A DOUBLY STABILIZED BUNDLE METHOD FOR NONSMOOTH CONVEX OPTIMIZATION

WELINGTON OLIVEIRA AND MIKHAIL SOLODOV

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 new variant is able to update the lower bound for the optimal value, thus providing an additional useful stopping test based on the optimality gap. Furthermore, the level set constraint gives 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 seven hundred instances of different types of problems are presented (the model unit-commitment problem in the energy sector, two-stage stochastic linear programming, and some standard nonsmooth optimization test problems). Our experiments show that the doubly stabilized bundle method inherits useful features of the level and the proximal versions, and compares favourably to both of them.

Keywords: Nonsmooth Optimization, Proximal Bundle Method, Level Bundle Method, Inexact Oracle.

1. Introduction

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

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

where $f : \Re^n \to \Re$ is a nonsmooth convex function and $\mathcal{X} \subset \Re^n$ 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., [12, Chap. XIV], [3, Part II], and the analytic-center cutting-plane methods, e.g., [9, 10]. 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);

Date: April 13, 2013
Instituto Nacional de Matemática Pura e Aplicada, Rio de Janeiro, Brazil, wlo@impa.br and solodov@impa.br
Research of the second author is supported in part by CNPq Grant 302637/2011-7, by PRONEX-Optimization, and by FAPERJ.

(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 three main classes are the proximal, level, and trust-region, whose simplified conceptual versions we state next.

Proximal bundle method: e.g., [13, 17, 8, 21],

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

where $\tau_k > 0$ is the proximal parameter.

Level bundle method: e.g., [16, 14, 4, 19],

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

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

Trust-region bundle method: e.g., [24],

$$x_{k+1} := \arg\min \left\{ \check{f}_k(x) : \ |x - \hat{x}_k| \leq \delta_k, \ x \in \mathcal{X} \right\}, \tag{4}$$

where $\delta_k > 0$ is the trust-region 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, level and trust-region parameters (τ_k, ℓ_k and δ_k) such that the three versions above generate the same next iterate x_{k+1}. In other words, for some choice of paremeters, the solution of subproblems (2), (3) and (4) is the same. In this (formal, theoretical) sense the three 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. We next discuss some details of the proximal and level variants, as these are the two ingredients relevant for our developments.

First, it is worth to mention that updating the stability center $\hat{x}_k$ in item (ii) above is mandatory for the proximal bundle methods, but it may not be so for level methods. In some variants of level methods one can update $\hat{x}_k$ at each iteration [16, 7], or keep $\hat{x}_k = \hat{x}$ fixed for all iterations [5] (in which case $\hat{x}_k$ does not have the role of the "best" point computed so far). See also [14, 2] for different rules to manage the stability center $\hat{x}_k$.

Second, 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 [13] and [17] 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 [23] that for constrained problems the rule of [17] 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 are thoretically 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 $f^{\mathtt{inf}}$); see [16, 7] and (19) 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:

$$x_{k+1} := \arg\min \left\{ \check{f}_k(x) + \frac{1}{2\tau_k}|x - \hat{x}_k|^2 : \ \check{f}_k(x) \le \ell_k, \ x \in \mathcal{X} \right\} . \tag{5}$$

We immediately comment that (5) 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 (9) below. Thus, the subproblem (5) is no harder to solve than (2) or (3). Moreover, it turns out that the (unique) solution to problem (5) is also a solution to at least one of the problems (2) or (3); see Lemma 2.2 below. This reveals that the proposed method indeed combines the proximal and the level approaches, choosing between the two at every step. Our numerical results put in evidence that the choices made are "intelligent".

The advantages derived from (5) can be summarized as follows:

- the level parameter ℓ_k is easily updated, and can take into account a lower bound for $f^{\mathtt{inf}}$ if it is available;
- the level constraint $\check{f}_k(x) \le \ell_k$ 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 $\{x \in \mathcal{X} \ : \check{f}_k(x) \le \ell_k\}$, and not only on its boundary, as the level method does.

Thus, 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 $f^{\mathtt{inf}}$, giving an additional stopping test based on a certain optimality gap, something not present in the proximal bundle methods. It should be noted that having this additional stopping test is useful, as usual proximal bundle methods sometimes "take time" to accumulate enough information to recorgnize that an acceptable approximate solution had been already computed. Our numerical experiments confirm that the novel features of the proposed algorithm do give improvements with respect to both the proximal and the level variants.

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 Section 4. Section 5 contains numerical experiments comparing our bundle method with: the proximal bundle methods using the updating rules for τ_k given in [13] and [17]; and the level method given in [4]. 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, and some standard nonsmooth optimization test problems. Finally, Section 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 $\{x_k\} \subset \mathcal{X}$. For each point x_k an oracle (black-box) is called to compute $f(x_k)$ and an arbitrary subgradient $g_k \in \partial f(x_k)$. With this information, the method creates the linearization

$$\bar{f}_k(x) := f(x_k) + \langle g_k, x - x_k\rangle \leqslant f(x)\,, \tag{6}$$

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

$$\check{f}_k(x) := \max_{j\in J_k} \bar{f}_j(x) \leqslant f(x)\,, \tag{7}$$

where the set J_k may index some of the linearizations $\bar{f}_j$, $j \leqslant k$, of the form in (6), but also affine functions obtained as certain convex combinations of such previous linearizations (the so-called aggregate linearizations, defined below). Note that (6) implies that the inequality in (7) 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 J_k simply enumerates the affine functions comprising $\check{f}_k$, and thus J_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

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

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

$$\mathbb{X}_k := \{x \in \mathcal{X} \,:\, \check{f}_k(x) \leqslant \ell_k\}\,, \tag{8}$$

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

We first observe that in the standard (via slack variable) reformulation of the doubly stabilized subproblem (5) given by

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

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 maintaing feasibility (with the same x part of the solution). This observation implies that the solution to the latter problem, and thus to (5), can be alternatively obtained by solving the simpler

$$\min_{(x,r)\in\Re^{n+1}} \left\{ r + \frac{1}{2\tau_k}|x-\hat{x}_k|^2 :\, \check{f}_k(x) \leqslant r\,,\, r \leqslant \ell_k\,,\, x \in \mathcal{X} \right\}. \tag{9}$$

We now state some properties of the minimizer x_{k+1} in (5), or equivalently of the x part of the solution in (9).

Proposition 2.1. *If $\mathbb{X}_k \neq \emptyset$ then problem (5) has the unique solution x_{k+1}.*

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

(10)

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

In addition, the aggregate linearization

$$\bar{f}_k^a(\cdot) := \check{f}_k(x_{k+1}) + \langle \hat{g}_k, \cdot - x_{k+1}\rangle \ \ \textit{satisfies}\ \bar{f}_k^a(x) \leq \check{f}_k(x) \leq f(x)\ \textit{for all}\ x \in \mathcal{X}. \tag{11}$$

Proof. The existence and uniqueness of solution x_{k+1} to (5) 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 [12] (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 (9) assert that there exist $\mu_k \geq 0$ and $\lambda_k \geq 0$ such that

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

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

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

which completes the proof of all the relations in (10).

To show (11), note that for all $x \in \mathcal{X}$ 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 $h_{k+1} \in N_{\mathcal{X}}(x_{k+1})$. □

The next result shows that the solution x_{k+1} of the doubly stabilized problem (5) solves at least one of "singly" stabilized problems: the proximal (2) or the level (3).

Lemma 2.2. *For $\tau_k > 0$ and $\ell_k \in \Re$, let $x^\tau \in \Re^n$ and $x^\ell \in \Re^n$ be the (unique) solutions of problems (2) and (3), respectively. Let $x_{k+1} \in \Re^n$ be the unique solution of problem (5). Then it holds that*

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

where μ_k is the Lagrange multiplier defined in Proposition 2.1.

Proof. Let $\mu_k = 1$. Similarly to the proof of Proposition 2.1, writing the optimality conditions for (5) with $\mu_k = 1$ gives

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

which shows that x_{k+1} satisfies the optimality condition for (2).

Suppose now that $\mu_k > 1$. Since x^ℓ solves (3), x_{k+1} solves (5), and x^ℓ is feasible in (5), we obtain that

$$\ell_k + \frac{1}{2\tau_k}|x^\ell - \hat{x}_k|^2 \geqslant \check{f}_k(x^\ell) + \frac{1}{2\tau_k}|x^\ell - \hat{x}_k|^2 \geqslant \check{f}_k(x_{k+1}) + \frac{1}{2\tau_k}|x_{k+1} - \hat{x}_k|^2 . \tag{12}$$

As $\mu_k > 1$, it follows from (10) that $\lambda_k > 0$, and thus $\check{f}_k(x_{k+1}) = \ell_k$. We then obtain that

$$\check{f}_k(x_{k+1}) + \frac{1}{2\tau_k}|x_{k+1} - \hat{x}_k|^2 = \ell_k + \frac{1}{2\tau_k}|x_{k+1} - \hat{x}_k|^2 \geqslant \ell_k + \frac{1}{2\tau_k}|x^\ell - \hat{x}_k|^2 , \tag{13}$$

where the inequality is by the optimality of x^ℓ and feasibility of x_{k+1} in problem (3). We conclude from (12) and (13) that all the inequalities in those relations hold as equalities, and thus x^ℓ is also a solution to (5). Since problems (3) and (5) have unique solutions, it holds that that $x_{k+1} = x^\ell$. □

According to Lemma 2.2, 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 method makes either a step of the associated proximal bundle method, or of the level method. At every iteration this choice is made "automatically", and our numerical results confirm that over the course of the iterative process the choices are overall "intelligent".

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

$$v_k^\tau := f(\hat{x}_k) - \check{f}_k(x_{k+1}) \geqslant 0, \tag{14}$$

where the inequality follows from x_{k+1} being the solution of (5) via

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

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

$$f(x_{k+1}) \leqslant f(\hat{x}_k) - m_f v_k^\tau , \tag{15}$$

where $m_f \in (0,1)$. Accordingly, each iteration results either
- in a *descent step* when (15) holds, in which case $\hat{x}_k$ is moved to x_{k+1}; or
- in a *null step* when (15) 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 (5) and thus to the proximal parameter τ_k, and the aggregate linearization error given by

$$\hat{e}_k := f(\hat{x}_k) - \bar{f}_k^a(\hat{x}_k)\,. \tag{16}$$

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

Proposition 2.3. *It holds that*

$$\hat{e}_k \geqslant 0, \quad \hat{e}_k + \tau_k \mu_k |\hat{g}_k|^2 = v_k^\tau \geqslant f(\hat{x}_k) - \ell_k = v_k^\ell\,, \tag{17}$$

where μ_k is Lagrange multiplier defined in Proposition 2.1. Moreover, if $\mu_k > 1$ then $v_k^\tau = v_k^\ell$.

Furthermore, the following inequality holds:

$$f(\hat{x}_k) + \langle \hat{g}_k, x - \hat{x}_k\rangle - \hat{e}_k \leqslant f(x) \quad \textit{for all } x \in \mathcal{X}. \tag{18}$$

Proof. The fact that $\hat{e}_k \geqslant 0$ follows directly from (11). To show (17), note that

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

where the last equality follows from (10). In addition, since x_{k+1} is feasible in (5), 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 (17).

Note also that if $\mu_k > 1$ then $\lambda_k > 0$, in which case (10) implies $\check{f}_k(x_{k+1}) = \ell_k$, so that $v_k^\ell = v_k^\tau$.

We next verify (18). Using again (11), for all $x \in \mathcal{X}$ it holds that

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

and (18) follows taking into account (17). □

The relation (18) motivates one of the alternative stopping tests for our algorithm, which is in the spirit of standard bundle methods. Note that if $\mathcal{X}$ were the whole space, (18) would mean that $\hat{g}_k$ is an ε-subgradient of f at $\hat{x}_k$ (with $\varepsilon = \hat{e}_k \geqslant 0$). It is then natural to 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 in the form of a pseudo-code, and comment on some of its ingredients.

(a) Observe that the lower bound f_k^{low} is updated either when the level set $\mathbb{X}_k$ is empty, on line 12 of Algorithm 1, or on line 22. In the second case, it is explicit that $f_k^{low} \leqslant f^{\texttt{inf}}$. In the first case, $\mathbb{X}_k = \emptyset$ means that $\ell_k < \check{f}_k(x) \leqslant f(x)$ for all $x \in \mathcal{X}$. And since the update sets $f_k^{low} \leftarrow \ell_k$, it again holds that $f_k^{low} \leqslant f^{\texttt{inf}}$.

Algorithm 1 Doubly Stabilized Bundle Method

▷ **Step 0: initialization**

Choose parameters $m_\ell, m_f, m_e \in (0,1)$, and stopping tolerances $\texttt{Tol}_\Delta, \texttt{Tol}_e, \texttt{Tol}_g > 0$.
Given $x_1 \in \mathcal{X}$, set $\hat{x}_1 \leftarrow x_1$. Compute the corresponding $f(x_1)$ and $g_1 \in \partial f(x_1)$.
If a lower bound f_1^{low} for $f^{\texttt{inf}}$ is available, set $v_1^\ell \leftarrow (1-m_\ell)(f(\hat{x}_1) - f_1^{low})$; otherwise, set $f_1^{low} \leftarrow -\infty$ and choose $v_1^\ell > 0$.
Choose $\tau_1 > 0$ and set $J_1 = \{1\}$.
for $k = 1, 2, \ldots$ **do**

▷ **Step 1: first stopping test**

  set optimality gap $\Delta_k \leftarrow f(\hat{x}_k) - f_k^{low}$
  if $\Delta_k \leqslant \texttt{Tol}_\Delta$ **then**
    return $\hat{x}_k$ and $f(\hat{x}_k)$
  end if

▷ **Step 2: trial point finding**

  set level parameter $\ell_k \leftarrow f(\hat{x}_k) - v_k^\ell$
  if the level set $\mathbb{X}_k$ defined by (8) is detected to be empty **then**
    set $f_k^{low} \leftarrow \ell_k$, $v_k^\ell \leftarrow (1-m_\ell)(f(\hat{x}_k) - f_k^{low})$ and go back to line 6
  else
    solve (9) to obtain (x_{k+1}, r_{k+1}) and a Lagrange multiplier λ_k associated to the level constraint $r \leqslant \ell_k$
    set $\mu_k \leftarrow \lambda_k + 1$ and $v_k^\tau \leftarrow f(\hat{x}_k) - r_{k+1}$
    set $\hat{g}_k \leftarrow (\hat{x}_k - x_{k+1})/\tau_k\mu_k$ and $\hat{e}_k \leftarrow v_k^\tau - \tau_k\mu_k|\hat{g}_k|^2$
  end if

▷ **Step 3: second stopping test**

  if $\hat{e}_k \leqslant \texttt{Tol}_e$ and $|\hat{g}_k| \leqslant \texttt{Tol}_g$ **then**
    return $\hat{x}_k$ and $f(\hat{x}_k)$
  end if

▷ **Step:4 oracle call**

  compute $f(x_{k+1})$ and $g_{k+1} \in \partial f(x_{k+1})$

▷ **Step 5: descent test**

  choose $f_{k+1}^{low} \in [f_k^{low}, f^{\texttt{inf}}]$ (one option is to set $f_{k+1}^{low} \leftarrow f_k^{low}$)
  if $f(x_{k+1}) \leqslant f(\hat{x}_k) - m_f v_k^\tau$ **then** ▷ descent step
    set $\hat{x}_{k+1} \leftarrow x_{k+1}$ and $v_{k+1}^\ell \leftarrow \min\{v_k^\ell, (1-m_\ell)(f(\hat{x}_{k+1}) - f_{k+1}^{low})\}$
    set $\tau_{k+1} \leftarrow \tau_k\mu_k$
    choose model $\check{f}_{k+1}$ satisfying $\bar{f}_{k+1}(\cdot) \leqslant \check{f}_{k+1}(\cdot) \leqslant f(\cdot)$
  else ▷ null step
    set $\hat{x}_{k+1} \leftarrow \hat{x}_k$
    if $\mu_k > 1$ **then**
      set $\tau_{k+1} \leftarrow \tau_k$
      if $\hat{e}_k \geqslant -m_e\tau_k\mu_k|\hat{g}_k|^2$ **then**
        set $v_{k+1}^\ell \leftarrow m_\ell v_k^\ell$
      else
        set $v_{k+1}^\ell \leftarrow v_k^\ell$
      end if
    else
      choose $\tau_{k+1} \in [\tau_{\min}, \tau_k]$ and set $v_{k+1}^\ell \leftarrow v_k^\ell$
    end if
    choose model $\check{f}_{k+1}$ satisfying $\max\{\bar{f}_{k+1}(\cdot), \bar{f}_k^a(\cdot)\} \leqslant \check{f}_{k+1}(\cdot) \leqslant f(\cdot)$
  end if
end for

Therefore, $f_k^{low} \leqslant f^{\mathtt{inf}}$ for all k, and if the algorithm stops at Step 1, we have that

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

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

Note also that when the level set $\mathbb{X}_k$ is empty, the update rules over the pass through lines 6–12 of Algorithm 1, and back to 6, decrease the optimality gap Δ_k by the factor of $(1 - m_\ell)$.

(b) To identify if the level set is empty, the most natural is probably to proceed as usual with solving (5) 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 a linear program (if $\mathcal{X}$ is a polyhedron)

$$\min |s|_1 \;\text{ s.t. }\; \bar{f}_j(x) + s_j \leqslant \ell_k \;\forall\, j \in J_k,\; x \in \mathcal{X},\; s \in \Re^{|J_k|}\,.$$

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 $\mathcal{X}$ is bounded or f_k^{low} is finite, it is enough to solve a linear program (LP) to update f_k^{low} on line 22:

$$\text{set } f_{k+1}^{low} \leftarrow \min\; r \;\text{ s.t. }\; \bar{f}_j(x) \leqslant r \;\forall\, j \in J_k\,,\; f_k^{low} \leqslant r\,,\; x \in \mathcal{X}\,,\;\; r \in \Re\,. \tag{19}$$

This strategy is particularly effective when solving an LP is cheaper than evaluating the function and subgradient values; it is tested in a part of our numerical experiments in Section 5.3.

(d) If $\mathcal{X}$ 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 $\mathcal{X} = [0, +\infty)$). In that case, the algorithm will not stop at Step 1, unless the initial lower bound f_1^{low} is within the $\mathtt{Tol}_\Delta$-tolerance of $f^{\mathtt{inf}}$.

(e) The QP formulation of subproblem (9) is given by

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

It can also be seen that (if $\mathcal{X} = \Re^n$) its dual has $|J_k|$ variables, the size of 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 J_k) should be kept bounded, without impairing convergence. For this, the usual aggregation techniques of proximal bundle can be employed here. Specifically, 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 on lines 26 and 39 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 $\bar{f}_k^a$.

(g) Step 5 increases the proximal parameter τ_k only after descent steps resulting from level iterations. On the other hand, τ_k can be decreased only after null steps resulting from proximal iterations. In this manner, the level parameter ℓ_k indicates how to update the proximal parameter τ_k. This is precisely the novel strategy to manage proximal parameter, proposed in this work.

(h) Because $v_k^\tau \geqslant v_k^\ell$ (recall Proposition 2.3), after null steps v_k^ℓ should be decreased at level iterations in order to allow v_k^τ to approach zero, an important matter for establishing convergence. This is the aim of the rule $v_{k+1}^\ell = m_\ell v_k^\ell$ on line 32 of Algorithm 1. When the oracle information $(f(x_k), g_k)$ is exact (as considered up to now), the linearization error $\hat{e}_k$ is nonnegative by (17). Hence, in the case of the exact oracle information the inequality on line 31 holds automatically. It is introduced in Algorithm 1 having in mind possibly inexact data, the case analyzed in Section 4 below.

(i) 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).

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;

– infinitely many descent steps are generated;

– 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 (18), 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 3.1. *Suppose the level set $\mathbb{X}_k$ is empty infinitely many times.*

Then $\Delta_k \to 0$, $\{f(\hat{x}_k)\} \to f^{\texttt{inf}}$, 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 $f_k^{low} > -\infty$ and thus $\Delta_k < +\infty$. Also, by Steps 2 and 5, $v_k^\ell \leqslant (1 - m_\ell)\Delta_k$. Thus,

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

which shows that if $\mathbb{X}_k = \emptyset$ at iteration k, then the update $f_k^{low} \leftarrow \ell_k$ 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 $f^{\texttt{inf}} = -\infty$, in the case under consideration $f^{\texttt{inf}} > -\infty$. We can then write $\Delta_k = f(\hat{x}_k) - f_k^{low} \geqslant f(\hat{x}_k) - f^{\texttt{inf}}$, 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 mostly follows the arguments for proximal bundle methods. We include a proof for completeness.

Lemma 3.2. *Suppose Algorithm 1 generates infinitely many descent steps.*

Then $\{f(\hat{x}_k)\} \to f^{\texttt{inf}}$ 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 j-th descent step. Defining $i(j) = k(j+1) - 1$, it follows by the descent test (15) that

$$f(\hat{x}_{k(j)}) - f(\hat{x}_{k(j+1)}) \geqslant m_f v^\tau_{i(j)} \geqslant 0 \quad \text{for all } j \geqslant 1.$$

As $\{f(\hat{x}_{k(j)})\}$ is a nonincreasing sequence, its limit is either finite or it is $-\infty$. In the second case, clearly $\{f(\hat{x}_{k(j)})\} \to f^{\texttt{inf}} = -\infty$. In the first case, it holds that

$$\begin{aligned} +\infty &> f(\hat{x}_{k(1)}) - \lim_{j\to\infty} f(\hat{x}_{k(j+1)}) \\ &= \sum_{j=1}^{\infty} \big(f(\hat{x}_{k(j)}) - f(\hat{x}_{k(j+1)})\big) \\ &\geqslant m_f \sum_{j=1}^{\infty} v^\tau_{i(j)} \geqslant 0\,. \end{aligned} \tag{20}$$

It follows that $v^\tau_{i(j)} \to 0$ as $j \to \infty$. Then, using (17) and the facts that $\tau_k \geqslant \tau_{\min} > 0$ and $\mu_k \geqslant 1$ for all k, we conclude that

$$\hat{e}_{i(j)} \to 0 \;\text{ and }\; |\hat{g}_{i(j)}| \to 0 \quad \text{as } j \to \infty. \tag{21}$$

Let $x \in \mathcal{X}$ be arbitrary. Using (10), we obtain that

$$\begin{aligned} |\hat{x}_{k(j+1)} - x|^2 &= |\hat{x}_{k(j)} - x|^2 + (\tau_{i(j)}\mu_{i(j)})^2|\hat{g}_{i(j)}|^2 + 2\tau_{i(j)}\mu_{i(j)}\langle \hat{g}_{i(j)}, x - \hat{x}_{k(j)}\rangle \\ &\leqslant |\hat{x}_{k(j)} - x|^2 + (\tau_{i(j)}\mu_{i(j)})^2|\hat{g}_{i(j)}|^2 + 2\tau_{i(j)}\mu_{i(j)}(f(x) - f(\hat{x}_{k(j)}) + \hat{e}_{i(j)}) \\ &\leqslant |\hat{x}_{k(j)} - x|^2 + 2\tau_{i(j)}\mu_{i(j)}v^\tau_{i(j)} + 2\tau_{i(j)}\mu_{i(j)}(f(x) - f(\hat{x}_{k(j)}))\,, \end{aligned} \tag{22}$$

where the first inequality is by (18) (notice that $\hat{x}_{k(j)} = \hat{x}_{i(j)}$) and the last is by (17).

Suppose that $\lim_{j\to\infty} f(\hat{x}_{k(j)}) > f^{\texttt{inf}}$. Then there exist $t > 0$ and $\tilde{x} \in \mathcal{X}$ such that $f(\hat{x}_{k(j)}) - t \geqslant f(\tilde{x})$ for all j. As $v^\tau_{i(j)} \to 0$ as $j \to \infty$, it holds that $v^\tau_{i(j)} \leqslant t/2$ for all j large enough. Then taking j large enough and $x = \tilde{x}$ in (22), we obtain that

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

where we used the fact that $\tau_k\mu_k \geq \tau_k \geq \tau_{\min}$. The above gives a contradiction when $j \to \infty$, thus proving the first assertion.

Suppose now that there exists a solution $\bar{x}$ to (1). Taking $x = \bar{x}$ in (22), we obtain that

$$|\hat{x}_{k(j+1)} - \bar{x}|^2 \leqslant |\hat{x}_{k(j)} - \bar{x}|^2 + 2\tau_{i(j)}\mu_{i(j)}v^\tau_{i(j)}. \tag{23}$$

If $\{\tau_{i(j)}\mu_{i(j)}\}$ is bounded above then, in view of (20), it holds that

$$\sum_{j=1}^{\infty} \tau_{i(j)}\mu_{i(j)}v^\tau_{i(j)} < +\infty.$$

In the latter case, (23) implies that $\{|\hat{x}_{k(j)} - \bar{x}|\}$ converges and so $\{\hat{x}_{k(j)}\}$ is bounded. Since all cluster points of the latter are solutions, we can take one of these as the solution $\bar{x}$ in (23). But then the limit of $\{|\hat{x}_{k(j)} - \bar{x}|\}$ must be zero, which establishes the claim.

We finally 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. □

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 3.3. *Suppose there exists an index $k_1 \geqslant 1$ such that the descent test* (15) *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 line 32 of Algorithm 1, $v_{k+1}^\ell = m_\ell v_k^\ell$ (recall that in the case of exact oracle under consideration, $\hat{e}_k \geqslant 0$ and inequality on line 31 holds automatically). 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 2.3, 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 (17),

$$v_k^\tau \geq v_{k_2}^\ell > 0 \quad \text{for all} \quad k \geq k_2. \tag{24}$$

Moreover, by Lemma 2.2, 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 [12][Chap. XV, Thm. 3.2.4] that $v_k^\tau \to 0$, in contradiction with (24).

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 $v_k^\tau = v_k^\ell$ (recall Proposition 2.3), 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 (17) that

$$e_k \to 0 \text{ and } |\hat{g}_k| \to 0 \quad \text{as } K \ni k \to \infty. \tag{25}$$

Now fixing an arbitrary $x \in \mathcal{X}$ in (18), using (25) and passing onto the limit in (18) along the indices in K, gives $f(\hat{x}_{k_1}) \leqslant f(x)$. This completes the proof. □

Summarizing Lemmas 3.1–3.3, we state the following convergence properties of Algorithm 1.

Theorem 3.4. *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).

Finally, we point out that the analysis presented above shows that for positive stopping tolerances $\texttt{Tol}_\Delta, \texttt{Tol}_e, \texttt{Tol}_g$, Algorithm 1 always terminates with an appropriate approximate solution of the problem. Indeed, in the three possible cases considered above, either $\Delta_k \to 0$

(Lemma 3.1) or there exists $K \subset \{1, 2, \ldots\}$ such that $\hat{e}_k \to 0$ and $|\hat{g}_k| \to 0$ as $K \ni k \to \infty$ (see (21) in Lemma 3.2 and (25) in Lemma 3.3).

4. Handling inexact data

In various real-world applications, the objective function and/or its subradient can be 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 Section 5.3 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 [11, 25, 15, 21, 20]. 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

$$\left\{ \begin{array}{l} f_x = f(x) - \eta_x \ \text{ and} \\ f(\cdot) \geqslant f_x + \langle g_x, \cdot - x \rangle - \eta_x^g, \end{array} \right. \tag{26}$$

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

$$|\eta_x| \leqslant \eta \quad \text{and} \quad \eta_x^g \leqslant \eta^g \quad \text{for all } x \in \mathcal{X}. \tag{27}$$

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 (7). However, because of the inexactness, we now have the weaker property $\check{f}_k(\cdot) \leqslant f(\cdot) + \eta^g$. Naturally, the predicted descent must employ only the available (inexact) information; instead of (14), it is thus given by

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

and the level parameter is

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

Solving the doubly stabilized bundle subproblem (5) 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 (10) and (11), 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 (17)), the aggregate linearization error $\hat{e}_k$ can be negative due to inaccuracy in the data. However, given the oracle assumptions (26), the following lower bound holds:

$$\hat{e}_k \geqslant f(\hat{x}_k) - \eta - \bar{f}_k^a(\hat{x}_k) \geqslant f(\hat{x}_k) - \eta - (f(\hat{x}_k) + \eta^g) = -(\eta + \eta^g)\,. \tag{28}$$

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 2.3, 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

$$f_{x_{k+1}} \leqslant f_{\hat{x}_k} - m_f v_k^\tau, \tag{29}$$

mimicking (15), is not meaningful. The methods in [15, 21, 20] 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 is called *noise attenuation* [15]. Once the linearization error $\hat{e}_k$ becomes (relatively) not too nonnegative, i.e., $\hat{e}_k \geqslant -m_e\tau_k|\hat{g}_k|^2$ holds for some $m_e \in (0,1)$, the expected decrease v_k^τ becomes positive. Then the inexact proximal bundle method proceeds just like the exact one. We refer to [20] for more details on the inexact proximal bundle methods. It turns out that our doubly stabilized method does not require the noise attenuation modification to handle inexact data.

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 (26) and (27). 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 following additional rule for managing the bundle is introduced:

(30)

If k is a null proximal iteration and $k+1,\ldots,k+p$ are consecutive null level iterations such that $\hat{e}_{k+i} < -m_e\tau_{k+i}\mu_{k+i}|\hat{g}_{k+i}|^2 \;\forall\, i=1,\ldots,p$, then the model $\check{f}_{k+i}$ satisfies $\max\{\bar{f}_{k+1}(\cdot), \bar{f}_k^a(\cdot), \bar{f}_{k+i}(\cdot), \bar{f}_{k+i-1}^a(\cdot)\} \leqslant \check{f}_{k+i}(\cdot) \leqslant f(\cdot), i=1,\ldots,p.$

In other words, on consecutive null level steps that follow a null proximal step and for which the inequality on line 31 of Algorithm 1 does not hold, the information about the last null proximal step is kept in the bundle.

An interesting feature of our doubly stabilized bundle method is that the predicted decrease v_k^τ is *always* positive, even if inexact information is used to build the cutting-plane model. We next verify this claim. First note that if v_k^ℓ becomes nonpositive at some iteration k due to the updates on line 12 or line 24 of Algorithm 1, then so does the inexact optimality gap Δ_k on line 6 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. The same argument as that in Proposition 2.3 shows that

$$v_k^\tau = \hat{e}_k + \tau_k\mu_k|\hat{g}_k|^2 \geqslant f_{\hat{x}_k} - \ell_k = v_k^\ell\,. \tag{31}$$

Hence,

$$v_k^\tau \geqslant v_k^\ell > 0 \quad \forall\, k\,.$$

Therefore, a descent test like (29) is always meaningful, unlike for the proximal bundle approach with inexact data. This ensures that in the doubly stabilized method essentially the same algorithm/code can be applied both in the exact and inexact cases, subject to the minor modification (30) in managing the bundle (actually, (30) could also be used in the exact case without any harm, of course).

We proceed with convergence analysis in the inexact case. First note that if the errors of approximation do not vanish in the limit, of course only approximate solutions to (1) can be expected in general. This is natural, and similar to [25, 15, 21, 20].

We can proceed as in Proposition 2.1 to show that $\bar{f}_k^a(x) \leqslant \check{f}_k(x)$ for all $x \in \mathcal{X}$. Since by the inexact oracle definition (26) we have that $\bar{f}_j(\cdot) \leqslant f(\cdot) + \eta^g$ for all $j \in J_k$, we conclude

that for all $x \in \mathcal{X}$ it holds that

$$\begin{aligned} f(x)+\eta^g &\geqslant \check{f}_k(x) \geqslant \bar{f}_k^a(x) = \check{f}_k(x_{k+1}) + \langle \hat{g}_k, x - x_{k+1}\rangle \\ &= f_{\hat{x}_k} - (f_{\hat{x}_k} - \check{f}_k(x_{k+1})) + \langle \hat{g}_k, x-\hat{x}_k\rangle + \langle \hat{g}_k, \hat{x}_k - x_{k+1}\rangle \\ &= f_{\hat{x}_k} - v_k^\tau + \langle \hat{g}_k, x-\hat{x}_k\rangle + \tau_k\mu_k|\hat{g}_k|^2 \qquad (32) \\ &= f_{\hat{x}_k} - \hat{e}_k + \langle \hat{g}_k, x-\hat{x}_k\rangle \qquad (33) \\ &\geqslant f_{\hat{x}_k} - v_k^\tau + \langle \hat{g}_k, x-\hat{x}_k\rangle\,. \qquad (34) \end{aligned}$$

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 (31) we have that $v_k^\ell = v_k^\tau$.

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

Lemma 4.1. *Suppose the level set $\mathbb{X}_k$ is empty infinitely many times.*

Then $\Delta_k \to 0$,

$$\lim_{k\to\infty} f_{\hat{x}_k} \leqslant f^{\mathit{inf}} + \eta^g, \qquad (35)$$

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 $f^{\mathtt{inf}} > -\infty$. The same argument as that of Lemma 3.1 shows that $\Delta_k \to 0$. Also, on the iterations in question we have that $\ell_k < \check{f}_k(x)$ for all $x \in \mathcal{X}$, and thus the update in Step 2 and (32) ensure that $f_k^{low} \leqslant f^{\mathtt{inf}} + \eta^g$. As $\{f_{\hat{x}_k}\}$ is decreasing and bounded below (since $f^{\mathtt{inf}} > -\infty$), we conclude that

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

which gives (35).

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

$$f^{\mathtt{inf}} + \eta^g \geqslant \lim_{j\to\infty} f_{\hat{x}_{k_j}} = \lim_{j\to\infty}(f(\hat{x}_{k_j}) + \eta_{\hat{x}_{k_j}}) \geqslant f(\tilde{x}) - \eta\,, \qquad (36)$$

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 (29) holds).

Lemma 4.2. *Suppose Algorithm 1 generates infinitely many descent steps.*

Then (35) 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 j-th descent step, and define $i(j) = k(j+1)-1$. It follows from (29) that $\{f_{\hat{x}_{k(j)}}\}$ is decreasing and either $\{f_{\hat{x}_{k(j)}}\} \to -\infty$, in which case (26), (27) 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 (29) implies that

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

Let $x \in \mathcal{X}$ be arbitrary. Using (34) and the fact that $\hat{x}_{k(j)} = \hat{x}_{i(j)}$, we then obtain that

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

Suppose that (35) does not hold. Then there exist $t > 0$ and $\tilde{x} \in \mathcal{X}$ such that $f_{\hat{x}_{k(j)}} \geqslant f(\tilde{x}) + \eta^g + t$ for all j. Taking j large enough so that $v^\tau_{i(j)} \leqslant t/2$, and choosing $x = \tilde{x}$ in the chain of inequalities above, we obtain that

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

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 (35) holds. The last assertion then follows by the same argument as in Lemma 4.1. □

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

Lemma 4.3. *Suppose that for Algorithm 1, with the additional bundle management rule* (30)*, there exists an index $k_1 \geqslant 1$ such that the descent test* (29) *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 $\{v^\ell_k\}$ is monotone; thus either $v^\ell_k \to 0$ or $v^\ell_k = v^\ell > 0$ for all k large enough.

If $v^\ell_k \to 0$ then the update on line 32 of Algorithm 1 occurs infinitely many times, which means that there exists an infinite index set K such that $\mu_k > 1$ and the inequality on line 31 is valid for $k \in K$. For such indices, it then holds that

$$0 \leqslant (1 - m_e)\tau_{\min}|\hat{g}_k|^2 \leqslant (1 - m_e)\tau_k\mu_k|\hat{g}_k|^2 \leqslant \hat{e}_k + \tau_k\mu_k|\hat{g}_k|^2 = v^\tau_k = v^\ell_k\,, \tag{37}$$

where the last equality follows from Proposition 2.3, because $\mu_k > 1$ for all $k \in K$. It follows from (37) 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 (33) as $K \ni k \to \infty$, with $x \in \mathcal{X}$ fixed but arbitrary and $\hat{x}_k = \hat{x}_{k_1}$ fixed, implies the assertion.

We now consider the second case: $v^\ell_k = v^\ell > 0$ 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 inbetween, we can consider the sequence $\{x_{k(j)}\}$ as that generated by the proximal bundle method, where the model satisfies, by the bundle management rule (30), the key conditions

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

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:

$$0 \geqslant \limsup_{j\to\infty} \left(f_{x_{k(j)}} - \check{f}_{k(j)-1}(x_{k(j)})\right). \tag{38}$$

(See [20, Theorem 4.6] and/or [15, Lemma 3.3 and Section 4.1].) On the other hand, as the descent condition (29) does not hold,

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

which gives a contradiction with (38).

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 $\mu_k > 1$, $\lambda_k > 0$, $\hat{x}_k = \hat{x}$, $v_k^\ell = v^\ell > 0$ and $\ell_k = \ell$ for all $k \geqslant k_3$.

Note that

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

By Proposition 2.1, 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

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

It then follows that

$$|x_{k+1} - \hat{x}|^2 \geqslant |x_k - \hat{x}|^2 + |x_{k+1} - x_k|^2 . \tag{39}$$

Note that

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

Using the Cauchy-Schwarz inequality, we obtain that

$$|g_{x_k}||x_{k+1} - x_k| \geqslant f_{x_k} - \ell . \tag{40}$$

Since this is a null step, it holds that

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

and since it is a level step, $v_{k-1}^\tau = v_{k-1}^\ell = v^\ell > 0$. 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 (40) 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 (39), it follows that

$$|x_{k+1} - \hat{x}|^2 \geqslant |x_k - \hat{x}|^2 + \left(\frac{(1 - m_f)v^\ell}{|g_{x_k}|} \right)^2 . \tag{41}$$

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 (26), (27)). But then (41) 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 (41), 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 (28) we have that

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

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

In the case under consideration, by line 31 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 (28), 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 $x \in \mathcal{X}$, and passing onto the limit in (33) 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 4.4. *If Algorithm 1 (with the additional rule* (30)*) 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,* (35) *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).

We finalize this section by stressing that differently from [15], Algorithm 1 does not increase the proximal parameter τ_k when the oracle error is excessive, i.e., there is no need for the noise attenuation procedure. We showed that for the case of finitely many descent steps with excessive oracle error, a subsequence of the Lagrange multipliers $\{\lambda_k\}$ associated to the level constraint tends to infinity (equivalently, $\{\mu_k\}$ tends to infinity). It follows by (10) that this is somehow equivalent to increasing τ_k as in noise attenuation, except that this is done by our method "automatically" as a by-product of the iterations. It is worth mentioning also that while the method of [15] may have to solve many QPs (2) for different proximal parameters τ_k before an iterate satisfying the inequality on line 31 of Algorithm 1 is obtained, our algorithm solves only one QP (9) per iteration.

Another alternative that can be tried in practice is to include noise attenuation in Algorithm 1, i.e., insert the additional command $\tau_{k+1} \leftarrow 10\tau_k$ on line 34, and not allow decreasing the proximal parameter τ_k until a new descent step is performed. In this case, the rule (30) is no longer needed. Notice that this procedure would still be not exactly the noise attenuation of [15], as the QP is still solved only once per iteration.

5. Numerical results

In this section we report computational experiments on a wide variety of different types of problems: the model unit-commitment problem in the energy sector, two-stage stochastic linear programming, and some standard nonsmooth test problems. Moreoever, for the stochastic case both exact and inexact oracles are considered. We compare the following four solvers:

– PBM-1 - proximal bundle method using the rule to update τ_k given in [13];
– PBM-2 - proximal bundle method using the rule to update τ_k given in [17];
– LBM - level bundle method of [4];
– DSBM - doubly stabilized bundle method, i.e., Algorithm 1.
All the four solvers terminate if the number of oracle calles reaches 1000 (considered a failure) or when

$$\hat{e}_k \leqslant \texttt{Tol}_e \;\text{ and }\; |\hat{g}_k| \leqslant \texttt{Tol}_g, \;\text{ with }\; \texttt{Tol}_g = \texttt{Tol}_e = 10^{-5}\sqrt{n}\,.$$

The solvers LBM and DSBM also use the following additional stopping test available to them, based on the (relative) optimality gap:

$$\frac{\Delta_k}{1+|f(\hat{x}_k)|} \leqslant \texttt{Tol}_\Delta, \;\text{ with }\; \texttt{Tol}_\Delta = 10^{-5}\,.$$

In all the solvers and runs, at each iteration k only strongly active linearizations are kept in the model $\check{f}_{k+1}$, if the bundle did not reach its maximal allowed size, adding to it also the last linearization $\bar{f}_{k+1}$. The maximal allowed size is 100, i.e., if there are more than 100 strongly active linearizations, the bundle is compressed: the two "oldest" linearizations are replaced by the latest $\bar{f}_{k+1}$ and by the aggregate linearization $\bar{f}^a_k$. Other parameters of Algorithm 1 are as follows: $m_f = 0.1$, $m_\ell = 0.5$, $m_e = 0.999$, $\tau_{\min} = 10^{-5}$. Those parameters that play the same roles in the other solvers, have the same values. In LBM, there is an additional parameter $t_{\max} = 10$ (in the notation of [4]).

The runs were performed on a computer with Intel(R) Core(TM) i3 CPU, 2.27 GHz, 4G (RAM), under Windows 7, 68 Bits. The QPs (and also LPs in § 5.3) are solved by the MOSEK (`http://www.mosek.com/`) Version 6.0.0.114 toolbox, the MATLAB version is R2008b.

We start with results for exact oracles in model energy problems and standard nonsmooth optimization test problems. Our analysis of the outcomes concerns 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. For problems with known optimal values (those are the standard nonsmooth test problems in Section 5.2) we also compare the quality of solutions obtained at termination (regardless of the stopping test that triggered it). To get some further insight, we report also the numbers of descent steps for all the solvers, the number of level iterations (for DSBM which has iterations of different kinds) and of empty level sets encountered (for LBM and DSBM).

5.1. **Unit-commitment energy problems.** In this subsection we consider a model unit-commitment problem, called `Unitoy`, created and coded (in Matlab) by Claudia Sagastizábal (`http://w3.impa.br/~sagastiz/`). According to [26], we consider stabilized versions of the problem: $f_{S_j}(x) = f(x) + S_j(x)$, where

$$\begin{array}{rcl} S_1(x) &=& L\sum_i |x_{i+1} - x_i| \\ S_2(x) &=& L\max_i |x_{i+1} - x_i| \\ S_3(x) &=& L\sum_i |-x_{i-1} + 2x_i - x_{i+1}| \\ S_4(x) &=& L\max_i |-x_{i-1} + 2x_i - x_{i+1}|\,. \end{array}$$

This problem is unconstrained. In our configuration, the decision variable has dimension $n = 20$. The constant L runs through the values

$$L \in \{1,\, 10,\, 20,\, 30,\, 40,\, \ldots,\, 500\}\,.$$

This gives 51 different instances for each stability function S_j, with $j = 1, 2, 3, 4$. Therefore, we considered 204 instances of the `Unitoy` problem.

In this battery of problems, all the runs were sucessful, i.e., a stopping test was satisfied before the maximal number of iterations was reached. Table 1 shows the total number of oracle calls and CPU time required to stop the four solvers over each family of problems, corresponding to different stability functions S_j. The solver DSBM is the fastest for each

	# oracle calls				CPU time in seconds			
Stability function	LBM	PBM-1	PBM-2	DSBM	LBM	PBM-1	PBM-2	DSBM
S_1	14200	14765	9132	5144	331	394	213	118
S_2	14346	16269	10134	5859	330	438	241	138
S_3	9049	14345	6689	4028	189	379	145	83
S_4	6809	12751	6997	3728	135	331	166	75
Sum	44404	58130	32952	18759	986	1541	764	413

TABLE 1. Total number of oracle calls and CPU time: sum over 51 instances.

family of stability functions, PBM-2 being the second fastest. When compared to LBM, PBM-1 and PBM-2, the solver DSBM reduced the CPU time needed to solve all 204 instances of `Unitoy` in about 58% , 73% and 45%, respectively. Reduction in oracle calls is approximately 57%, 67% and 43%, respectively.

Table 2 shows the number of descent steps, level steps and empty level sets. We complement

	# decent steps				# empty level sets		# level steps
Stability function	LBM	PBM-1	PBM-2	DSBM	LBM	DSBM	DSBM
S_1	1900	2104	2096	1855	93	51	1279
S_2	1930	2222	2336	2118	93	50	1374
S_3	1444	1905	1895	1439	132	51	1287
S_4	1235	1861	1766	1306	195	60	1222
Sum	6509	8092	8093	6718	513	212	5162

TABLE 2. Total number of descent and level steps, and of empty level sets: sum over 51 instances.

these results by the following comments.

LBM: 15% of all iterations are descent steps; the level set was empty in 1.15% of all iterations.

PBM-1: 14% of all iterations are descent steps.

PBM-2: 25% of all iterations are descent steps.

DSBM: 35% of all iterations are descent steps; the level set was empty in 1.13% of all iterations; about 72% of all iterations were of proximal type (28% of level type).

Figure 1 gives performance profiles [6] of the four solvers over 204 instances of `Unitoy`. The top graphic considers the CPU time, and the bottom one considers the number of oracle calls (iterations). 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 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).

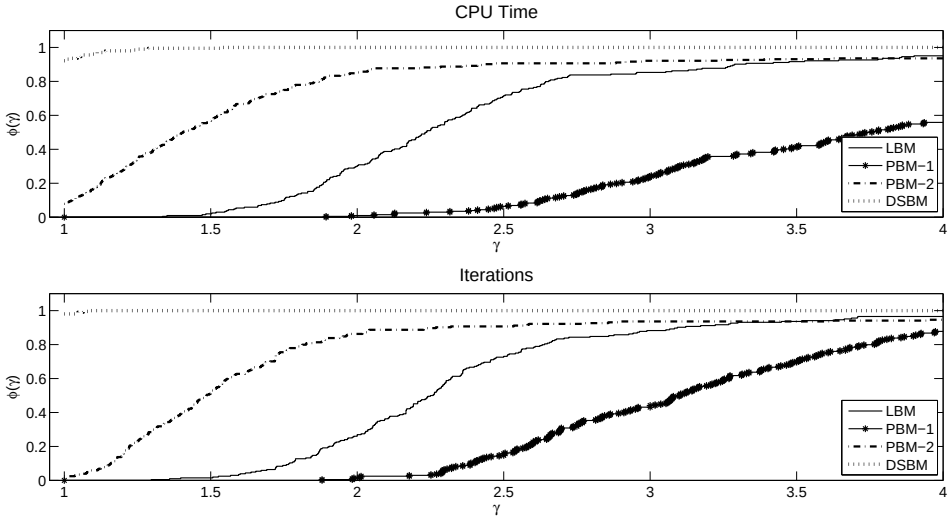

FIGURE 1. Performance profile of the four solvers over 204 instances of `Unitoy`.

We observe that DSBM required less oracle calls in approximately 98% of the 204 instances, followed by PBM-1 (2%). DSBM is also more robust: it achieves $\phi(\gamma) = 1$ for lower values of γ. For this type of problem, PBM-2 and LBM are more robust than PBM-1.

5.2. **Classical unconstrained nonsmooth test problems.** In this subsection we consider some typical functions for nonsmooth optimization benchmarking, such as `MaxQuad` [3, p. 131], `BadGuy` [12, p. 227, Vol. II], `TR48` [12, p. 21, Vol. II] and others. All the problems are unconstrained and have known optimal values. We refer to [22] for more information on these test problems.

Tables 3-5 report on results obtained by the four solvers on this type of problems, using default dimensions and starting points; they would be varied later on. Note that there are some failures (1000 iterations before satisfaction of a stopping test).

	# oracle calls				CPU time in seconds			
Problem	LBM	PBM-1	PBM-2	DSBM	LBM	PBM-1	PBM-2	DSBM
TR48	223	452	347	227	8.3	16.7	13.0	8.6
MaxQuad	99	248	480	87	1.7	4.7	11.4	1.0
Ury	69	87	153	57	1.2	1.1	2.6	0.9
CPS	144	237	1000	81	10.4	5.2	27.3	1.0
TiltedMax	74	19	58	111	1.3	0.2	0.7	1.4
Check	46	123	68	54	0.9	1.6	0.8	0.7
NK	60	85	47	63	0.9	1.3	0.6	1.0
BadGuy	340	1000	1000	1000	5.8	12.9	12.6	17.4
Sum	1055	2251	3153	1680	30.5	43.7	69.0	32.0

TABLE 3. Total number of oracle calls and CPU time.

As the stopping tests for some of the solvers are (in part) of different nature, it is important to compare the quality of the obtained solutions, which is possible in this battery of problems since the optimal values are known (see, e.g., [22]). Table 4 shows the true optimal value of each problem (column $f^{\texttt{inf}}$) and the values obtained by the four solvers at termination, with six decimal digits. In bold are the best obtained values.

Problem	LBM	PBM-1	PBM-2	DSBM	$f^{\texttt{inf}}$
TR48	-638564.825917	-638564.999595	-638564.999681	**-638564.999810**	-638565
MaxQuad	**-0.841408**	**-0.841408**	**-0.841408**	**-0.841408**	-0.841408
Ury	500.000262	**500.000000**	500.000001	500.000023	500
CPS	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0
TiltedMax	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0
NK	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0
BadGuy	-2047.999365	-2047.999965	-2046.093532	**-2048.000000**	-2048

TABLE 4. Best objective function values.

We can conclude from Table 4 that the quality of solutions obtained by the doubly stabilized method does not deteriorate because of the additional stopping test, i.e., this test does not make the method stop prematurely. In fact, there is only one case when DSBM did not stop with a solution at least as good as the best of the other solvers, while this number is greater for the other solvers (although the overall differences are not too significant in this respect). As additional information, we comment that due to the relative optimality gap stopping test, for large $|f^{\texttt{inf}}|$ (and thus $f(\hat{x}_k)$) the solvers LBM and DSBM are more likely to terminate after less iterations. Consider, for instance, `BadGuy` (which, incidentally, is a difficult problem): its optimal value is -2048. LBM solver achieved -2047.999365 in 340 iterations. The other solvers terminated by the maximal number of iterations. However, DSBM did achieve the best function value (with six digits of precision of the optimal); but it did not quite satisfy the stopping test. That said, we verified that after a few more iterations the stopping test at Step 3 of Algorithm 1 would be satisfied.

For `CPS` the optimal value is zero. All solvers achieved the optimal value with 6 digits of precision. However, PBM-2 did not terminate with the stopping test.

The CPU time reduction provided by DSBM when compared to PBM-1 and PBM-2 is, respectively, 27% and 53%. For this set of problems, DSBM was slower than LBM in around 5%.

	# decent steps				# empty level sets		# level steps
Problem	LBM	PBM-1	PBM-2	DSBM	LBM	DSBM	DSBM
TR48	8	100	130	113	1	1	46
MaxQuad	23	34	171	29	0	0	56
Ury	20	15	77	23	14	4	22
CPS	66	81	86	26	1	0	41
TiltedMax	36	12	26	53	7	3	22
Check	19	66	41	27	12	3	20
NK	29	45	24	30	21	12	28
BadGuy	287	17	22	151	3	0	52
Sum	488	370	577	452	59	23	287

TABLE 5. Total number of descent and level steps, and of empty level sets.

Some additional comments on the data in Table 5.

LBM: 46% of all iterations are descent steps; the level set was empty in 2.9% of all iterations.

PBM-1: 16% of all iterations are descent steps.

PBM-2: 18% of all iterations are descent steps.

DSBM: 27% of all iterations are descent steps; the level set was empty in 2.6% of iterations; about 83% of all iterations were of the proximal type (17% of level type).

We next consider the problems above varying their dimensions (up to $n = 50$), and randormly generating starting points with components in $[-1, 1]$. Overall, 83 instances are obtained. Table 6 reports results for this battery of problems.

	LBM	PBM-1	PBM-2	DSBM
CPU Time	382	429	578	277
# oracle calls	19979	17714	25414	13252
# descent steps	11129	6795	12625	4573
# empty level sets	600	-	-	174
# level steps	-	-	-	6862

TABLE 6. Sum over 83 instances (varying dimensions and starting points).

DSBM was 27% faster than LBM, 35% faster than PBM-1 and 52% faster than PBM-2.

We complement Table 6 with the following information.

LBM: 56% of all iterations are descent steps; the level set was empty in 3% of all iterations.

PBM-1: 38% of all iterations are descent steps.

PBM-2: 49% of all iterations are descent steps.

DSBM: 35% of all iterations are descent steps; the level set was empty in 1.3% of iterations; about 48% of all iterations were of the proximal type (52% of level level).

We give in Figure 2 performance profiles of the four solvers over the 83 instances. Note that in some cases the maximum number of iterations (1000) was reached; this is taken into account: to measure robustness those instances were considered as not solved (notice that lines on Figure 2 do not reach the value 1).

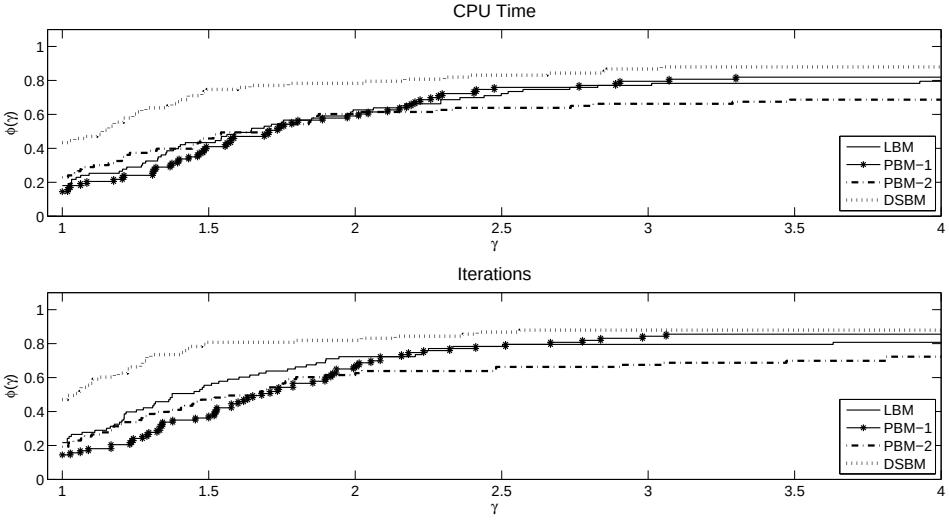

FIGURE 2. Performance profile of the four solvers over 83 instances.

As the line of DSBM is above the other lines, it was the more robust solver on the problems considered. Moreover, it was the fastest in about 40% of the (solved) instances, followed by PBM-2 (21%). Although PBM-1 was the solver which performed more oracle calls for the solved instances (see the lower value $\phi(1)$ in the bottom graphic of Figure 2), it was the second more robust solver: PBM-1 and DSBM coincide at $\phi(4)$ (bottom graphic).

5.3. **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,

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

where

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

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

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

Table 7 shows the total number of oracle calls and CPU time for solving (successively) all the twenty instances of each of the 10 problems (in total, 200) by the four methods. DSBM

	# oracle calls				CPU time in seconds			
Problem	LBM	PBM-1	PBM-2	DSBM	LBM	PBM-1	PBM-2	DSBM
1	392	359	355	398	163	146	141	158
2	213	268	269	215	102	119	119	105
3	608	1117	732	528	345	662	406	299
4	489	618	627	461	201	256	261	190
5	366	536	528	349	157	242	240	148
6	855	1256	1038	661	495	764	620	389
7	1616	1949	4258	1423	666	895	1913	570
8	2176	2678	5404	1994	987	1269	2333	963
9	1006	1640	1487	867	613	1080	945	531
10	2236	2384	2909	2327	922	970	1198	959
Sum	9957	12805	17607	9223	4651	6402	8177	4313

TABLE 7. Total number of oracle calls and CPU time: sum over 20 instances.

is the fastest solver, followed by LBM. Recall that for the unconstrained `Unitoy` problem the second fastest solver was PBM-2. In part this confirms the earlier comments that for constrained problems level methods might be preferable. When compared to LBM, PBM-1 and PBM-2, the solver DSBM reduced the CPU time needed to solve all the 200 instances in about 7% , 32% and 47%, respectively. Reduction of oracle calls was approximately 7%, 28% and 47%, respectively.

Information on descent, level steps, and empty level sets, is given in Table 8. Additional information is as follows.

LBM: 24% of all iterations are descent steps; the level set was empty in 30% of all iterations.

PBM-1: 23% of all iterations are descent steps.

	# decent steps				# empty level sets		# level steps
Problem	LBM	PBM-1	PBM-2	DSBM	LBM	DSBM	DSBM
1	167	164	162	185	309	151	167
2	95	100	103	91	310	197	108
3	147	266	260	166	311	54	304
4	217	199	205	200	311	155	170
5	145	195	186	141	305	157	151
6	201	298	447	233	300	64	332
7	391	362	2880	644	302	25	470
8	345	403	4120	782	281	21	808
9	216	353	540	334	304	41	388
10	487	652	1418	1112	249	12	790
Sum	2411	2992	10321	3888	2982	877	3688

TABLE 8. Total number of descent and level steps, and of empty level sets: sum over 20 instances.

PBM-2: 58% of all iterations are descent steps.
DSBM: 42% of all iterations are descent steps; the level set was empty in 10% of iterations; about 60% of all iterations were of the proximal type (40% of level type).

When compared to the previous results for unconstrained problems, the level set is empty much more often for (this class of) constrained problems. Once again, DSBM performed more proximal than level iterations. However, the percentage of level iterations increased, when compared to `Unitoy` by about 28%.

We next consider Algorithm 1 where, on line 22, the lower bound f_{k+1}^{low} is set by solving LP (19). We shall call this method DSBM-LP. As mentioned above, this updating rule is applicable if $\mathcal{X}$ is bounded (which is the case here), or if a lower bound for $f^{\texttt{inf}}$ is available. Solving (19) at each iteration to update the lower bound f_{k+1}^{low} has its cost, of course. But it pays off, at least for the kind of problems under consideration here, where the objective function is more difficult to evaluate than solving LP (19). Specifically, we observed that the solver DSBM-LP performed 10.8% less iterations than its counterpart DSBM; 44.3% of all iterations gave descent steps; 34.4% of all iterations were level steps. Recall that solving LP (19) prevents empty level sets.

In Figure 3 we give performance profiles of the five solvers over the 200 instances. We conclude from Figure 3 that among the main four solvers, DSBM used less oracle calls (and CPU time) in approximately 60% of the 200 different problems, followed by LBM (30%). Solvers PBM-1 and PBM-2 are comparable, PBM-1 being slightly more robust than PBM-2. Furthermore, we observe that for this type of problems, the variant DSBM-LP is yet faster and more robust than DSBM.

To finalize our numerical experiments, we consider an inexact oracle for the same ten families of two-stages stochastic linear programs. The inexact oracle here means the following: for each given $x \in \mathcal{X}$ and each instance with N scenarios, the optimization problems that compute $Q(x, h_i)$ were solved only for $i = 1, 2, \ldots, N_{small}$ scenarios, with $N_{small} = \min\{0.67N, 50\}$. The values of $Q(x, h_i)$ for $i = N_{small} + 1, \ldots, N$ were approximated as in [19], taking vectors that are feasible for the dual of the problem that computes $Q(x, h_i)$.

We mention that solver LBM is not suitable for dealing with inexact oracles. We thus used the inexact extension of LBM recently introduced in [18].

For each solver and each family of problems we give in Figure 4 the total number of oracle calls and CPU time required to solve all the twenty instances, using both exact and inexact

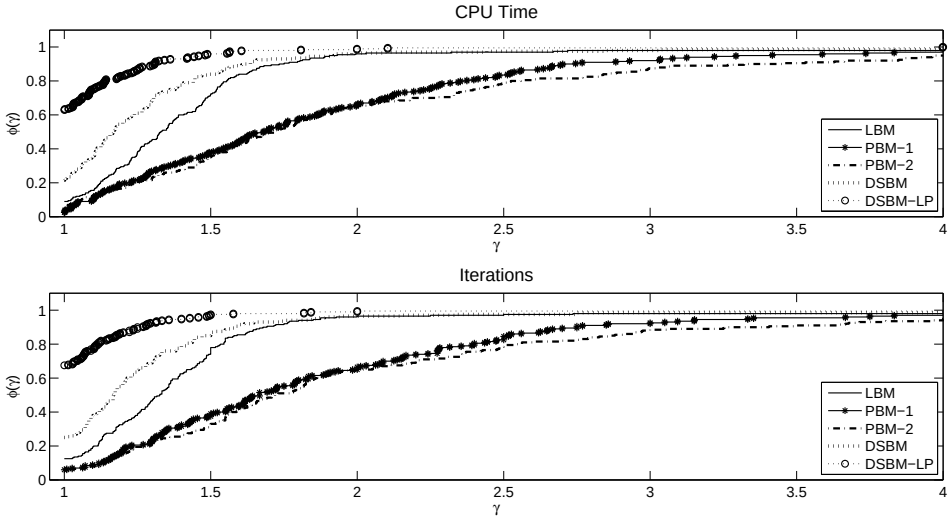

FIGURE 3. Performance profile: 200 instances of two-stage stochastic problems.

oracles. For the inexact oracle, the errors for the optimal values are comparable among the

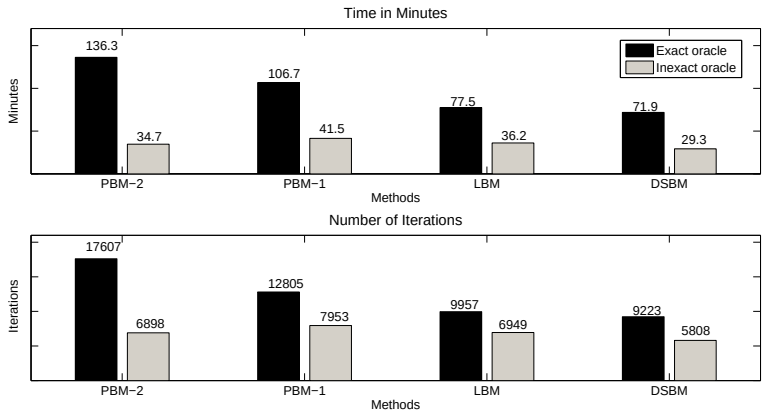

FIGURE 4. Total number of oracle calls and CPU time. Exact and inexact oracles.

four solvers: all the errors are lower than 1%. DSBM again performed better than the other solvers in this inexact case. Figure 4 shows that DSBM is the fastest solver, being able to solve (approximately) all the 200 instances with the smallest number of oracle calls compared to LBM, PBM-1 and PBM-2. We stress that the values for the exact oracle in Figure 4 are precisely the ones at the last line in Table 7.

5.4. **Summary over all the problems considered.** We now give in Figure 5 the performance profiles of the four main solvers over all the 487 different problems with exact oracle in Sections 5.1–5.3. We observe that DSBM is the fastest solver in approximately 70% of the cases, and is more robust than the other solvers.

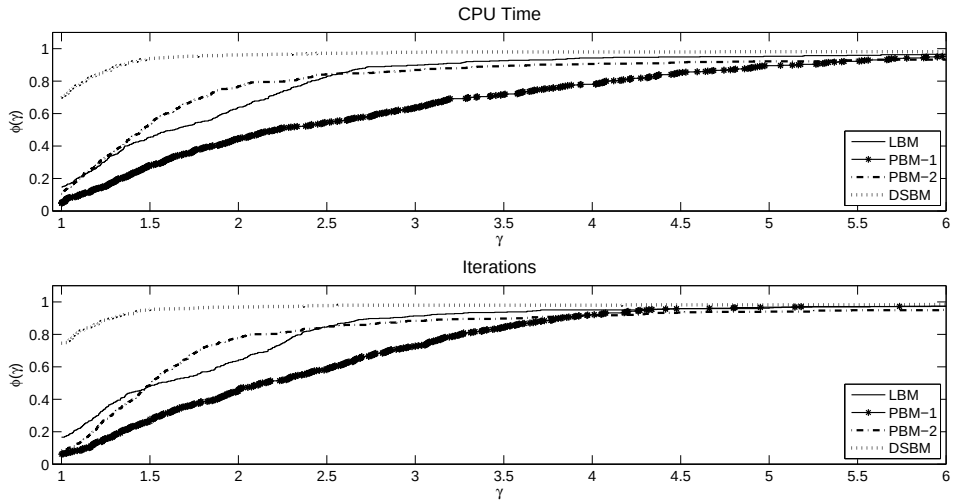

FIGURE 5. Performance profile on 487 problems with exact oracle in Sections 5.1–5.3.

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, something not present in the proximal bundle methods.

The algorithm appears to perform well in computation, as validated in Section 5, where almost seven hundred instances of various types of problems were considered (487 with exact oracle and 200 with inexact oracle). 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.

Acknowledgments

The authors would like to acknowledge helpful comments of Claudia Sagastizábal.

References

[1] A. Astorino, A. Frangioni, M. Gaudioso, and E. Gorgone, *Piecewise-quadratic approximations in convex numerical optimization*, SIAM Journal on Optimization, 21 (2011), pp. 1418–1438.

[2] A. Ben-Tal and A. Nemirovski, *Non-euclidean restricted memory level method for large-scale convex optimization*, Math. Program., 102 (2005), pp. 407–456.

[3] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical Optimization. Theoretical and Practical Aspects*, Universitext, Springer-Verlag, Berlin, 2006. Second edition, xiv+490 pp.

[4] U. Brannlund, K. C. Kiwiel, and P. O. Lindberg, *A descent proximal level bundle method for convex nondifferentiable optimization*, Operations Research Letters, 17 (1995), pp. 121 – 126.

[5] J. Y. B. Cruz and W. Oliveira, *Level bundle-like algorithms for convex optimization*. Submitted for publication, 2012.

[6] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[7] C. Fábián, *Bundle-type methods for inexact data*, Central European Journal of Operations Research, 8 (special issue, T. Csendes and T. Rapcsk, eds.) (2000), pp. 35–55.

[8] A. Frangioni, *Generalized bundle methods*, SIAM Journal on Optimization, 13 (2002), pp. 117–156.

[9] J. L. Goffin, A. Haurie, and J. P. Vial, *Decomposition and nondifferentiable optimization with the projective algorithm*, Management Science, 38 (1992), pp. 284–302.

[10] J.-L. Goffin and J.-P. Vial, *Interior points methods for nondifferentiable optimization*, In Operations Research Proceedings 1997 (Jena), (1998), pp. 35–49.

[11] M. Hintermüller, *A proximal bundle method based on approximate subgradients*, Computational Optimization and Applications, 20 (2001), pp. 245–266.

[12] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, no. 305-306 in Grund. der math. Wiss, Springer-Verlag, 1993. (two volumes).

[13] K. C. Kiwiel, *Proximity control in bundle methods for convex nondiferentiable minimization*, Mathematical Programming, 46 (1990), pp. 105–122.

[14] ———, *Proximal level bubdle methods for convex nondiferentiable optimization, saddle-point problems and variational inequalities*, Math. Program., 69 (1995), pp. 89–109.

[15] ———, *A proximal bundle method with approximate subgradient linearizations*, SIAM Journal on Optimization, 16 (2006), pp. 1007–1023.

[16] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov, *New variants of bundle methods*, Math. Program., 69 (1995), pp. 111–147.

[17] C. Lemaréchal and C. Sagastizábal, *Variable metric bundle methods: From conceptual to implementable forms*, Mathematical Programming, 76 (1997), pp. 393–410. 10.1007/BF02614390.

[18] J. Malick, W. Oliveira, and S. Zaourar, *Nonsmooth optimization using uncontrolled inexact information*. Submitted for publication, 2013.

[19] W. Oliveira and C. Sagastizábal, *Level bundle methods for oracles with on-demand accuracy*. Submitted for publication, 2013.

[20] W. Oliveira, C. Sagastizábal, and C. Lemaréchal, *Bundle methods in depth: a unified analysis for inexact oracles*. Submitted for publication, 2013.

[21] W. Oliveira, C. Sagastizábal, and S. Scheimberg, *Inexact bundle methods for two-stage stochastic programming*, SIAM Journal on Optimization, 21 (2011), pp. 517–544.

[22] C. Sagastizábal, *Composite proximal bundle method*, Mathematical Programming, (2012). 10.1007/s10107-012-0600-5.

[23] C. Sagastizbal and M. Solodov, *An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter*, SIAM Journal on Optimization, 16 (2005), pp. 146–169.

[24] H. Schramm and J. Zowe, *A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results*, SIAM Journal on Optimization, 2 (1992), pp. 121–152.

[25] M. Solodov, *On approximations with finite precision in bundle methods for nonsmooth optimization*, Journal of Optimization Theory and Applications, 119 (2003), pp. 151–165.

[26] S. Zaourar and J. Malick, *Prices stabilization for inexact unit-commitment problems*. In preparation, 2012.