(4), 770–784 (2008)
35. Custódio, A.L., Rocha, H., Vicente, L.N.: Incorporating minimum Frobenius norm models in direct search. Comput. Optim. Appl. **46**(2), 265–278 (2010)
36. Custódio, A.L., Scheinberg, K., Vicente, L.N.: Methodologies and software for derivative-free optimization. In: Terlaky, T., Anjos, M.F., Ahmed, S. (eds.) Advances and Trends in Optimization with Engineering Applications, MOS-SIAM Book Series on Optimization, Chapter 37, SIAM, Philadelphia (2017)
37. Davis, C., Hare, W.: Exploiting known structures to approximate normal cones. Math. Oper. Res. **38**(4), 665–681 (2013)
38. Echeverrá Ciaurri, D., Isebor, O.J., Durlofsky, L.J.: Application of derivative-free methodologies to generally constrained oil production optimization problems. Proc. Comput. Sci. **1**(1), 1301–1310 (2010)
39. Fasano, G. Liuzzi, G., Lucidi, S., Rinaldi, F.: A line search-based derivative-free approach for nonsmooth constrained optimization. SIAM J. Optim. **24**(3), 959–992 (2014)
40. Fowler, K.R., Reese, J.P., Kees, C.E., Dennis Jr., J.E., Kelley, C.T., Miller, C.T., Audet, C., Booker, A.J., Couture, G., Darwin, R.W., Farthing, M.W., Finkel, D.E., Gablonsky, J.M., Gray, G., Kolda, T.G.: Comparison of derivative-free optimization methods for ground water supply and hydraulic capture community problems. Adv. Water Resour. **31**(5), 743–757 (2008)
41. Gheribi, A.E., Robelin, C., Le Digabel, S., Audet, C., Pelton, A.D.: Calculating all local minima on liquidus surfaces using the FactSage software and databases and the mesh adaptive direct search algorithm. J. Chem. Thermodyn. **43**(9), 1323–1330 (2011)
42. Gheribi, A.E., Audet, C., Le Digabel, S., Bélisle, E., Bale, C.W., Pelton, A.D.: Calculating optimal conditions for alloy and process design using thermodynamic and properties databases, the FactSage software and the Mesh Adaptive Direct Search algorithm. CALPHAD: Comput. Coupling Phase Diagrams and Thermochem. **36**, 135–143 (2012)
43. Giuliani, C.M., Camponogara, E.: Derivative-free methods applied to daily production optimization of gas-lifted oil fields. Comput. Chem. Eng. **75**, 60–64 (2015)
44. Han, L., Liu, G.: On the convergence of the UOBYQA method. J. Appl. Math. Comput. **16**(1–2), 125–142 (2004)
45. Hare, W.: Compositions of convex functions and fully linear models. Optim. Lett. **11**(7), 1217–1227 (2017)
46. Hare, W.L., Lewis, A.S.: Estimating tangent and normal cones without calculus. Math. Oper. Res. **30**(4), 785–799 (2005)
47. Hare, W., Macklem, M.: Derivative-free optimization methods for finite minimax problems. Optim. Methods Softw. **28**(2), 300–312 (2013)
48. Hare, W., Nutini, J.: A derivative-free approximate gradient sampling algorithm for finite minimax problems. Comput. Optim. Appl. **56**(1), 1–38 (2013)
49. Hare, W.L., Lucet, Y.: Derivative-free optimization via proximal point methods. J. Optim. Theory Appl. **160**(1), 204–220 (2014)
50. Hare, W., Jaberipour, M.: Adaptive interpolation strategies in derivative-free optimization: a case study. Pac. J. Optim. **14**(2), 327–347 (2018)
51. Hare, W., Planiden, C., Sagastizábal, C.: A derivative-free $\mathcal{VU}$-algorithm for convex finite-max problems (2018). arXiv:1903.11184
52. Hooke, R., Jeeves, T.A.: "Direct search" solution of numerical and statistical problems. J. Assoc. Comput. Mach. **8**(2), 212–229 (1961)
53. Huyer, W., Neumaier, A.: SNOBFIT—stable noisy optimization by branch and fit. ACM Trans. Math. Softw. **35**(2), 9:1–9:25 (2008)
54. Kelley, C.T.: Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. SIAM J. Optim. **10**, 43–55 (1999)
55. Kelley, C.T.: Implicit Filtering. Society for Industrial and Applied Mathematics, Philadephia (2011)

56. Khan, K., Larson, J., Wild, S.: Manifold sampling for optimization of nonconvex functions that are piecewise linear compositions of smooth components. SIAM J. Optim. **28**(4), 3001–3024 (2018)
57. Kiwiel, K.C.: A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **20**(4), 1983–1994 (2010)
58. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: New perspectives on some classical and modern methods. SIAM Rev. **45**(3), 385–482 (2003)
59. Larson, J., Menickelly, M., Wild, S.M.: Manifold sampling for L_1 nonconvex optimization. SIAM J. Optim. **26**(4), 2540–2563 (2016)
60. Meza, J.C., Martinez, M.L.: On the use of direct search methods for the molecular conformation problem. J. Comput. Chem. **15**, 627–632 (1994)
61. Mifflin, R.: A superlinearly convergent algorithm for minimization without evaluating derivatives. Math. Program. **9**(1), 100–117 (1975)
62. Mifflin, R., Sagastizábal, C.: A $\mathcal{VU}$-algorithm for convex minimization. Math. Program. **104**(2–3, Ser.B), 583–608 (2005)
63. Minville, M., Cartier, D., Guay, C., Leclaire, L.-A., Audet, C., Le Digabel, S., Merleau, J.: Improving process representation in conceptual hydrological model calibration using climate simulations. Water Resour. Res. **50**, 5044–5073 (2014)
64. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009)
65. Mugunthan, P., Shoemaker, C.A., Regis, R.G.: Comparison of function approximation, heuristic and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models. Water Resour. Res. **41**(11) (2005)
66. Müller, J., Woodbury, J.: GOSAC: global optimization with surrogate approximation of constraints. J. Global Optim. **69**(1), 117–136 (2017)
67. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
68. Oeuvray, R.: Trust-Region Methods Based on Radial Basis Functions with Application to Biomedical Imaging. Ph.D. thesis, Institut de Mathématiques, École Polytechnique Fédérale de Lausanne, Switzerland (2005)
69. Oeuvray, R., Bierlaire, M.: A new derivative-free algorithm for the medical image registration problem. Int. J. Model. Simul. **27**(2), 115–124 (2007)
70. Poissant, C.: Exploitation d'une structure monotone en recherche directe pour l'optimisation de boîtes grises. Master's thesis, École Polytechnique de Montréal (2018)
71. Powell, M.J.D.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.P. (eds.) Advances in Optimization and Numerical Analysis, Mathematics and Its Applications, vol. 275, pp. 51–67, Springer, Dordrecht (1994)
72. Powell, M.J.D.: UOBYQA: Unconstrained optimization by quadratic approximation. Math. Program. **92**(3), 555–582 (2002)
73. Powell, M.J.D.: On trust region methods for unconstrained minimization without derivatives. Math. Program. **97**(3), 605–623 (2003)
74. Powell, M.J.D.: Least Frobenius norm updating of quadratic models that satisfy interpolation conditions. Math. Program. **100**(1), 183–215 (2004)
75. Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. In: Di Pillo, G., Roma, M. (eds.) Large-Scale Nonlinear Optimization, Nonconvex Optimization and Its Applications, vol. 83, Springer, Boston (2006)
76. Powell, M.J.D.: The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, Cambridge University, UK (2009)
77. Price, C.J., Coope, I.D., Byatt, D.: A convergent variant of the Nelder Mead algorithm. J. Optim. Theory Appl. **113**(1), 5–19 (2002)
78. Regis, R.G.: The calculus of simplex gradients. Optim. Lett. **9**(5), 845–865 (2015)

79. Regis, R.G., Wild, S.M.: CONORBIT: constrained optimization by radial basis function interpolation in trust regions. Optim. Methods Softw. **32**(3), 552–580 (2017)
80. Renaud, E., Robelin, C., Gheribi, A.E., Chartrand, P.: Thermodynamic evaluation and optimization of the Li, Na, K, Mg, Ca, Sr, F, Cl reciprocal system. J. Chem. Thermodyn. **43**(8), 1286–1298 (2011)
81. Rockafellar, R.T., Wets, R. J.-B.: Variational Analysis, Part of the Grundlehren der mathematischen Wissenschaften book series (GL), vol. 317. Springer, Berlin (1998)
82. Sampaio, Ph.R., Toint, Ph.L.: A derivative-free trust-funnel method for equality-constrained nonlinear optimization. Comput. Optim. Appl. **61**(1), 25–49 (2015)
83. Sauer, Th., Xu, Y.: On multivariate Lagrange interpolation. Math. Comput. **64**, 1147–1170 (1995)
84. Schoute, P.H.: Mehrdimensionale Geometrie, vol. 1. Cornell University Library, Ithaca (1902)
85. Serafini, D.B.: A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions. Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston (1998)
86. Talgorn, B., Audet, C., Kokkolaras, M., Le Digabel, S.: Locally weighted regression models for surrogate-assisted design optimization. Optim. Eng. **19**(1), 213–238 (2018)
87. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. **58**(1), 267–288 (1996)
88. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**(1), 1–25 (1997)
89. Tseng, P.: Fortified-descent simplicial search method: A general approach. SIAM J. Optim. **10**(1), 269–288 (1999)
90. Wild, S.M., Regis, R.G., Shoemaker, C.A.: ORBIT: optimization by radial basis function interpolation in trust-regions. SIAM J. Sci. Comput. **30**(6), 3197–3219 (2008)
91. Wild, S.M., Shoemaker, C.A.: Global convergence of radial basis function trust region derivative-free algorithms. SIAM J. Optim. **21**(3), 761–781 (2011)
92. Winfield, D.: Function and Functional Optimization by Interpolation in Data Tables. Ph.D. thesis, Harvard University, USA (1969)
93. Winfield, D.: Function minimization by interpolation in a data table. J. Inst. Math. Appl. **12**, 339–347 (1973)
94. Wright, M.H.: Direct search methods: Once scorned, now respectable. In: Griffiths, D.F., Watson, G.A. (eds.) Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis, pp. 191–208. Addison-Wesley, Harlow (1996)
95. Xu, J., Audet, C., DiLiberti, C.E., Hauck, W.W., Montague, T.H., Parr, A.F., Potvin, D., Schuirmann, D.J.: Optimal adaptive sequential designs for crossover bioequivalence studies. Pharm. Stat. **15**(1), 15–27 (2016)

Final Words

The book is an updated state of the art of the numerical methods for solving optimization problems in finite dimension, under no differentiability assumptions. Such methods date back to the 60s of last Century and we have tried to provide an idea on how many branches did stem out from the pioneering works that, both, in Eastern and Western countries were conducted at that time. In fact, despite of the current wide diversification of models, methods and application fields, it is apparent that some very basic ideas (e.g. subgradient direction and cutting plane) are still alive and capable to indicate new directions of research. Experts from several countries, who have been invited to illustrate their points of view on specific aspects of NSO, have covered, in practice, the main research areas related to numerical methods.

The dualism and the fruitful competition between subgradient and cutting plane-like methods have been fully reflected in Part I of the book. In this part, the modern and accelerated versions of the subgradient method, bundle methods in their numerous variants and gradient-sampling approaches, capable to handle nonconvexity too, have been discussed in depth.

Part II has focused on treatment of problems equipped with some known structure. It is worth mentioning that the analysed methods, although structure-based, cover an extremely vast area of interest. For instance, minmax (sometimes minmaxmin) problems are the most common ones in applications where either worst-case analysis or robustness issues or stochastic scenarios appear within decision making processes. Not to say that the often occurring possibility of expressing nonconvex nonsmooth objective functions in DC form allows us to use most of the machinery coming from convex optimization to deal with such special class of (global) optimization problems. This is particularly relevant in many modern applications of optimization, for example within the impressively growing area of machine learning. Structure exploiting methods are also those based on detecting domain areas where function smoothness is concentrated or on constructing piecewise linear model reflecting derivative discontinuities.

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3

Part III has been devoted to the study of special problems, in particular multiobjective and mixed integer. As for the former, the analysis has involved the general case of LLC functions, as well as the special case of multiobjective DC functions. Coming to mixed integer optimization (both linear and nonlinear), it suffices to recall the strong demand for effective NSO methods that such an area has expressed in last decades, motivated by the diffusion of dual ascent methods in large-scale settings. Moreover, in the latter sector (but also in other applications of relevant importance) time-consuming function evaluation has led to design methods which take into account explicitly inexact data evaluations of functions and/or subgradients.

Finally, Part IV has been concerned with the rapidly growing research area of the derivative-free methods.

We firmly believe that nonsmooth optimization is becoming an increasingly important research area in optimization. It will find many interesting applications in very diverse areas of human activity in the future. We are confident that this book will stimulate further research in this fascinating area and in its many applications.

Adil M. Bagirov
Manlio Gaudioso
Napsu Karmitsa
Marko M. Mäkelä
Sona Taheri

Index

© Springer Nature Switzerland AG 2020

A. M. Bagirov et al. (eds.), *Numerical Nonsmooth Optimization*,
https://doi.org/10.1007/978-3-030-34910-3