Math. Program., Ser. A (2016) 156:125–159
DOI 10.1007/s10107-015-0873-6

FULL LENGTH PAPER

A doubly stabilized bundle method for nonsmooth convex optimization

Welington de Oliveira · Mikhail Solodov

Received: 13 April 2013 / Accepted: 9 February 2015 / Published online: 20 February 2015

Abstract We propose a bundle method for minimizing nonsmooth convex functions that combines both the level and the proximal stabilizations. Most bundle algorithms use a cutting-plane model of the objective function to formulate a subproblem whose solution gives the next iterate. Proximal bundle methods employ the model in the objective function of the subproblem, while level methods put the model in the subproblem's constraints. The proposed algorithm defines new iterates by solving a subproblem that employs the model in both the objective function and in the constraints. One advantage when compared to the proximal approach is that the level set constraint provides a certain Lagrange multiplier, which is used to update the proximal parameter in a novel manner. We also show that in the case of inexact function and subgradient evaluations, no additional procedure needs to be performed by our variant to deal with inexactness (as opposed to the proximal bundle methods that require special modifications). Numerical experiments on almost 1,000 instances of different types of problems are presented. Our experiments show that the doubly stabilized bundle method inherits useful features of the level and the proximal versions, and compares favorably to both of them.

Keywords Nonsmooth optimization · Proximal bundle method · Level bundle method · Inexact oracle

Mikhail Solodov is supported in part by CNPq Grant 302637/2011-7, by PRONEX-Optimization and by FAPERJ.

W. de Oliveira (✉) · M. Solodov
IMPA – Instituto de Matemática Pura e Aplicada, Estrada Dona Castorina 110,
Jardim Botânico, Rio de Janeiro, RJ 22460-320, Brazil
e-mail: wlo@impa.br

M. Solodov
e-mail: solodov@impa.br

Mathematics Subject Classification 90C25 · 90C30 · 65K05

1 Introduction

In this work, we are interested in solving problems of the form

$$f^{\inf} := \inf_{x \in \mathscr{X}} f(x), \tag{1}$$

where $f : \Re^n \to \Re$ is a nonsmooth convex function and $\mathscr{X} \subset \Re^n$ is a nonempty convex and closed set, typically polyhedral. As is widely accepted, the most efficient optimization techniques to solve such problems are the bundle methods, e.g., [18, Chap. XIV], [4, Part II], and the analytic-center cutting-plane methods, e.g., [15,16]. All bundle methods make use of the following three ingredients:

(i) a convex model $\check{f}_k$ of f (usually, $\check{f}_k$ is a cutting-plane approximation satisfying $\check{f}_k \leqslant f$);
(ii) a stability center $\hat{x}_k$ (some previous iterate, usually the "best" point generated by the iterative process so far);
(iii) a certain algorithmic parameter updated at each iteration (proximal, level, or trust-region, depending on the variant of the method).

The new iterate x_{k+1} of a bundle method depends on the above three ingredients, whose organization defines different methods. The main classes are the proximal, level, and trust-region. We next discuss some details of the proximal and level variants, as these are the two strategies relevant for our developments. The simplified conceptual versions can be stated as follows.
Proximal bundle method e.g., [10,13,19,23],

$$x_{k+1} := \arg\min \left\{ \check{f}_k(x) + \frac{1}{2\tau_k}|x - \hat{x}_k|^2 : \ x \in \mathscr{X} \right\}, \tag{2}$$

where $\tau_k > 0$ is the proximal parameter.
Level bundle method e.g., [5,9,20,22],

$$x_{k+1} := \arg\min \left\{ \frac{1}{2}|x - \hat{x}_k|^2 : \ \check{f}_k(x) \leq \ell_k, \quad x \in \mathscr{X} \right\}, \tag{3}$$

where $\ell_k \in \Re$ is the level parameter.

As is well known, for the same model $\check{f}_k$ and the same stability center $\hat{x}_k$, one can find the proximal and level parameters τ_k and ℓ_k such that the two versions above generate the same next iterate x_{k+1} (i.e., for some choice of parameters, the solution of the subproblems (2) and (3) is the same). In this (formal, theoretical) sense the two approaches can be considered equivalent. Details of the implementation and practical performance can be quite different, however. In particular, because the parameters are updated by strategies specific to each of the methods, and the corresponding rules are not related in any direct way.

It is worth to mention that updating the stability center $\hat{x}_k$ in item (ii) above is mandatory for (at least the most standard) versions of proximal bundle methods, but it may not be necessary for level methods. In some variants of level methods one can update $\hat{x}_k$ at each iteration [12,22], or keep $\hat{x}_k = \hat{x}$ fixed for all iterations [2] (in which case $\hat{x}_k$ does not have the role of the "best" point computed so far). See also [3,5,20] for various rules to manage the stability center $\hat{x}_k$ in level methods.

It should be stressed that the choice of the parameter τ_k in the proximal variant is quite a delicate task. Although the simplest choice $\tau_k = \tau > 0$ (for all k) is enough to prove theoretical convergence, it is well understood that for practical efficiency τ_k must be properly updated along iterations. We refer to [19] and [23] for some strategies that usually work well in practice. However, the former has some heuristic features, while the latter (based on quasi-Newton formulas) is designed for unconstrained problems and needs some safeguards to fit the general convergence theory. Also, it was noticed during numerical experimentation in [25] that for constrained problems the rule of [23] does not work as well as for the unconstrained.

Continuing the discussion of choosing parameters, a fixed level parameter ℓ_k is not possible, of course, as this may give infeasible subproblems (3). But there exist strategies that manage ℓ_k by simple explicit calculations (whether the problem is unconstrained or constrained), and which are theoretically justified. As a somewhat more costly but very efficient option, the level parameter ℓ_k can be adjusted by solving a linear program (when the feasible set $\mathcal{X}$ is polyhedral, and is either bounded or there is a known lower bound f^{low} for the optimal value f^{inf}); see [12,22] and (17) below.

Overall, there seems to be a consensus that for solving unconstrained problems proximal bundle methods are very good choices, although the updating rule for τ_k is somewhat of an issue (at least from the viewpoint of combining theory and efficiency). On the other hand, there is some evidence that for constrained problems level bundle methods might be preferable. Also, strategies for updating the level parameter ℓ_k are readily available. It is thus appealing to try to combine the attractive features of both approaches in a single algorithm that performs for unconstrained (respectively, constrained) problems as well as proximal bundle methods (respectively, level bundle methods), or maybe even better in some cases. To this end, we propose what we call a *doubly stabilized bundle method*, that combines both proximal and level stabilizations in the same subproblem, namely:

$$x_{k+1} := \arg\min \left\{ \check{f}_k(x) + \frac{1}{2\tau_k}|x - \hat{x}_k|^2 : \ \check{f}_k(x) \le \ell_k, \quad x \in \mathcal{X} \right\}. \tag{4}$$

We immediately comment that (4) can be reformulated as a quadratic program (if $\mathcal{X}$ is polyhedral), just like (2) or (3), with just one extra scalar bound constraint compared to (2), or one extra scalar variable and scalar bound constraint compared to (3); see (8) below. The dual for (4) is also very similar in structure to the duals of (2) or (3). Thus, the subproblem (4) is no harder (or at least, cannot be much harder) to solve than (2) or (3). Moreover, it turns out that the (unique) solution to problem (4) is also a solution to at least one of the problems (2) or (3); see Lemma 1 below. This reveals that the proposed method indeed combines the proximal and the level approaches, "automatically" choosing between the two at every step.

The advantages derived from (4) can be summarized as follows:

- the level parameter ℓ_k is easily updated, and can take into account a lower bound for $f^{\inf}$ if it is available;
- the level constraint $\check{f}_k(x) \leq \ell_k$ provides:
  - a Lagrange multiplier useful to update the proximal parameter τ_k;
  - an additional useful stopping test, based on a certain optimality gap;
- the objective function $\check{f}_k(x) + \frac{1}{2\tau_k}|x - \hat{x}_k|^2$ with proximal regularization allows for searching for good points *inside* of the level set $\{x \in \mathcal{X} : \check{f}_k(x) \leq \ell_k\}$, and not only on its boundary, as the level method does.

(It should be noted here that proximal bundle methods can also exploit known lower bounds for $f^{\inf}$ by adding certain associated linearizations [14]).

Among other things, our new variant aims at taking advantage of the simplicity of managing the level parameter ℓ_k to produce a simple and efficient rule to update the proximal parameter τ_k. In particular, this update depends on whether or not the level constraint is active. In this sense, activity of this constraint (and the associated Lagrange multiplier) can be seen as a tool indicating when and how to update τ_k. Furthermore, depending on the feasible set $\mathcal{X}$ (for example if it is bounded), the management of the level parameter can provide a lower bound for $f^{\inf}$, giving an additional stopping test based on a certain optimality gap. It will be shown in Sect. 2 that the lower bound can be updated in a computationally cheap way.

The idea to combine proximal and level bundle methods was first suggested in [8] (giving some limited numerical experiment, without proof of convergence and without handling inexact data). To the best of our knowledge, the only other bundle-type method which employs some kind of double stabilization is [1], where the proximal and trust-region features are present for piecewise quadratic models of f. However, the motivation for this and the resulting algorithm are rather different from ours. For example, the subproblems in [1] are that of minimizing a quadratic function subject to *quadratic constraints*.

The rest of this paper is organized as follows. Section 2 introduces the doubly stabilized bundle algorithm more formally. Section 3 is devoted to convergence analysis of the method. Inexactness of the function and subgradient evaluations is addressed in Sect. 4. Section 5 contains numerical experiments comparing the proposed algorithm with: the proximal bundle methods using the updating rules for τ_k based on [19] and [23]; and the level method given in [5]. A variety of different types of problems are used to validate our proposal: the model unit-commitment problem in the energy sector, two-stage stochastic linear programming, nonsmoothly-regularized maxima of quadratic functions, and some standard nonsmooth optimization test problems. Finally, Sect. 6 gives some concluding comments and remarks.

Our notation is standard. For any points $x, y \in \Re^n$, $\langle x, y\rangle$ stands for the Euclidean inner product, and $|\cdot|$ for the associated norm, i.e., $|x| = \sqrt{\langle x, x\rangle}$. For a set $\mathcal{X} \subset \Re^n$, we denote by $i_{\mathcal{X}}$ its indicator function, i.e., $i_{\mathcal{X}}(x) = 0$ if $x \in \mathcal{X}$ and $i_{\mathcal{X}}(x) = +\infty$ otherwise. For a convex set $\mathcal{X}$, $\mathtt{ri}\,\mathcal{X}$ stands for its relative interior, and $N_{\mathcal{X}}(x)$ for its normal cone at the point x, i.e., the set $\{y : \langle y, z - x\rangle \leqslant 0\ \forall z \in \mathcal{X}\}$ if $x \in \mathcal{X}$ and the empty set otherwise. Given a convex function f, we denote its subdifferential at the point x by $\partial f(x) = \{g : f(y) \geqslant f(x) + \langle g, y - x\rangle\ \forall y\}$.

2 A doubly stabilized bundle method

The method generates a sequence of feasible iterates $\{x_k\} \subset \mathscr{X}$. For each point x_k an oracle (black-box) is called to compute the function value $f(x_k)$ and one arbitrary subgradient $g_k \in \partial f(x_k)$. With this information, the method creates the linearization

$$\bar{f}_k(x) := f(x_k) + \langle g_k, x - x_k \rangle \leqslant f(x), \tag{5}$$

where the inequality holds by the definition of the subgradient of f. At iteration k, a polyhedral *cutting-plane* model of f is available:

$$\check{f}_k(x) := \max_{j \in \mathscr{B}_k} \bar{f}_j(x) \leqslant f(x), \tag{6}$$

where the set $\mathscr{B}_k$ may index some of the linearizations $\bar{f}_j$, $j \leqslant k$, of the form in (5), but also affine functions obtained as certain convex combinations of such previous linearizations (the so-called aggregate linearizations, defined below). Note that (5) implies that the inequality in (6) holds for such a construction. Some additional (standard) conditions on the model $\check{f}_k$ will be imposed further below, when needed. Note finally that in our notation $\mathscr{B}_k$ simply enumerates the affine functions comprising $\check{f}_k$, and thus $\mathscr{B}_k$ need not be a subset of $\{1, \ldots, k\}$ even though $\check{f}_k$ is, of course, built with information computed on those previous iterations. In particular, the aggregate linearization mentioned above may be indexed by some $j \notin \{1, \ldots, k\}$ (this gives some notational convenience; for example, we do not have to worry about assigning to an aggregate linearization an index already taken by the "usual" previous cutting plane).

Let $\hat{x}_k$ be the current stability center (the best past iterate), and let v_k^ℓ be a nonnegative scalar representing how much we aim to reduce the value $f(\hat{x}_k)$ at the current iteration. Define the corresponding level parameter by

$$\ell_k := f\left(\hat{x}_k\right) - v_k^\ell.$$

Then the level set associated with the model $\check{f}_k$ and the parameter ℓ_k is given by

$$\mathbb{X}_k := \{x \in \mathscr{X} : \check{f}_k(x) \leqslant \ell_k\}, \tag{7}$$

which is polyhedral if $\mathscr{X}$ is polyhedral.

We first observe that in the standard (via slack variable) reformulation of the doubly stabilized subproblem (4) given by

$$\min_{(x,r) \in \Re^{n+1}} \left\{ r + \frac{1}{2\tau_k} |x - \hat{x}_k|^2 : \check{f}_k(x) \leqslant r, \quad \check{f}_k(x) \leqslant \ell_k, \quad x \in \mathscr{X} \right\},$$

the first constraint must be active at the solution ($\check{f}_k(x) = r$), as otherwise the r term in the objective can be reduced maintaining feasibility (with the same x part of the

solution). This observation implies that the solution to the latter problem, and thus to (4), can be alternatively obtained by solving the simpler

$$\min_{(x,r)\in\Re^{n+1}} \left\{ r + \frac{1}{2\tau_k}|x-\hat{x}_k|^2 : \check{f}_k(x) \leqslant r, \quad r \leqslant \ell_k, \quad x \in \mathscr{X} \right\}. \tag{8}$$

We now state some properties of the minimizer x_{k+1} in (4), or equivalently of the x part of the solution in (8).

Proposition 1 *If $\mathbb{X}_k \neq \emptyset$ then problem (4) has the unique solution x_{k+1}.*

In addition, if $\mathscr{X}$ is polyhedral or $\mathtt{ri}\,\mathscr{X} \cap \{x \in \Re^n : \check{f}_k(x) \leqslant \ell_k\} \neq \emptyset$ then there exist $s_{k+1} \in \partial f_k(x_{k+1})$ and $h_{k+1} \in N_{\mathscr{X}}(x_{k+1}) = \partial i_{\mathscr{X}}(x_{k+1})$, and (scalar) Lagrange multipliers $\mu_k \geqslant 1$ and $\lambda_k \geqslant 0$ such that

$$\begin{aligned} x_{k+1} = \hat{x}_k - \tau_k\mu_k\hat{g}_k, \ \ \textit{with}\ \hat{g}_k = s_{k+1} + \frac{1}{\mu_k}h_{k+1}, \\ \mu_k = \lambda_k + 1 \ \textit{and}\ \ \lambda_k(\check{f}_k(x_{k+1}) - \ell_k) = 0. \end{aligned} \tag{9}$$

In addition, for all $x \in \mathscr{X}$ the aggregate linearization

$$\bar{f}_k^a(\cdot) := \check{f}_k(x_{k+1}) + \langle \hat{g}_k, \cdot - x_{k+1}\rangle \quad \textit{satisfies} \quad \bar{f}_k^a(x) \leqslant \check{f}_k(x) \leqslant f(x). \tag{10}$$

Proof The existence and uniqueness of solution x_{k+1} to (4) follow from the assumption that the problem is feasible and the fact that its objective function is strongly convex.

Next, under the stated assumptions, combining the results from [18] (specifically, Thm. 1.1.1 on p. 293, Prop. 5.3.1 and Remark 5.3.2 on p. 139, Prop. 2.2.2 on p. 308), the optimality conditions for (8) assert that there exist $\mu_k \geqslant 0$ and $\lambda_k \geqslant 0$ such that

$$\begin{aligned} &0 \in \frac{1}{\tau_k}\left(x_{k+1} - \hat{x}_k\right) + \mu_k\partial\check{f}_k(x_{k+1}) + N_{\mathscr{X}}(x_{k+1}), \\ &0 = 1 - \mu_k + \lambda_k, \\ &\mu_k\left(\check{f}_k(x_{k+1}) - r_{k+1}\right) = 0, \quad \lambda_k(r_{k+1} - \ell_k) = 0. \end{aligned}$$

In particular, $\mu_k = 1 + \lambda_k \geq 1$ and thus $r_{k+1} = \check{f}_k(x_{k+1})$, and there exist $s_{k+1} \in \partial f_k(x_{k+1})$ and $h_{k+1} \in N_{\mathscr{X}}(x_{k+1})$ such that

$$x_{k+1} = \hat{x}_k - \tau_k(\mu_k s_{k+1} + h_{k+1}) = \hat{x}_k - \tau_k\mu_k\left(s_{k+1} + \frac{1}{\mu_k}h_{k+1}\right),$$

which completes the proof of all the relations in (9).

To show (10), note that for all $x \in \mathscr{X}$ it holds that

$$\bar{f}_k^a(x) = \check{f}_k(x_{k+1}) + \langle s_{k+1}, x - x_{k+1}\rangle + \frac{1}{\mu_k}\langle h_{k+1}, x - x_{k+1}\rangle \leqslant \check{f}_k(x) \leqslant f(x),$$

where the first inequality follows from the facts that $s_{k+1} \in \partial\check{f}_k(x_{k+1})$ and $h_{k+1} \in N_{\mathscr{X}}(x_{k+1})$. □

The next result shows that the solution x_{k+1} of the doubly stabilized problem (4) solves at least one of the "singly" stabilized problems: the proximal (2) or the level (3).

Lemma 1 *For $\tau_k > 0$ and $\ell_k \in \Re$, let $x_k^\tau \in \Re^n$ and $x_k^\ell \in \Re^n$ be the (unique) solutions of problems (2) and (3), respectively. Let $x_{k+1} \in \Re^n$ be the unique solution of problem (4). Then it holds that*

$$x_{k+1} = \begin{cases} x_k^\tau & \text{if } \mu_k = 1 \\ x_k^\ell & \text{if } \mu_k > 1, \end{cases}$$

where μ_k is the Lagrange multiplier defined in Proposition 1.

Proof Let $\mu_k = 1$. Similarly to the proof of Proposition 1, writing the optimality conditions for (4) with $\mu_k = 1$ gives

$$0 \in \frac{1}{\tau_k}\left(x_{k+1} - \hat{x}_k\right) + \partial \check{f}_k(x_{k+1}) + N_{\mathscr{X}}(x_{k+1}),$$

which shows that x_{k+1} satisfies the optimality condition for (2). Since the solutions of the respective problems are unique, it holds that $x_{k+1} = x_k^\tau$.

If $\mu_k > 1$ then $\lambda_k > 0$, and hence $\check{f}_k(x_{k+1}) = \ell_k$ by (9). Clearly, the solution x_k^ℓ of (3) is also the unique solution of

$$\min\left\{\ell_k + \frac{1}{2\tau_k}|x - \hat{x}_k|^2 : \check{f}_k(x) \leqslant \ell_k, \quad x \in \mathscr{X}\right\}. \tag{11}$$

Observe that the optimal value of (11) is bounded below by the optimal value of the problem (4), due to the level constraint $\ell_k \geqslant \check{f}_k(x)$. As the solution x_{k+1} of (4) is feasible in (11) and achieves this lower bound (since $\ell_k = \check{f}_k(x_{k+1})$), it follows that x_{k+1} solves (11). Since problems (11) and (4) have unique solutions, it holds that $x_{k+1} = x_k^\ell$. □

According to Lemma 1, we shall call x_{k+1} a *proximal iterate* if $\mu_k = 1$, and otherwise ($\mu_k > 1$), we shall call it a *level iterate*. Similarly, an iteration k will be referred to as a proximal or a level iteration. It is thus clear that each iteration of the doubly stabilized algorithm makes either a step of the associated proximal bundle method, or of the level method. At every iteration, the algorithm makes this choice automatically.

We now define the *predicted decrease* by the model $\check{f}_k$ by

$$v_k^\tau := f(\hat{x}_k) - \check{f}_k(x_{k+1}) \geqslant 0, \tag{12}$$

where the inequality follows from x_{k+1} being the solution of (4) via

$$f(\hat{x}_k) \geqslant \check{f}_k(\hat{x}_k) \geqslant \check{f}_k(x_{k+1}) + \frac{1}{2\tau_k}\left|x_{k+1} - \hat{x}_k\right|^2.$$

As discussed in [11], to define the predicted decrease quantity there are alternatives other than (12). We have chosen (12) because of its direct connection with the level parameter ℓ_k, established in (15) below.

Once the iterate x_{k+1} is computed, the oracle provides the new function value $f(x_{k+1})$. As is usual in bundle methods, we shall change the stability center when the new iterate gives sufficient descent with respect to the predicted one. Namely, when

$$f(x_{k+1}) \leqslant f(\hat{x}_k) - m_f v_k^\tau, \tag{13}$$

where $m_f \in (0, 1)$. Accordingly, each iteration results either

- in a *descent step* when (13) holds, in which case $\hat{x}_k$ is moved to x_{k+1}; or
- in a *null step* when (13) does not hold, and the stability center is maintained.

We next provide useful connections between the predicted decrease $v_k^\ell = f(\hat{x}_k) - \ell_k$ related to the level parameter ℓ_k, the predicted decrease $v_k^\tau = f(\hat{x}_k) - \check{f}_k(x_{k+1})$ related to the solution of (4) and thus to the proximal parameter τ_k, and the aggregate linearization error given by

$$\hat{e}_k := f(\hat{x}_k) - \bar{f}_k^a(\hat{x}_k). \tag{14}$$

We also establish a key relation that would be the basis for the subsequent convergence analysis.

Proposition 2 *It holds that*

$$\hat{e}_k \geqslant 0, \quad \hat{e}_k + \tau_k \mu_k \left|\hat{g}_k\right|^2 = v_k^\tau \geqslant f(\hat{x}_k) - \ell_k = v_k^\ell, \tag{15}$$

where μ_k is the Lagrange multiplier defined in Proposition 1. *Moreover, if $\mu_k > 1$ then $v_k^\tau = v_k^\ell$.*

Furthermore, for all $x \in \mathscr{X}$ it holds that

$$f(\hat{x}_k) + \left\langle \hat{g}_k x - \hat{x}_k \right\rangle - \hat{e}_k \leqslant f(x). \tag{16}$$

(*In other words, $\hat{g}_k$ is $\hat{e}_k$-subgradient of the essential objective $(f + i_{\mathscr{X}})$ at $\hat{x}_k$*).

Proof The fact that $\hat{e}_k \geqslant 0$ follows directly from (10). To show (15), note that

$$\begin{aligned}
\hat{e}_k &= f(\hat{x}_k) - \bar{f}_k^a(\hat{x}_k) \\
&= f(\hat{x}_k) - \left(\check{f}_k(x_{k+1}) + \left\langle \hat{g}_k \hat{x}_k - x_{k+1} \right\rangle\right) \\
&= v_k^\tau - \left\langle \hat{g}_k \hat{x}_k - x_{k+1} \right\rangle \\
&= v_k^\tau - \tau_k \mu_k |\hat{g}_k|^2,
\end{aligned}$$

where the last equality follows from (9). In addition, since x_{k+1} is feasible in (4), we have that $\check{f}_k(x_{k+1}) \leqslant \ell_k = f(\hat{x}_k) - v_k^\ell$, which implies $v_k^\ell \leqslant v_k^\tau$. This completes the proof of (15). (Recall also that if $\mu_k > 1$ then $\lambda_k > 0$, in which case (9) implies $\check{f}_k(x_{k+1}) = \ell_k$, so that $v_k^\ell = v_k^\tau$).

The relation (16) follows from the fact that $\hat{g}_k$ is $\hat{e}_k$-subgradient of the essential objective at $\hat{x}_k$, which is verified as follows. Using again (10), for all $x \in \mathscr{X}$ it holds that

$$\begin{aligned} f(x) &\geqslant \bar{f}_k^a(x) \\ &= \check{f}_k(x_{k+1}) + \langle \hat{g}_k, x - x_{k+1} \rangle \\ &= f(\hat{x}_k) - \Big(f(\hat{x}_k) - \check{f}_k(x_{k+1}) \Big) + \langle \hat{g}_k, x - \hat{x}_k \rangle + \langle \hat{g}_k, \hat{x}_k - x_{k+1} \rangle \\ &= f(\hat{x}_k) - v_k^\tau + \langle \hat{g}_k, x - \hat{x}_k \rangle + \tau_k \mu_k |\hat{g}_k|^2, \end{aligned}$$

and (16) follows taking into account (15). □

The relation (16) motivates one of the alternative stopping tests for our algorithm, which is in the spirit of standard bundle methods: stop the algorithm when both $|\hat{g}_k|$ and $\hat{e}_k$ are small enough, i.e., an approximate optimality condition holds.

We now state the algorithm in full detail, and then comment on some of its ingredients.

Algorithm 1: Doubly stabilized bundle algorithm

Step 0 (initialization) *Choose parameters* $m_\ell, m_f \in (0,1)$, *and stopping tolerances* $\texttt{Tol}_\Delta, \texttt{Tol}_e, \texttt{Tol}_g > 0$. *Given* $x_1 \in \mathscr{X}$, *set* $\hat{x}_1 \leftarrow x_1$. *Compute* $f(x_1)$ *and* $g_1 \in \partial f(x_1)$. *If a lower bound* f_1^{low} *for* f^{inf} *is available, set* $v_1^\ell \leftarrow (1 - m_\ell)(f(\hat{x}_1) - f_1^{\text{low}})$; *otherwise, set* $f_1^{\text{low}} \leftarrow -\infty$ *and choose* $v_1^\ell > 0$. *Choose* $\tau_{\min} > 0$, $\tau_1 \geqslant \tau_{\min}$ *and set* $k = 1$.

Step 1 (first stopping test) *Set the optimality gap by* $\Delta_k \leftarrow f(\hat{x}_k) - f_k^{\text{low}}$.
If $\Delta_k \leqslant \texttt{Tol}_\Delta$, *stop. Return* $\hat{x}_k$ *and* $f(\hat{x}_k)$.

Step 2 (trial point finding) *Define the level parameter by* $\ell_k \leftarrow f(\hat{x}_k) - v_k^\ell$.

Step 2.1 (feasibility detection) *If the level set* $\mathbb{X}_k$ *defined by (7) is detected to be empty, set* $f_k^{\text{low}} \leftarrow \ell_k$, $v_k^\ell \leftarrow (1 - m_\ell)(f(\hat{x}_k) - f_k^{\text{low}})$ *and go back to Step 1.*

Step 2.2 (next iterate) *Solve (8) to obtain* (x_{k+1}, r_{k+1}) *and a Lagrange multiplier* λ_k *associated to the level constraint* $r \leqslant \ell_k$. *Set* $\mu_k \leftarrow \lambda_k + 1$, $v_k^\tau \leftarrow f(\hat{x}_k) - r_{k+1}$, $\hat{g}_k \leftarrow (\hat{x}_k - x_{k+1})/\tau_k\mu_k$ *and* $\hat{e}_k \leftarrow v_k^\tau - \tau_k \mu_k |\hat{g}_k|^2$.

Step 3 (second stopping test) *If* $\hat{e}_k \leqslant \texttt{Tol}_e$ *and* $|\hat{g}_k| \leqslant \texttt{Tol}_g$, *stop. Return* $\hat{x}_k$ *and* $f(\hat{x}_k)$.

Step 4 (oracle call) *Compute* $f(x_{k+1})$ *and* $g_{k+1} \in \partial f(x_{k+1})$.

Step 5 (descent test) *Choose* $f_{k+1}^{\text{low}} \in [f_k^{\text{low}}, f^{\text{inf}}]$.
If (13) holds, declare a descent step; otherwise a null step.

Step 5.1 (descent step) *Set* $\hat{x}_{k+1} \leftarrow x_{k+1}$, $\tau_{k+1} \leftarrow \tau_k \mu_k$ *and* $v_{k+1}^\ell \leftarrow \min\{v_k^\ell, (1 - m_\ell)(f(\hat{x}_{k+1}) - f_{k+1}^{\text{low}})\}$.
Choose a model $\check{f}_{k+1}$ *satisfying* $\check{f}_{k+1}(\cdot) \leqslant f(\cdot)$.

Step 5.2 (null step) *Set* $\hat{x}_{k+1} \leftarrow \hat{x}_k$ *and choose* $\tau_{k+1} \in [\tau_{\min}, \tau_k]$.
If $\mu_k > 1$ *(level iterate), set* $v_{k+1}^\ell \leftarrow m_\ell v_k^\ell$; *otherwise set* $v_{k+1}^\ell \leftarrow v_k^\ell$.
Choose a model $\check{f}_{k+1}$ *satisfying* $\max\{\bar{f}_{k+1}(\cdot), \bar{f}_k^a(\cdot)\} \leqslant \check{f}_{k+1}(\cdot) \leq f(\cdot)$.

Step 6 (loop) *Set* $k \leftarrow k + 1$ *and go back to Step 1.*

Some comments are in order.

(a) Observe that the lower bound f_k^{low} is updated either when the level set $\mathbb{X}_k$ is empty in Step 2.1, or in Step 5. In the second case, it is explicit that $f_k^{\text{low}} \leqslant f^{\text{inf}}$.

In the first case, $\mathbb{X}_k = \emptyset$ means that $\ell_k < \check{f}_k(x) \leqslant f(x)$ for all $x \in \mathscr{X}$. And since the update sets $f_k^{\text{low}} \leftarrow \ell_k$, it again holds that $f_k^{\text{low}} \leqslant f^{\text{inf}}$. Therefore, $f_k^{\text{low}} \leqslant f^{\text{inf}}$ for all k, and if the algorithm stops at Step 1, we have that

$$\texttt{Tol}_\Delta \geqslant f\left(\hat{x}_k\right) - f_k^{\text{low}} \geqslant f\left(\hat{x}_k\right) - f^{\text{inf}},$$

i.e., $\hat{x}_k$ is a $\texttt{Tol}_\Delta$-approximate solution to problem (1).

Note that when the level set $\mathbb{X}_k$ is empty, the update rules in the pass through Step 2.1 and back through Step 1, decrease the optimality gap Δ_k by the factor of $(1 - m_\ell)$. A simple update of the lower bound in Step 5 is $f_{k+1}^{\text{low}} \leftarrow f_k^{\text{low}}$.

(b) To identify if the level set is empty, the most natural is probably to proceed as usual with solving (4) and let the solver return with the infeasibility flag. Note that this is not a wasteful computation, as it leads to adjusting the level parameter as well as improving the lower bound f_k^{low}. Alternatively, to detect infeasibility we can solve the linear program (if $\mathscr{X}$ is a polyhedron)

$$\min\ s \text{ s.t. } \bar{f}_j(x) - s \leqslant \ell_k\ \forall j \in \mathscr{B}_k, \quad x \in \mathscr{X}, \quad s \geqslant 0.$$

If its optimal value is positive then $\mathbb{X}_k = \emptyset$.

(c) If one prefers to avoid infeasible level sets $\mathbb{X}_k$, then when $\mathscr{X}$ is bounded or f_k^{low} is finite, it is enough to update f_k^{low} in Step 5 as follows, solving the linear program:

$$\text{set } f_{k+1}^{\text{low}} \leftarrow \min\ r \quad \text{s.t. } \bar{f}_j(x) \leqslant r\ \forall j \in \mathscr{B}_k \quad f_k^{\text{low}} \leqslant r \quad x \in \mathscr{X} \quad r \in \Re. \tag{17}$$

This strategy is especially effective when solving LP is not too expensive relative to other tasks of the algorithm (in particular, the oracle computations).

(d) If $\mathscr{X}$ is unbounded, the level set $\mathbb{X}_k$ can be nonempty for all k, and f_k^{low} will never be updated (for example, for problem (1) with $f(x) = e^{-x}$ and $\mathscr{X} = [0, +\infty)$). In that case, the algorithm will not stop at Step 1, unless the initial lower bound f_1^{low} is within the $\texttt{Tol}_\Delta$-tolerance of f^{inf}.

(e) Step 5 increases the proximal parameter τ_k only after descent steps resulting from level iterations ($\mu_k > 1$). On the other hand, τ_k can be decreased only after null steps. A simple rule used in the numerical experiments of Sect. 5 is

$$\tau_{k+1} \leftarrow \max\left\{\tau_{\min},\ \tau_k v_k^\ell / v_k^\tau\right\},$$

which decreases the proximal parameter only after null steps resulting from proximal iterations ($v_k^\tau > v_k^\ell$ is only possible when $\mu_k = 1$, see Proposition 2). In this manner, the level parameter ℓ_k and the multiplier μ_k indicate how to update the proximal parameter τ_k. This is precisely the novel strategy to manage proximal parameter, proposed in this work.

(f) If at Step 2 (for all k) the rule $\ell_k = f(\hat{x}_k) - v_k^\ell$ is replaced by $\ell_k = +\infty$, Algorithm 1 becomes a proximal bundle algorithm (all iterations are proximal iterations).

(g) The QP formulation of subproblem (8) is given by

$$\min_{(x,r)\in\Re^{n+1}} \left\{ r + \frac{1}{2\tau_k} \left| x - \hat{x}_k \right|^2 : \bar{f}_j(x) \leqslant r \ \forall j \in \mathscr{B}_k \quad r \leqslant \ell_k, \ x \in \mathscr{X} \right\}.$$

It can be seen that (if $\mathscr{X} = \Re^n$) its dual has the number of variables equal to the number of cutting-planes in the bundle.
To keep the size of this QP (or of its dual) manageable, the number of elements in the bundle (the cardinality of the set $\mathscr{B}_k$) should be kept bounded, without impairing convergence. For this, the usual aggregation techniques of proximal bundle can be employed here. After a serious step, the only requirement is that the model should be below the objective function (which means that elements from the bundle can be deleted arbitrarily); this is reflected in Step 5.1 of Algorithm 1. During a sequence of consecutive nulls steps, the model $\check{f}_k$ can be composed of as few as only two cutting planes, corresponding to the new linearization $\bar{f}_{k+1}$ and the aggregate linearization $\bar{f}_k^a$ (or any number of cutting planes, as long as these two are included). This is reflected in the choice of the model specified in Step 5.2 of Algorithm 1. If the next model contains all the linearizations for which the constraint $\bar{f}_j(x) \leqslant r$ of the above QP is active at its solution (x_{k+1}, r_{k+1}), then there is no need to include the aggregate linearization $\bar{f}_k^a$.

3 Convergence analysis

Convergence analysis of the doubly stabilized bundle method has to account for all the possible combinations of level and proximal steps, whether null or descent, and the possibility of empty level sets. To that end, we consider the following three possible cases:

- The level sets $\mathbb{X}_k$ are empty infinitely many times;
- The above does not happen, and infinitely many descent steps are generated;
- In the same situation, finitely many descent steps are generated.

In what follows, we assume that $\texttt{Tol}_\Delta = \texttt{Tol}_e = \texttt{Tol}_g = 0$ and that Algorithm 1 does not stop. (If the algorithm stops for zero tolerance in Step 1, then the last descent step is, by comment (a) above, a solution to the problem. The same conclusion holds, by (16), if the method stops for zero tolerances in Step 3.) As a by-product of our convergence analysis, it would also follow that if the stopping rules parameters are positive then the method terminates in a finite number of iterations, with an appropriate approximate solution.

Lemma 2 *Suppose the level set $\mathbb{X}_k$ is empty infinitely many times.*

Then $\Delta_k \to 0$, $\{f(\hat{x}_k)\} \to f^{\inf}$, and every cluster point of the sequence $\{\hat{x}_k\}$ (if any exists) is a solution to problem (1); or the last $\hat{x}_k$ is a solution if this sequence is finite.

Proof It follows by Step 2 that for all k after the first $\mathbb{X}_k = \emptyset$ is encountered, we have $f_k^{\text{low}} > -\infty$ and thus $\Delta_k < +\infty$. Also, by Steps 2 and 5, $v_k^\ell \leqslant (1 - m_\ell)\Delta_k$. Thus,

$$f(\hat{x}_k) - \ell_k = f(\hat{x}_k) - \left(f(\hat{x}_k) - v_k^\ell\right) = v_k^\ell \leqslant (1 - m_\ell)\Delta_k,$$

which shows that if $\mathbb{X}_k = \emptyset$ at iteration k, then the update $f_k^{\text{low}} \leftarrow \ell_k$ decreases the optimality gap Δ_k by a factor of at least $(1-m_\ell)$. Hence, if this happens infinitely many times, we have that $\Delta_k \to 0$. Moreover, as no level set can be empty if $f^{\text{inf}} = -\infty$, in the case under consideration $f^{\text{inf}} > -\infty$. We can then write $\Delta_k = f(\hat{x}_k) - f_k^{\text{low}} \geqslant f(\hat{x}_k) - f^{\text{inf}}$, which implies the assertion as $\Delta_k \to 0$. □

From now on, we consider the case when $\mathbb{X}_k \neq \emptyset$ for all k large enough. Clearly, without loss of generality, we can simply assume that $\mathbb{X}_k \neq \emptyset$ for all k.

Analysis in the case of infinitely many descent steps essentially follows the theory for proximal bundle methods; in particular the argument in [6] can be readily applied.

Lemma 3 *Suppose* Algorithm 1 *generates infinitely many descent steps.*

Then $\{f(\hat{x}_k)\} \to f^{\text{inf}}$ *and every cluster point of the sequence* $\{\hat{x}_k\}$ *(if any exist) is a solution to problem* (1).

In addition, if the solution set of (1) *is nonempty and the sequence* $\{\tau_k \mu_k\}$ *is bounded above (for example, this is the case when there are finitely many level iterations) then the sequence* $\{\hat{x}_k\}$ *converges to a solution of* (1).

Proof Let $\{\hat{x}_{k(j)}\}$ be the subsequence of $\{\hat{x}_k\}$ such that $k(j)$ corresponds to the jth descent step. Define $i(j) = k(j+1) - 1$. Recalling (13), (15) and Proposition 2, we then have an iterative sequence satisfying, for all $j \geqslant 1$, the relations

$$\hat{x}_{k(j+1)} = \hat{x}_{k(j)} - \tau_{i(j)}\mu_{i(j)}\hat{g}_{i(j)}, \quad \hat{g}_{i(j)} \in \partial_{e_{i(j)}}(f + i_{\mathscr{X}})\left(\hat{x}_{k(j)}\right), \quad \tau_{i(j)}\mu_{i(j)} \geq \tau_{\min},$$
$$f\left(\hat{x}_{k(j)}\right) - f\left(\hat{x}_{k(j+1)}\right) \geqslant m_f\left(e_{i(j)} + \tau_{i(j)}\mu_{i(j)}\left|\hat{g}_{i(j)}\right|^2\right).$$

We are thus in the setting of the ϵ-subgradient method with an additional descent condition along the iterations. The announced convergence properties follow from [6].

For the last assertion, recall that τ_k can increase only on descent steps resulting from level iterations (in the case of $\mu_k > 1$). Thus, if the number of such iterations is finite, the sequence $\{\mu_k \tau_k\}$ is bounded above. Then, [6, Prop. 2.2] with t_k therein replaced by $\mu_k \tau_k$ can be invoked to obtain the stated assertion. □

Now we consider the last case, when $\hat{x}_k$ is eventually fixed and the last descent step is followed by an infinite number of null steps (note also that in this case the level sets $\mathbb{X}_k$ are nonempty).

Lemma 4 *Suppose there exists an index* $k_1 \geqslant 1$ *such that the descent test* (13) *is not satisfied for all* $k \geqslant k_1$.

Then there is an infinite number of level iterations, and the last descent iterate $\hat{x}_{k_1}$ *is a solution to problem* (1).

Proof Note that the sequence $\{v_k^\ell\}$ is nonincreasing. Let K be the set of indices k such that $\mu_k > 1$ (level iterations), and so according to Step 5.2 of Algorithm 1, $v_{k+1}^\ell = m_\ell v_k^\ell$. We then have that the values in $\{v_k^\ell\}$ only reduce on indices in K and do not change otherwise.

Suppose first that K is a finite set. Then, by Proposition 2, there exists an index $k_2 \geq k_1$ such that $\mu_k = 1$, $\lambda_k = 0$ and $v_k^\ell = v_{k_2}^\ell > 0$ for all $k \geq k_2$. Thus, by (15),

$$v_k^\tau \geq v_{k_2}^\ell > 0 \quad \text{for all} \quad k \geq k_2. \tag{18}$$

Moreover, by Lemma 1, all such iterations are proximal iterations. Hence, all iterations of Algorithm 1 indexed by $k \geq k_2$ can be considered as those of the classical proximal bundle method applied to the same problem. It then follows from [18] [Chap. XV, Thm. 3.2.4] that $v_k^\tau \to 0$, in contradiction with (18).

Hence, K must have infinitely many indices. But then the values of v_k^ℓ are reduced by the factor of m_ℓ infinitely many times, so that $\{v_k^\ell\} \to 0$ as $k \to \infty$. Since for $k \in K$ it holds that $v_k^\tau = v_k^\ell$ (c.f. Proposition 2), we conclude that $\{v_k^\tau\} \to 0$ as $K \ni k \to \infty$. As $\tau_k \geqslant \tau_{\min} > 0$ and $\mu_k \geqslant 1$, it follows from (15) that

$$\hat{e}_k \to 0 \text{ and } |\hat{g}_k| \to 0 \quad \text{as } K \ni k \to \infty. \tag{19}$$

As $\hat{g}_k$ is $\hat{e}_k$-subgradient of the essential objective $(f + i_{\mathscr{X}})$ at $\hat{x}_{k_1}$, (19) implies that $\hat{x}_{k_1}$ is a solution to (1). This completes the proof. □

Summarizing Lemmas 2–4, we state the following convergence properties of Algorithm 1.

Theorem 1 *If for the sequence generated by* Algorithm 1 *it holds that* $\hat{x}_k = \hat{x}_{k_1}$ *for all* $k \geq k_1$, *then* $\hat{x}_{k_1}$ *is a solution to* (1). *Otherwise,* $\{f(\hat{x}_k)\} \to f^{\inf}$ *as* $k \to \infty$, *and every cluster point of* $\{\hat{x}_k\}$ *(if any exist) is a solution to problem* (1). *In addition, if the solution set of* (1) *is nonempty, and an infinite number of descent steps is generated among which the number of level iterations is finite, then the sequence* $\{\hat{x}_k\}$ *converges to a solution of* (1).

An interesting question is whether the level bundle methods' lower worst-case complexity (when compared to the proximal versions) extends to the doubly stabilized algorithm. At this time, we conjecture this is probably not the case, as there does not seem to be a way to estimate the number of proximal iterations between level iterations.

We finish this section by considering a more general strategy of managing the level parameter, which we found useful in our numerical experiments. Note that Step 5.2 of Algorithm 1 reduces the predicted decrease v_k^ℓ by a factor of m_ℓ on null level iterations ($\mu_k > 1$), and keeps it unchanged on null proximal ones. Decreasing v_k^ℓ implies increasing the level parameter ℓ_k (Step 2 in Algorithm 1). The idea is that it may be sometimes useful to keep ℓ_k fixed for some null level iterations, because this can lead to infeasible level sets which, in turn, leads to updating the lower bound f_k^{low} thus decreasing the optimality gap Δ_k. The idea itself can be implemented in a number of different ways. For example, by decreasing v_k^ℓ after some fixed number of

consecutive null steps. Note, however, that the argument in Lemma 4 would not apply (because not all null level iterations reduce v_k^ℓ, which is an important ingredient in the proof). Thus the implementation should be such that convergence can still be justified by other tools.

3.1 Managing the level parameter

Consider an additional parameter $\mu_{\max} \geq 1$ as input for the algorithm, and replace the update rule for v_k^ℓ in Step 5.2 of Algorithm 1 by the following:

$$\text{If } \mu_k > \mu_{\max}, \text{ set } v_{k+1}^\ell \leftarrow m_\ell v_k^\ell; \quad \text{otherwise} \quad \text{set } v_{k+1}^\ell \leftarrow v_k^\ell. \tag{20}$$

Note that $\mu_{\max} = 1$ recovers the original formulation of Algorithm 1. The parameter v_k^ℓ remains fixed for null level iterations that result in a multiplier μ_k not large enough; when it is sufficiently large, v_k^ℓ is decreased and the level parameter ℓ_k is increased. The motivation for keeping v_k^ℓ fixed on some iterations is outlined above. The reason for updating v_k^ℓ when $\mu_k > \mu_{\max} > 1$ has to do with using [5, Thm. 3.7] to show convergence in the corresponding case. Additionally, an intuition as to why it is reasonable that the update of v_k^ℓ depends on μ_k can be derived from Lemma 7 below. The arguments in the proof of Lemma 7 (it is not important that it considers the more general case with inexact data) show that if v_k^ℓ is fixed over a sequence of null steps then μ_k is increasing (tends to $+\infty$ if the sequence is continued infinitely). Thus, if $\mu_{\max}$ is large enough, the rule (20) is likely to keep v_k^ℓ fixed, but only for some iterations so that the parameter is eventually updated.

As the modified rule (20) plays a role only on null steps, to verify convergence of this version of the algorithm we only have to consider the case when all the level sets are nonempty and there is a finite number of descent steps, i.e., all iterations from some point on are null steps. Apart from the condition $\mu_{\max} > 1$, we need the following stronger (but not at all restrictive from the practical viewpoint) condition on managing the bundle during null steps. Let $p(k)$ be the last proximal iteration performed up to iteration k. Choose $\check{f}_{k+1}$ to satisfy

$$\max\left\{\bar{f}_{k+1}(\cdot), \bar{f}_k^a(\cdot), \bar{f}_{p(k)+1}(\cdot), \bar{f}_{p(k)}^a(\cdot)\right\} \leqslant \check{f}_{k+1}(\cdot) \leqslant f(\cdot). \tag{21}$$

In particular, if k is a null proximal iteration, then $p(k) = k$ and the above rule is the same as for usual proximal bundle methods [6,13]. However, (21) differs from standard rules in the case of null level steps: during null level iterations information about the last proximal iteration is kept in the bundle.

If there are infinitely many null proximal iterations, the algorithm can be interpreted as a proximal bundle method in the case of a finite number of descent steps followed by null steps, with level iterates seen as merely enriching the cutting-plane model. In particular, the key conditions (4.7)–(4.9) in [6] are satisfied. Convergence then follows from [18, Chap. XV, Thm. 3.2.4]; see also [6,11].

On the other hand, if there are only finitely many proximal iterations, the algorithm becomes essentially a level bundle method in the case of a finite number of descent steps followed by null steps. In this case, [5, Thm. 3.7] provides the assertion on convergence [we note that for this it is important that $\mu_{\max} > 1$, because λ_k in [5] is required to be bounded by some $\lambda_{\max} > 0$, and we have $\mu_k = \lambda_k + 1$ in (9)].

4 Handling inexact data

In various real-world applications, the objective function and/or its subgradient can be too costly (sometimes, impossible) to compute. This is particularly true when f is given by some optimization problem, e.g., $f(x) = \max_{u \in U} \varphi(u, x)$, as in numerical experiments in Sect. 5.2 for example. In such situations, approximate values must be used.

Various inexact bundle methods that use approximate function and subgradient evaluations have been studied in [10,11,17,21,26]. The natural setting is to assume that, given any $x \in \Re^n$, the oracle provides some approximate values $f_x \in \Re$ and $g_x \in \Re^n$ of the objective function and its subgradient, respectively, such that

$$\begin{cases} f_x = f(x) - \eta_x \quad \text{and} \\ f(\cdot) \geqslant f_x + \langle g_x, \cdot - x \rangle - \eta_x^g, \end{cases} \tag{22}$$

where $\eta_x \in \Re$ and $\eta_x^g \geqslant 0$ are some unknown but uniformly bounded errors. Specifically, there exist $\eta \geqslant 0$ and $\eta^g \geqslant 0$ such that

$$|\eta_x| \leqslant \eta \quad \text{and} \quad \eta_x^g \leqslant \eta^g \quad \text{for all } x \in \mathscr{X}. \tag{23}$$

Remark 1 Assumptions (22) and (23) are also present in [21] and [27]. They are weaker than the assumptions employed by the level bundle methods given in [9], which require $\eta^g = 0$, and further the bound η to be known, controllable, and asymptotically vanishing in a suitable sense. Thus, the ingredients of our analysis concerning level iterations are certainly applicable to the setting of [9], and lead to new results under the weaker oracle assumptions. On the other hand, using stronger assumptions [9] is able to compute exact solutions, rather than inexact as in our case.

In [27], nonlinearly constrained problems are considered, which require the use of non-static merit functions (specifically, of improvement functions as in [25]). Thus, even considering level iterations only, [27] is very different from our case. Also, [27] requires boundedness of the feasible set $\mathscr{X}$ for convergence analysis, and in fact for convergence itself (there are examples which show that the method therein can fail for unbounded $\mathscr{X}$).

With given oracle information, the inexact linearization of f at iteration k is defined accordingly by

$$\bar{f}_k(x) := f_{x_k} + \langle g_{x_k} x - x_k \rangle \quad (\leqslant f(x) + \eta^g),$$

and the inexact model $\check{f}_k$ is then defined as in (6). However, because of the inexactness, we now have the weaker property $\check{f}_k(\cdot) \leqslant f(\cdot) + \eta^g$. The predicted decrease must now employ only the available (inexact) information; the counterpart of (12) is thus given by

$$v_k^\tau := f_{\hat{x}_k} - \check{f}_k(x_{k+1}),$$

and the level parameter is

$$\ell_k := f_{\hat{x}_k} - v_k^\ell \quad \text{for a given } v_k^\ell > 0.$$

Solving the doubly stabilized bundle subproblem (4) for the inexact model $\check{f}_k$, the direction of change $\hat{g}_k$ and the aggregate linearization $\bar{f}_k^a$ are defined exactly as before, i.e., by (9) and (10), respectively. The aggregate linearization error is now given by

$$\hat{e}_k := f_{\hat{x}_k} - \bar{f}_k^a(\hat{x}_k).$$

The first observation is that, unlike in the exact case [recall (15)], the aggregate linearization error $\hat{e}_k$ can be negative due to inaccuracy in the data. However, given the oracle assumptions (22), the following lower bound holds:

$$\hat{e}_k \geqslant f(\hat{x}_k) - \eta - \bar{f}_k^a(\hat{x}_k) \geqslant f(\hat{x}_k) - \eta - \left(f(\hat{x}_k) + \eta^g\right) = -(\eta + \eta^g). \tag{24}$$

Most inexact proximal bundle methods work the following way. In the proximal setting the predicted decrease has the form $v_k^\tau = \hat{e}_k + \tau_k|\hat{g}_k|^2$ (recall Proposition 2, where the proximal method corresponds to $\mu_k = 1$). Then $v_k^\tau < 0$ means that $\hat{e}_k$ is too negative (the oracle error is excessive). In such a case, the descent test

$$f_{x_{k+1}} \leqslant f_{\hat{x}_k} - m_f v_k^\tau, \tag{25}$$

mimicking (13), is not meaningful. The methods in [10,11,21] deal with this situation using the following simple idea. To make v_k^τ positive (when $\hat{g}_k \neq 0$), the strategy is then to increase the proximal parameter τ_k and solve again the QP with the same model $\check{f}_k$ to get another candidate x_{k+1}. This procedure, called *noise attenuation* [21], ensures that:

(i) the predicted decrease v_k^τ is always nonnegative before testing for descent;
(ii) if the noise is persistently excessive (an infinite number of noise attenuation steps is required) then the associated parameter is driven to infinity, which ensures in turn that $\hat{g}_k$ tends to zero.

With exact oracles, the predicted decrease v_k^τ can be seen as an optimality measure: if the proximal parameter $\tau_k > 0$ is bounded away from zero, (15) ensures that

$$v_k^\tau = 0 \iff \hat{e}_k = 0 \quad \text{and} \quad \hat{g}_k = 0.$$

The above is no longer true for inexact oracles. For the proximal version (corresponding to $\mu_k = 1$ above), one has the following (much weaker) relation:

$$v_k^\tau \leq 0 \implies \tau_k |\hat{g}_k|^2 \leq -\hat{e}_k \quad \left(\leq \eta + \eta^g\right).$$

It then holds that

$$|\hat{g}_k|^2 \leq \frac{(\eta + \eta^g)}{\tau_k}.$$

And this is where the property (ii) above comes into play. To ensure that $\hat{g}_k$ goes to zero in the case of excessive oracle errors, [21] drives τ_k to infinity. In principle, a similar strategy can be implemented in Algorithm 1. However, this clearly has some disadvantages. To start with, the QP has to be solved again with the same model $\check{f}_k$ and (sharply) increased prox-parameter, to obtain another candidate x_{k+1}. And this may need to be done more than once consecutively. Also, it may eventually turn out that this increase of the prox-parameter is harmful, or at least unnecessary in some sense (note that there are only heuristic rules for this update). It turns out that the doubly stabilized method does not require such procedures to ensure that $\hat{g}_k$ always tends to zero. Instead of "manually" increasing τ_k, the algorithm controls the steps automatically and properly via the multipliers μ_k (as is revealed by the analysis in Lemma 7 below). This is an interesting, and clearly desirable property. Another interesting feature of the doubly stabilized method is that the predicted decrease v_k^τ is *always* positive, i.e., property (i) above holds true. To that end, first note that if v_k^ℓ becomes nonpositive at some iteration k due to the updates in Steps 2 and 5 of Algorithm 1, then so does the inexact optimality gap Δ_k in Step 1 and the algorithm stops immediately (and it can be seen that an appropriate approximate solution is obtained). We can thus consider that $v_k^\ell > 0$ for all k. Then the same argument as that in Proposition 2 shows that

$$v_k^\tau = \hat{e}_k + \tau_k \mu_k |\hat{g}_k|^2 \geqslant f_{\hat{x}_k} - \ell_k = v_k^\ell > 0 \quad \forall\, k. \tag{26}$$

Therefore, a descent test like (25) is always meaningful, unlike for the proximal bundle approach with inexact data. In conclusion, our doubly stabilized method does not require the noise attenuation modification to handle inexact data: the property (i) is automatic, while the assertion of (ii) is obtained as a consequence of the algorithm's behavior (the iterates it generates) rather than driving some parameter to extreme values by "brute-force".

In what follows, we consider Algorithm 1 with the change of notation in that $\check{f}_k$ refers to the inexact model with the data satisfying (22) and (23). Accordingly, $f(\hat{x}_k)$ in Algorithm 1 is replaced by $f_{\hat{x}_k}$, etc. The quantities v_k^τ, ℓ_k and $\hat{e}_k$ are as defined in this section above. Finally, for the current inexact setting the bundle management rule given in (21) becomes

$$\max\left\{\bar{f}_{k+1}(\cdot),\, \bar{f}_k^a(\cdot),\, \bar{f}_{p(k)+1}(\cdot),\, \bar{f}_{p(k)}^a(\cdot)\right\} \leqslant \check{f}_{k+1}(\cdot) \leqslant f(\cdot) + \eta^g, \tag{27}$$

where $p(k)$ once again stands for the last proximal iteration performed up to iteration k.

As standard in inexact proximal bundle methods, the linearization error $\hat{e}_k$ is declared not too negative when the inequality

$$\hat{e}_k \geqslant -m_e \tau_k \mu_k |\hat{g}_k|^2 \tag{28}$$

holds for some parameter $m_e \in (0, 1)$. This inequality, together with a parameter $\mu_{\max} \geq 1$, is employed to update v_k^ℓ in Step 5.2 of Algorithm 1 as follows:

$$\text{If } \mu_k > \mu_{\max} \text{ and (28) holds, set } v_{k+1}^\ell \leftarrow m_\ell v_k^\ell; \quad \text{otherwise set } v_{k+1}^\ell \leftarrow v_k^\ell. \tag{29}$$

Since our method does not use noise attenuation, we cannot invoke the results from [21] and [11] for the case of infinitely many proximal iterations. For the case of finitely many proximal iterations, we cannot invoke previous results on inexact level bundle methods either; see comments in Remark 1. Therefore, convergence analysis largerly independent of previous literature is in order (although, naturally, a few ingredients would be familiar). First note that if the oracle errors do not vanish in the limit, of course only approximate solutions to (1) can be expected in general. This is natural, and similar to [10,11,21,26].

4.1 Convergence analysis for the inexact case

We can proceed as in Proposition 1 to show that $\bar{f}_k^a(x) \leqslant \check{f}_k(x)$ for all $x \in \mathscr{X}$. Since by the inexact oracle definition (22) we have that $\bar{f}_j(\cdot) \leqslant f(\cdot) + \eta^g$ for all $j \in \mathscr{B}_k$, we conclude that for all $x \in \mathscr{X}$ it holds that

$$\begin{aligned} f(x) + \eta^g &\geqslant \check{f}_k(x) \geqslant \bar{f}_k^a(x) = \check{f}_k(x_{k+1}) + \langle \hat{g}_k x - x_{k+1} \rangle && (30)\\ &= f_{\hat{x}_k} - \Big(f_{\hat{x}_k} - \check{f}_k(x_{k+1}) \Big) + \langle \hat{g}_k x - \hat{x}_k \rangle + \langle \hat{g}_k \hat{x}_k - x_{k+1} \rangle \\ &= f_{\hat{x}_k} - v_k^\tau + \langle \hat{g}_k x - \hat{x}_k \rangle + \tau_k \mu_k |\hat{g}_k|^2 \\ &= f_{\hat{x}_k} - \hat{e}_k + \langle \hat{g}_k x - \hat{x}_k \rangle && (31)\\ &\geqslant f_{\hat{x}_k} - v_k^\tau + \langle \hat{g}_k x - \hat{x}_k \rangle . && (32) \end{aligned}$$

Note also that as in the exact case, if $\check{f}_k(x_{k+1}) = \ell_k$ (which holds if $\mu_k > 1$), then in (26) we have that $v_k^\ell = v_k^\tau$.

As in Sect. 3, we consider separately the same three possible cases.

Lemma 5 *Suppose the level set $\mathbb{X}_k$ is empty infinitely many times.*

Then $\Delta_k \to 0$,

$$\lim_{k \to \infty} f_{\hat{x}_k} \leqslant f^{\inf} + \eta^g, \tag{33}$$

and every cluster point of the sequence $\{\hat{x}_k\}$ (if any exist) is a $(\eta + \eta^g)$-approximate solution to problem (1), or the last $\hat{x}_k$ is a $(\eta+\eta^g)$-approximate solution if this sequence is finite.

Proof Recall that in the case under consideration $f^{\inf} > -\infty$. The same argument as that of Lemma 2 shows that $\Delta_k \to 0$. Also, on the iterations in question we have that $\ell_k < \check{f}_k(x)$ for all $x \in \mathscr{X}$, and thus the update in Step 2 and (30) ensure that $f_k^{\text{low}} \leqslant f^{\inf} + \eta^g$. As $\{f_{\hat{x}_k}\}$ is decreasing and bounded below (since $f^{\inf} > -\infty$), we conclude that

$$\lim_{k\to\infty} f_{\hat{x}_k} - f^{\inf} - \eta^g \leqslant \lim_{k\to\infty} \left(f_{\hat{x}_k} - f_k^{\text{low}}\right) = \lim_{k\to\infty} \Delta_k = 0,$$

which gives (33).

Now let $\tilde{x}$ be any cluster point of $\{\hat{x}_k\}$, and let $\{\hat{x}_{k_j}\}$ be a subsequence converging to $\tilde{x}$ as $j \to \infty$. Then

$$f^{\inf} + \eta^g \geqslant \lim_{j\to\infty} f_{\hat{x}_{k_j}} = \lim_{j\to\infty} \left(f\left(\hat{x}_{k_j}\right) - \eta_{\hat{x}_{k_j}}\right) \geqslant f(\tilde{x}) - \eta, \tag{34}$$

which establishes the last assertion. □

Consider now the case where $\mathbb{X}_k \neq \emptyset$ for all k large enough, and there is an infinite number of descent steps [for which (25) holds].

Lemma 6 *Suppose* Algorithm 1 *generates infinitely many descent steps.*

Then (33) *holds and every cluster point of the sequence* $\{\hat{x}_k\}$ *(if any exist) is a* $(\eta + \eta^g)$*-approximate solution to problem* (1).

Proof Let $\{\hat{x}_{k(j)}\}$ be the subsequence of $\{\hat{x}_k\}$ such that $k(j)$ corresponds to the jth descent step, and define $i(j) = k(j+1)-1$. It follows from (25) that $\{f_{\hat{x}_{k(j)}}\}$ is decreasing and either $\{f_{\hat{x}_{k(j)}}\} \to -\infty$, in which case (22), (23) imply that $\{f(\hat{x}_{k(j)})\} \to -\infty$ and the conclusions are obvious, or the limit of $\{f_{\hat{x}_{k(j)}}\}$ is finite. In the second case (25) implies that

$$\lim_{j\to\infty} v_{i(j)}^{\tau} = 0.$$

Let $x \in \mathscr{X}$ be arbitrary. Using (32) and the fact that $\hat{x}_{k(j)} = \hat{x}_{i(j)}$, we then obtain that

$$\begin{aligned}
\left|\hat{x}_{k(j+1)} - x\right|^2 &= \left|\hat{x}_{k(j)} - x\right|^2 + \left(\tau_{i(j)}\mu_{i(j)}\right)^2 \left|\hat{g}_{i(j)}\right|^2 \\
&\quad + 2\tau_{i(j)}\mu_{i(j)} \left\langle \hat{g}_{i(j)} x - \hat{x}_{k(j)} \right\rangle \\
&\leqslant \left|\hat{x}_{k(j)} - x\right|^2 + \left(\tau_{i(j)}\mu_{i(j)}\right)^2 \left|\hat{g}_{i(j)}\right|^2 \\
&\quad + 2\tau_{i(j)}\mu_{i(j)} \left(f(x) + \eta^g - f_{\hat{x}_{k(j)}} + v_{i(j)}^{\tau}\right).
\end{aligned}$$

Suppose that (33) does not hold. Then there exist $t > 0$ and $\tilde{x} \in \mathscr{X}$ such that $f_{\hat{x}_{k(j)}} \geqslant f(\tilde{x}) + \eta^g + t$ for all j. Taking j large enough so that $v_{i(j)}^{\tau} \leqslant t/2$, and choosing $x = \tilde{x}$ in the chain of inequalities above, we obtain that

$$\begin{aligned}\left|\hat{x}_{k(j+1)}-\tilde{x}\right|^2 &\leqslant \left|\hat{x}_{k(j)}-\tilde{x}\right|^2-\tau_{i(j)}\mu_{i(j)}t\\ &\leqslant \left|\hat{x}_{k(1)}-\tilde{x}\right|^2-t\sum_{q=1}^{j}\tau_{i(q)}\mu_{i(q)}\\ &\leqslant \left|\hat{x}_{k(1)}-\tilde{x}\right|^2-jt\tau_{\min},\end{aligned}$$

where we used the fact that $\tau_k\mu_k \geq \tau_k \geq \tau_{\min}$. The above gives a contradiction when $j \to \infty$. We conclude that (33) holds. The last assertion then follows by the same argument as in Lemma 5. □

We now consider the case of finitely many descent steps, with the level set $\mathbb{X}_k$ nonempty (for all k large enough).

Lemma 7 *Suppose that for* Algorithm 1*, with the additional bundle management rule* (27) *and Step 5 employing* (29) *with* $\mu_{\max} \geqslant 1$*, there exists an index* $k_1 \geqslant 1$ *such that the descent test* (25) *is not satisfied for all* $k \geqslant k_1$*.*

Then the last descent iterate $\hat{x}_{k_1}$ *is a* $(\eta+\eta^g)$*-approximate solution to problem* (1)*.*

Proof The sequence $\{v_k^\ell\}$ is monotone and when its elements decrease, they decrease by a fixed fraction $m_e \in (0,1)$. Thus either $v_k^\ell \to 0$ or $v_k^\ell = v^\ell > 0$ for all k large enough.

Consider first the case of $v_k^\ell \to 0$. Then by rule (29) there exists an infinite index set K such that $\mu_k > \mu_{\max}$ and the inequality (28) is valid for $k \in K$. For such indices, it then holds that

$$0 \leqslant (1-m_e)\tau_{\min}|\hat{g}_k|^2 \leqslant (1-m_e)\tau_k\mu_k|\hat{g}_k|^2 \leqslant \hat{e}_k+\tau_k\mu_k|\hat{g}_k|^2 = v_k^\tau = v_k^\ell, \quad (35)$$

where the last equality follows from Proposition 2, because $\mu_k > \mu_{\max} \geqslant 1$ for all $k \in K$. It follows from (35) that

$$\tau_k\mu_k|\hat{g}_k|^2 \to 0, \quad \hat{g}_k \to 0, \quad \hat{e}_k \to 0 \quad \text{as } K \ni k \to \infty.$$

Now passing onto the limit in (31) as $K \ni k \to \infty$, with $x \in \mathscr{X}$ fixed but arbitrary and $\hat{x}_k = \hat{x}_{k_1}$ fixed, implies the assertion.

We now consider the second case: $v_k^\ell = v^\ell > 0$ for all $k \geqslant k_2$.

Suppose first that there exists an infinite subsequence of null proximal steps ($\mu_k = 1$), indexed by $\{k(j)\}$. "Ignoring" the possible null level steps in between, we can consider the sequence $\{x_{k(j)}\}$ as that generated by the proximal bundle method, where the model satisfies, by the rule (27), the key conditions

$$\max\left\{\bar{f}_{k(j)}(\cdot),\, \bar{f}^a_{k(j)-1}(\cdot)\right\} \leqslant \check{f}_i(\cdot) \leqslant f(\cdot)+\eta^g, \quad \text{for } k(j) \leqslant i \leqslant k(j+1)-1.$$

Of specific importance here is the relation for $i = k(j+1)-1$, which shows that on consecutive null proximal steps the model satisfies the conditions which, together

with $\{\tau_{k(j)}\}$ being nonincreasing, guarantee the following property of the (inexact) proximal bundle method:

$$0 \geqslant \limsup_{j\to\infty} \left(f_{x_{k(j)}} - \check{f}_{k(j)-1}(x_{k(j)}) \right). \tag{36}$$

(See [11, Theorem 6.4] and/or [21, Lemma 3.3 and Section 4.1].) On the other hand, as the descent condition (25) does not hold,

$$\begin{aligned} f_{x_{k(j)}} - \check{f}_{k(j)-1}(x_{k(j)}) &> f_{\hat{x}_{k_1}} - m_f v^{\tau}_{k(j)-1} - \check{f}_{k(j)-1}(x_{k(j)}) \\ &= (1-m_f) v^{\tau}_{k(j)-1} \\ &\geqslant (1-m_f) v^{\ell} > 0, \end{aligned}$$

which gives a contradiction with (36).

Therefore, in the case under consideration, there can only be a finite number of null proximal steps. Hence, all iterations indexed by $k \geqslant k_3$ are of the null level type, and it holds that $\mu_k > 1$, $\lambda_k > 0$, $\hat{x}_k = \hat{x}$, $v^{\ell}_k = v^{\ell} > 0$ and $\ell_k = \ell$ for all $k \geqslant k_3$.

Note that

$$\ell \geqslant \check{f}_k(x_{k+1}) \geqslant \bar{f}^a_{k-1}(x_{k+1}) = \check{f}_{k-1}(x_k) + \left\langle \hat{g}_{k-1} x_{k+1} - x_k \right\rangle.$$

By Proposition 1, as $\lambda_{k-1} > 0$ it holds that $\check{f}_{k-1}(x_k) = \ell$. Hence, $0 \geqslant \langle \hat{g}_{k-1}, x_{k+1} - x_k \rangle$, and since $\hat{x} - x_k = \tau_{k-1}\mu_{k-1}\hat{g}_{k-1}$, it holds that

$$0 \geqslant \left\langle \hat{x} - x_k, x_{k+1} - x_k \right\rangle.$$

It then follows that

$$\left| x_{k+1} - \hat{x} \right|^2 \geqslant \left| x_k - \hat{x} \right|^2 + \left| x_{k+1} - x_k \right|^2. \tag{37}$$

Note that

$$\ell \geqslant \check{f}_k(x_{k+1}) \geqslant \bar{f}_k(x_{k+1}) = f_{x_k} + \left\langle g_{x_k} x_{k+1} - x_k \right\rangle.$$

Using the Cauchy–Schwarz inequality, we obtain that

$$|g_{x_k}||x_{k+1} - x_k| \geqslant f_{x_k} - \ell. \tag{38}$$

Since this is a null step, it holds that

$$f_{x_k} > f_{\hat{x}} - m_f v^{\tau}_{k-1},$$

and since it is a level step, $v^{\tau}_{k-1} = v^{\ell}_{k-1} = v^{\ell} > 0$. Using further the definition $\ell = f_{\hat{x}} - v^{\ell}$, we conclude that

$$f_{x_k} - \ell \geqslant (1-m_f) v^{\ell} > 0.$$

In view of (38) and the last inequality above, it holds that $g_{x_k} \neq 0$ and we obtain that

$$|x_{k+1} - x_k| \geqslant \frac{f_{x_k} - \ell}{|g_{x_k}|} \geqslant \frac{(1 - m_f)v^\ell}{|g_{x_k}|}.$$

Using now (37), it follows that

$$|x_{k+1} - \hat{x}|^2 \geqslant |x_k - \hat{x}|^2 + \left(\frac{(1 - m_f)v^\ell}{|g_{x_k}|}\right)^2. \tag{39}$$

If the sequence $\{x_k\}$ were to be bounded, there would exist a constant $C > 0$ such that $|g_{x_k}| \leqslant C$ for all k [by boundedness of the ε-subdifferential on bounded sets and (22), (23)]. But then (39) would mean that the monotone sequence $\{|x_k - \hat{x}|^2\}$ is increasing at every iteration by a fixed positive quantity $((1 - m_f)v^\ell/C)^2$, and thus $\{x_k\}$ cannot be bounded. Hence, $\{x_k\}$ is unbounded. Since $\{|x_k - \hat{x}|^2\}$ is monotone by (39), it follows that $|x_k - \hat{x}| \to +\infty$ as $k \to \infty$.

We next show that $\limsup_{k\to\infty} \mu_k = +\infty$. Suppose the contrary, i.e., that there exists $\bar{\mu} > 0$ such that $\mu_k \leqslant \bar{\mu}$ for all k. As $\{\tau_k\}$ is nonincreasing for $k \geqslant k_3$ and $v_k^\tau = v_k^\ell = v^\ell$, using (24) we have that

$$\begin{aligned}\tau_{k_3}\bar{\mu}v^\ell &\geqslant \tau_k\mu_k v_k^\tau = \tau_k\mu_k\hat{e}_k + (\tau_k\mu_k)^2|\hat{g}_k|^2\\ &\geqslant -\tau_{k_3}\bar{\mu}(\eta + \eta^g) + \left|x_{k+1} - \hat{x}\right|^2,\end{aligned}$$

in contradicton with $|x_k - \hat{x}| \to +\infty$. Hence, $\limsup_{k\to\infty} \mu_k = +\infty$.

In the case under consideration, by rule (29) of Algorithm 1, $\hat{e}_k < -m_e\tau_k\mu_k|\hat{g}_k|^2$ for all $k \geqslant k_3$. In particular, $\limsup_{k\to\infty} \hat{e}_k \leqslant 0$. Also, using again (24), from $\hat{e}_k < -m_e\tau_k\mu_k|\hat{g}_k|^2$ it follows that

$$\frac{(\eta + \eta^g)}{\tau_{\min}\mu_k} > m_e|\hat{g}_k|.$$

As $\limsup_{k\to\infty} \mu_k = +\infty$, this implies that $\liminf_{k\to\infty} |\hat{g}_k| = 0$. Now fixing an arbitrary $x \in \mathscr{X}$, and passing onto the limit in (31) along a subsequence for which the last relation above holds (taking also into account that in the case under consideration $\hat{e}_k \leqslant 0$), concludes the proof. □

Combining all the cases considered above, we conclude the following.

Theorem 2 *If* Algorithm 1 [*with the additional rules* (29) *and* (27)] *generates a sequence such that* $\hat{x}_k = \hat{x}_{k_1}$ *for all* $k \geq k_1$, *then* $\hat{x}_{k_1}$ *is a* $(\eta + \eta^g)$*-approximate solution to* (1). *Otherwise,* (33) *holds and every cluster point of the sequence* $\{\hat{x}_k\}$ *(if any exist) is a* $(\eta + \eta^g)$*-approximate solution to problem* (1).

The analysis above also shows that in all the cases either $\Delta_k \to 0$ or there exists a subsequence $K \subset \{1, 2, \ldots\}$ such that $\limsup_{K\ni k\to\infty} \hat{e}_k \leqslant 0$ and $\lim_{K\ni k\to\infty} |\hat{g}_k| = 0$. This means that, for positive tolerances, some stopping rule in Algorithm 1 is eventually satisfied (at which time an appropriate approximate solution is obtained).

5 Numerical results

In this section we report computational experiments on different types of problems: two-stage stochastic linear programming, nonsmoothly-regularized maxima of quadratic functions, the model unit-commitment problem in the energy sector, and some standard nonsmooth test problems (about 1000 instances overall). We compare the following four solvers:

- PBM-1—proximal bundle method using a rule to update τ_k based on [19];
- PBM-2—proximal bundle method using a rule to update τ_k based on [23];
- LBM—level bundle method of [5];
- DSBM—doubly stabilized bundle method (the algorithm described in this article).

The runs were performed on a computer with Intel(R) Core(TM), i3-3110M CPU @ 2.40, 4G (RAM), under Windows 8, 64 Bits. The QPs (and also LPs) were solved by the MOSEK 7 toolbox for MATLAB (http://www.mosek.com/). The MATLAB version is R2012a.

Our analysis of the outcomes reports success or failure (i.e., whether a stopping test was eventually satisfied or the maximal number of iterations was reached), the number of oracle calls (here, the same as number of iterations), and CPU time to termination. We also compare the quality of solutions obtained at termination. To get some further insight, we report the numbers of descent steps for all the solvers, the number of empty level sets encountered for LBM and DSBM, and for DSBM which has various possibilities—the number of level iterations and which stopping criterion triggered termination.

We start with describing some details of implementations, tuning, and stopping rules of the algorithms in question.

5.1 Implementations, tuning the parameters, and stopping criteria

Many parameters need to be set for the solvers: the constant for the descent test $m_f \in (0, 1)$ in (13) (used in all four solvers), the constant $m_\ell \in (0, 1)$ for adjusting the level parameter (for LBM and DSBM), and some further parameters for updating τ_k in the proximal solvers PBM-1, PBM-2 and DSBM.

Some specific parameters of each solver are listed below.

5.1.1 The level bundle algorithm LBM

The algorithm is as described in [5]. The initial predicted decrease is given by $v_1^\ell = f(x_1) - \check{f}_1(\tilde{x})$, where $\tilde{x}$ is the solution of the QP (2) with $k = 1$ and τ_1 given. When a lower bound f_k^{low} for the optimal value f^{inf} is found, the subsequent iterations solve the LP (17) to update f_k^{low} to f_{k+1}^{low}.

As in the rule (20), the LBM method of [5] employs the parameter $\mu_{\max} > 0$. For this solver, we need to set mainly the parameters m_ℓ, $\mu_{\max}$ and τ_1 (the latter defines v_1^ℓ as explained above).

5.1.2 The proximal bundle solvers PBM-1 and PBM-2

The rule to update the prox-parameter τ_k is as follows: let $a > 1$ and $\tau_{\min} > 0$ be two given parameters, and τ_{aux}^k be an auxiliary prox-parameter at iteration k (different for PBM-1 and PBM-2).

- If null step, set $\tau_{k+1} \leftarrow \min\{\tau_k, \max\{\tau_{\text{aux}}^k, \tau_k/a, \tau_{\min}\}\}$
- If descent step:
  - if more than five consecutive descent steps, set $\tau_{\text{aux}}^k \leftarrow a\tau_{\text{aux}}^k$
  - set $\tau_{k+1} \leftarrow \min\{\tau_{\text{aux}}^k, 10\tau_k\}$.

In PBM-1 [19], one sets

$$\tau_{\text{aux}}^k \leftarrow 2\tau_k \left(1 + \frac{f(\hat{x}_k) - f(x_{k+1})}{v_k^\tau}\right).$$

In PBM-2 [23], one sets

$$\tau_{\text{aux}}^k \leftarrow \tau_k \left(1 + \frac{\langle g_{k+1} - g_k, x_{k+1} - x_k\rangle}{|g_{k+1} - g_k|^2}\right),$$

under some safeguards [23, Section 4.2].

The essential parameters to tune in the updates above are a, τ_1 and $\tau_{\min}$. Parameters taken as 10 and 2 in the setting of τ_k could also be tuned, but we use here their standard values.

5.1.3 The doubly stabilized DSBM solver

This is Algorithm 1 employing rule (20) in Step 5. The initial predicted decrease is given by $v_1^\ell = f(x_1) - \check{f}_1(\tilde{x})$, where $\tilde{x}$ is the solution of the QP (2) (the same as for LBM). When a lower bound f_k^{low} for the optimal value f^{inf} is found, the subsequent iteration solves the LP (17) to update f_k^{low} to f_{k+1}^{low} (the same as in LBM).

The essential parameters to tune in the updates above are $\mu_{\max}$, m_ℓ, τ_1 and $\tau_{\min}$.

5.1.4 Tuning the parameters

The parameters were tuned for each problem class separately. To decide on the "best" settings of parameters, we first ran each solver on representative instances (a subset of about 10 %) of each considered family of problems, with various possible combinations of the solvers' parameters.

- *Setting the stopping tolerances.* Depending on the solver, the tolerances involved in stopping tests are: $\texttt{Tol}_e$ for the aggregate error $\hat{e}_k$, $\texttt{Tol}_g$ for the norm of the aggregate subgradient $\hat{g}_k$ and $\texttt{Tol}_\Delta$ for the optimality gap Δ_k. As it is natural to have the optimality measures $\hat{e}_k$ and Δ_k of the same magnitude, we set $\texttt{Tol}_e = \texttt{Tol}_\Delta = \texttt{Tol}$. On the other hand, $|\hat{g}_k|$ is a dimension-dependent measure, which can be different. To set $\texttt{Tol}_g$ we performed the following steps for each class of problems:

- first, the sample of problems was solved by Algorithm 1 with the stopping test $\hat{e}_k \leq 10^{-8}$ (checking also that $\hat{g}_k$ is small enough at termination);
- at the last iteration k_i of the given method on problem i, we performed a linear regression on the data $\{\hat{e}_{k_i}\}$ and $\{|\hat{g}_{k_i}|\}$ to estimate the best constant $\rho > 0$ that minimizes the mean square error $\sum_i (\rho \hat{e}_{k_i} - |\hat{g}_{k_i}|)^2$;
- given tolerance $\texttt{Tol}$ for $\hat{e}_k$ and Δ_k, we then set $\texttt{Tol}_g := \rho \texttt{Tol}$.

In the final experiments reported, the solvers terminate either if the number of oracle calls reaches 1000 (considered a failure) or when

$$\hat{e}_k \leqslant \texttt{Tol} \quad \text{and} \quad |\hat{g}_k| \leqslant \texttt{Tol}_g, \quad \text{or} \quad \Delta_k \leqslant \texttt{Tol} \quad \text{with} \quad \texttt{Tol} = \left(1 + \left|\bar{f}\right|\right) 10^{-8}. \tag{40}$$

Here, $\bar{f}$ is a good approximation of the optimal value $f^{\inf}$, obtained by running one solver in advance, and the stopping tolerances are set as described above. The last stopping test, based on the optimality gap, is employed only by the solvers LBM and DSBM.

- *Setting the initial prox-parameter.* As mentioned, all the solvers employ an initial prox-parameter τ_1 (solvers LBM and DSBM use τ_1 to define v_1^ℓ). For each class of problems we tested $\tau_1 \in \{1, 5, 10\}$.
- *Lower bound for the prox-parameter.* Except for solver LBM: $\tau_{\min} \in \{10^{-6}, 10^{-5}, 10^{-3}\}$.
- *Parameter a to update τ_k during null steps*. Only for solvers PBM-1 and PBM-2: $a \in \{2, 4, 5\}$.
- *Level parameter m_ℓ*. Only for solvers LBM and DSBM: $m_\ell \in \{0.2, 0.5, 0.7\}$.
- *Descent parameter m_f*. All solvers: $m_f \in \{0.1, 0.5, 0.7\}$.
- *Parameter $\mu_{\max}$ in* (20). Only for solvers LBM and DSBM: $\mu_{\max} \in \{1, 5, 10\}$.

As expected, the standard choice $m_f = 0.1$ for the descent test proved adequate for all the solvers. Another adequate choice was $\mu_{\max} = 5$. Other parameters take different values depending on the class of problems, as shown below.

In all the solvers, all linearizations are kept in the model until the bundle reaches its maximal allowed size, which was set at 334 (approximately one third of the maximum number of iterations). When the maximal allowed size is reached, the solvers eliminate inactive linearizations, if any exist. If there are no inactive linearizations, the bundle is compressed: the two "less active" linearizations (with the smallest Lagrange multipliers) are replaced by the latest $\bar{f}_{k+1}$ and by the aggregate linearization $\bar{f}_k^a$.

5.2 Two-stage stochastic linear programming problems

We consider ten families of problems available at http://web.uni-corvinus.hu/~ideak1/kut_en.htm, by I. Deák. They result in convex linearly-constrained nonsmooth problems, of the form (1). Specifically,

$$f(x) := \langle c, x \rangle + \sum_{i=1}^{N} p_i \, Q(x, h_i) \quad \text{and} \quad \mathscr{X} := \left\{ x \in \Re^n_+ : \; Ax = b \right\},$$

Table 1 Total number of oracle calls and CPU time: sum over 200 instances

	PBM-1	PBM-2	LBM	DSBM
CPU time (min)	139	118	86	84
# Oracle calls	18,007	15,168	10,521	11,125
# Descent steps	4030	4234	4638	4649
# Level steps	0	0	10,521	3643

where

$$Q(x, h_i) := \min_{y \in \Re^{n_2}_+} \langle q, y \rangle \quad \text{s.t. } Tx + Wy = h_i$$

is the recourse function corresponding to the ith scenario $h_i \in \Re^{m_2}$ with probability $p_i > 0$ (W and T above are matrices of appropriate dimensions); $c \in \Re^n$, matrix $A \in \Re^{m_1 \times n}$ and vector $b \in \Re^{m_1}$ are such that the set $\mathscr{X}$ is bounded. We consider twenty instances corresponding to scenarios

$$N \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100\}.$$

The best configuration found for the parameters is the following: PBM-1 and PBM-2: $\tau_1 = 10$, $\tau_{\min} = 10^{-6}$ and $a = 2$; LBM: $\tau_1 = 10$, and $m_\ell = 0.2$; DSBM: $\tau_1 = 1$, $\tau_{\min} = 10^{-6}$ and $m_\ell = 0.2$. All the solvers employed the tolerances $\texttt{Tol}_g = 100\texttt{Tol}$ and $\texttt{Tol}$ as in (40).

Table 1 shows the total number of oracle calls and CPU times for solving (successively) all the twenty instances of each of the 10 problems (in total, 200) by the four methods.

DSBM is the fastest solver, followed by LBM. There were no failures in this benchmark. DSBM solver stopped by the relative optimality gap in 93 % of the instances, whereas LBM in around 96 %.

Optimality measures are reported in Table 2, for a subset of the instances. Ideally, both measures $\hat{e}_k/(1 + |\bar{f}|)$ and $\hat{g}_k/(1 + |\bar{f}|)$, or the measure $\Delta_k/(1 + |\bar{f}|)$, should be zero. Table 2 presents the number of digits of accuracy for these quantities. For instance, the number 09 for $\hat{e}_k/(1 + |\bar{f}|)$ means that the quantity in question has the value $c\, 10^{-09}$, with some $c \in (1, 10)$.

In Fig. 1 we give performance profiles [7] of the four solvers over the 200 instances. The top graphic considers the number of oracle calls (iterations), and the bottom one considers the CPU time. For example, let the criterion be CPU time. For each algorithm, we plot the proportion of problems that it solved within a factor of the time required by the best algorithm. In other words, denoting by $t_s(p)$ the time spent by solver s to solve problem p and by $t^*(p)$ the best time for the same problem among all the solvers, the proportion of problems solved by s within a factor γ is

$$\phi_s(\gamma) = \frac{\text{number of problems } p \text{ such that } t_s(p) \leq \gamma t^*(p)}{\text{total number of problems}}.$$

Therefore, the value $\phi_s(1)$ gives the probability of the solver s to be the best by a given criterion. Furthermore, unless $t_s(p) = \infty$ (which means that solver s failed to solve

Table 2 Comparison of the optimality measures: digits of accuracy

n	$\hat{e}_k/(1+\lvert\bar{f}\rvert)$				$\hat{g}_k/(1+\lvert\bar{f}\rvert)$				$\Delta_k/(1+\lvert\bar{f}\rvert)$	
	PBM-1	PBM-2	LBM	DSBM	PBM-1	PBM-2	LBM	DSBM	LBM	DSBM
5	09	09	08	08	13	10	08	08	09	09
10	09	09	08	09	13	10	06	10	09	08
15	09	09	08	06	13	11	05	05	09	09
20	09	09	09	09	13	09	06	08	09	08
25	09	09	09	07	14	10	04	06	09	09
30	09	09	09	06	13	07	05	05	09	09
35	09	09	13	08	12	09	05	06	09	09
40	09	10	08	08	14	08	06	10	09	10
45	09	09	08	07	08	08	06	05	09	09
50	09	09	08	08	11	08	07	09	09	08
55	09	09	08	08	10	08	06	08	09	09
60	09	09	09	09	08	07	06	07	09	08
65	09	08	10	08	08	07	05	07	09	09
70	08	09	09	07	07	08	05	06	09	09
75	09	09	10	09	08	07	06	08	09	08
80	09	09	09	09	12	08	06	09	09	09
85	09	09	09	09	11	07	08	07	09	08
90	09	09	08	08	11	09	06	07	09	09
95	10	07	09	09	08	08	07	06	08	09
100	09	08	10	08	08	07	05	07	09	09

problem p), it follows that $\lim_{\gamma\to\infty}\phi_s(\gamma)=1$. Thus, the higher is the line, the better is the solver (by this criterion).

We conclude from Fig. 1 that among the four solvers, LBM used less oracle calls (and CPU time) in approximately 60 % (58 %) of the 200 different instances, followed by DSBM (40 %) that was better than both solvers PBM-1 and PBM-2.

5.3 `RandMaxQuad` problems

In this subsection we consider a family of randomly generated problems of the form (1) with the objective function given by

$$f(x)=\max_{i=1,\ldots,10}\left\{\langle Q_i x x\rangle+\langle q_i x\rangle\right\}+\alpha|x|_1 \quad \text{and}$$
$$f(x)=\max_{i=1,\ldots,10}\left\{\langle Q_i x x\rangle+\langle q_i x\rangle\right\}+\alpha|x|_\infty,$$

where $Q_i\in\Re^{n\times n}$ and $q_i\in\Re^n$ are randomly generated, Q_i being symmetric positive semidefinite, $i=1,\ldots,10$. The problem's dimension n varies according to

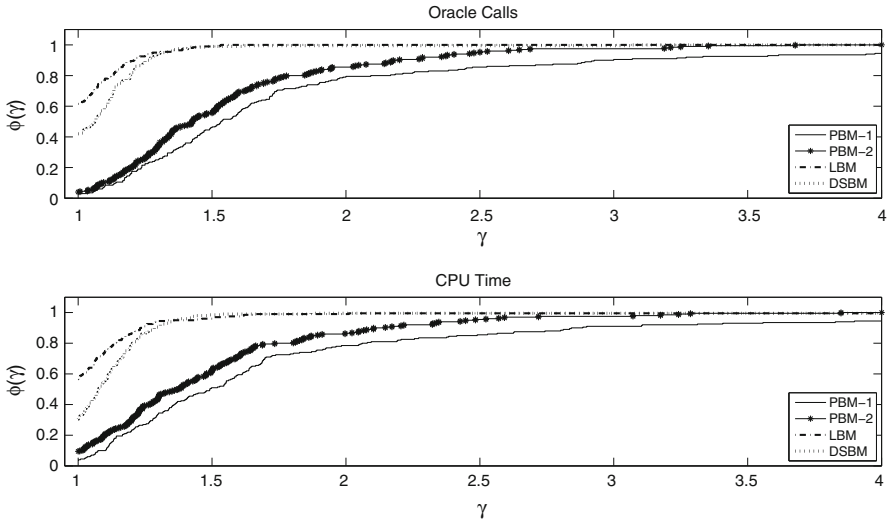

Fig. 1 Performance profile: 200 instances of two-stage stochastic problems

$$n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, 500\}.$$

Parameter α runs through the values $\alpha \in \{0.1, 0.5, 1\}$. Two settings were considered for the feasible set $\mathscr{X}$ in (1): $\mathscr{X} = \Re^n$ (unconstrained setting) and $\mathscr{X} = \{x \in \Re^n_+ : \sum_{i=1}^n x = 1\}$ (simplex setting). In total, 708 different instances of problem (1) were obtained by using different seeds for the MATLAB random number generator: 354 unconstrained and 354 constrained.

5.3.1 Unconstrained `RandMaxQuad`*: 354 instances*

The best configurations found for the parameters were: PBM-1: $\tau_1 = 1$, $\tau_{\min} = 10^{-6}$ and $a = 5$; PBM-2: $\tau_1 = 1$, $\tau_{\min} = 10^{-6}$ and $a = 2$; LBM: $\tau_1 = 1$, and $m_\ell = 0.2$; DSBM: $\tau_1 = 1$, $\tau_{\min} = 10^{-6}$ and $m_\ell = 0.2$. Tolerances were set as $\texttt{Tol}_g = 1000\texttt{Tol}$, with `Tol` given in (40).

Among other information, Table 3 shows the total number of CPU time (in min) and oracles calls required to solve all the 354 unconstrained instances. Notice that the less demanding with respect to oracle calls and CPU time is the DSBM solver, followed by PBM-2. Table 3 also shows that around 75 % of the DSBM iterations were of the level type.

Table 4 presents optimality measures at termination for each solver on some instances. DSBM stopped by the relative optimality gap in 29 % of the instances, whereas LBM triggered this additional stopping test in 38 %.

Figure 2 gives performance profiles [7] of the four solvers over 354 instances of the unconstrained `RandMaxQuad` problem.

We observe that DSBM required less oracle calls in approximately 48 % of the 354 instances, followed by PBM-2 and PBM-1 (around 40 and 10 %, respectively). Besides, DSBM is more robust in terms of oracle calls: it achieves $\phi(\gamma) = 1$ for

Table 3 Total number of oracle calls and CPU time: sum over 354 instances

	PBM-1	PBM-2	LBM	DSBM
CPU time (min)	302	154	462	143
# Oracle calls	121,155	80,261	202,922	77,741
# Descent steps	14,320	18,324	15,786	19,237
# Level steps	0	0	202,922	58,452
# Empty level sets	0	0	143	106
% Failure	2	1	25	0

Table 4 Comparison of the optimality measures: digits of accuracy

n	$\hat{e}_k/(1+\lvert\bar{f}\rvert)$				$\hat{g}_k/(1+\lvert\bar{f}\rvert)$				$\Delta_k/(1+\lvert\bar{f}\rvert)$	
	PBM-1	PBM-2	LBM	DSBM	PBM-1	PBM-2	LBM	DSBM	LBM	DSBM
10	09	09	09	09	08	06	05	05	09	09
20	09	09	09	09	06	06	05	05	09	09
30	09	09	09	10	06	06	05	06	09	–
40	09	09	09	09	06	06	06	06	–	–
50	09	09	09	09	06	06	06	06	–	–
60	09	09	09	10	06	06	06	06	–	–
70	09	09	09	09	06	06	06	06	–	–
80	09	09	09	09	06	06	06	06	–	–
90	09	09	09	09	06	06	06	06	–	–
100	09	09	09	09	06	06	06	06	–	–
150	09	10	07	09	06	06	06	05	–	–
200	09	09	07[a]	09	06	06	05[a]	06	–	–
250	08	09	06[a]	09	06	06	05[a]	06	–	–
300	09	09	06[a]	09	06	06	04[a]	06	–	–
500	10	09	05[a]	09	06	06	05[a]	06	–	–

– $f_k^{\text{low}} = -\infty$

[a] Means failure

lower values of γ. For this type of problems, both PBM-1 and PBM-2 are more robust than LBM, that failed to satisfy the stopping test in around 25 % of the instances, as reported in Table 3.

5.3.2 Constrained `RandMaxQuad`*: 354 instances*

We now consider problems with the same 354 objective functions, but constrained on a simplex. The employed solver parameters are the following: PBM-1: $\tau_1 = 1$, $\tau_{\min} = 10^{-3}$ and $a = 5$; PBM-2: $\tau_1 = 1$, $\tau_{\min} = 10^{-5}$ and $a = 2$; LBM: $\tau_1 = 1$, and $m_\ell = 0.7$; DSBM: $\tau_1 = 1$, $\tau_{\min} = 10^{-6}$ and $m_\ell = 0.7$. Tolerances were set as $\texttt{Tol}_g = 1000\texttt{Tol}$, with `Tol` given in (40).

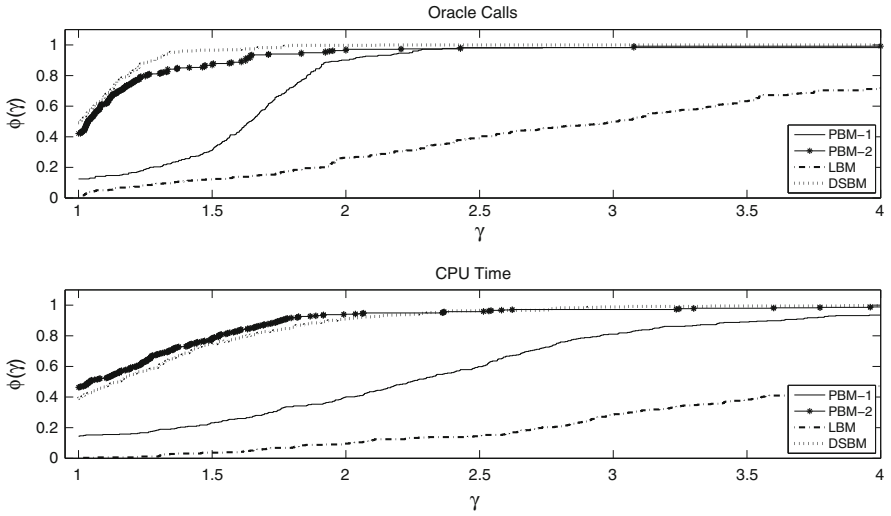

Fig. 2 Performance profile of the four solvers over 354 instances of `MaxQuad`

Table 5 Total number of oracle calls and CPU time: sum over 354 instances

	PBM-1	PBM-2	LBM	DSBM
CPU time (min)	319	309	35	31
# Oracle calls	110,495	103,494	35,892	31,617
# Descent steps	11,206	11,599	15,586	11,263
# Level steps	0	0	35,892	14,607
# Empty level sets	0	0	355	358
% Failure	2	0	0	0

Table 5 shows the total number of oracle calls, CPU time, descent steps and level steps required to solve all the constrained instances. We observe that the solvers LBM and DSBM are much more effective on the constrained problems than PBM-1 and PBM-2. Around 45 % of the DSBM iterations were of the level type.

LBM triggered the optimality gap stopping test in 62 % of the instances, while DSBM in 72 %. Note that these percentages were smaller for the unconstrained instances: 38 and 29 %, respectively. Table 6 reports (for some selected instances) the optimality measures at the last iteration.

Performance profiles of the four solvers on these 354 constrained instances are presented in Fig. 3. Among the considered solvers, we notice that DSBM is both the fastest and the most robust one, followed by LBM.

5.4 Unit-commitment energy problems

In this subsection we consider a unit-commitment problem for a power system operation model with four power plants. For each given point x, an oracle must solve four mixed-integer linear programming problems to compute $f(x)$ and $g \in \partial f(x)$. The

Table 6 Comparison of the optimality measures: digits of accuracy

n	$\hat{e}_k/(1+\lvert\bar{f}\rvert)$				$\hat{g}_k/(1+\lvert\bar{f}\rvert)$				$\Delta_k/(1+\lvert\bar{f}\rvert)$	
	PBM-1	PBM-2	LBM	DSBM	PBM-1	PBM-2	LBM	DSBM	LBM	DSBM
10	09	10	09	09	15	09	05	06	09	08
20	09	09	09	09	07	06	05	05	09	09
30	09	09	09	09	06	06	05	05	09	09
40	09	09	09	09	10	06	05	05	09	09
50	09	07	09	09	09	06	05	05	09	09
60	09	09	09	09	06	06	05	05	09	09
70	09	09	09	09	07	06	05	05	09	09
80	09	09	09	09	06	06	05	05	09	09
90	09	09	09	09	06	06	05	06	09	08
100	09	09	09	09	06	05	05	05	09	09
150	09	09	09	09	06	06	06	06	08	08
200	09	09	09	09	06	06	07	06	08	08
250	09	09	09	09	06	06	06	06	08	08
300	09	09	09	09	06	06	06	06	08	08
500	07	09	09	09	06	06	06	06	08	08

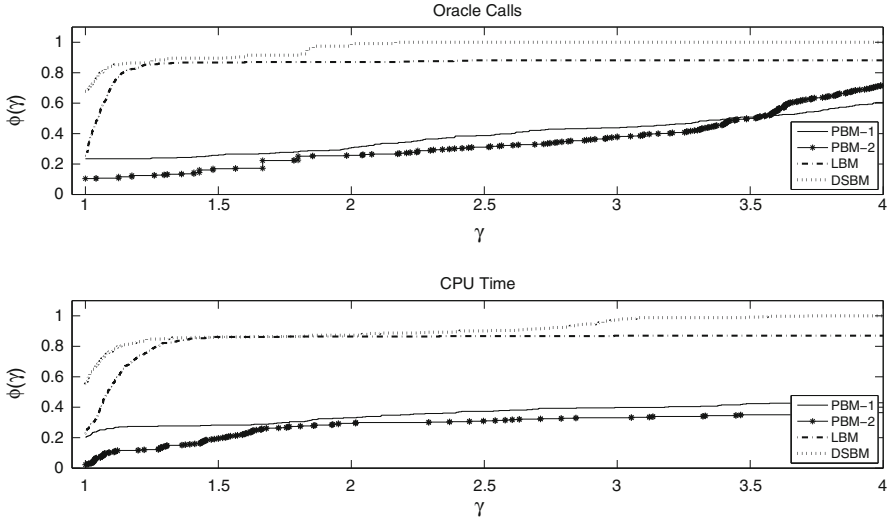

Fig. 3 Performance profile of the four solvers over 354 instances of constrained `MaxQuad`

feasible set for this problem is the positive orthant $\mathscr{X} = \Re^n_+$. In our configuration, the problem's dimension ranges in $n \in \{12, 24, 36, 48, 60\}$. The electricity demands for the unit-commitment problem were chosen randomly, using ten different seeds for the random number generator. In total, 50 instances of the problem were considered.

Table 7 Total number of oracle calls and CPU times: sum over 50 instances

	PBM-1	PBM-2	LBM	DSBM
CPU time (min)	78	69	91	77
# Oracle calls	4540	4050	3976	3087
# Descent steps	1052	1381	2274	1526
# Level steps	0	0	3976	1807
# Empty level sets	0	0	50	50

Table 8 Comparison of the optimality measures: digits of accuracy

n	$\hat{e}_k/(1+\lvert\bar{f}\rvert)$				$\hat{g}_k/(1+\lvert\bar{f}\rvert)$				$\Delta_k/(1+\lvert\bar{f}\rvert)$	
	PBM-1	PBM-2	LBM	DSBM	PBM-1	PBM-2	LBM	DSBM	LBM	DSBM
12	11	10	09	05	11	10	08	05	09	10
24	12	09	09	06	12	11	06	06	09	09
36	09	09	09	07	10	10	08	06	09	09
48	09	09	09	08	11	09	08	09	09	09
60	09	09	09	06	11	10	07	06	09	09

The employed solver parameters are the following: PBM-1: $\tau_1 = 10$, $\tau_{\min} = 10^{-6}$ and $a = 2$; PBM-2: $\tau_1 = 1$, $\tau_{\min} = 10^{-5}$ and $a = 4$; LBM: $\tau_1 = 1$, and $m_\ell = 0.7$; DSBM: $\tau_1 = 1$, $\tau_{\min} = 10^{-6}$ and $m_\ell = 0.2$. Tolerances were set as $\texttt{Tol}_g = \texttt{Tol}$, with `Tol` given in (40).

In this battery of problems, all the runs were successful, i.e., a stopping test was satisfied before the maximal number of iterations was reached. Table 7 shows the total number of oracle calls and CPU times required to stop the four solvers over all instances of the problem.

PBM-2 was the fasted solver on these problems, followed by DSBM. DSBM terminated by the optimality gap in 98 % of the instances, whereas LBM in 96 %. Moreover, around 58 % of the DSBM's iterations were of the level type. In Table 8 we present (for some instances) the optimality measures at the last iteration.

Figure 4 gives performance profiles of the four solvers over 50 instances of the problem.

We observe that DSBM required less oracle calls in approximately 90 % of cases, while PBM-2 was the fastest solver in 20 % of the instances.

5.5 Classical unconstrained nonsmooth test problems

In this subsection we consider some typical functions for nonsmooth optimization benchmarking, such as `MaxQuad` [4, p. 153], `TR48` [18, p.21, vol. II] and others. All the problems are unconstrained and have known optimal values. We refer to [24] for more information on these test problems.

Tables 9 and 10 report on results obtained by the four solvers on this type of problems, using default dimensions and starting points.

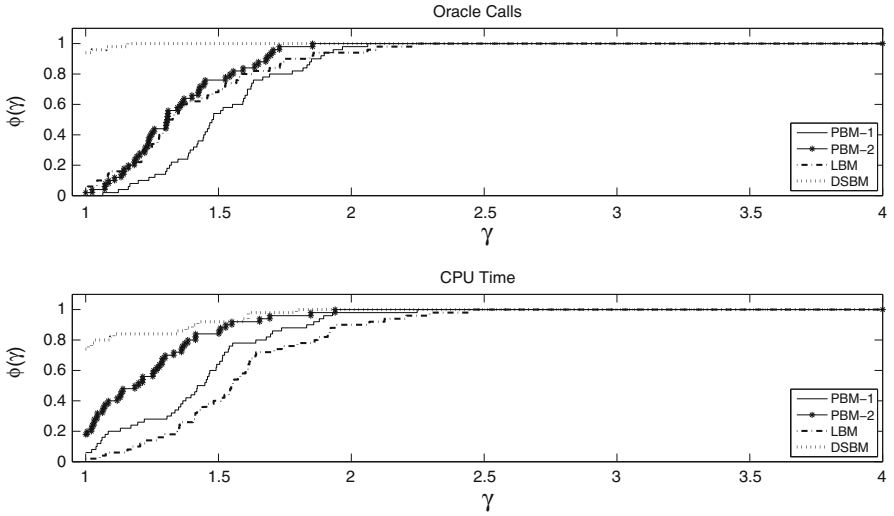

Fig. 4 Performance profile of the four solvers over 40 instances

Table 9 Total number of oracle calls and CPU time

Problem	# Oracle calls				CPU time (s)			
	LBM	PBM-1	PBM-2	DSBM	LBM	PBM-1	PBM-2	DSBM
TR48	260	157	127	139	30.399	9.986	6.873	8.669
MaxQuad	78	231	111	96	4.834	16.767	5.074	3.595
Ury	65	64	58	60	3.361	2.224	1.981	2.214
CPS	183	63	83	65	31.812	1.910	2.794	2.546
TiltedMax	50	14	18	15	2.285	0.434	0.601	0.461
Check	50	61	72	44	2.954	1.913	2.818	1.803
NK	58	62	81	61	2.938	1.754	2.543	3.053
Sum	744	652	550	480	78.583	34.988	22.684	22.341

Table 10 Digits of accuracy in the difference $f(\hat{x}) - f^{\text{inf}}$

Problem	LBM	PBM-1	PBM-2	DSBM	f^{inf}
TR48	1	5	3	3	−638,565
MaxQuad	6	9	9	7	−0.84140833459641
Ury	4	4	5	5	500
CPS	8	7	9	7	0
TiltedMax	8	9	7	9	0

Table 10 shows the true optimal value of each problem (column f^{inf}) and the number of digits of accuracy in the difference $f(\hat{x}) - f^{\text{inf}}$ for the four solvers at termination, where $\hat{x}$ is the obtained solution.

We can conclude from Table 10 that the quality of solutions obtained by the doubly stabilized method is as good as the other solvers.

6 Concluding remarks

We proposed a new algorithm for nonsmooth convex minimization, called doubly stabilized bundle method. It combines the level and proximal stabilizations in a single subproblem, and at each iteration automatically "chooses" between a proximal and a level step. The aim is to take advantage of good properties of both, depending on the problem at hand, and also use the simplicity of updating the level parameter to produce a simple and efficient rule to update the proximal parameter, thus speeding up the optimization process. In addition, the method provides a useful stopping test based on the optimality gap.

The algorithm appears to perform well in computation, as validated in Sect. 5, where almost 1,000 instances of various types of problems were considered. Numerical results show that the proposed method compares favorably with both the proximal and level bundle methods.

The new doubly stabilized algorithm can also handle inexactness of data in a natural way, without introducing special modifications to the iterative procedure (such as noise attenuation).

Acknowledgments The authors would like to acknowledge helpful comments of Claudia Sagastizábal. The authors also thank the anonymous referees for constructive suggestions that considerably improved the original version of this article.

References

1. Astorino, A., Frangioni, A., Gaudioso, M., Gorgone, E.: Piecewise-quadratic approximations in convex numerical optimization. SIAM J. Optim. **21**(4), 1418–1438 (2011). doi:10.1137/100817930
2. Bello-Cruz, J.Y., de Oliveria, W.: Level bundle-like algorithms for convex optimization. J. Glob. Optim. **59**(4), 787–809 (2014). doi:10.1007/s10898-013-0096-4
3. Ben-Tal, A., Nemirovski, A.: Non-euclidean restricted memory level method for large-scale convex optimization. Math. Program. **102**, 407–456 (2005). doi:10.1007/s10107-004-0553-4. http://dl.acm.org/citation.cfm?id=1057781.1057789
4. Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Theoretical and Practical Aspects. Universitext. Springer, Berlin (2006). Second edition, xiv+490 pp
5. Brannlund, U., Kiwiel, K.C., Lindberg, P.O.: A descent proximal level bundle method for convex nondifferentiable optimization. Oper. Res. Lett. **17**(3), 121–126 (1995). doi:10.1016/0167-6377(94)00056-C. http://www.sciencedirect.com/science/article/pii/016763779400056C
6. Correa, R., Lemaréchal, C.: Convergence of some algorithms for convex minimization. Math. Program. **62**(2), 261–275 (1993)
7. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002). doi:10.1007/s101070100263
8. de Oliveira, W.: Combining level and proximal bundle methods for convex optimization in energy problems. In: 3rd International Conference on Engineering Optimization—EngOpt, Rio de Janeiro, Mathematical Optimization Techniques, vol. 404, pp. 1–10 (2012). http://www.engopt.org/authors/404.html
9. de Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on-demand accuracy. Optim. Methods Softw. **29**(6), 1180–1209 (2014)

10. de Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM J. Optim. **21**(2), 517–544 (2011). doi:10.1137/100808289
11. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex bundle methods in depth: a unified analysis for inexact oracles. Math. Program. **148**(1–2), 241–277 (2014)
12. Fábián, C.: Bundle-type methods for inexact data. Cent. Eur. J. Oper. Res. **8**, 35–55 (2000)
13. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **13**(1), 117–156 (2002). doi:10.1137/S1052623498342186
14. Frangioni, A., Gorgone, E.: Bundle methods for sum-functions with 'easy' components: applications to multicommodity network design. Math. Program. **145**(1–2), 133–161 (2014)
15. Goffin, J.L., Haurie, A., Vial, J.P.: Decomposition and nondifferentiable optimization with the projective algorithm. Manag. Sci. **38**(2), 284–302 (1992). http://EconPapers.repec.org/RePEc:inm:ormnsc:v:38:y:1992:i:2:p:284-302
16. Goffin, J.L., Vial, J.P.: Interior points methods for nondifferentiable optimization. In: Proceedings of the Operations Research 1997 (Jena), pp. 35–49 (1998). http://EconPapers.repec.org/RePEc:fth:ehecge:97.24
17. Hintermüller, M.: A proximal bundle method based on approximate subgradients. Comput. Optim. Appl. **20**, 245–266 (2001)
18. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. No. 305–306 in Grund. der math. Wiss (two volumes). Springer, Berlin (1993)
19. Kiwiel, K.C.: Proximity control in bundle methods for convex nondiferentiable minimization. Math. Program. **46**, 105–122 (1990)
20. Kiwiel, K.C.: Proximal level bubdle methods for convex nondiferentiable optimization, saddle-point problems and variational inequalities. Math. Program. **69**(1), 89–109 (1995). doi:10.1007/BF01585554
21. Kiwiel, K.C.: A proximal bundle method with approximate subgradient linearizations. SIAM J. Optim. **16**(4), 1007–1023 (2006)
22. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**(1), 111–147 (1995). doi:10.1007/BF01585555
23. Lemaréchal, C., Sagastizábal, C.: Variable metric bundle methods: from conceptual to implementable forms. Math. Program. **76**, 393–410 (1997)
24. Sagastizábal, C.: Composite proximal bundle method. Math. Program. **140**(1), 189–233 (2013)
25. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**(1), 146–169 (2005). doi:10.1137/040603875
26. Solodov, M.: On approximations with finite precision in bundle methods for nonsmooth optimization. J. Optim. Theory Appl. **119**(1), 151–165 (2003). doi:10.1023/B:JOTA.0000005046.70410.02
27. van Ackooij, W., de Oliveira, W.: Level bundle methods for constrained convex optimization with various oracles. Comput. Optim. Appl. **57**(3), 555–597 (2014). doi:10.1007/s10589-013-9610-3