Math. Program., Ser. B (2013) 140:189–233
DOI 10.1007/s10107-012-0600-5

FULL LENGTH PAPER

Composite proximal bundle method

Claudia Sagastizábal

Received: 30 July 2009 / Accepted: 5 August 2012 / Published online: 25 October 2012

Abstract We consider minimization of nonsmooth functions which can be represented as the composition of a positively homogeneous convex function and a smooth mapping. This is a sufficiently rich class that includes max-functions, largest eigenvalue functions, and norm-1 regularized functions. The bundle method uses an oracle that is able to compute separately the function and subgradient information for the convex function, and the function and derivatives for the smooth mapping. With this information, it is possible to solve approximately certain proximal linearized subproblems in which the smooth mapping is replaced by its Taylor-series linearization around the current serious step. Our numerical results show the good performance of the Composite Bundle method for a large class of problems.

Keywords Nonconvex Optimization · Nonsmooth optimization · Bundle methods · Composite functions

Mathematical Subject Classification Primary 90C26 · 49J52; Secondary 65K10 · 49J53 · 49M05

To C. Lemaréchal, a nonsmooth optimization giant who kindly guided the author through ridges of nondifferentiability until a superlinear path of knowledge. Merci, Claude.

On leave from INRIA, France. Visiting researcher at IMPA.

Research partially supported by Grants CNPq 303840/2011-0, AFOSR FA9550-08-1-0370, and NSF DMS 0707205, as well as by PRONEX-Optimization and FAPERJ.

C. Sagastizábal (✉)
IMPA, Estrada Dona Castorina 110, Jardim Botânico,
Rio de Janeiro, RJ 22460-320, Brazil
e-mail: sagastiz@impa.br

1 Introduction and motivation

For some years already, nonsmooth optimization research has focused on exploiting structure in the objective function as a way to speed up numerical methods. Indeed, for convex optimization, complexity results establish that oracle based methods have a sublinear rate of convergence; [28]. The $\mathscr{U}$-Lagrangian theory [16,17], and the $\mathscr{V}\mathscr{U}$-space decomposition [24], can be seen as tools to "extract" smooth structure from general convex functions. The approach was extended to a class of nonconvex functions in [26]. Similar ideas for more general nonconvex functions, having a smooth representative on a certain manifold corresponding to the $\mathscr{U}$-subspace, are explored in [10,18]. Another line of work concentrates efforts on identifying classes of functions structured enough to have some type of second-order developments, such as the primal-dual gradient structured functions from [24,25], or the composite functions in [32]. Composite functions were initially considered in [34]; see also [8, Ch. 14]. Later on these functions were studied in [4,21], and have been more recently revisited in [19,29], in the convex and nonconvex settings, respectively.

Most of the work above is conceptual, in the sense that if some algorithmic framework is considered, often it is not implementable. Among the exceptions, we find the convex optimization algorithms in [27,29,31], and the $\mathscr{V}$-space identification method in [6]. In this paper, we develop an implementable bundle method that exploits structure for certain nonconvex composite optimization problems. More precisely, given a smooth mapping $c : \Re^n \to \Re^m$ and a positively homogeneous (of degree 1) convex function $h : \Re^m \to \Re$, we consider how to solve the unconstrained problem

$$\min_{x\in\Re^n} (h \circ c)(x). \tag{1}$$

In this composite objective, we refer to c as the *inner* mapping and to h as the *outer* function. Note in particular that the outer function is real-valued on $\Re^m$, an assumption that excludes indicator functions, but still covers a rich enough family of interesting problems; see Sect. 2.

For solving the possibly nonconvex problem (1) we develop a specialized bundle method that takes full advantage of the composite structure of the objective function. Specifically, rather than using composite data:

($\mathtt{bb}_{\mathtt{h\circ c}}$) For $x \in \Re^n$ a composite black-box computes $(h \circ c)(x)$ and $\gamma \in \partial(h \circ c)(x)$,

we suppose the oracle has the ability to make separate computations. More precisely,

($\mathtt{bb}_{\mathtt{c}}$) For $x \in \Re^n$ an inner black-box computes $c(x)$ and its Jacobian $Dc(x)$
($\mathtt{bb}_{\mathtt{h}}$) For $C \in \Re^m$ an outer black-box computes $h(C)$ and $G \in \partial h(C)$.

While ($\mathtt{bb}_{\mathtt{h\circ c}}$) would build a classical cutting-plane model for the composite objective, having the richer oracle ($\mathtt{bb}_{\mathtt{c}}$)-($\mathtt{bb}_{\mathtt{h}}$) at hand, we can make a better approximation and build a cutting-plane model for the so-called *conceptual model*. This model from [8], considered by [19] for general composite functions, and given by (4) below, defines certain proximal linearized subproblems in which the smooth mapping is replaced by its Taylor-series linearization. The corresponding algorithmic framework, named ProxDescent in [19], is rather broad (outer functions can be extended-valued and prox-regular), but relies heavily on the exact computation of proximal points at each iteration. In this work, we provide a fully implementable variant of such an algorithmic framework for a less general, but still large enough, class of outer functions (real-valued, positively homogeneous, and convex).

In order to deal with approximate proximal computations, our fully implementable variant, given by Composite Algorithm 1 below, incorporates typical features from bundle methods, such as *serious* and *null* step iterates. But this is not the only modification of ProxDescent: there is an additional level of approximation, introduced by the need to estimate the conceptual model, which is handled in the implementable algorithm by backtracking. For these reasons, our convergence analysis deviates significantly from that of [19]. In addition, instead of a standard proximal term as in ProxDescent, in our development we consider a variable prox-metric that opens the way to exploiting second-order information of the smooth mapping, when available.

An alternative line of work, based on somewhat different richer oracles (that can compute more than one subgradient at any given point), refers to the proximity control bundle method [30]. These methods, devised for nonconvex optimization problems appearing in eigenvalue optimization and automatic control, use certain "first-order local models" that contain the conceptual model from [19] as a special case.

Our paper is organized as follows. In Sect. 2 we give several classes of functions with composite structure. The conceptual model and the proposed cutting-plane function, that makes the conceptual model computationally tractable, are described in Sect. 3. Sections 4 and 5 contain, respectively, the Composite Bundle method and its convergence results. In Sect. 6 we report our numerical experience, on many functions, including some described in Sect. 2. Our results show the good performance of the composite algorithm, when compared to several solvers, on a large number of problems of various dimensions. In the final section with concluding remarks, we compare the main features of our composite method with the proximity control bundle method from [30].

2 Composite functions in the family

Our class of composite functions includes many examples from [19,29,32]. We mention in passing that the wording "composite" means slightly different things for different authors. Specifically, while we refer to a composition that can be nonconvex (as in [19,32]), composite functions in [29] are the sum of two terms, one convex and the other one smooth.

We suppose the outer function is positively homogeneous and convex, so it is also sublinear. In [13, Lesson C] there are many examples of real-valued sublinear functions, such as norms, quadratic seminorms, gauges of closed convex sets containing 0 in their interior, infimal convolutions of sublinear functions, and support functions of bounded sets. The composition of any of such (outer) function with any smooth (inner) mapping defines a function in our family. We review below some interesting special cases. In what follows, the notation c, with lower capitals, refers to the smooth mapping while C, with upper capitals, denotes an m-dimensional vector in the mapping image, so that $C = c(x)$ for some $x \in \Re^n$.

Example 1 (*max-functions*) The function given by the pointwise maximum of a finite collection of C^2-functions in $\Re^n$, $\{c_1(\cdot), \dots, c_m(\cdot)\}$, can be defined by composing the outer function $h(C) = \max(C_1, \dots, C_m)$ with the inner mapping with components $c_j(\cdot)$, for $j = 1, \dots, m$.

For any $x \in \Re^n$, the Jacobian mapping $Dc(x)$ is the $m \times n$ matrix with rows given by the transposed gradients $\nabla c_j(x)^\top$. As for the second-order derivative, $D^2c(x)$, for any $d, \tilde{d} \in \Re^n$, $[D^2c(x)\tilde{d}]d$ is the vector in $\Re^m$ with jth component given by $\tilde{d}^\top \nabla^2 c_j(x) d$. □

We now give a more involved function, appearing in semidefinite programming.

Example 2 (*eigenvalue optimization*) Let $\mathscr{S}^p$ denote the linear space of symmetric matrices of order p, and consider the function $\lambda_{\max}(X)$, given by the maximum eigenvalue of a symmetric matrix $X \in \mathscr{S}^p$, whose elements are C^2-functions of a vector $x \in \Re^n$. Suppose that for a fixed X, r denotes the multiplicity of the maximum eigenvalue. Then from the analysis in [2, Ex. 3.98], the orthogonal projection onto the eigenspace corresponding to the maximum eigenvalue is an analytic function near X. By applying Gram–Schmidt orthonormalization to the columns of such projection, it is possible to define a "diagonalizing" analytic mapping $\mathscr{C} : \mathscr{S}^p \to \mathscr{S}^r$, such that: $\mathscr{C}(X) = \lambda_{\max}(X)\mathrm{I}_r$ where I_r is the identity matrix of order r; the Jacobian $D\mathscr{C}(X)$ is surjective; and the eigenvalues $\lambda(\mathscr{C}(X))$ coincide with $\lambda_{\max}(X)$. The inner smooth mapping corresponds to $\mathscr{C}$, while the outer function is the eigenvalue function $h(C) = \lambda(C)$, a positively homogeneous convex function. □

The next function is typically used in applications such as wavelet-based image restoration, sparse representations, sparse regression, and compressive sensing problems.

Example 3 (*Regularized minimization maps*) Many signal or image reconstruction problems, as well as approximation problems, minimize a function of the form

$$\frac{1}{2}|Ax - y|^2 + \tau|x|\,,$$

for $x \in \Re^n$, $y \in \Re^k$, A a matrix $k \times n$, τ a positive parameter, and for given appropriate norms. In the expression above, the first *data fidelity* term is smooth, and the second term corresponds to some regularization or penalty, for example ensuring sparsity of the solution. To fit our composite framework, it suffices to take $m = n + 1$,

$$c_j(x) = x_j \text{ for } j = 1, \dots, n\,, \qquad c_{n+1}(x) = \frac{1}{2}|Ax - y|^2\,,$$

and define the outer function $h(C_1, \dots, C_{n+1}) = C_{n+1} + \tau|(C_1, \dots, C_n)|$.

For any $x \in \Re^n$, the Jacobian mapping $Dc(x)$ is the $(n+1) \times n$ matrix made by appending to the identity matrix of order n a row with the transpose of $A^\top(Ax-y)$. As for the second-order derivative, $D^2c(x)$, for any $d, \tilde{d} \in \Re^n$, $[D^2c(x)\tilde{d}]d$ is the vector with its first n components equal to 0, and its last component equal to $\tilde{d}^\top A^\top Ad$, a constant with respect to x. □

We now give a possibly nonconvex function, appearing in approximation problems.

Example 4 (*sum of Euclidean norms*) Given a collection of smooth vector functions, $\{\phi_1, \ldots, \phi_J\}$ with $\phi_j : \Re^n \to \Re^{m_j}$ for $j = 1, \ldots, J$, the function $\sum_{j=1}^J |\phi_j(x)|$ is the composition of the following smooth mapping with $m = \sum_{j=1}^J m_j$ components,

$$c(x) = (\phi_1(x), \ldots, \phi_J(x))$$

and outer function:

$$h(C_1, \ldots, C_{m_1}, C_{m_1+1}, \ldots, C_m) = \sum_{j=1}^J \left|(C_{\sum_{k=1}^{j-1} m_k+1}, \ldots, C_{\sum_{k=1}^{j} m_k})\right| .$$

When $n = J = m_1 = 1$, letting $\phi_1(x) = 0.5a_2x^2 + a_1x + a_0$ for $a_{2,1,0}$ given scalars with $a_2 \neq 0$, it is easy to see that the function f is convex if and only if $a_1^2 \leq 2a_0a_2$. □

The next function shows how to cast ℓ_1-penalizations of nonlinear programming problems into the composite framework (provided no indicator function is used for polyhedral sets, to preserve finite-valuedness of the outer function).

Example 5 (*Uryasev's exact penalty function*) This is an optimal control problem over a discrete horizon with n time steps. The control variable $x \in \Re^n$ is constrained to the box $[-0.2, 0.2]^n$. Given initial values (ξ_0, ψ_0), the state variables $(\xi, \psi) \in \Re^2$ follow a trajectory given by a recursive state equation of the form

$$\begin{aligned} &\xi_1(x_1) = \xi_0 + 0.2\psi_0 , && \psi_1(x_1) = -0.004 + 0.2x_1, \text{ and} \\ &\xi_{i+1}(x_{1:i+1}) = \xi_i(x_{1:i}) + 0.2\psi_i(x_{1:i}) , && \psi_{i+1}(x_{1:i+1}) = \psi_i(x_{1:i}) - \texttt{cubic}\,\psi_i(x_{1:i})^2 \\ &&& \quad -0.004\xi_i(x_{1:i}) + 0.2x_{i+1} , \end{aligned}$$

for $i = 1, \ldots, n-1$ and where $\texttt{cubic} \geq 0$ is a given parameter. In the expressions above, we use the notation $x_{1:i}$ to refer to the dependence of the ith-state variables on the first ith components of the control variable x. The control problem to be solved is:

$$\begin{cases} \min_x \frac{1}{2} \sum_{i=1}^n \xi_i(x_{1:i})^2 \\ \quad x_i \in [-0.2, 0.2] \quad \text{for } i = 1, \ldots, n \\ \quad \psi_i(x_{1:i}) \geq -1 \quad\ \ \text{for } i = 1, \ldots, n-1 \\ \quad \psi_n(x_{1:n}) = 0 . \end{cases}$$

The corresponding ℓ_1^+-penalty function uses parameters $\mathtt{c}_1, \mathtt{c}_2, \mathtt{c}_3 > 0$, and can be written in composite form by letting $m = 3n + 1$,

$$c_j(x) = \begin{cases} -1 - \psi_j(x_{1:j}) & \text{for } j = 1, \ldots, n-1 \\ \psi_n(x_{1:n}) & \text{for } j = n \\ \frac{1}{2}\sum_{i=1}^{n} \xi_i(x_{1:i})^2 & \text{for } j = n+1 \\ x_{j-n-1} - 0.2 & \text{for } j = n+2, \ldots, 2n+1 \\ -x_{j-2n-1} - 0.2 & \text{for } j = 2n+2, \ldots, 3n+1\,, \end{cases}$$

and taking the outer function

$$h(C) = C_{n+1} + \mathtt{c}_1(|C_{n+2:2n+1}|^+ + |C_{2n+2:3n+1}|^+) + \mathtt{c}_2|C_n| + \mathtt{c}_3|C_{1:n-1}|^+\,.$$

The Jacobian mapping $Dc(x)$ can be computed using the adjoint state equations, to obtain the trajectory derivatives $\nabla\xi_i$ and $\nabla\psi_i$. When the parameter `cubic` is null, the control problem is linear-quadratic and the composite function $h \circ c$ is convex. In this case, the second-order derivative, $[D^2c(x)\tilde{d}]d$ is the vector with all of its components equal to 0, except for the $(n+1)$th component, equal to $\tilde{d}^\top \sum_{i=1}^{n} \nabla\xi_i \nabla\xi_i^\top d$ for any $d, \tilde{d} \in \Re^n$. When `cubic > 0`, the function is nonconvex, and second-order derivatives are difficult to compute for higher dimensions. For $n = 2$, only $c_2(x) = \psi_2$ and $c_3(x) = \frac{1}{2}(\xi_1^2 + \xi_2^2)$ have a nonzero Hessian, constantly equal to a scalar factor of the identity matrix of order n (the corresponding factors of cubic are -0.08 and 0.0016, respectively). □

Our final function, modifying [18, Sec.7] as in [25] to have a smooth inner mapping, is an example of a composite function having a nonconvex positively homogeneous outer function.

Example 6 (*Modified Lewis function*) For $x = (x_1, x_2)^\top$ consider the following function, defined on a partition of $\Re^2$:

$$f(x) := \begin{cases} x_1^2 - x_2 & \text{on } \{(x_1, x_2) \in \Re^2 : x_2 \le 0\} \\ x_1^2 + x_2 & \text{on } \{(x_1, x_2) \in \Re^2 : 0 < x_2 \le x_1^2\} \\ 3x_1^2 - x_2 & \text{on } \{(x_1, x_2) \in \Re^2 : 0 < x_1^2 \le x_2 \le 4x_1^2\} \\ -5x_1^2 + x_2 & \text{on } \{(x_1, x_2) \in \Re^2 : 4x_1^2 < x_2\}\,. \end{cases}$$

This nonconvex function has the origin as unique critical point. The composite form, $f = h \circ c$, has smooth inner mapping $c : \Re^2 \to \Re^2$ defined by $c_1(x) = x_1^2$ and $c_2(x) = x_2$; and outer function $h : \Re^2 \to \Re$ given by

$$h(C) := \begin{cases} |C_1|^+ - C_2 & \text{on } C \in \{(C_1, C_2) \in \Re^2 : C_2 \le 0\} \\ |C_1|^+ + C_2 & \text{on } C \in \{(C_1, C_2) \in \Re^2 : 0 < C_2 \le C_1\} \\ 3|C_1|^+ - C_2 & \text{on } C \in \{(C_1, C_2) \in \Re^2 : 0 < C_1 \le C_2 \le C_1\} \\ -5|C_1|^+ + C_2 & \text{on } C \in \{(C_1, C_2) \in \Re^2 : 4C_1 < C_2\}\,. \end{cases}$$

This outer function is posivitely homogeneous, but not convex, so $h \circ c$ is not included in our family. □

3 Outer function and conceptual model

Even though the composite function $h \circ c$ may be nonconvex, it is always locally Lipschitz continuous and directionally differentiable. Furthermore, at any point $x \in \Re^n$ such that $c(x) = 0$, the composite function is semismooth [23], and strongly semismooth if the inner mapping is twice continuously differentiable; see [32, Prop.3.2]. Moreover, since the outer function is assumed to be positively homogeneous of degree 1, the relation $h'(0; \cdot) = h(\cdot)$ holds. As a result, whenever $c(x) = 0$, the directional derivative of the composite function satisfies the relation $(h \circ c)'(x, d) = h'(c(x), Dc(x)d) = h(Dc(x)d)$, [32, Sec.3].

The optimality condition for $\bar{x}$ to be a critical point of (1) is $0 \in \partial(h \circ c)(\bar{x})$. Being real-valued and convex, the outer function is continuous and, using a chain rule [2, Ch.3.4.1],

$$\forall x \in \Re^n \quad \partial(h \circ c)(x) = Dc(x)^\top \partial h(C) \quad \text{for } C = c(x)\,. \tag{2}$$

In particular, critical points satisfy the inclusion

$$0 \in Dc(\bar{x})^\top \partial h(\bar{c}) \quad \text{for } \bar{c} = c(\bar{x}).$$

Another important consequence of positive homogeneity is the generalized *Euler formula*, stating that

$$\forall C \in \Re^m \text{ and for any } G \in \partial h(C) \text{ it holds that } h(C) = G^\top C. \tag{3}$$

(the result follows from the subgradient inequality $h((\alpha + 1)C) \geq h(C) + G^\top((\alpha + 1)C - C)$, using that $h((\alpha + 1)C) = (\alpha + 1)h(C)$ for any $\alpha \geq -1$). This particular structural property will be exploited in the bundle method to define a cutting-plane approximation of the conceptual model.

The conceptual model from [19] composes the outer function with the linearization of the inner mapping around a point x^k. Specifically, if $D_k = Dc(x^k)$ denotes the Jacobian mapping at x^k, then for any $d \in \Re^n$, letting

$$c_k(d) := c(x^k) + D_k d\,,$$

the conceptual model is given by the function

$$h(c_k(d)). \tag{4}$$

To make tractable this conceptual model, we use a cutting-plane approximation $\check{h}$ for the outer function and, for each pair $(H^i := h(C^i), G^i \in \partial h(C^i))$, define the plane

$$H^i + G^{i\,\top}(c_k(d) - C^i) = h(c(x^k)) - \Delta_i^k + G^{i\,\top} c_k(d),$$

with

$$\Delta_i^k := h(c(x^k)) - H^i + G^{i\,\top} C^i \geq G^{i\,\top} c(x^k), \tag{5}$$

by the subgradient inequality. The cutting-plane model for the conceptual model has the form

$$\check{h}(c_k(d)) = h(c(x^k)) + \max_{i \in \mathtt{B}} \left\{ -\Delta_i^k + G^{i\top} c_k(d) \right\}, \tag{6}$$

where the set $\mathtt{B}$ represents a bundle of information, varying along iterations. By convexity of h, the cutting-plane model is always a lower approximation for the conceptual model.

When h is positively homogeneous, equality (3) implies in (5) that $\Delta_i^k := h(c(x^k))$, so each plane has the form $G^{i\top} c_k(d)$, and the bundle only needs to keep the subgradient information (G^i). Accordingly, the composite cutting-plane model can be rewritten as

$$\check{h}(c_k(d)) = \max_{i \in \mathtt{B}} \left\{ G^{i\top} c_k(d) \right\} \tag{7}$$

where the symbolism $i \in \mathtt{B}$ means for all i such that G^i is in the current bundle $\mathtt{B}$.

The composite model may not be a lower approximation for $h \circ c$. However, since the conceptual model is a good estimate for $h \circ c$, eventually the composite model is a good approximation for the objective function. This feature is particularly important for nonconvex bundle methods, which need to handle very carefully cutting-plane models (in some cases a tangent line of a nonconvex function can cut-off a section of the graph of the function, leaving out a critical point, for example), [11].

4 Composite bundle method

At iteration number ℓ, the composite model $\check{h}_\ell(c_k(d))$, given by (7) written with bundle set $\mathtt{B} = \mathtt{B}_\ell$, is used to generate a direction d^ℓ. The inner mapping is linearized at a point x^k with $k = k(\ell)$. It is possible for k to be fixed for several consecutive values of ℓ. The quadratic programming problem (QP) solved by d^ℓ is nothing but the computation of the proximal point of the composite model, with prox-center x^k and a variable prox-metric depending on matrices M_k^ℓ detailed below.

The bundle of information $\mathtt{B}_\ell$ is formed from (some of the) past information, corresponding to past directions d^i with $i < \ell$, generated using a prox-center $x^{k(i)}$ and for which a call to the outer black-box ($\mathtt{bb_h}$) with $C^i = c_{k(i)}(d^i)$ gave the value $H^i = h(C^i)$ and the subgradient $G^i \in \partial h(C^i)$.

The variable prox-metric is induced by a positive definite matrix M_k^ℓ of order n having the form

$$M_k^\ell = M_k + \mu_\ell \mathrm{I} \text{ with } M_k \text{ a symmetric } n \times n \text{ matrix}, \mu_\ell \geq 0,$$

and I the identity matrix of order n. The corresponding primal and dual metrics in $\Re^n$ are

$$\forall x, \gamma \in \Re^n \quad |x|_{k,\ell}^2 := x^\top M_k^\ell x \qquad \text{and} \qquad \|\gamma\|_{k,\ell}^2 := \gamma^\top \left(M_k^\ell\right)^{-1} \gamma,$$

respectively. The eigenvalues of M_k^ℓ are obtained by adding μ_ℓ to those of M_k, while the eigenvectors of M_k^ℓ coincide with those of M_k. As a result, if for a matrix M the symbol $\lambda(M)$ (respectively $\lambda_{\min/\max}(M)$) denotes an eigenvalue (resp., minimum/maximum eigenvalues), then $\lambda(M_k^\ell) = \lambda(M_k) + \mu_\ell$,

$$\lambda_{\min}\left(M_k^\ell\right) = \lambda_{\min}(M_k) + \mu_\ell\,, \quad \text{and} \quad \lambda_{\min}\left((M_k^\ell)^{-1}\right) = \frac{1}{\lambda_{\max}(M_k) + \mu_\ell}\,.$$

In particular, letting $|\cdot|$ denote the Euclidean norm in $\Re^n$,

$$|x|_{k,\ell}^2 \geq (\lambda_{\min}(M_k) + \mu_\ell)|x|^2 \quad \text{and} \quad \|\gamma\|_{k,\ell}^2 \geq \frac{1}{\lambda_{\max}(M_k) + \mu_\ell}|\gamma|^2\,. \tag{8}$$

Having the elements above, the next direction $d^\ell \in \Re^n$ solves the quadratic programming problem

$$\min_{d\in\Re^n} \check{h}_\ell(c_k(d)) + \frac{1}{2}|d|_{k,\ell}^2\,. \tag{9}$$

Letting $C^\ell = c_{k(\ell)}(d^\ell)$, the corresponding optimality condition with $k = k(\ell)$ is

$$\exists \hat{G}^\ell \in \partial\check{h}_\ell(C^\ell) = conv\{G^i \in \mathtt{B}_\ell\} : \begin{cases} M_k^\ell d^\ell + D_k^\top \hat{G}^\ell = 0 \\ \check{h}_\ell(C^\ell) = \hat{G}^{\ell\top} C^\ell = \hat{G}^{\ell\top} c(x^k) - \|D_k^\top \hat{G}^\ell\|_{k,\ell}^2\,. \end{cases} \tag{10}$$

The optimal gradient $\hat{G}^\ell$ is a convex combination of active G^i's such that $\check{h}_\ell(C^\ell) = G^{i\top} C^\ell = \hat{G}^{\ell\top} C^\ell$.

Without loss of clarity, and depending on the context, we sometimes use the short notations k and C^ℓ, instead of the longer ones $k(\ell)$ and $c_{k(\ell)}(d^\ell)$.

Algorithm 1 (COMPOSITE BUNDLE) Inner and outer black-boxes, ($\mathtt{bb_c}$) and ($\mathtt{bb_h}$) respectively, are available.

Step 0 (Input and initialization)

Select a stopping tolerance $\mathtt{tol_{stop}} \geq 0$, two parameters $m_1, m_2 > 0$, and a minimum positive threshold $0 < \mu_{\min} < +\infty$.
Initialize the iteration and serious step counters to $\ell = 0$ and $k = k(\ell) = 0$.
For a starting point $x^0 \in \Re^n$, call ($\mathtt{bb_c}$) to compute the inner oracle values $C^0 = c(x^0)$ and $D_0 = Dc(x^0)$. Call ($\mathtt{bb_h}$) to compute the outer oracle values $H^0 = (h \circ c)(x^0)$, $G^0 \in \partial h(C^0)$. Set $\mathtt{B}_0 := \{G^0\}$.
Choose a symmetric matrix M_0 of order n and a prox-parameter $\mu_0 \geq 0$ ensuring that the matrix $M_0^0 = M_0 + \mu_0 \mathrm{I}$ is positive definite.

Step 1 (Model Generation and QP Subproblem)

Having the current serious step iterate x^k, $c(x^k)$, its associated Jacobian D_k, and the bundle $\{G^i\}_{i\in\mathtt{B}_\ell}$, define the composite cutting-plane model function $\check{h}_\ell(c_k(\cdot))$ from (7). A matrix M_k and a scalar $\mu_\ell \geq 0$ such that $M_k^\ell = M_k + \mu_\ell \mathrm{I}$ is positive definite are also available.
Compute d^ℓ by solving the quadratic program (9). These calculations include finding optimal simplicial multipliers α^ℓ such that in (10)

$$\hat{G}^\ell = \sum_{i\in\mathtt{B}_\ell} \alpha_i^\ell G^i \,.$$

Define the aggregate error $\hat{e}_\ell = (h \circ c)(x^k) - \hat{G}^{\ell\top} c(x^k)$, noting that $\hat{e}_\ell \geq 0$ by (5) and (3). Compute the predicted decrease

$$\delta_\ell = (h\circ c)(x^k) - \check{h}_\ell(c_k(d^\ell)) - \frac{1}{2}|d^\ell|_{k,\ell}^2 = \hat{e}_\ell + \frac{1}{2}\|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 \,, \tag{11}$$

where the last equality follows from (10).

Step 2 (Stopping test)

If $\max(\hat{e}_\ell, \|D_k^\top \hat{G}^\ell\|_{k,\ell}^2) \leq \mathtt{tol}_{\mathtt{stop}}$, stop.
Otherwise, call $(\mathtt{bb}_\mathtt{h})$ to obtain $h(c_k(d^\ell))$ and $G^\ell \in \partial h(c_k(d^\ell))$.

Step 3 (Serious/backtrack/null step test)

Check the descent condition

$$h(c_k(d^\ell)) \leq (h\circ c)(x^k) - m_1\delta_\ell \,. \tag{12}$$

If (12) does not hold, declare a null step: take $\mu_{\ell+1} \geq \mu_\ell$, $M_k^{\ell+1} = M_k + \mu_{\ell+1}\mathrm{I}$, and go to Step 4.
If (12) is true, call the inner oracle $(\mathtt{bb}_\mathtt{c})$ to compute $c(x^k + d^\ell)$. Call the outer oracle $(\mathtt{bb}_\mathtt{h})$ to compute $\Gamma^\ell \in \partial h(c(x^k + d^\ell))$, and check the condition below:

$$\Gamma^{\ell\top}[c_k(d^\ell) - c(x^k + d^\ell)] \geq -m_2\delta_\ell \,. \tag{13}$$

If (13) holds, declare a serious step, and go to Step 4. Otherwise, if (13) does not hold, declare a backtracking step: take $\mu_{\ell+1} \geq \mu_{\min} + \mu_\ell$, $M_k^{\ell+1} = M_k + \mu_{\ell+1}\mathrm{I}$, $\mathtt{B}_{\ell+1} = \mathtt{B}_\ell$, set $k(\ell+1) = k(\ell)$, increase ℓ by 1, and loop to Step 1.

Step 4 (Bundle update and management)

If needed, compress $\mathtt{B}_\ell$ either by keeping all strongly active elements ($G^i \in \mathtt{B}_\ell$: $\alpha_i^\ell > 0$), or by replacing some strongly active subgradients by the aggregate gradient $\hat{G}^\ell$. Define $\mathtt{B}_{\ell+1}$ by adding to the possibly compressed set the new outer gradient G^ℓ.

If the step was declared null, set $k(\ell+1) = k(\ell)$, increase ℓ by 1, and loop to Step 1. If the step was declared serious, set $k(\ell+1) = k+1$, $x^{k+1} = x^k + d^\ell$, call $(\mathtt{bb}_\mathtt{c})$ to compute $D_{k+1} = Dc(x^{k+1})$, compute a new symmetric matrix M_{k+1} and a prox-parameter $\mu_{\ell+1} \geq 0$ such that $M_{k+1}^{\ell+1} = M_{k+1} + \mu_{\ell+1}\mathrm{I}$ is positive definite. Increase k and ℓ each by 1, and loop to Step 1.

In Step 3, the decision to declare a null-step is different from standard bundle methods, and exploits the structure knowledge in the composite case. Namely, while condition (12) checks if there is descent for the outer function, (13) checks adequacy between the inner mapping and its conceptual model.

The prox-parameter increases at backtracking steps, and can be increased at null steps or decreased at serious steps. In this context, the wording "backtrack" may sound odd, since it corresponds to choosing a larger prox-parameter. The explanation comes from noticing that, due to (10) and the definition of M_k^ℓ, the value $1/\mu_\ell$ can be seen as a stepsize. In this sense, when the algorithm detects the need for backtracking, increasing the prox-parameter results indeed in a smaller stepsize. Our setting allows the prox-parameter to be zero if the matrix M_k is positive definite; for this reason in the backtracking step we force positivity of the prox-parameter by means of the positive threshold $\mu_{\min}$, set at the Initialization step.

In order to ensure that matrices M_k^ℓ remain positive definite, the prox-parameter update can be done as follows:

$$\mu_\ell = \begin{cases} 0 & \text{if } \lambda_{\min}(M_{k(\ell)}) \geq \mu_{\min}\,, \\ -\lambda_{\min}(M_{k(\ell)}) + \mu_{\min} & \text{otherwise.} \end{cases} \tag{14}$$

We mention that there is a difference between the Composite Algorithm 1 and usual bundle methods, in terms of oracle calls. A standard bundle method makes one call to both the inner and outer oracles per iteration, independent of getting a serious or a null step. Instead, Step 2 in Composite Algorithm 1 makes one call to the outer oracle $(\mathtt{bb}_\mathtt{h})$ at all iterations, needing additional calls to both oracles $(\mathtt{bb}_{\mathtt{h}/\mathtt{c}})$ at Step 3, for deciding between serious or backtracking steps. In this case, the additional subgradient $\Gamma^\ell \in \partial h(c(x^k + d^\ell))$ may enter the bundle. For comparisons to be fair, an appropriate measure for (bb)-calls is used in our numerical experience; see Sect. 6.3.

In Step 4 we see that the bundle is formed only by outer gradients, corresponding either to $G^i \in \partial h(C^i)$ or to some past aggregate gradient $\hat{G}^i$. By convexity of h, the cutting-plane model is always a lower approximation to h:

$$\text{for all} \quad \ell \geq 1 \quad \text{and} \quad C \in \Re^m \quad \check{h}_\ell(C) \leq h(C). \tag{15}$$

Moreover, since $\hat{G}^\ell \in \partial \check{h}_\ell(C^\ell)$ by (10), after some algebra involving δ_ℓ and $\hat{e}_\ell$, we see that

$$h(c(x^k)) - \check{h}_\ell(C^\ell) - \|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 = \delta_\ell - \frac{1}{2}\|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 = \hat{e}_\ell,$$

In particular, since $h(C) \geq \hat{G}^{\ell\top}C$, the definition of $\hat{e}_\ell$ in Step 1 implies that and, hence,

$$\forall \ell \text{ and } C \in \Re^m \quad h(C) \geq (h \circ c)(x^k) + \hat{G}^{\ell\top}(C - c(x^k)) - \hat{e}_\ell. \tag{16}$$

Remark 1 (*The trivial composite case*) To see how a classical bundle method compares to the Composite Algorithm 1, consider a convex nonsmooth function f, and the, somewhat artificial, outer function $h \equiv f$ and inner mapping $c(x) = x$. With respect to our composite structure, h is not necessarily positively homogeneous, $Dc = I$, $c_k(d) = x^k + d = c(x^k + d)$, and, instead of (7), the cutting-plane model has the form (6). Since the inner mapping is affine, there is no second order information to exploit, and the matrices $M_k \equiv 0$ in Composite Algorithm 1. The QP optimality condition in (10) becomes

$$\exists \hat{G}^\ell \in \partial \check{h}_\ell(x^\ell) = conv\{G^i \in \mathrm{B}_\ell\} : \begin{cases} \mu_\ell d^\ell + \hat{G}^\ell = 0 \\ \check{h}_\ell(x^\ell) = h(x^k) - \hat{\Delta}^k_\ell + \hat{G}^{\ell\top}x^\ell \,, \end{cases}$$

for $\hat{\Delta}^k_\ell := \sum_{i \in \mathrm{B}_\ell} \alpha^\ell_i \Delta^k_i$. The aggregate error is defined by $\hat{e}_\ell := \hat{\Delta}^k_\ell - \hat{G}^{\ell\top}x^k$, consistent with the error definition in Step 1, because when h is positively homogenous, $\Delta^k_i = h(x^k)$, by (3). As a result, the expression for the predicted decrease in (11) remains true. Finally, in Step 3, because $c_k(d^\ell) = x^k + d^\ell = c(x^k + d^\ell)$, there is no backtracking step, because the inequality in (13) trivially holds:

$$0 = \Gamma^{\ell\top}(c_k(d^\ell) - c(x^k + d^\ell)) \leq -m_2\delta_\ell \,.$$

From the above it follows that Composite Algorithm 1 with $M_k \equiv 0$ is applicable for the trivial composite case, becoming a classical bundle method for convex functions without any particular structure. $\square$

5 Convergence results

Since matrices M^ℓ_k are always positive definite, they induce a norm and (8) holds. Together with (11) and (10), this means that the relations

$$\delta_\ell \geq \begin{cases} \max\left(\frac{1}{2}\|D_k^\top \hat{G}^\ell\|^2_{k,\ell}, \hat{e}_\ell\right) \geq \dfrac{1}{2(\lambda_{\max}(M_k) + \mu_\ell)}|D_k^\top \hat{G}^\ell|^2 & \text{(a)} \\ \max\left(\frac{1}{2}|d^\ell|^2_{k,\ell}, \hat{e}_\ell\right) \geq \dfrac{1}{2}\left(\lambda_{\min}(M_k) + \mu_\ell\right)|d^\ell|^2 & \text{(b)} \end{cases} \tag{17}$$

are always satisfied.

We start our convergence analysis showing that there can only be a finite number of consecutive backtracking steps.

Proposition 1 (Finite backtracking loop) *Suppose the inner mapping is* C^2. *If after some iteration* $\hat{\ell}$, *Composite Algorithm 1 makes a last serious step and thereafter generates only null and backtracking steps, the following holds:*

(i) If the sequence $\{d^\ell\}_{\ell\geq\hat\ell}$ is bounded, then there exists $\mu_{noBT} > 0$ such that (13) *holds for any $\mu_\ell > \mu_{noBT}$.*
(ii) There cannot be infinitely many consecutive backtracking steps.

Proof For convenience, let $\hat k = k(\hat\ell)$, $\hat x = x^{\hat k}(= x^{k(\ell)}$ for all $\ell \geq \hat\ell)$, and $\hat M = M_{\hat k}$ denote, respectively, the k-iteration index, prox-center and matrix corresponding to the last serious step.

Since c is a C^2-mapping, a mean-value theorem applies to each component c_j, $j = 1, \ldots, m$:

$$c_j(\hat x + d^\ell) - c_j(\hat x) - \nabla c_j(\hat x)^\top d^\ell = \frac{1}{2}\nabla^2 c_j(\xi_j)(d^\ell, d^\ell) \quad \text{for some } \xi_j \in [\hat x, \hat x + d^\ell]\,.$$

Recall that $|\cdot|$ is the Euclidean matrix norm. By assumption, $\{\hat x + d^\ell\}$ is bounded so $|\Gamma^\ell| \leq L$ for all ℓ and some constant L. Similarly, boundedness of $\{d^\ell\}$ implies that $|\nabla^2 c_j(\xi_j)| \leq D$ for all $j = 1, \ldots, m$, for some constant D. By Cauchy–Schwarz inequality,

$$\Gamma^{\ell\,\top}[c(\hat x + d^\ell) - c(\hat x) - D_{\hat k} d^\ell] \leq \frac{\sqrt{m}}{2} LD|d^\ell|^2\,.$$

Using the rightmost inequality in (17)(b), we obtain

$$\delta_\ell \geq \frac{1}{2}|d^\ell|^2_{\hat k,\ell} \geq \frac{1}{2}(\lambda_{\min}(\hat M) + \mu_\ell)|d^\ell|^2\,.$$

In the notation used in this proof, $c(x^k + d^\ell) = c(\hat x + d^\ell)$ and $c_k(d^\ell) = c(\hat x) + D_{\hat k} d^\ell$. Suppose that (13) is not satisfied and multiply by -1 the corresponding inequality:

$$m_2\delta_\ell < -\Gamma^{\ell\,\top}[c_k(d^\ell) - c(x^k + d^\ell)] = \Gamma^{\ell\,\top}[c(\hat x + d^\ell) - c(\hat x) - D_{\hat k} d^\ell]\,.$$

Using the bounds above we see that

$$\frac{1}{2}m_2(\lambda_{\min}(\hat M) + \mu_\ell)|d^\ell|^2 \leq m_2\delta_\ell < \Gamma^{\ell\,\top}[c(\hat x + d^\ell) - c(\hat x) - D_{\hat k} d^\ell] \leq \frac{\sqrt{m}}{2} LD|d^\ell|^2\,.$$

Therefore, nonsatisfaction of (13) implies $\mu_\ell < \sqrt{m}LD/m_2 - \lambda_{\min}(\hat M)$. Item (i) follows, by taking $\mu_{\text{noBT}} \geq \sqrt{m}LD/m_2 - \lambda_{\min}(M_k)$.
The claim in item (ii) is shown by contradiction, assuming that (13) does not hold, with $\ell \to \infty$ due to backtracking. An infinite backtracking loop drives μ_ℓ to infinity, as well as the minimum eigenvalue of the matrix $M^\ell_{\hat k}$, because $M^\ell_{\hat k} = \hat M + \mu_\ell \mathrm{I}$. Since there are only backtracking steps, the bundle does not change, and the composite model $\check h_\ell(c_{\hat k}(\cdot))$ is a fixed function, say φ. Hence, by [12, Prop. XV.4.1.5], the minimand in (9) converges to $\varphi(0)$ with $d^\ell \to 0$ as $\ell \to \infty$. Therefore, $\hat x + d^\ell \to \hat x$ with $\varphi(d^\ell) + \frac{1}{2}|d^\ell|^2_{\hat k,\ell} \to \varphi(0)$. In particular, this means that the sequence $\{d^\ell\}$ is bounded, and the contradiction follows from item (i). □

As a consequence, if the algorithm loops forever, it generates either an infinite sequence of serious steps, or a finite one, followed by infinitely many null steps (possibly nonconsecutive, due to backtracking). Both situations are considered below, adapting classical developments for bundle methods [12, Ch.XV.3, XV.4] to the composite setting.

We start with the case of infinitely many serious steps.

Lemma 1 (Infinitely many serious steps) *Suppose the Composite Algorithm 1 generates an infinite sequence $\{x^k\}$ of serious steps. Denote by ℓ_k an iteration index yielding a serious step: $x^{k+1} = x^k + d^{\ell_k}$. The following holds:*

(i) If $0 < m_2 < m_1$ then, as $k \to \infty$, either $(h \circ c)(x^k) \to -\infty$ or $\delta_{\ell_k} \to 0$.
(ii) Suppose $\delta_{\ell_k} \to 0$. Then $\lim \hat{e}_{\ell_k} = 0$ and,

$$\textit{if the series } \sum_k \frac{1}{\mu_{\ell_k} + \lambda_{\max}(M_k)} \textit{ is divergent,} \tag{18}$$

then $\liminf \|D_k^\top \hat{G}^{\ell_k}\|_{k,\ell_k} = 0$.
(iii) If, in addition, the sequence $\{x^k\}$ is bounded, then it has at least one accumulation point that is critical for (1).
(iv) If, instead of (18), the stronger condition

$$\textit{the sequence } \{\mu_{\ell_k} + \lambda_{\max}(M_k)\} \textit{ is bounded above} \tag{18'}$$

holds, then all accumulation points are critical for (1).

Proof By (13), since $\Gamma^{\ell_k} \in \partial h(c(x^k + d^{\ell_k}))$, we have that

$$h\left(c(x^k) + D_k d^{\ell_k}\right) \geq h(c(x^k + d^{\ell_k})) - m_2\delta_{\ell_k} = (h \circ c)(x^{k+1}) - m_2\delta_{\ell_k}\,. \tag{19}$$

Together with satisfaction of (12), this means that $(h \circ c)(x^{k+1}) \leq (h \circ c)(x^k) - (m_1 - m_2)\delta_{\ell_k}$. The telescopic sum yields that either $(h \circ c)(x^k) \searrow -\infty$ or $0 \leq \delta_{\ell_k} \to 0$, because $m_1 - m_2 > 0$. In the first case, problem (1) does not have a minimizer and there is nothing to prove. The second case is addressed by the next item. The first assertion in (ii) follows from (11). The rightmost inequality in (17)(a), together with our assumption (18) gives the second result. To show item (iii), consider a k-subsequence such that $\|D_k^\top \hat{G}^{\ell_k}\|_{k,\ell_k} \to 0$ and extract a further subsequence of serious steps with accumulation point x^{acc}. By boundedness of $\{x^k\}$ and local Lipschitzianity of h, there is an associated subsequence $\{\hat{G}^\ell\}$ with accumulation point $\hat{G}^{acc}$. Passing to the limit in (16) and using that $\hat{e}_{\ell_k} \to 0$, we obtain in the limit that $\hat{G}^{acc} \in \partial h(C^{acc})$ for $C^{acc} = c(x^{acc})$. By smoothness of c, $D_k \to D_{acc} = Dc(x^{acc})$, and by (2), $D_{acc}^\top \hat{G}^{acc} \in \partial(h \circ c)(x^{acc})$. The result follows from item (ii). Item (iv) is similar to item (iii), noticing that if $L > 0$ denotes an upper bound for $\{\mu_{\ell_k} + \lambda_{\max}(M_k)\}$, then the relation $\frac{1}{2(\mu_{\ell_k} + \lambda_{\max}(M_k))} \geq 1/2L$ in (17)(a) gives that $|D_k^\top \hat{G}^{\ell_k}| \to 0$ as $k \to \infty$ (for the whole sequence). □

The remaining case refers to an infinite null-step loop and makes use of the *aggregate linearization*

$$h_\ell^{\mathrm{agg}}(c_k(d)) = \check{h}_\ell(c_k(d^\ell)) + \hat{G}^{\ell\top} D_k(d - d^\ell)\,,$$

the associated strongly convex function

$$\mathrm{H}_\ell(d) = h_\ell^{\mathrm{agg}}(c_k(d)) + \frac{1}{2}|d|_{k,\ell}^2\,, \tag{20}$$

and the result [12, Lem. XV.4.3.3]:

$$\mathrm{H}_{\ell-1}(d) = \mathrm{H}_{\ell-1}(d^{\ell-1}) + \frac{1}{2}|d - d^{\ell-1}|_{k,\ell-1}^2\,. \tag{21}$$

Finally, and similar to [5, Sec. 4], note that Step 4 in the Composite Algorithm 1 updates the bundle of information in a way ensuring that not only

$$\text{if iteration } \ell-1 \text{ was declared a null step, then} \quad h_{\ell-1}^{\mathrm{agg}}(c_k(d)) \leq \check{h}_\ell(c_k(d)), \tag{22}$$

but also

$$\text{if iteration } \ell-1 \text{was declared a null step, then} \quad \check{h}_\ell(c_k(d)) \geq G^{\ell-1\top} c_k(d) \tag{23}$$

for all $d \in \Re^n$.

Lemma 2 (Finitely many serious steps) *Suppose that, after some iteration $\hat{\ell}$, the Composite Algorithm 1 makes a last serious step $\hat{x} = \hat{x}^{k(\ell)}$ and thereafter generates an infinite number of null steps, possibly nonconsecutive, due to intermediate backtracking steps. The following holds:*

(i) The sequence $\{d^\ell\}_{\ell > \hat{\ell}}$ is bounded and there is an iteration $\ell' > \hat{\ell}$ such that only null steps are done for all $\ell \geq \ell'$.
(ii) If $m_1 < 1$, then $\delta_\ell \to 0$ and $\hat{e}_\ell \to 0$.
(iii) If, in addition, for the (fixed) matrix $\hat{M} = M_{k(\hat{\ell})}$.

$$\textit{the series} \sum_{\ell \geq \ell'} \frac{\mu_{\ell-1} + \lambda_{\min}(\hat{M})}{(\mu_\ell + \lambda_{\max}(\hat{M}))^2} \textit{is divergent,} \tag{24}$$

then $\liminf |Dc(\hat{x})^\top \hat{G}^\ell| = 0$*, $\hat{x} + d^\ell \to \hat{x}$ for some ℓ-subsequence, and $\hat{x}$ is critical for* (1).

Proof For convenience, let $\hat{k} = k(\hat{\ell})$, $\hat{x} = x^{\hat{k}} (= x^{k(\ell)}$ for all $\ell \geq \hat{\ell})$, and $\hat{M} = M_{\hat{k}}$ denote, respectively, the k-iteration index, prox-center and matrix corresponding to the last serious step.

Consider $\ell > \hat\ell$ and recall that, since $M_k^\ell = \hat M + \mu_\ell \mathrm{I}$ and $\{\mu_\ell\}$ is nondecreasing at null and backtracking steps,

$$\frac{1}{2}|d|^2_{\hat k,\ell-1} \le \frac{1}{2}|d|^2_{\hat k,\ell}\,.$$

The sum of this inequality and (22), together with (20) written with ℓ replaced $\ell-1$, results in the relation

$$\mathtt{H}_{\ell-1}(d) \le \check h_\ell(c_{\hat k}(d)) + \frac{1}{2}|d|^2_{\hat k,\ell}\,.$$

In particular, for $d = d^\ell$, we obtain that $\mathtt{H}_{\ell-1}(d^\ell) \le \mathtt{H}_\ell(d^\ell)$ from (20), using the identity $\check h_\ell(c_{\hat k}(d^\ell)) = h_\ell^{\rm agg}(c_{\hat k}(d^\ell))$. Together with (21), written at $d = d^\ell$, we see that

$$\mathtt{H}_{\ell-1}(d^{\ell-1}) \le \mathtt{H}_{\ell-1}(d^{\ell-1}) + \frac{1}{2}|d^\ell - d^{\ell-1}|^2_{\hat k,\ell-1} = \mathtt{H}_{\ell-1}(d^\ell) \le \mathtt{H}_\ell(d^\ell)\,. \quad (25)$$

By the definitions of $h_\ell^{\rm agg}$ and $\mathtt{H}_\ell$, the optimal value in (9) equals $\mathtt{H}_\ell(d^\ell)$. Hence, (25) implies that the sequence of optimal values in (9) is strictly increasing, with $\mathtt{H}_\ell(d^\ell) \le \check h_\ell(c_k(0)) = \check h_\ell(c(\hat x))$. Since, by (15), $\check h_\ell \le h$, then

$$\mathtt{H}_\ell(d^\ell) \le (h \circ c)(\hat x)\,. \quad (26)$$

So $\{\mathtt{H}_\ell(d^\ell)\} \uparrow \mathtt{H}_\infty$ for some $\mathtt{H}_\infty \le (h\circ c)(\hat x)$, with $|d^\ell - d^{\ell-1}|^2_{\hat k,\ell-1} \to 0$ as $\ell \to \infty$, by (25). But $\mu_\ell \ge \mu_{\hat\ell}$ (at null and backtracking steps prox-parameters are nondecreasing), so the left relation in (8) implies that

$$d^\ell - d^{\ell-1} \to 0\,. \quad (27)$$

Using (20) with $d = 0$, we see that $\mathtt{H}_\ell(0) = h_\ell^{\rm agg}(c_{\hat k}(0))$. Together with the definition of $h_\ell^{\rm agg}$, the left inequality in the second line in (10), and the definition of $c_k(d^\ell)$, this implies that that $\mathtt{H}_\ell(0) = \check h_\ell(c_{\hat k}(d^\ell)) - \hat G^{\ell\top} D_{\hat k} d^\ell = \hat G^{\ell\top} c(\hat x)$. Since $\hat G^\ell \in conv\{G^i : i \in \mathtt{B}_\ell\}$, using (5) and (3) we obtain that $\mathtt{H}_\ell(0) \le (h \circ c)(\hat x)$. Therefore, writing (21) with $\ell - 1$ replaced by ℓ at $d = 0$ yields the relations

$$\frac{1}{2}|d^\ell|^2_{\hat k,\ell} = \mathtt{H}_\ell(0) - \mathtt{H}_\ell(d^\ell) \le (h \circ c)(\hat x) - \mathtt{H}_{\hat\ell+1}(d^{\hat\ell+1}),$$

because the sequence $\{\mathtt{H}_\ell(d^\ell)\}$ is increasing, by (25), since $\ell > \hat\ell$. Using once more the left relation in (8) we conclude that the sequence $\{d^\ell\}$ is bounded, and item (i) follows from Proposition 1(i).

To show item (ii), consider iteration indices $\ell - 1, \ell \ge \ell'$, giving two consecutive null steps, and set $C^\ell = c(\hat x) + Dc(\hat x)d^\ell$ and $C^{\ell-1} = c(\hat x) + Dc(\hat x)d^{\ell-1}$.

Since $\delta_\ell \geq 0$, we substract the inequality $\delta_\ell \leq (h \circ c)(\hat{x}) - \check{h}_\ell(C^\ell)$, obtained from (11), from nonsatisfaction of (12), both with $x^k = \hat{x}$, to see that

$$0 \leq (1 - m_1)\delta_\ell \leq h(C^\ell) - \check{h}_\ell(C^\ell). \tag{28}$$

By (3),

$$h(C^{\ell-1}) = G^{\ell-1\top} C^{\ell-1},$$

and by (23),

$$\check{h}_\ell(c_{\hat{k}}(d^\ell)) = \check{h}_\ell(C^\ell) \geq G^{\ell-1\top} C^\ell.$$

Since $\{d^\ell\}$ is bounded, any Lipschitz constant L for h gives an upper bound for $\{|G^\ell|\}$; as a result,

$$\begin{aligned} h(C^\ell) - \check{h}_\ell(C^\ell) &= h(C^\ell) - h(C^{\ell-1}) + h(C^{\ell-1}) - \check{h}_\ell(C^\ell) \\ &\leq L|C^\ell - C^{\ell-1}| + G^{\ell-1\top}(C^{\ell-1} - C^\ell) \\ &\leq 2L|Dc(\hat{x})||d^\ell - d^{\ell-1}|. \end{aligned} \tag{29}$$

From (28) and (27) it follows that $\delta_\ell \to 0$, and by the right hand side expression in (11), $\hat{e}_\ell \to 0$, as stated.

Finally, to see item (iii), the left hand side expression in (11) of δ_ℓ and the definitions of h_ℓ^{agg} and $\mathtt{H}_\ell$ give the identity $\delta_\ell = (h \circ c)(\hat{x}) - \mathtt{H}_\ell(d^\ell)$. Therefore, by the right inequality in (25), (21) with $d = d^\ell$, and the left relation in (8),

$$\begin{aligned} \delta_{\ell-1} &\geq \delta_\ell + \frac{1}{2}|d^\ell - d^{\ell-1}|^2_{\hat{k},\ell-1} \\ &\geq \delta_\ell + \frac{\lambda_{\min}(\hat{M}) + \mu_{\ell-1}}{2}|d^\ell - d^{\ell-1}|^2. \end{aligned}$$

From (29) and (28), we obtain that $\delta_\ell \leq \frac{2L|Dc(\hat{x})|}{1-m_1}|d^\ell - d^{\ell-1}|$, so

$$\delta_{\ell-1} - \delta_\ell \geq \frac{\lambda_{\min}(\hat{M}) + \mu_{\ell-1}}{2}\left(\frac{1 - m_1}{2L|Dc(\hat{x})|}\right)^2 \delta_\ell^2.$$

Letting $K := (1 - m_1)^2 / \left(8|Dc(\hat{x})|^2 L^2\right)$, and summing over $\ell > \hat{\ell}$,

$$K \sum_{\ell > \hat{\ell}} (\lambda_{\min}(\hat{M}) + \mu_{\ell-1})\delta_\ell^2 \leq \delta_{\hat{\ell}} < +\infty.$$

Furthermore, using (17)(a), the series

$$\sum_{\ell>\hat{\ell}} \frac{\mu_{\ell-1} + \lambda_{\min}(\hat{M})}{(\mu_\ell + \lambda_{\max}(\hat{M}))^2} |Dc(\hat{x})^\top \hat{G}^\ell|^4$$

converges too. With our assumption (24), this implies that $\liminf |Dc(\hat{x})^\top \hat{G}^\ell|^4 = 0$. Consider indices ℓ in a corresponding convergent subsequence of $\{d^\ell\}$. Since from the expression for d^ℓ in (10), $|Dc(\hat{x})^\top \hat{G}^\ell|^2 = |M_k^\ell d^\ell|^2 \geq (\lambda_{\min}(\hat{M}) + \mu_{\hat{\ell}})^2 |d^\ell|^2$, we see that the subsequence fo $\{d^\ell\}$ converges to zero and, hence, $\hat{x} + d^\ell \to \hat{x}$ on this subsequence. Finally, by boundedness of $\{C^\ell = c(\hat{x}) + Dc(\hat{x})d^\ell\}$, all outer subgradients are bounded and, hence, the subsequence $\{\hat{G}^\ell\}$ has some accumulation point $\hat{G}^{acc}$ such that $Dc(\hat{x})^\top \hat{G}^{acc} = 0$. The result follows from passing to the limit in (16) to give $\hat{G}^{acc} \in \partial h(c(\hat{x}))$, and using the chain rule (2). □

Remark 2 (*The trivial composite case, suite and end.*) In the setting of Remark 1, when h is not positively homogeneous but merely convex, and c is the identity, the cutting-plane model having the form (6), then (23) states that

$$\text{if iteration } \ell - 1 \text{ was declared null, then } \check{h}_\ell(c_k(d)) \geq h(x^k) - \Delta^k_{\ell-1} + G^{\ell-1\,\top} c_k(d)\,,$$

which is consistent with the fact that $h(x^k) = \Delta^k_{\ell-1}$ when h is positively homogenous, by (5) and (3).

In fact, when $M_k \equiv 0$ for all k, as considered in the trivial structure, Lemmas 1 and 2 boil down to [12, pp.309 and 311, vol.II, Thms.XV.3.2.2 and XV.3.2.4]. □

Putting together Proposition 1 and Lemmas 1 and 2, we can show convergence for objective functions that are *inf-compact*, i.e., functions having some level set that is nonempty and compact (sometimes also referred to as *level-bounded* functions). In this case, by lower semicontinuity, $h \circ c$ always attains its minimum and the sequence of serious steps is bounded.

Theorem 1 (Convergence) *Consider solving problem* (1) *with Composite Algorithm 1 and suppose that in* (1) *the objective function* $h \circ c$ *is inf-compact. If* $0 < m_2 < m_1 < 1$, $\texttt{tol}_{\texttt{stop}} = 0$, *with both* (18) *and* (24) *being satisfied, the following holds.*

(i) Either the sequence of serious steps is infinite and bounded, and at least one of its accumulation points is critical for (1).

(ii) Or the last serious step $\hat{x}$ *is critical for* (1), *with* $\{\hat{x} + d^\ell\} \to \hat{x}$ *for some* ℓ-*subsequence.*

If, instead of (18), *the stronger condition* (18') *holds, item* (i) *can be replaced by*

(i') Either the sequence of serious steps is infinite and bounded, and all its accumulation points are critical for (1). □

Our conditions (18) and (24), on the variable prox-metric, are fairly general and not difficult to enforce. For (18) to hold, it is enough to take matrices M_k^ℓ that are uniformly bounded from above at serious steps (when $\ell = \ell_k$). As for null steps, choosing $\mu_{\ell+1} \in [\mu_\ell, \mu_{\max}]$ for some finite bound $\mu_{\max} > -\lambda_{\min}(\hat{M})$, rule (14) ensures satisfaction of (24). Depending on the particular problem, the more general condition (24) may help in preventing bad choices (too small) for the bound $\mu_{\max}$.

We finish our analysis by supposing the stopping tolerance is positive. In this case, if (18') and (24) hold, by Lemmas 1 and 2, the nominal decrease goes to 0. Then, by (11), both $\hat{e}_\ell$ and $\hat{G}^\ell$ go to 0, and eventually the stopping test will be triggered. For such an iteration index, say ℓ_{best}, (16) gives the following approximate optimality condition for the last serious step $x^{\text{best}} = x^{k(\ell\text{best})}$:

$$\begin{aligned}(h \circ c)(x) &\geq (h \circ c)(x^{\text{best}}) + \hat{G}^{\ell\text{best}\top}\Big(c(x) - c(x^{\text{best}})\Big) - \hat{e}_{\ell\text{best}} \\ &\geq (h \circ c)(x^{\text{best}}) - \hat{G}^{\ell\text{best}\top}\Big(Dc(x^{\text{best}})(x - x^{\text{best}}) + o(|x - x^{\text{best}}|)\Big) - \hat{e}_{\ell\text{best}} \\ &\geq (h \circ c)(x^{\text{best}}) - \texttt{tol}_{\texttt{stop}}|x - x^{\text{best}}| - \texttt{tol}_{\texttt{stop}} + o(|x - x^{\text{best}}|)\,.\end{aligned}$$

for all $x \in \Re^n$. The relation above ensures approximate optimality for x^{best}, even when the composite function is nonconvex.

6 Numerical experience

To assess practical performance of the Composite Bundle method, we coded the Composite Algorithm 1 in MATLAB and ran it on many collections of functions, including some from Sect. 2, using a one 3GHz processor and 1.49GB RAM computer.

6.1 Solvers in the benchmark

We compared the performance of the Composite Bundle method with that of the MATLAB HANSO package, implementing a "Hybrid Algorithm for NSO", and downloadable from http://cs.nyu.edu/overton/software/index.html. In [20] it is explained that, for nonsmooth optimization problems, BFGS may fail theoretically. However, our results below showthe that HANSO package can provide good benchmarks for the purpose of comparison, helping to shed some light on important NSO issues. The HANSO package is organized in two phases:

– Input to a first phase consists of multiple starting points. For each starting point this phase runs a BFGS method for smooth minimization, with a linesearch that attemps to avoid kinks as explained in [20]. If the termination test is not satisfied at the best point found by BFGS, HANSO continues to the next phase.
– Starting from the best point found by BFGS, the second phase executes up to three runs of the Gradient Sampling method in [3] with decreasing sampling radii. For initial information, this methods uses *BFGS final bundle* defined as the last $\min[100, 2n, n + 10]$ generated points and their `(bb)` information. For locally Lipschitz functions, the method converges to Clarke critical points in a probabilistic sense; [3].

We also created another hybrid algorithm, the *Hybrid Composite Bundle*, which after the BFGS phase switches to the Composite Bundle method, starting as does HANSO from BFGS's final bundle.

Therefore, the benchmark considers the following four solvers:

- CBUN, the Composite Bundle method,
- BFGS, the first phase in the HANSO package,
- HANSO, the hybrid variant combining BFGS and Gradient Sampling methods; and
- HYCB, the hybrid variant combining BFGS and CBUN.

6.2 Parameters for the different solvers

Letting n be the problem dimension, the maximum number of iterations and calls to the oracle were set to $\texttt{maxit} = 150\min(n, 20)$ and $\texttt{maxsim} = 300\min(n, 20)$, respectively. We set the stopping tolerance $\texttt{tol}_{\texttt{stop}} = 10^{-5}$ if $n < 50$ and multiply it by $\sqrt{n}$ when $n \geq 50$.

6.2.1 Parameters for CBUN *and* HYCB

The two parameters in (12) and (13) are $m_1 = 0.9$ and $m_2 = 0.55$, respectively. The minimum and maximum positive thresholds are $\mu_{\min} = 10^{-6}$ and $\mu_{\max} = 10^8$. In all of our runs, only strongly active bundle elements are kept at each iteration, but, when there are more than $|\mathtt{B}\max| = 50$ strongly active elements, the bundle is compressed.

If no second-order information is available for the inner mapping, in the variable prox-metric we take $M_k = 0$. Otherwise, we use the combination of Hessian matrices

$$M_k = \sum_{j=1}^{m} \hat{G}_j^{\ell_k} \nabla^2 c_j(x^k), \tag{30}$$

exploiting sparsity patterns, if they exist. To update the prox-parameter, we use an estimate for $\lambda_{\min}(M_k)$, obtained by a modified Cholesky factorization.

The prox-parameter was started at

$$\mu_0 = \frac{5|\gamma(x^0)|^2 10^{-\texttt{fact}}}{1 + |(h \circ c)(x^0)|}\,.$$

The parameter $\texttt{fact}$ is 0 if the function is convex, and 3 otherwise (such a value constrains the search of the next iterate in a region close to x^0, which makes sense for a nonconvex function). At later iterations, every time a serious step is declared, the update is done as follows with $k = k(\ell)$:

$$\mu_{\ell+1} = \min(\mu, \mu_{\max}) \text{ for } \mu = \begin{cases} \max(\mu_{\min}, \mu_\ell, -1.01\lambda_{\min}(M_k)) & \text{if } \lambda_{\min}(M_k) < 0 \\ \max(\mu_{\min}, \mu_\ell^{\text{qN}}) & \text{if } \lambda_{\min}(M_k) = 0 \\ \max(0, \lambda_{\min}(M_k) - \mu_\ell^{\text{AN}}) & \text{if } \lambda_{\min}(M_k) > 0. \end{cases}$$

In these relations, μ_ℓ^{qN} is computed using the *reversal quasi-Newton* scalar update in [1, § 9.3.3]:

$$\mu_\ell^{\rm qN} := \min\left\{ \frac{|\gamma^k - \gamma^{k-1}|^2}{(\gamma^k - \gamma^{k-1})^\top (x^k - x^{k-1})}, \text{ for } \begin{array}{l} \gamma^k \in \left\{ Dc_k^\top \hat{G}^{\ell_k}, Dc_k^\top G^{\ell_k} \right\} \\ \gamma^{k-1} \in \left\{ Dc_k^\top \hat{G}^{\ell_{k-1}}, Dc_k^\top G^{\ell_{k-1}} \right\} \end{array} \right\},$$

recalling that ℓ_k and ℓ_{k-1} denote the last two iterations declaring a serious step ($x^k + d^{\ell_k}$ and $x^{k-1} + d^{\ell_{k-1}}$, respectively). In all cases, if $\lambda_{\min}(M_k) < 0$, then $\mu_{\ell+1} \geq -1.1\lambda_{\min}(M_k)$.

Backtracking steps multiply the current prox-parameter by a factor of two. At consecutive null steps, the prox-parameter is defined as

$$\mu_{\ell+1} = \max(\mu_\ell, \sqrt{(\mu_\ell + \lambda_{\min}(M_k))\texttt{nul}}),$$

for `nul` a counter of null steps. This update satisfies the convergence condition (24). If necessary, $\mu_{\ell+1}$ is projected so that $\mu_{\ell+1} \in [\mu_{\min}, \mu_{\max}]$.

6.2.2 *Parameters for* BFGS *and* Hanso

Both tolerances `normtol` and `evaldist` are set to $\sqrt{\texttt{tol}_{\texttt{stop}}}$. In the linesearch, Armijo's and Wolfe's parameters were taken equal to 0.01 and 0.5, respectively. The option for a so-called Strong Wolfe criterion is not activated, and the method quits when the linesearch fails. As for the quasi-Newton updates, they are of the full memory type, with scaling of the initial Hessian. Finally, the *final bundle* keeps $\min[100, 2n, n+10]$ past gradients.

6.3 Benchmark rules

In an effort to make comparisons fair, we adopted the following rules:

- All solvers use the same black-boxes.
- Each solver has a different computational effort per iteration, which depends not only on the solver, but also on how many times the black-box is called per iteration. The number of iterations is not a meaningful measure for comparison, and the number of (bb) calls is different for each solver. While each iteration of Hanso calls ($\texttt{bb}_\texttt{c}$) and ($\texttt{bb}_\texttt{h}$), CBun calls ($\texttt{bb}_\texttt{h}$) at Step 2, and may or may not call ($\texttt{bb}_\texttt{c}$) and ($\texttt{bb}_\texttt{h}$) at Step 3. Moreover, Hanso always uses gradient values, but CBun requires additional first order information for the inner mapping only at a serious step, to define a new inner model. Striving for fairness, we defined the counters below:
  - For Hanso, one call to both ($\texttt{bb}_\texttt{c}$) and ($\texttt{bb}_\texttt{h}$) counted as one (bb)-call.
  - For CBun, we kept separate counters for the number of c, Dc, and h/G evaluations: `nc`, `nDc`, `nf`, respectively. To each counter corresponds a number of scalars needed for the corresponding calculation: m, mn, and $1+m$, respectively. This gives the number of scalars required for a single evaluation of $h \circ c$

and a subgradient (as in Hanso):

$$\texttt{OneEval} = m(1+n) + 1 + m.$$

Accordingly, the number of oracle evaluations in CBun was defined as

$$\#(\texttt{bb})\text{-calls} = \frac{\texttt{nc}m + \texttt{nDc}mn + \texttt{nf}(1+m)}{\texttt{OneEval}}.$$

At this point a potential advantage of CBun becomes clear: when for a given function the method makes many consecutive null steps, this is not expensive in terms of (bb) calls, since in this case only ($\texttt{bb}_h$) is needed.

- We also computed total CPU time for each solver to reach its stopping test. This information should mostly be taken as a complement to the counter of (bb) calls. The reason is that CPU times can be misleading, because for almost all the tested functions the (bb) calls take a negligible amount of time, a feature that is rare in real-life problems. For example, for the nuclear generation planning problems in [7, Sec.7], 95 % of the total CPU time is spent in producing the black-box information. By contrast, for all but one instance in all of our runs, (bb) calls represent less than 10 % of the total CPU times (taking up to 40 % for one test-function, written in Fortran and requiring a mex-interface).
- All solvers use the same quadratic programming packages, for which there are two possibilities: a Fortran library with a mex-interface, or a special Matlab solver. Quadratic programs such as (9) can be solved by quadprog, the built-in matlab QP solver, but we prefer the method in [15], and made a mex-interface for the Fortran code developed by the author. This method is specially tailored for quadratic minimization over a simplex, as it is the case for the problem dual to (9) and, hence, often outperforms quadprog, which is a general solver. Hanso also has a special QP solver, written in Matlab. Since Hanso QP problems amount to setting $M_k^\ell \equiv 0$ in the inclusion $0 \in conv\{G^i \in \mathtt{B}_\ell\} + M_k^\ell d^\ell$, which is the optimality condition for (9), we could modify the Hanso QP solver to handle (9). Reciprocally, it is possible to use the Fortran QP solver based on [15] to solve Hanso QP problems.
- For large-scale instances, Hanso offers a limited memory BFGS method explained in [33]. However, since for the prox-variable metric the calculation of M_k corresponds to that of a full Hessian, the limited-memory option was not activated in the comparisons.
- Each solver has specific stopping tests, and since BFGS uses a smooth method, the triggers terminating its runs are only heuristic. For each solver we declare a run a success as follows:
  - For BFGS and Hanso, when the tolerance on the shortest vector in the convex hull of certain subgradients is met (for BFGS, these are the subgradients in its *final bundle*).
  - For CBun and HyCB, when the stopping test in Step 2 is reached.
- All non-successful runs are declared failures, with a special counter for when a solver reached the maximum number of iterations or calls to the black-box

(`maxit` or `maxsim`, denoted by `max` in the tables). HANSO does not check if `max` is attained during the Gradient Sampling cycles. Other possible reasons for failures are detection of unboundedness in x or in the function values, errors in the QP solver, and, for BFGS, a nondescent direction, or a problem in the linesearch subroutine.
- The hybrid variants are not initiated if BFGS detected unboundedness. They are started when BFGS succeeds, reaches `maxit`, cannot descend from the generated direction, or if the linesearch or the QP solver failed.
- Both BFGS and CBUN use the same starting points, each function was run for 10 different starting points. Unless otherwise specified, the starting points have all components randomly drawn in $[-1, 1]$.
- To measure the accuracy reached by each solver, we only considered test-functions either with a known optimal value $\bar{f}$, or such that all the solvers converged to the same final value within a tolerance of 10^{-5}. In this case, the optimal value $\bar{f}$ is given by the smallest function value found by all the solvers. Letting f^{best} denote the function value of the analyzed case, then

$$RA := -\log_{10} \max\left(10^{-16}, \frac{f^{\text{best}} - \bar{f}}{1 + |\bar{f}|}\right)$$

  measures the number of digits of accuracy achieved by the solver.
- We exclude from the tables results for those nonconvex cases for which different solvers found different critical points, see Table 8 in the “Appendix” for details.
- Since full tables are large, for the reader’s convenience in this section we only report the overall results of each full table in the “Appendix”.

6.4 Battery of convex test problems

We first consider the typical functions for convex NSO benchmarking in Table 1.

The composite structure $h \circ c$ for functions Maxquad and Ury is the one described by the respective examples in Sect. 2. Both TR48 and TSP are the piecewise maximum of affine functions, so the inner mapping is affine. However, our Fortran `(bb)` code for TSP was too involved to identify the c-components (corresponding to the *minimal 1-trees* in the underlying graph), so we just took the trivial composite case for TSP, as in Remark 1. Similarly for BadGuy, because it does not have a positively homogeneous outer mapping.

Table 2 summarizes the Group 1 results from Table 9 in the “Appendix”. In Table 9, each column corresponds to one of the four solvers, CBUN, BFGS, HANSO, HYCB, run with the Fortran or Matlab special QP solver (denoted by `f` and `m`, respectively). For all of our runs we observed that the Matlab QP special solver is systematically less efficient and/or less reliable than the Fortran one, the shorter Table 2 contains the indicators for the Fortran variants only: CBUN`f`, BFGS`f`, HANSO`f`, HYCB`f`.

To each of the ten functions and each case in Table 1 there corresponds a row in Table 9, reporting the numbers obtained with each solver, averaged over ten runs, with random starting points. All solvers used the same starting points, with components

Table 1 Convex functions in Group 1

Name	n	$\bar{f}$	Reference
BadGuy	10	−2,048	[12, p. 277, vol.II Ex.XV.1.1.2]
Maxquad	10	−0.84140833459641	[1, p. 153]
TR48	48	−638,565	[12, p. 21, vol.II, Ex.IX.2.2.6]
TSP family in [12, p. 22, vol.II, Ex.IX.2.2.7], data from TSPLIB95			
TSP	29	−9,015	`bayg29`
TSP	442	−50,500	`pcb442`
TSP	1,173	−56,349	`pcb1173`
TSP	3,038	−136,601	`pcb3038`
Convex Ury family, low dimension			Example 5 with `cubic`=0
Ury-cvx	10	500	
Ury-cvx	20	911.833349450300716	
Ury-cvx	30	1,118.219919518173128	

Table 2 Group 1 overall results: functions in Table 1

	CBUN`f`			BFGS`f`			HANSO`f`			HYCB`f`		
	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$
BadGuy (10)	8	1.7	135	9	1.9	289	9	2.5	377	9	2.0	301
MQ (10)	11	0.2	40	8	0.2	432	8	0.3	683	12	0.2	443
TR48 (10)	12	3.5	48	13	32.6	2,738	13	291	29,439	15	33	2,746
TSP (40)	8	102	1,045	13	1,947	8,188	13	1,986	18,050	14	2,015	8,686
Ury (30)	11	1.8	45	9	2.6	1,469	11	20	17,762	15	2.7	1,475
Mean (100)	10	22	262	10	397	2,623	11	460	13,262	13	411	2,730
Max/fails	1/1			3/38			0/38			0/0		

in $[-1, 1]$ except for BadGuy, taken in $[-512, 512]$. Results are displayed in three columns, with the accuracy RA, the mean CPU time in seconds $\overline{\texttt{sec}}$, and the average number of black-box calls $\overline{\texttt{(bb)}}$, respectively. At the end of each function family there are two lines. A first line with the mean values averaged for all the considered cases with different n-values (in this line the second column reports between parenthesis the considered number of runs). The second line gives the number of failures and how many of these failures corresponded to having reached the maximum number allowed for iterations or evaluations (`max`/fails). Finally, the bottom two lines of Table 9 in the "Appendix" contain the same indicators, averaged over all the problems in the group, as well as of the total number of instances considered for the test. These bottom lines and the average for each function family are reproduced in Table 2 as a summary.

For Group 1, we observe that all methods are very accurate. When compared to CBUN, BFGS exhibits a significant increase both in CPU times and number of `(bb)` calls; failing to trigger its termination criteria 38 % of the times (only thrice the reason was `max`). For the TSP family, BFGS is 60 % more accurate than CBUN, but for getting 5 more digits, BFGS spends 19 (8) times more CPU ((bb) calls) than CBUN does. We conjecture that endowing TSP with a nontrivial composite structure can improve CBUN's average figures (we observed a significant change for TR48, when comparing CBUN performance on TR48 black-boxes with and without composite structure). HANSO is the slowest solver, and uses the most (bb) calls, but it is not the most accurate method: this hybrid variant did not seem to be adequate for this set of problems, probably because they are all convex. By contrast, HYCB eliminated all of BFGS's failures, with a relatively low additional computational effort: HYCB extra CPU times and (bb) calls represent less than 5 % of BFGS totals. With respect to CBUN, the HYCB gain of 30 % in accuracy is obtained at the cost of an increase of almost 20 and 10 times, respectively, in CPU seconds and (bb) calls. For this group of problems, CBUN performs better than all the other solvers.

6.5 Convex and nonconvex problems

The two other groups include a mix of convex and nonconvex problems given in Table 3. Group 2 gathers together functions with low dimensions ($n \leq 50$), while Group 3 contains high dimensional ones ($n \in \{100, 500\}$). For each instance in Groups 2 and 3, we give the optimal values when known, or the lowest function value found by all solvers, in Table 8 in the "Appendix".

For the CPS, MQ, EucSum, and TiltedNorm-collections, matrices and vectors were generated randomly. All A-matrices are symmetric positive semidefinite, with condition number equal to $rank\ A^2$. The B-matrices in CPS are symmetric positive semidefinite with condition number equal to n^2. To make calculations possible in our computer, for all the sparse matrices the density was set to 0.1, 0.01, 0.01, 0.001 for $n = 10, 50, 100, 500$, respectively. For GenModRos, five different starting points were considered, namely $x^0 \in \Big\{[-0.1, +0.1], [-1, 1], [-2, 0], [0, 2], [-2, 2]\Big\}$.

Tables 4 and 5 summarize the results for Groups 2 and 3, respectively. In the appendix, Tables 10 and 11 report the respective full details.

For Group 2 the instances excluded because different solvers found different critical points correspond to two variants of the functions NK and LV, namely F8 and T3 in [33]; and GenModRos with starting point in $[0, 2]$ and $[-2, 2]$. For this group, the overall results show again that CBUN performs better on average than the other three solvers. However, BFGS did better for some instances of LV, as well as for GenModRos and ModRos. CBUN had difficulties solving the second instance of LV, corresponding to T3 in [33] with $n = 10$. For the excluded instances of GenModRos, the optimal value (5.337) was found often by BFGS while CBUN found only a critical point (with value 9.3283). For the function NesChebRos, nine out of the thirty starting points are very difficult to handle by BFGS, but not by CBUN, explaining the huge difference in accuracy obtained by these solvers. For these problems, we observed that BFGS got stuck at a nonoptimal kink and exited having triggered its heuristic stopping test (the projection of zero on its final bundle was smaller than the tolerance). For contrast, the Rosenbrock modifications GenModRos and ModRos put CBUN into trouble: these are the only problems for which CBUN systematically makes more (bb) evaluations than BFGS. For these functions, CBUN finds a very precise minimizer, after taking many

Table 3 Convex and nonconvex functions in Groups 2 and 3

Name	Parameters	Reference
CPS	$n \in \{10, 50, 100, 500\}$	[33, Sec. 4.2.2]
CVX	$rank\ A \in \{0.2, 0.8\}n$	$f(x) = \sqrt{x^\top Ax} + x^\top Bx$
MQ	$n \in \{10, 50, 100, 500\}$	Example 1,with
CVX	$rank\ A_j \in \{0.2, 0.8\}n$	$c_j(x) = \frac{1}{2}x^\top A_j x + b_j^\top x$
	$m \in 5, 403$	$\{A_j\} \geq 0$ and $\{b_j\}_{j=1}^m$ LI
EucSum	Same than MQ, but $n \in \{4, 10, 50, 100\}$	Example 4 with the ℓ_1-norm, $m_j = 1$, $J = m$, and
NCV		$\phi_j = c_j$ from MQ
TiltedNorm	$n \in \{10, 50, 100, 500\}$	[33, Sec. 4.2.1]
CVX	$w = 4$	$f(x) = w\|Ax\| + (w-1)e_1^\top Ax$
GenModRos	$n = 12$	[33, Sec. 4.2.4]
NCV	$U=1, V=10$	$f(x)=\sum_{i=1}^{n-1}\left(V\frac{i}{n}\|x_{i+1}-x_i^2/n\|+U\frac{i}{n}(1-x_i)^2\right)$
ModRos	$n = 2$	[20, Sec. 5.7]
NCV	$w \in \{1, 2, 4, 8\}$	$f(x) = w\|x_2 - x_1^2\| + (1-x_2)^2$
NesChebRos	$n \in \{5, 10, 50, 100\}$	[20, The nonsmooth variation in Sec. 5.8]
NCV	$x^0 \in [0, 2]$	$f(x) = \sum_{i=1}^{n-1} \|x_{i+1} - 2x_i^2 + 1\| + 0.25(x_1-1)^2$
Ferrier	$n \in \{10, 50, 100\}$	[11]
NCV	$case \in \{1, 3\}$	$f(x)=\begin{cases} \sum_{i=1}^n \|ix_i^2-2x_i+\sum_{j=1}^n x_j\| & \text{if } case=1 \\ \max_{i=1}^n \|ix_i^2-2x_i+\sum_{j=1}^n x_j\| & \text{if } case=3 \end{cases}$
NK	$n \in \{10, 50, 100, 500\}$	Problems F*case* in [33, Sec. 5.4.2]
CVX/NCV	$case \in \{1, 3, 4, 5, 8, 9\}$	See also [9, Sec. 3]
LV	$n \in \{10, 50, 100, 500\}$	Problems T *case* in [33, Sec. 5.4.3]
NCV	$case \in \{3, 4, 5, 6\}$	see also [22]
Ury	$n \in \{10, 20, 30, 100\}$	Example 5
CVX/NCV	`cubic` $\in \{0, 0.01\}$ if $n = 100$,	
	`cubic` = 0.01 otherwise	

serious steps (very short ones); since for each new serious step the mapping Jacobian $Dc(\hat{x}^k)$ is computed, this significantly increases the total (bb) counter. Finally, and as observed for Group 1, HyCB seems to be a better hybrid variant than Hanso.

For Group 3 the instances excluded because different solvers found different critical points correspond to four variants of the functions NK and LV, namely F8 and T3, T5, and T6 in [33]; and EucSum with $n = 100$ and $rank\ A_j = 400$.

As expected, functions in this higher dimensional group are more difficult for all the solvers. The low mean $\overline{\text{(bb)}}$ for CBun indicates that the methods often stalled making many null/backtracking steps, rather than serious steps. However, the second instance of Ferrier functions (corresponding to outer function $h(\cdot) = \max(\cdot)$ and $n = 100$) was difficult for CBun, which made many short serious steps, expensive in terms of (bb) calls. Function MQ with $n = 500$ and $rank\ A_j = 400$ was very difficult to minimize for all methods. The 100 runs of the NK family did not seem difficult for any solver. CBun exited problem TiltedNorm having reached the maximum number of iterations,

Table 4 Group 2 overall results: functions in Table 3, $n \leq 50$

	CBUNf			BFGSf			HANSOf			HYCBf		
	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$
CPS (60)	10	0.1	10	6	0.2	348	6	0.3	470	8	0.2	352
EucSum (50)	9	0.7	17	6	0.8	574	6	2.2	3,708	14	0.9	578
Ferrier (40)	6	0.5	21	4	0.6	777	4	0.9	1,775	7	0.6	783
GenModRos (30)	10	1.5	839	8	0.2	472	8	1.0	1,821	14	0.3	514
LV (60)	8	1.4	334	11	1.9	1,292	11	8.5	6,355	13	1.9	1,297
MQ (40)	9	0.1	11	7	3.8	2,212	7	20	9,611	9	3.9	2,217
ModRos (40)	6	0.1	130	6	0.1	104	6	0.1	124	9	0.2	260
NK (100)	11	0.5	28	8	0.7	577	8	6.1	6,195	14	0.9	589
NesChebRos (30)	16	0.2	71	1	0.7	775	2	8.8	10,335	2	1.2	1,021
TiltedNorm (30)	11	1.0	62	7	0.1	434	7	0.4	1,132	10	0.1	439
Ury (30)	12	2.0	58	11	2.7	1,444	11	20	19,460	14	3.0	1,456
Mean (510)	10	0.7	143	7	1.1	819	7	6.2	5,544	10	1.0	864
Max/fails	29/29			9/76			0/60			23/23		

Table 5 Group 3 overall results: functions in Table 3, $n = 100$ and $n = 500$

	CBUNf			BFGSf			HANSOf			HYCBf		
	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$
CPS (60)	6	0.7	7	5	83	1,945	5	107	2,925	7	84	1,948
EucSum (10)	16	0.3	8	5	0.8	433	5	1.08	734	6	0.9	435
Ferrier (20)	4	6.5	64	10	5.8	3,203	10	9.41	10,437	10	5.9	3,207
LV (40)	4	4.0	65	13	102	8,856	16	232	29,816	13	102	8,860
MQ (40)	4	1,398	40	3	3,951	6,646	9	20,484	33,343	8	5,350	6,657
NK (100)	9	6.5	19	10	86	5,189	10	195	21,102	14	87	5,193
NesChebRos (10)	16	0.9	19	1	29.6	5,247	1	113	44,548	1	29	5,251
TiltedNorm (10)	3	7.9	105	7	1.6	1,372	7	2.0	2,776	7	1.6	1,374
Ury (20)	0	48.0	92	12	62	5,770	12	274	45,070	16	92	5,836
Mean (310)	7	164	47	7	481	4,295	8	2,380	21,195	9	639	4,307
Max/fails	29/29			21/53			0/47			15/15		

while BFGS found a good point and triggered its heuristic stopping test. For Ury, both CBUN and BFGS ended by having reached the maximum number of iterations, but BFGS terminated at a point that is much better than the one found by CBUN. For NesChebRos, BFGS fails in the linesearch, stuck at a nonoptimal kink, and none of the hybrid variants succeeds in getting away from it.

For this group, and especially for its nonconvex functions, we see a more erratic behaviour of CBUN, even though it still has the best performance on average.

6.6 Performance profiles

Figures 1, 2 and 3 contain performance profiles over all the 920 runs, *excluding* cases converging to different critical points, but *including* failures, like in the tables of results. This choice was done not to handicap BFGS, whose heuristic stopping test may sometimes fail to be triggered.

Each curve in a performance profile can be interpreted as a cumulative probability distribution of a resource of interest: accuracy, CPU time, (bb) calls. In general, for a particular solver s, the ordinate $\phi_s(\theta)$ gives information on the percentage of problems that the solver will solve if given a maximum amount of resource. This maximum amount is equal to θ times the minimum amount of resource employed by all solvers. Therefore, for an abscissa $\theta = \theta_{\min}$ in a graph, the probability $\phi_s(\theta_{\min})$ of a particular solver is the probability that the solver will win over all the others. As a result, the solver with higher value of $\phi_s(\theta_{\min})$ should be preferred if a user is only interested in the number of actual wins. On the other hand, for large values of θ, the probability $\phi_s(\theta)$ informs if a solver actually solves a problem. Thus, if a user is concerned only in the probability that a solver will be succesful, the solver with highest $\phi_s(\theta)$ as θ becomes large should be considered. The above explanation supposes that "smaller" values of θ mean "better performance" of the considered resource. Since for accuracy such is not the case, for this indicator we plotted the reciprocal of the figures obtained by each solver. In this manner in all the profiles below, the solver with the highest curve is the best one for the given indicator of performance.

The first profile, in Fig. 1, shows the performance in terms of accuracy. Looking at the highest value for the leftmost abscissa, we conclude that the hybrid variant HyCB is the most precise solver in 72 % of the runs. The three other solvers, CBun, Hanso, and BFGS, are the most accurate solvers in 37, 36, 31 % runs, respectively.

Profile 2 measures the performance in terms of CPU time in seconds, and shows that CBun is the fastest solver in 56 % of the runs, followed by BFGS, which was fastest in 33 % of the runs.

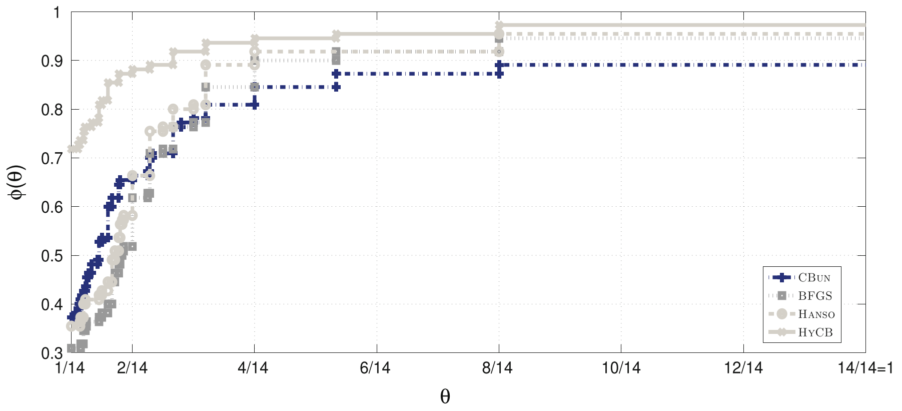

Fig. 1 Performance profile: (reciprocal of) accuracy

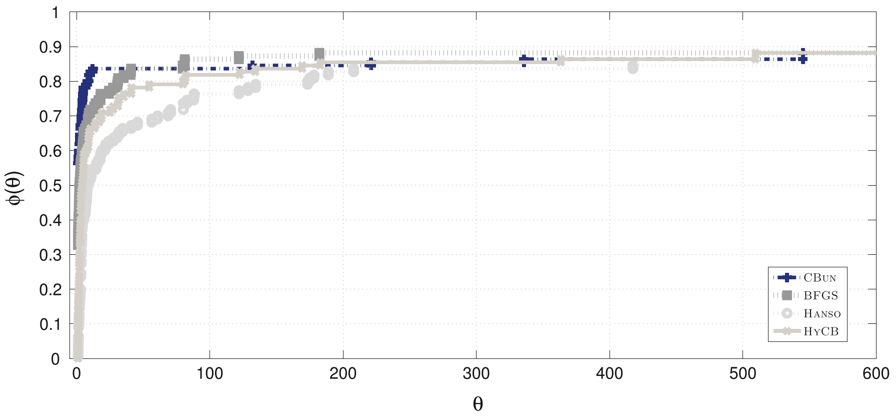

Fig. 2 Performance profile: CPU time

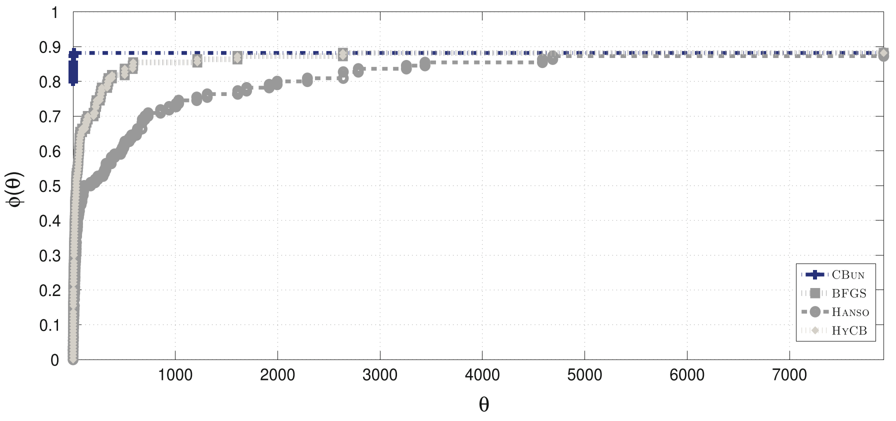

Fig. 3 Performance profile: (bb) calls

The final profile, in Fig. 3, measures the performance of the different solvers in terms of (bb) calls, and shows a clear superiority of CBun, which appears as the most economic solver in 88 % of the cases. Hanso makes extensive use of (bb) calls, so it should mostly be used for unstructured nonconvex functions that are not too difficult to evaluate (possibly like the matrix problems in [33, Sec.5.3]). The curves for BFGS and HyCB practically coincide, making both methods indistinguishable in terms of (bb) calls. Since BFGS is faster and HyCB is more precise, the choice between these two solvers should be driven by the user's preference (speed or accuracy), keeping in mind that HyCB is more reliable in terms of stopping test.

Finally, by examining the right end of the curves in the three profiles, we conclude that all solvers can be deemed similarly successful in solving the considered battery of problems.

6.7 Determining $\mathscr{V}$-dimension

Many composite functions are partly smooth [18], a notion that generalizes to the nonconvex setting the $\mathscr{V}\mathscr{U}$-space decomposition for convex functions in [16,24].

Identification of the $\mathscr{V}\mathscr{U}$ subspaces can be used to determine directions along which the function behaves smoothly so that a (manifold restricted) Newton-like method is likely to succeed. Such smooth directions lie in the $\mathscr{U}$-subspace; its orthogonal complement, the $\mathscr{V}$-subspace, concentrates all the relevant nonsmoothness of the function, at least locally. At a critical point $\bar{x}$, the $\mathscr{V}$-subpace is spanned by the subdifferential $Dc(\bar{x})^\top \partial h(\bar{C})$, with $\bar{C} = c(\bar{x})$, and the $\mathscr{U}$-subspace is the orthogonal complement of $\mathscr{V}$. Alternatively, in the wording of [18], the $\mathscr{U}$-subspace is the subspace tangent to the *smooth activity manifold* at $\bar{x}$, and $\mathscr{V} = \mathscr{U}^\perp$.

In [20,33] it is observed that BFGS can retrieve $\mathscr{V}\mathscr{U}$-information by analyzing the eigenvalues of the inverse Hessian used to define a new iterate. For comparison purposes, we consider CBUN and BFGS only and estimate the dimension of the respective generated $\mathscr{V}$-subspaces as follows:

– For CBUN we compute the dimension of the subspace spanned by the final strongly active gradients in the bundle:

$$dim\, \mathscr{V}_{\text{CBUN}} := rank \left\{ Dc(x^k)^\top (G^i - \hat{G}^\ell) : i \in \mathsf{B}_\ell \text{ with } \alpha_i^\ell > 0 \right\},$$

for x^k the last generated serious step and ℓ the iteration triggering the stopping test; recalling that in Step 4 of Composite Algorithm 1 the bundle sizes are kept controlled by a parameter $|\mathsf{B}\max|$,

– For BFGS we count how many eigenvalues of the final inverse Hessian H cluster near 0:

$$dim\, \mathscr{V}_{\text{BFGS}} := card \left\{ i \le n : \frac{\lambda_i(H)}{\lambda_{\max}(H)} \le \epsilon \right\},$$

for ϵ a given tolerance.

Table 6 reports the obtained results for some of the problems in Groups 1 and 2, with low dimension and $\mathscr{V}$-dimensionality depending on the case. The parameter settings were $|\mathsf{B}\max| = 50$ and $\epsilon \in \{0.1, 0.01\}$ (which gave identical results for this group of runs). Each problem was run 10 times with random starting points. For TR48 the exact $\mathcal{V}$-dimension is reported as ??, because it is unknown.

Table 6 $\mathcal{V}$ dimensions for BadGuy, EucSum, Maxquad, MQ, and TR48

	BadGuy	EucSum				Maxquad	MQ				TR48
n	10	10	10	10	10	10	10	10	10	10	48
$dim\, \mathcal{V}$	10	8	6	4	2	3	8	6	4	2	??
CBUN	10	10	8	6	4	4	8	7	4	3	47
BFGS	2.3	8	3.8	3.4	2.6	3.2	8	6.1	5	7	47

Even though the rules adopted for determining the $\mathscr{V}$-dimensions are rather rough, both CBUN and BFGS estimations are reasonable, with a few exceptions. For both solvers the worst results are those obtained for the nonconvex EucSum functions. The extremely low $\mathscr{V}$-dimension estimated by BFGS for BadGuy comes from the fact that it is hard to automatically determine when a very small eigenvalue should be considered equal to zero. An a posteriori (visual) examination of the eigenvalues obtained for each starting point shows a rather erratic behaviour of BFGS for this function over the different starting points, even though BFGS's heuristic stopping test was always triggered. Such oscillation could be explained by a lack of stability of the Hessian with respect to small perturbations, a common phenomenon for a nonsmooth function near a kink.

We made a second group of runs, to determine the impact of smaller or larger $\mathscr{V}$-dimension, with respect to the dimension of the full space. We considered the CPS function, with dimension $n \in \{10, 50, 100\}$ and varying $\mathscr{V}$-dimension. For this example, the $\mathscr{V}$-dimension coincides with the rank of the matrix A (taken with sparse density equal to 0.1 for all cases).

Table 7 reports the $\mathscr{V}$-dimensions estimated by CBUN and BFGS for different parameters. For CBUN, the maximum bundle size was set to 50 and 100: we expect results to be worse if $|\mathbb{B}\max| < dim\,\mathscr{V} + 1$ and the bundle needs to be compressed to an insufficient number of elements. For BFGS, we took two values of ϵ, as before. In the table, the parameter values appear between parentheses next to the name of each solver.

We observe that problems with larger $\mathscr{V}$-subspaces are more difficult for both solvers. In general, CBUN(100) seems to give a reasonable estimate, but this is not always true, especially when $n = 100$.

We conclude our analysis with Fig. 4, with the real and estimated $\mathscr{V}$-dimensions for all the 30 different functions considered in this subsection. In general, we observe that BFGS overestimates the size of the $\mathscr{V}$-space. We emphasize that this set of tests determining $\mathscr{V}$-dimensionality is only preliminary, and rather crude. For this reason, the conclusions above should not be taken as an indication of goodness or badness of a solver. The subject of determining $\mathscr{V}\mathscr{U}$ subspaces is still rather unexplored, with a few exceptions in [20,33], and the MQ functions considered in [6].

Table 7 $\mathcal{V}$ dimensions for CPS

	CPS																		
n	10	10	10	10	50	50	50	50	50	100	100	100	100	100	100	100	100	100	100
$dim\,\mathcal{V}$	8	6	4	2	40	30	20	10	2	90	80	70	60	50	40	30	20	10	2
CBUN (50)	8	6	4	2	40	28	20	10	2	8	46	36	44	44	30	29	20	10	2
CBUN (100)	8	6	4	2	40	30	20	10	2	77	78	70	56	50	40	30	20	10	2
BFGS (10^{-2})	8	6	4	3	44	40	36	36	36	96	91	77	88	86	85	86	86	87	86
BFGS (10^{-3})	8	6	4	2	43	37	31	24	19	93	86	81	80	75	71	67	61	63	64

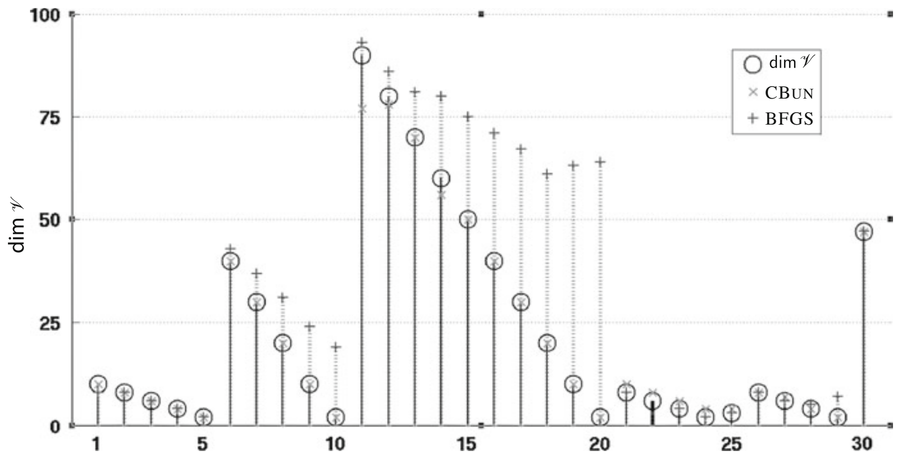

Fig. 4 True and estimated $\mathscr{V}$-dimensions for the 30 functions

7 Concluding remarks

The composite bundle method presented in this work makes tractable the algorithm ProxDescent in [19] for a large class of composite functions, having real-valued, positively homogeneous, and convex outer functions. In particular, the method can be applied to minimize some nonconvex nonsmooth functions, a challenging issue for bundle methods. Our composite cutting-plane model, approximating the conceptual model, avoids typical pitfalls in nonconvex bundle methods.

The numerical experience reported in this work shows the good performance of method for problems of moderate size. For large dimensions, the use of variable prox-metrics may increase the solution times too much, even if there are sparse patterns to exploit. The impact of such an increase is problem dependent: for some functions (such as CPS and TSP) there is a clear advantage in applying a bundle method ($n \leq 500$ in CPS, and $n \leq 3038$ in TSP, but $M_k \equiv 0$). The advantage is less clear for other functions, especially some of the nonconvex ones in Group 3. Since we sometimes also observed that too many short serious steps made CBUN stall, we conjecture that a linesearch (replacing or complementing the curved search modifying μ_ℓ) can improve the performance of Composite Algorithm 1 for nonconvex functions, but this is a subject of future research.

Although BFGS is accurate and fast (at least for our examples, with computationally light blackboxes), neither BFGS nor HANSO appeared as the best alternative for many classes of functions considered in our runs. However, conclusions can be different for a different set of test-functions. Also, the usefulness of a solver depends on the specific purpose sought by the user: since BFGS descends fast from a starting point, it could be an interesting alternative if not much accuracy is required, or if the user seeks a "better" point, without caring if it is the best one. For some problems, we observed that BFGS got stuck at a nonoptimal kink and exited having triggered the heuristic stopping test (the projection of zero onto its final bundle was smaller than the tolerance). If reliability is a concern, the output of BFGS can be plugged into a bundle method, as in HYCB, to satisfy a theoretical stopping test.

However, if too much accuracy is desired, the hybrid variant is likely to increase the computational effort of BFGS too much (at least when compared to applying directly CBUN). As for HANSO, since it makes extensive use of `(bb)` calls, we think it should mostly be used for unstructured nonconvex functions that are not too difficult to evaluate (possibly like the matrix problems in [33, Sec. 5.3]).

Another important issue to consider for a heavy duty application is that, even in the presence of a composite structure, the resulting smooth mapping may be large, or have no special second order sparse patterns to exploit. In this case, it can be sound to use null or diagonal matrices M_k in Composite Algorithm 1, or apply the limited memory variants in [9,33].

We mention the work [14], comparing several NSO general purpose solvers for different type and size of problems, as well as for different `(bb)` available information. Table 3 therein, analizing the efficiency and reliability of the considered solvers, can be useful as complementary information for the conclusions drawn from our numerical results, keeping in mind that solvers are different and that the test-functions are not exactly the same, although there is some intersection.

Comparison with [30]. The proximity control bundle algorithm [30] for nonconvex optimization considers models for functions such that several Clarke subgradients at one point can be computed at reasonable cost. The proposed scheme is fairly general and bears some resemblance to our composite approach, which we explain next.

Instead of assuming that the objective function enjoys some particular structure, in [30] the authors suppose there is available a certain *local model*, $\phi(\cdot, x^k)$, for the objective function f at the current iterate serious x^k. In our notation, $f = h \circ c$, and the local model is $\phi(x^k + \cdot, x^k) = h(c_k(\cdot))$. As explained in [30, Rems. 2.9 and 6.3], such a composite model is both a strong and strict first-order model for f. At each iteration ℓ the local model is approximated by a *working model*, $\phi_\ell(\cdot, x)$, which would correspond to our composite cutting-plane model, $\check{h}_\ell(c_k(\cdot))$, keeping in mind that $k = k(\ell)$. Contrary to our model, the cutting-plane model $\check{h}_\ell$ needs to satisfy both

$$\check{h}_\ell(c(x^k)) = (h \circ c)(x^k) \text{ and } conv\{G^i \in \mathtt{B}_\ell : \check{h}_\ell(c(x^k)) = G^{i\,\top} c(x^k)\} \subset \partial(h \circ c)(x^k).$$

Such relations (equivalent to the conditions $\phi_\ell(x, x) = (h \circ c)(x)$ and $\partial_1\phi_\ell(x, x) \subset \partial_1\phi(x, x)$ imposed to the first-order working model in [30, Def. 3.3]) only hold if the outer subgradient information for $C = c(x^k)$ was kept in the bundle. For our composite bundle method, such is not a requirement for convergence: only the aggregate and the last generated gradient ($\hat{G}^\ell$ and $G^{\ell+1}$, respectively) need to enter the bundle.

Furthermore, the method in [30] drops all the accumulated information every time there is a serious step. More precisely, whenever $k(\ell + 1) = k + 1$, the next working model $\phi_{\ell+1}$ uses the singleton bundle $\mathtt{B} = \{G^{\ell+1}\}$. This procedure, which can be seen as restarting the method from a different initial point, may be justified for general nonconvex functions. However, for our composite functions such a clearing of the bundle may harm efficiency: in our setting the outer function h is convex, and past information can (and should) be kept along iterations to improve the algorithm's performance. Indeed, thanks to the convexity of the outer function, our method ends up with a point that is approximately optimal (recall the last paragraphs in Sect. 5).

Another related important difference is that the working model in [30], in addition to (15), (22), and (23), needs to incorporate *exact* cutting planes, which corresponds to requiring that

$$\forall \ell \geq 1, \text{ given some } \gamma \in \partial(h \circ c)(x^k), \quad (h \circ c)(x^k) + \gamma^\top \cdot \leq \check{h}_\ell(c_k(\cdot)).$$

Instead, in (13) we use the outer subgradient $\Gamma^\ell \in \partial h(C^\ell)$ for $C^\ell = c(x^k + d^\ell)$ to detect if the linearization of the inner mapping is not good enough and trigger the backtracking process. But if $x^k + d^\ell$ is declared a serious step, the corresponding subgradient Γ^ℓ does not enter the bundle (but nothing prevents the bundle management step to incorporate this data).

Like ours, the quadratic programming subproblem in [30] includes a second-order term with a possibly nonpositive definite matrix M_k, augmented by a (positive enough) matrix μ_ℓ. The acceptance test in [30], corresponding to (12)–(13) in our method, does not distinguish between serious and backtracking steps. As for null steps, the decision on whether or not to increase the parameter μ_ℓ is done by checking if, for some parameter $m_3 \in (m_1 - m_2, 1)$,

$$h(c(x^k) + D_k d^\ell) \leq (h \circ c)(x^k) - m_3 \delta_\ell$$

(the prox-parameter is left unchanged if the inequality above does not hold).

Convergence results for the proximity control bundle method with strong first-order models are similar to ours. The method keeps matrices M_k bounded from above and below by $\pm q\mathrm{I}$ for some $0 < q < +\infty$, so (18) always holds. The case of an infinite number of null steps is treated in [30, Lem. 4.1], where it is shown that a subsequence of the prox-parameter sequence diverges and, hence, checking satisfaction of our condition (24) is not straightforward.

Finally, once again because the method does not exploit any underlying convexity, the stopping test [30, (10.7)] checks approximate criticality by computing the shortest element in the *full* subdifferential at x^k. In applications, to perform such test may be too expensive or simply impossible, depending on the function. Sections 7–10 in [30] contain several cases showing the good numerical behavior of the algorithm for difficult functions arising in H_∞-controller synthesis. An interesting subject of future research would be to compare the performance of both algorithms on composite objective functions.

Appendix

See Tables 8, 9, 10, and 11.

Table 8 Optimal (opt) or best (best) function values, for problems in Groups 2 and 3

Name	Parameters	$\overline{h \circ c}$
CPS	$n \in \{10, 50, 100, 500\}$	0 (opt)
CVX	$rank\ A \in \{0.2, 0.8\}n$	
MQ	$n \in \{10, 50, 100, 500\}$	0 (opt)
CVX	$rank\ A \in \{0.2, 0.8\}n$	
	$m = rank\ A + 3$	
EucSum		
NCV	$n = 4, rank\ A = 2$	0.930538450443740 (best)
NCV	$n = 10, rank\ A = 8$	0.465587005455171 (best)
NCV	$n = 10, rank\ A = 2$	0.666424291184390 (best)
NCV	$n = 50, rank\ A = 40$	0.399571129728750 (best)
NCV	$n = 50, rank\ A = 10$	0.002143493387185 (best)
NCV	$n = 100, rank\ A = 80$	Excluded
NCV	$n = 100, rank\ A = 20$	0.333869560359649 (best)
TiltedNorm	$n \in \{10, 50, 100, 500\}$	0 (opt)
CVX	$w = 4$	
GenModRos	$n = 12$	5.377690121369670 (best)
NCV	$U = 1, V = 10$	
ModRos	$n = 2$	0 (opt)
NCV	$w \in \{1, 2, 4, 8\}$	
NesChebRos	$n \in \{5, 10, 50, 100\}$	0 (opt)
NCV	$x^0 \in [0, 2]$	
Ferrier	$n \in \{10, 50, 100\}$	0 (opt)
NCV	$case \in \{1, 3\}$	
NK		
NCV	$case = 8$	Excluded
CVX/NCV	$case \in \{1, 3, 4, 5, 9\}$	$\{0, -\sqrt{2}(n-1), 2(n-1), 2(n-1), 0\}$ (opt)
LV		
NCV	$case = 3, n \in \{10, 50, 100, 500\}$	Excluded
NCV	$case = 4, n = 10$	106.059118520625645 (best)
NCV	$case = 4, n = 50$	587.997761620671213 (best)
NCV	$case = 4, n = 100$	1, 190.421065495747825 (best)
NCV	$case = 4, n = 500$	6, 009.807496496680869 (best)
NCV	$case \in \{5, 6\}, n \leq 100$	0 (best)
NCV	$case \in \{5, 6\}, n = 500$	(excluded)
Ury		
NCV	$n = 10$, `cubic=0.01`	500 (best)
NCV	$n = 20$, `cubic=0.01`	909.889558838787480 (best)
NCV	$n = 30$, `cubic=0.01`	1,114.734712066170232 (best)
CVX	$n = 100$, `cubic=0`	1,159.869805021747879 (best)
NCV	$n = 100$, `cubic=0.01`	1,162.455887489049701 (best)

Table 9 Results for Group 1: problems in Table 1

(bb)	#-n	CBUNf			CBUNm			BFGSf			BFGSm			HANSOf			HANSOm			HYCBf			HYCBm		
		RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$
BadGuy	1–10	8	1.70	135	1	24.32	1,501	9	1.92	289	9	1.94	289	9	2.53	377	9	2.52	378	9	2.00	301	9	22.12	1,793
Mean	(10)	8	1.70	135	1	24.32	1,501	9	1.92	289	9	1.94	289	9	2.53	377	9	2.52	378	9	2.00	301	9	22.12	1,793
Max/fails		0/0			10/10			0/0			0/0			0/0			0/0			0/0			10/10		
MQ	1–10	11	0.16	40	11	1.20	62	8	0.20	432	8	0.77	441	8	0.32	683	8	1.66	1,451	12	0.24	443	12	0.84	452
Mean	(10)	11	0.16	40	11	1.20	62	8	0.20	432	8	0.77	441	8	0.32	683	8	1.66	1,451	12	0.24	443	12	0.84	452
Max/fails		0/0			0/0			0/1			0/1			0/0			0/1			0/0			0/0		
TR48	1–48	12	3.51	48	3	133.67	196	13	32.57	2,738	13	46.02	2,738	13	290.53	29,439	13	232.43	20,537	15	33.16	2,746	13	115.59	2,816
Mean	(10)	12	3.51	48	3	133.67	196	13	32.57	2,738	13	46.02	2,738	13	290.53	29,439	13	232.43	20,537	15	33.16	2,746	13	115.59	2,816
Max/fails		0/0			10/10			0/10			0/10			0/10			0/10			0/0			4/4		
TSP	1–29	16	0.11	50	6	9.68	608	9	0.12	173	9	0.09	173	9	0.20	319	9	0.19	313	9	0.12	180	9	0.10	180
	2–442	7	13.38	996	7	20.30	1,000	16	60.39	5,145	16	68.11	5,145	16	217.17	44,446	16	241.89	44,446	16	66.89	5,660	16	77.13	5,664
	3–1,173	6	91.09	2,044	6	83.85	1,363	16	352.03	5,657	16	360.70	5,657	16	352.03	5,657	16	360.70	5,657	16	374.85	6,202	16	391.58	6,216
	4–3038	4	303.63	1,090	4	330.90	1,092	10	7,375.32	21,777	10	7,352.68	21,777	10	7,375.33	21,777	10	7,352.68	21,777	16	7,617.81	22,701	11	7,586.84	22,592
Mean	(40)	8	102.05	1,045	6	111.18	1,016	13	1,946.97	8,188	13	1,945.40	8,188	13	1,986.19	18,050	13	1,988.87	18,048	14	2,014.92	8,686	13	2,013.91	8,663
Max/fails		1/1			0/0			2/6			2/6			0/6			0/6			0/0			0/0		
Ury	1–10	11	0.18	18	0	66.20	384	8	0.51	5,78	14	3.40	823	14	7.39	9,855	14	6.60	3,932	16	0.53	584	14	13.20	961
	3–20	12	1.08	39	0	175.27	415	13	2.12	1,395	13	10.13	1,395	13	20.29	19,696	13	25.20	8,770	14	2.18	1,401	16	14.59	1,430
	5–30	10	4.20	77	−1	199.46	285	7	5.29	2,433	7	41.38	2,503	7	32.42	2,3734	7	82.82	18,543	16	5.44	2,440	12	84.60	2,635
Mean	(30)	11	1.82	45	−0	146.98	362	9	2.64	1,469	11	18.30	1,574	11	20.03	17,762	11	38.20	10,415	15	2.72	1,475	14	37.46	1,675
Max/fails		0/0			30/30			1/21			1/29			0/22			0/29			0/0			6/6		
Mean	(100)	10	22	262	4	83	627	10	397	2,623	11	402	2,646	11	460	13,262	11	453	10,166	13	411	2,730	12	438	3,080
Max/fails		1/1			50/50			3/38			3/46			0/38			0/46			0/0			20/20		

Table 10 Results for Group 2: problems in Table 3, $n \leq 50$

(bb)	#-n	CBUNf			BFGSf			HANSOf			HYCBf		
		RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$
CPS	1-10	12	0.07	14	7	0.07	184	7	0.11	245	8	0.09	187
	2-10	12	0.03	14	7	0.05	131	7	0.09	196	9	0.06	134
	3-10	11	0.02	8	6	0.04	94	6	0.07	156	9	0.05	98
	4-50	6	0.07	9	5	0.51	772	5	0.65	953	5	0.54	774
	5-50	6	0.08	9	6	0.39	560	6	0.53	741	10	0.42	563
	6-50	10	0.07	8	6	0.27	349	6	0.41	530	9	0.29	352
Mean	(60)	10	0.06	10	6	0.22	348	6	0.31	470	8	0.24	352
Max/fails		0/0			0/0			0/0			0/0		
EucSum	1-4	10	0.34	9	6	0.08	40	6	0.12	65	16	0.10	43
	2-10	7	0.09	27	7	0.21	590	7	6.36	15,622	16	0.25	601
	3-10	16	0.01	6	5	0.02	75	5	0.06	143	5	0.03	77
	4-50	5	3.02	30	6	3.35	1,411	6	3.79	1,626	16	3.49	1,416
	5-50	6	0.13	11	7	0.34	752	7	0.50	1,085	16	0.36	754
Mean	(50)	9	0.72	17	6	0.80	574	6	2.17	3,708	14	0.85	578
Max/fails		1/1			0/0			0/0			0/0		
Ferrier	1-10	8	0.04	8	4	0.11	142	4	0.16	254	7	0.12	148
	2-10	7	0.02	9	4	0.03	141	5	0.11	348	9	0.05	147
	3-50	5	0.26	13	2	1.65	1,865	2	2.76	4,820	6	1.76	1,874
	4-50	5	1.51	53	4	0.39	961	5	0.69	1,678	5	0.46	965
Mean	(40)	6	0.46	21	4	0.55	777	4	0.93	1,775	7	0.60	783
Max/fails		0/0			0/5			0/0			0/0		
GenModRos	1-12	9	1.93	1,050	7	0.24	489	7	1.10	2,044	16	0.35	547

Table 10 continued

(bb)	#-n	CBUNf			BFGSf			HANSOf			HYCBf		
		RA	$\overline{\texttt{sec}}$	$\overline{(\texttt{bb})}$	RA	$\overline{\texttt{sec}}$	$\overline{(\texttt{bb})}$	RA	$\overline{\texttt{sec}}$	$\overline{(\texttt{bb})}$	RA	$\overline{\texttt{sec}}$	$\overline{(\texttt{bb})}$
	2-12	6	2.19	1,229	8	0.21	447	8	0.83	1,568	16	0.32	503
	2-12	6	2.19	1,229	8	0.21	447	8	0.83	1,568	16	0.32	503
	3-12	16	0.43	237	9	0.24	480	9	1.03	1,850	10	0.26	492
	4-12	$\neq$ critical points											
	5-12	$\neq$ critical points											
Mean	(30)	10	1.52	839	8	0.23	472	8	0.99	1,821	14	0.31	514
Max/fails		2/2			0/0			0/0			0/0		
LV	1-10	$\neq$ critical points											
	2-10	4	4.70	1,740	8	0.17	347	8	1.52	2,639	16	0.18	351
	3-10	16	0.02	8	5	0.09	206	5	0.29	479	6	0.11	209
	4-10	16	0.03	15	4	0.15	305	5	0.24	453	7	0.16	311
	5-50	$\neq$ critical points											
	6-50	3	1.89	149	16	5.16	2,534	16	42.32	29,835	16	5.22	2,539
	7-50	3	1.21	55	16	2.07	1,582	16	2.33	1,764	16	2.11	1,586
	8-50	5	0.65	34	16	3.84	2,778	16	4.10	2,960	16	3.88	2,783
Mean	(60)	8	1.42	334	11	1.91	1,292	11	8.47	6,355	13	1.94	1,297
Max/fails		10/10			0/10			0/10			0/0		
MQ	2-10	10	0.06	13	8	0.30	680	9	1.66	3,755	10	0.32	687
	3-10	10	0.02	10	7	0.04	144	7	0.07	205	10	0.05	147
	4-50	7	0.44	11	4	13.74	5,769	4	76.94	31,841	8	14.11	5,775
	5-50	9	0.06	8	8	1.08	2,256	8	1.31	2,642	9	1.13	2,259
Mean	(40)	9	0.14	11	7	3.79	2,212	7	20.00	9,611	9	3.90	2,217

Table 10 continued

(bb)	#-n	CBunf			BFGSf			Hansof			HyCBf		
		RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$
Max/fails		0/0			7/20			0/10			0/0		
ModRos	1-2	9	0.07	38	6	0.02	37	6	0.03	68	10	0.04	67
	2-2	7	0.04	53	6	0.02	56	6	0.03	80	10	0.06	100
	3-2	4	0.13	150	6	0.03	109	6	0.03	122	9	0.27	382
	4-2	2	0.24	278	5	0.06	214	5	0.07	228	6	0.30	492
Mean	(40)	6	0.12	130	6	0.03	104	6	0.04	124	9	0.17	260
Max/fails		13/13			0/0			0/0			17/17		
NK	1-10	9	0.10	32	6	0.03	63	6	0.07	126	16	0.08	81
	10-50	16	0.55	23	8	0.06	75	8	0.20	298	11	0.17	79
	11-50	$\neq$ critical points											
	12-50	16	0.11	8	5	0.04	72	5	0.15	269	10	0.13	76
	2-10	16	0.30	82	6	0.07	191	7	0.17	408	9	0.32	237
	3-10	13	0.05	12	6	0.07	199	6	0.34	741	16	0.09	205
	4-10	14	0.09	31	7	0.02	70	7	0.06	140	16	0.05	77
	5-10	$\neq$ critical points											
	6-10	11	0.03	12	5	0.02	76	5	0.06	141	16	0.06	88
	7-50	8	0.44	22	5	0.09	238	5	0.18	431	16	0.43	251
	8-50	5	3.01	44	16	3.14	2,351	16	23.55	29,652	16	3.32	2,357
	9-50	6	0.45	11	16	3.79	2,438	16	36.04	29,739	16	3.98	2,443
Mean	(100)	11	0.51	28	8	0.73	577	8	6.08	6,195	14	0.86	589
Max/fails		0/0			0/19			0/19			1/1		
NesChebRos	1-5	16	0.36	188	1	0.16	415	2	0.58	1,886	2	0.45	614

Table 10 continued

(bb)	#-n	CBUNf			BFGSf			HANSOf			HYCBf		
		RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\texttt{(bb)}}$
	2-10	16	0.02	9	2	0.08	184	2	0.15	398	2	1.09	716
	3-50	16	0.21	14	1	1.87	1,727	1	25.59	28,720	1	1.91	1,732
Men	(30)	16	0.20	71	1	0.70	775	2	8.77	10,335	2	1.15	1,021
Max/fails		3/3			2/3			0/2			5/5		
TiltedMax	1-10	14	0.10	30	6	0.05	180	6	0.13	407	15	0.07	186
	2-20	13	0.25	46	7	0.09	358	7	0.30	1,043	9	0.12	365
	3-50	6	2.63	109	7	0.22	763	7	0.60	1,946	7	0.24	766
Mean	(30)	11	0.99	62	7	0.12	434	7	0.35	1,132	10	0.14	439
Max/fails		0/0			0/0			0/0			0/0		
Ury	2-10	16	0.18	20	7	0.29	399	7	10.16	14,847	10	0.30	405
	4-20	11	1.01	42	11	1.90	1,305	11	18.19	19,606	16	2.06	1,316
	6-30	10	4.66	110	16	5.81	2,627	16	30.98	23,928	16	6.61	2,649
Mean	(30)	12	1.95	58	11	2.67	1,444	11	19.78	19,460	14	2.99	1,456
Max/fails		0/0			0/19			0/19			0/0		
Mean	(510)	10	0.73	143	7	1.07	819	7	6.17	5,544	10	1.20	864
Max/fails		29/29			9/76			0/60			23/23		

Table 11 Results for Group 3: problems in Table 3, $n = 100$ and $n = 500$

(bb)	#-n	CBUNf			BFGSf			HANSOf			HYCBf		
		RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\texttt{sec}}$	$\overline{\text{(bb)}}$
CPS	10-500	6	1.31	7	3	106.27	2,409	3	227.11	6,780	3	106.60	2,410
	11-500	5	1.26	7	5	229.58	4,214	5	238.07	4,515	8	230.25	4,217
	12-500	5	1.25	7	5	152.19	2,011	5	160.79	2,312	9	152.93	2,014
	7-100	6	0.13	7	5	5.27	1,464	5	5.95	1,765	5	5.33	1,466
	8-100	6	0.14	8	5	4.26	1,005	5	4.85	1,306	9	4.34	1,008
	9-100	6	0.14	8	6	2.29	569	6	2.95	870	10	2.36	572
Mean	(60)	6	0.70	7	5	83.31	1,945	5	106.62	2,925	7	83.64	1,948
Max/fails		0/0			0/0			0/0			0/0		
EucSum	6-100	$\neq$ critical points											
	7-100	16	0.27	8	5	0.78	433	5	1.08	734	6	0.84	435
Mean	(10)	16	0.27	8	5	0.78	433	5	1.08	734	6	0.84	435
Max/fails		0/0			0/0			0/0			0/0		
Ferrier	5-100	5	0.88	14	14	9.07	4,657	14	15.64	17,958	14	9.16	4,661
	6-100	4	12.20	114	5	2.58	1,748	5	3.18	2,916	5	2.62	1,752
Mean	(20)	4	6.54	64	10	5.83	3,203	10	9.41	10,437	10	5.89	3,207
Max/fails		1/1			0/2			0/0			0/0		
LV	10-100	3	0.54	13	16	22.09	4,676	16	112.32	43,976	16	22.17	4,679
	11-100	3	5.44	113	16	17.69	3,676	16	18.38	3,978	16	17.81	3,680
	12-100	4	5.65	121	4	29.24	5,882	16	40.52	10,822	5	29.49	5,887
	13-500	$\neq$ critical points											
	14-500	4	4.25	13	16	338.32	21,189	16	757.08	60,490	16	338.82	21,191
	15-500	$\neq$ critical points											

Table 11 continued

(bb)	#-n	CBUNf			BFGSf			HANSOf			HYCBf		
		RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$
	16-500	$\neq$ critical points											
	9-100	$\neq$ critical points											
Mean	(40)	4	3.97	65	13	101.84	8,856	16	232.08	29,816	13	102.08	8,860
Max/fails		0/0			1/5			0/4			0/0		
MQ	6-100	2	90.94	105	4	165.89	6,536	16	804.07	42,648	5	172.30	6,544
	7-100	16	0.15	9	5	12.16	5,122	5	13.23	6,178	6	12.27	5,125
	8-500	−0	5,023.56	22	0	14, 634.68	8,352	0	75,735.20	42,086	16	20,200.74	8,384
	9-500	0	478.97	25	3	991.68	6,572	16	5,382.02	42,461	3	1,016.28	6,575
Mean	(40)	4	1,398.41	40	3	3, 951.10	6,646	9	20,483.63	33,343	8	5,350.40	6,657
Max/fails		6/6			6/8			0/5			2/2		
NK	13-100	16	1.35	23	5	0.63	410	5	0.80	711	5	0.80	413
	14-100	4	7.81	36	16	13.09	4,443	16	58.45	43,744	16	13.41	4,447
	15-100	5	0.91	9	16	16.35	4,752	16	79.97	44,052	16	16.86	4,756
	16-100	12	0.88	21	8	0.13	83	8	0.39	384	16	0.26	86
	17-100	$\neq$ critical points											
	18-100	10	0.21	8	4	0.09	69	4	0.29	370	16	0.31	75
	19-500	5	19.78	39	4	30.79	902	4	32.57	1,204	16	39.20	920
	20-500	2	11.11	7	16	342.25	19,620	16	763.23	58,920	16	343.85	19,621
	21-500	6	16.54	18	16	451.70	21,468	16	1,006.16	60,769	16	453.87	21,470
	22-500	16	5.44	22	8	2.25	68	8	6.47	489	11	3.20	72
	23-500	$\neq$ critical points											
	24-500	16	0.83	7	5	2.16	72	5	4.41	373	8	2.64	74

Table 11 continued

(bb)	#-n	CBUNf			BFGSf			HANSOf			HYCBf		
		RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$	RA	$\overline{\text{sec}}$	$\overline{\text{(bb)}}$
Mean	(100)	9	6.49	19	10	85.94	5,189	10	195.28	21,102	14	87.44	5,193
Max/fails		0/0			0/8			0/8			0/0		
NesChebRos	4-100	16	0.84	19	1	29.63	5,247	1	112.88	44,548	1	29.73	5,251
Mean	(10)	16	0.84	19	1	29.63	5,247	1	112.88	44,548	1	29.73	5,251
Max/fails		0/0			0/10			0/10			0/0		
TiltedMax	4-100	3	7.84	105	7	1.59	1,372	7	2.04	2,776	7	1.62	1,374
Mean	(10)	3	7.84	105	7	1.59	1,372	7	2.04	2,776	7	1.62	1,374
Max/fails		2/2			0/0			0/0			0/0		
Ury	7-100	1	70.85	94	7	61.95	5,854	7	269.76	45,155	16	95.00	5,897
	8-100	0	25.19	91	16	61.52	5,685	16	277.93	44,986	16	88.49	5,774
Mean	(20)	0	48.02	92	12	61.73	5,770	12	273.85	45,070	16	91.74	5,836
Max/fails		20/20			14/20			0/20			13/13		
Mean	(310)	7	163.67	47	7	480.19	4,295	8	2,379.65	21,195	9	639.26	4,307
Max/fails		29/29			21/53			0/47			15/15		

References

1. Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Theoretical and Practical Aspects. Universitext, 2nd edn, pp xiv+423. Springer, Berlin (2006)
2. Bonnans, J.F., Shapiro, A.: Perturbation Analysis of Optimization Problems. Springer Series in Operations Research, Springer, New York (2000)
3. Burke, J., Lewis, A., Overton, M.: Two numerical methods for optimizing matrix stability. Linear Algebra Appl. **351–352**, 117–145 (2002)
4. Burke, J.V., Ferris, M.C.: A Gauss-Newton method for convex composite optimization. Math. Program. **71**, 179–194 (1995). doi:10.1007/BF01585997
5. Correa, R., Lemaréchal, C.: Convergence of some algorithms for convex minimization. Math. Program. **62**(2), 261–275 (1993)
6. Daniilidis, A., Sagastizábal, C., Solodov, M.: Identifying structure of nonsmooth convex functions by the bundle technique. SIAM J. Optim. **20**(2), 820–840 (2009). doi:10.1137/080729864. http://link.aip.org/link/?SJE/20/820/1
7. Emiel, G., Sagastizábal, C.: Incremental-like bundle methods with application to energy planning. Comput. Optim. Appl. **46**, 305–332 (2010). doi:10.1007/s10589-009-9288-8
8. Fletcher, R.: Practical Methods of Optimization, 2nd edn. Wiley, Chichester (1987)
9. Haarala, M., Miettinen, K., Makela, M.M.: New limited memory bundle method for large-scale nonsmooth optimization. Optim. Methods Softw. (6), 673–692 (2004). http://www.informaworld.com/10.1080/10556780410001689225
10. Hare, W.: Nonsmoth Optimization with Smooth Substructure. Ph.D. thesis, Department of Mathematics, Simon Fraser University (2003)
11. Hare, W., Sagastizábal, C.: A redistributed proximal bundle method for nonconvex optimization. SIAM J. Optim. **20**(5), 2442–2473 (2010). doi:10.1137/090754595
12. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. No. 305–306 in Grund. der math. Wiss, vol. 2. Springer, Berlin (1993)
13. Hiriart-Urruty, J.B., Lemaréchal, C.: Fundamentals of Convex Analysis. Grundlehren Text Editions, Springer, Berlin (2001)
14. Karmitsa, N., Bagirov, A., Makela, M.M.: Comparing different nonsmooth minimization methods and software. Optim. Methods Softw. (2010). http://www.informaworld.com/10.1080/10556788.2010.526116
15. Kiwiel, K.: A method for solving certain quadratic programming problems arising in nonsmooth optimization. IMA J. Numer. Anal. **6**, 137–152 (1986)
16. Lemaréchal, C., Oustry, F., Sagastizábal, C.: The $\mathcal{U}$-Lagrangian of a convex function. Trans. Am. Math. Soc. **352**(2), 711–729 (2000)
17. Lemaréchal, C., Sagastizábal, C.: Practical aspects of the Moreau-Yosida regularization: theoretical preliminaries. SIAM J. Optim. **7**(2), 367–385 (1997). http://link.aip.org/link/?SJE/7/367/1
18. Lewis, A.: Active sets, nonsmoothness and sensitivity. SIAM J. Optim. **13**(3), 702–725 (2002)
19. Lewis, A., Wright, S.: A Proximal Method for Composite Minimization (2008). Available at http://www.optimization-online.org/DB_HTML/2008/12/2162.html
20. Lewis, A.S., Overton, M.L.: Nonsmooth optimization via quasi-Newton methods. Math. Program. (2011). Accepted for publication
21. Li, C., Wang, X.: On convergence of the gauss-newton method for convex composite optimization. Math. Program. **91**, 349–356 (2002). doi:10.1007/s101070100249
22. Luksan, L., Vlcek, J.: Test Problems for Nonsmooth Unconstrained and Linearly Constrained Optimization. Technical Report 798, Institute of Computer Science, Academy of Sciences of the, Czech Republic (2000)
23. Mifflin, R.: Semi-smooth and semi-convex functions in constrained optimization. SIAM J. Control Optim. **15**, 959–972 (1977)
24. Mifflin, R., Sagastizábal, C.: On $\mathcal{VU}$-theory for functions with primal-dual gradient structure. SIAM J. Optim. **11**(2), 547–571 (2000). http://siamdl.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=SJOPE8000011000002000547000001\&idtype=cvips\&gifs=Yes
25. 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)
26. Mifflin, R., Sagastizábal, C.: $\mathcal{VU}$-smoothness and proximal point results for some nonconvex functions. Optim. Methods Softw. **19**(5), 463–478 (2004)

27. Mifflin, R., Sagastizábal, C.: A $\mathscr{V}U$-algorithm for convex minimization. Math. Program. Ser. A **104**(2–3), 583–608 (2005). doi:10.1007/s10107-005-0630-3
28. Nemirovsky, A., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. A Wiley-Interscience Publication (1983)
29. Nesterov, Y.: Gradient methods for minimizing composite objective functions. Math. Program. (2011). Accepted for publication
30. Noll, D., Prot, O., Rondepierre, A.: A proximity control algorithm to minimize nonsmooth and non-convex functions. Pac. J. Optim. **4**(3), 569–602 (2008)
31. Oustry, F.: A second-order bundle method to minimize the maximum eigenvalue function. Math. Program. **89**(1, Ser. A), 1–33 (2000)
32. Shapiro, A.: On a class of nonsmooth composite functions. Math. Oper. Res. **28**(4), 677–692 (2003)
33. Skajaa, A.: Limited Memory BFGS for Nonsmooth Optimization. Master's thesis, Courant Institute of Mathematical Science (2010). http://cs.nyu.edu/overton/mstheses/skajaa/msthesis.pdf
34. Womersley, R.: Local properties of algorithms for minimizing nonsmooth composite functions. Math. Program. **32**, 69–89 (1985)