Math. Program., Ser. B (2009) 116:297–320
DOI 10.1007/s10107-007-0123-7

FULL LENGTH PAPER

A bundle-filter method for nonsmooth convex constrained optimization

Elizabeth Karas · Ademir Ribeiro · Claudia Sagastizábal · Mikhail Solodov

Received: 28 June 2005 / Accepted: 26 April 2006 / Published online: 28 April 2007

Abstract For solving nonsmooth convex constrained optimization problems, we propose an algorithm which combines the ideas of the proximal bundle methods with the filter strategy for evaluating candidate points. The resulting algorithm inherits some attractive features from both approaches. On the one hand, it allows effective control of the size of quadratic programming subproblems via the compression and aggregation techniques of proximal bundle methods. On the other hand, the filter criterion for accepting a candidate point as the new iterate is sometimes easier to satisfy than the usual descent condition in bundle methods. Some encouraging preliminary computational results are also reported.

Dedicated to Alfred Auslender on the occasion of his 65th birthday.

This work has been supported by CNPq-PROSUL program. Claudia Sagastizábal was also supported by CNPq, by PRONEX–Optimization, and by FAPERJ. Mikhail Solodov was supported in part by CNPq Grants 300734/95-6 and 471780/2003-0, by PRONEX-Optimization, and by FAPERJ.
Claudia Sagastizábal is on leave from INRIA-Rocquencourt, BP 105, 78153 Le Chesnay, France.

E. Karas · A. Ribeiro
Universidade Federal do Paraná, Departamento de Matemática,
CP 19081, 81531-980 Curitiba, PR, Brazil
e-mail: karas@mat.ufpr.br

A. Ribeiro
e-mail: ademir@mat.ufpr.br

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

C. Sagastizábal
e-mail: sagastiz@impa.br

Keywords Constrained optimization · Nonsmooth convex optimization · Bundle methods · Filter methods

Mathematics Subject Classification (2000) 90C30 · 65K05 · 49D27

1 Introduction

We consider the optimization problem

$$\begin{array}{l} \text{minimize} \quad f(x) \\ \text{subject to} \quad c(x) \leq 0, \end{array} \tag{1}$$

where $f, c : \mathbb{R}^n \to \mathbb{R}$ are convex functions, in general nondifferentiable. It is very appropriate to dedicate this work to Professor Alfred Auslender, who made numerous profound contributions to the theory and numerical methods for nonsmooth optimization. In fact, to the best of our knowledge, Auslender [1] was the first to use the so-called *improvement function* [defined by (3) below] to construct numerical methods for solving (1). This function will also play a central role in our algorithmic development.

We note that there is no loss of generality in formulating problem (1) with only one scalar constraint: if necessary, c can be defined as the pointwise maximum of finitely many convex functions, thus covering the case of multiple inequality constraints. We assume that an *oracle* is available, which for any given $x \in \mathbb{R}^n$ computes the values $f(x)$ and $c(x)$, and *one* subgradient for each of the functions, i.e., some $g_f \in \partial f(x)$ and some $g_c \in \partial c(x)$. As usual in nonsmooth optimization, we do not assume that there is any control over which particular subgradients are computed by the oracle.

Any nonlinear programming algorithm must deal with two different (and possibly conflicting!) criteria related, respectively, to optimality and to feasibility. Optimality is naturally measured by the objective function f; feasibility is typically measured by penalization of constraint violation, for example, by the function $c^+ : \mathbb{R}^n \to \mathbb{R}_+$, where

$$c^+(x) = \max\{0, c(x)\}.$$

Both measures must be optimized eventually, and the algorithm should follow a certain balance between the two criteria, at every step of the iterative process. Traditionally, this task was approached by minimizing a weighted sum of f and c^+, i.e., by minimizing a penalty function. In the context of bundle methods for nonsmooth optimization, the use of penalty functions has been explored in [14,17]. More recently, for smooth problems an alternative idea, called *filter* strategy, has been introduced in [6]. Global convergence of various filter-based methods can be found in [4,10].

Filter algorithms define a *forbidden region* in a clever way, by memorizing optimality and infeasibility pairs $(f(x^i), c^+(x^i))$ from well chosen past iterations i, and then avoiding points dominated by these pairs according to the usual Pareto domination rule:

"x dominates y if and only if $f(y) \geq f(x)$ and $c^+(y) \geq c^+(x)$".

Roughly speaking, a candidate point is accepted by the filter as the next iterate whenever it is not dominated by any of the previous iterates (as a practical matter, a slightly stronger condition is required: with respect to every pair in the filter, at least one of the measures should be reduced by a sufficient margin, measured in terms of c^+).

At least part of the motivation for filter algorithms consists in avoiding the need to estimate a suitable value of penalty parameter, which is often a delicate task. Also, if a large value of the parameter is required to guarantee exactness of a given penalty function, then numerical difficulties may arise.

The use of filter strategy in the nonsmooth case is, up to now, limited to [5]. The method of [5] is based on solving linear programming subproblems obtained by replacing the objective and constraint functions by their respective cutting-planes models, subject to a box-constrained trust region. Specifically, if y^i, $i = 1, \ldots, k$, are the points observed prior to the kth iteration, [5] solves the following subproblem:

$$\begin{array}{ll} \text{minimize} & \eta \\ \text{subject to} & \eta \geq f(y^i) + \langle g_f^i, y - x^k \rangle, \quad i = 1, \ldots, k, \\ & 0 \geq c(y^i) + \langle g_c^i, y - x^k \rangle, \quad i = 1, \ldots, k, \\ & \|y - x^k\|_\infty \leq \rho_k, \end{array} \tag{2}$$

where x^k is the last serious iterate, $g_f^i \in \partial f(y^i)$, $g_c^i \in \partial c(y^i)$ and $\rho_k > 0$ is the trust region parameter. We note that, as stated, the method of [5] does not guarantee that the number of affine constraints in the subproblems can be kept smaller than a given desired bound, even if "inactive cuts" are discarded at each iteration. Without this feature, a method cannot be guaranteed to be practical.

Another approach for solving nonsmooth constrained problems, which uses neither penalty functions nor filters, has recently been proposed in [30]. Given a current ("serious") iterate $x \in \mathbb{R}^n$, the method of [30] applies the usual unconstrained bundle technique to the *improvement function* $h_x(\cdot)$, where

$$\mathbb{R}^n \ni y \mapsto h_x(y) := \max\{f(y) - f(x),\, c(y)\}, \tag{3}$$

until descent for this function with respect to the value $h_x(x)$ is achieved (the so-called serious step condition). At this time, the corresponding candidate is accepted as the next (serious) iterate and the algorithm proceeds, working with the new improvement function, corresponding to the new (serious) iterate. This algorithm builds on the theory of the well-developed unconstrained bundle methods [3,11,13] and in particular, it allows effective control of (the quadratic programming) subproblem sizes by suitably revised compression and aggregation techniques.

In this paper, we propose a new approach which combines some of the ideas of filter methods and of the bundle method of [30]. More specifically, instead of using the usual descent condition to evaluate candidate points, we shall employ the filter criterion. The resulting algorithm preserves the possibility of using compression and aggregation techniques. In some parts, the required convergence analysis becomes quite different from [30], as well as from the usual filter analysis [5,10]. Ignoring for

now the aggregation feature, subproblems in our method are based on the cutting-planes model of the improvement function (3) and have the following structure:

$$\begin{array}{lll} \text{minimize} & \eta + \mu_k \|y - x^k\|^2/2 & \\ \text{subject to} & \eta \geq f(y^i) - f(x^k) + \langle g_f^i, y - x^k \rangle, & i \text{ s.t.} f(y^i) - f(x^k) \geq c(y^i), \\ & \eta \geq c(y^i) + \langle g_c^i, y - x^k \rangle, & i \text{ s.t.} f(y^i) - f(x^k) < c(y^i), \end{array} \tag{4}$$

where $\mu_k > 0$ is the proximal parameter, x^k is the last serious iterate, and y^i, $i = 1, \ldots, k$, are the previously observed points. When compared to (2), note that apart from using the proximal term instead of a trust region, subproblems (4) are also structurally different. Not only the number of constraints in (4) is twice less than in (2), but all the constraints are, in fact, different (the "f-constraints" are shifted by the value $f(x^k)$, while the "c-constraints" contain the variable η).

Some other bundle-type methods for constrained nonsmooth optimization, not cited above, are [9,13,16,18,26,27]. A more detailed discussion and comparison of these methods can be found in [30]. As for the improvement function (3), its use as a theoretical tool in convergence analysis of bundle-type methods can be traced back to [26], see also [13]. However, in none of these works the improvement function is used in the algorithms themselves. As already mentioned above, the first (and apparently the only, apart from the recent proposal [30]) work where the improvement function is directly involved in constructing numerical methods for nonsmooth optimization is Auslender's paper [1]. The methods of [1] are of inexact proximal point type and can be considered as predecessors of [30] and of the current proposal. Indeed, our methods can be thought of as constructive realizations of inexact proximal point algorithms of [1], where the requirement of solving proximal subproblems with some prescribed precision is replaced by making one descent step with respect to the proximal center in [30] or one filter-acceptable step in the method introduced below.

The paper is organized as follows. Our constrained proximal bundle filter method is described in Sect. 2. Sects. 3 and 4 contain, respectively, convergence analysis and numerical results.

Our notation is fairly standard. The Euclidean inner product in $\mathbb{R}^n$ is denoted by $\langle x, y \rangle = \sum_{j=1}^n x_j y_j$, and the associated norm by $\|\cdot\|$. The positive-part function is denoted by $x^+ := \max\{x, 0\}$. For a set X in $\mathbb{R}^n$, conv X denotes its convex hull. Given some $\varepsilon \geq 0$, we denote the ε-subdifferential of a convex function h at the point $x \in \mathbb{R}^n$ by $\partial_\varepsilon h(x) = \{g \in \mathbb{R}^n \mid h(y) \geq h(x) + \langle g, y - x \rangle - \varepsilon\}$, with $\partial_0 h(x) = \partial h(x)$ being the usual subdifferential.

Before proceeding, we recall some useful properties of the improvement function. Directly by the definition (3), we have that

$$\partial h_x(y) = \begin{cases} \partial f(y) & \text{if } f(y) - f(x) > c(y), \\ \text{conv}\{\partial f(y) \bigcup \partial c(y)\} & \text{if } f(y) - f(x) = c(y), \\ \partial c(y) & \text{if } f(y) - f(x) < c(y). \end{cases} \tag{5}$$

In addition,

$$h_x(x) = c^+(x) = \max\{c(x), 0\} \quad \text{for all } x \in \mathbb{R}^n.$$

In our convergence analysis, we shall assume that the Slater constraint qualification [25] holds, i.e., there exists $x \in \mathbb{R}^n$ such that $c(x) < 0$. Under this assumption, the following holds (e.g., [13, Lemma 2.16, p.17]).

Theorem 1 *Suppose that the Slater constraint qualification is satisfied for* (1)*. Then the following statements are equivalent*:

(i) $\bar{x}$ *is a solution to* (1);
(ii) $\min\{h_{\bar{x}}(y) \mid y \in \mathbb{R}^n\} = h_{\bar{x}}(\bar{x}) = 0$;
(iii) $0 \in \partial h_{\bar{x}}(\bar{x})$*, i.e.,* $0 \in \partial\varphi(\bar{x})$*, where* $\varphi(\cdot) := h_{\bar{x}}(\cdot)$.

2 Description of the algorithm

The description of the algorithm is quite involved, which seems to be unavoidable given the relative complexity both of bundle methods and of filter methods. We shall start with conceptual comments, passing to technical details gradually.

The main ideas of the method are as follows. Given the last serious iterate x^k (the starting point x^0 is regarded as the first serious iterate), we generate candidate points by iterating with the unconstrained proximal bundle method applied to the function $h_{x^k}(\cdot)$ until the next serious iterate x^{k+1} is obtained. However, to decide whether some candidate point can be declared the next serious iterate, we use a filter strategy instead of the usual descent condition in bundle methods. Once the new serious iterate x^{k+1} has been computed, we start working with the new function $h_{x^{k+1}}(\cdot)$. Note, however, that because the sequence $\{f(x^k)\}$ need not be monotone, at this stage some special care should be taken to obtain a valid cutting-planes approximation for the new improvement function, as will be shown later.

2.1 The bundle technique

The bundle subroutine of our algorithm consists in performing null steps of the unconstrained proximal bundle method applied to the function

$$h_k(\cdot) := h_{x^k}(\cdot) = \max\left\{f(\cdot) - f(x^k),\ c(\cdot)\right\},$$

where x^k is a given serious iterate, until an acceptable new serious iterate x^{k+1} is generated (our criteria for declaring a serious step will be specified later).

Generating candidate points. Let ℓ be the current iteration index, where iterations include both the serious steps x^k and the null steps y^ℓ (candidate points which were not declared serious). In particular, $\{x^k\} \subset \{y^\ell\}$, with $x^0 = y^0$. Let $k = k(\ell)$ be the index of the last serious iterate preceding iteration ℓ. Given a proximal parameter $\mu_\ell > 0$, to find a new serious iterate x^{k+1}, our bundle subroutine generates candidate points y^ℓ by solving quadratic programming problems derived from the problem

$$\min_{y\in\mathbb{R}^n} \psi_\ell(y) + \frac{1}{2}\mu_\ell\|y - x^k\|^2, \tag{6}$$

where $\psi_\ell(\cdot)$ is a cutting-planes approximation for the improvement function $h_k(\cdot)$. Subproblem (6) has the structure described by (4). Introducing its dual, it can be solved as a quadratic program with simplicial feasible set. This structure is effectively exploited by specialized solvers, [8,15].

The following characterization of the solution to (6) is well-known, (e.g., [3, Lemma 9.8]).

Lemma 1 *The unique solution y^ℓ to (6) satisfies the following relations*:

(i) $y^\ell = x^k - \frac{1}{\mu_\ell}\hat{g}^\ell$, *where* $\hat{g}^\ell \in \partial\psi_\ell(y^\ell)$;
(ii) $\hat{g}^\ell \in \partial_{\hat{\varepsilon}_\ell^k} h_k(x^k)$, *where* $0 \le \hat{\varepsilon}_\ell^k := h_k(x^k) - \psi_\ell(y^\ell) - \mu_\ell\|y^\ell - x^k\|^2$.

To define the cutting-planes approximation, the subroutine accumulates information from past points y^i, $i \le \ell$, in a bundle $\mathcal{B}_\ell$ formed by two subsets. The first subset, denoted by $\mathcal{B}_\ell^{oracle}$, collects information corresponding to (some of the) previously computed oracle data:

$$f_i = f(y^i),\ g_f^i \in \partial f(y^i) \quad \text{and} \quad c_i = c(y^i),\ g_c^i \in \partial c(y^i) \quad i \le \ell.$$

The second subset, denoted by $\mathcal{B}_\ell^{agg}$, refers to information of the form $(\hat{\varepsilon}_\ell^k, \hat{g}^\ell)$, defining the so-called *aggregate function* (see [3, Chap. 9]):

$$l_{k,\ell}(y) := h_k(x^k) - \hat{\varepsilon}_\ell^k + \langle \hat{g}^\ell, y - x^k\rangle.$$

Just as ψ_ℓ, this aggregate function is also a lower approximation to h_k. This function is fundamental for keeping the number of elements in the bundle [and hence, the size of subproblems (6)] computationally manageable, as we discuss next.

Compressing the bundle. Whenever the number of elements in the bundle reaches some chosen upper bound, the bundle has to be *compressed*, i.e, some elements need to be deleted, without impairing convergence of the algorithm. Actually, any number of elements can be deleted, provided we include in the subset $\mathcal{B}_\ell^{agg}$ information about the aggregate function $l_{k,\ell}$, and in the subset $\mathcal{B}_\ell^{oracle}$ information about the cutting-plane at the last point y^ℓ, as is done in Step 7 of Algorithm 1 below. In fact, the total number of elements can be kept as small as two, at every step of the method. That said, the less information is used, the slower is convergence. We refer the reader to [3, Chap.9] for a more detailed discussion of aggregation in bundle methods.

To see how aggregation should be done in our setting, we consider the compact representation for the oracle bundle data, which makes use of *linearization errors* at y^i with respect to x^k. In particular, we define

$$\begin{aligned} e_f^{k,i} &:= f(x^k) - f_i - \langle g_f^i, x^k - y^i\rangle, \\ e_c^{k,i} &:= c(x^k) - c_i - \langle g_c^i, x^k - y^i\rangle, \end{aligned} \tag{7}$$

which are nonnegative, by the convexity of f and c.

In [30, Lemma 3.1] it is shown that for elements in $\mathcal{B}_\ell^{oracle}$ choosing

$$\begin{cases} e_i^k := {e_f}^{k,i} + c^+(x^k) & \text{and} \quad g_{h_k}^i := g_f^i, \quad \text{if } f_i - f(x^k) \geq c_i, \\ e_i^k := {e_c}^{k,i} + c^+(x^k) - c(x^k) & \text{and} \quad g_{h_k}^i := g_c^i, \quad \text{if } f_i - f(x^k) < c_i, \end{cases} \tag{8}$$

where the linearization errors are shifted, guarantees that

$$g_{h_k}^i \in \partial_{e_i^k} h_k(x^k) \quad \text{with } e_i^k \geq 0.$$

With this notation, bundle subsets can be rewritten in the compact form

$$\begin{aligned} \mathcal{B}_\ell^{oracle} &\subseteq \bigcup_{i<\ell} \left\{ \left(f_i, c_i, {e_f}^{k,i}, {e_c}^{k,i}, g_f^i \in \partial_{{e_f}^{k,i}} f(x^k), g_c^i \in \partial_{{e_c}^{k,i}} c(x^k) \right) \right\}, \\ \mathcal{B}_\ell^{agg} &\subseteq \bigcup_{i<\ell} \left\{ \left(\hat{\varepsilon}_i^k, \hat{g}^i \in \partial_{\hat{\varepsilon}_i^k} h_k(x^k) \right) \right\}. \end{aligned} \tag{9}$$

Furthermore, the cutting-planes function

$$\begin{aligned} \psi_\ell(y) := c^+(x^k) & \\ + \max & \left\{ \max_{i \in \mathcal{B}_\ell^{oracle}} \left\{ -e_i^k + \langle g_{h_k}^i, y - x^k \rangle \right\}, \max_{i \in \mathcal{B}_\ell^{agg}} \left\{ -\hat{\varepsilon}_i^k + \langle \hat{g}^i, y - x^k \rangle \right\} \right\}, \\ k &= k(\ell), \end{aligned} \tag{10}$$

is a correct lower approximation for h_k, see [30].

The bundle subroutine is terminated and y^ℓ is declared to be the next serious step x^{k+1} when it is accepted by the filter strategy described below. Otherwise, the data computed at the new point is added to the bundle, the bundle is compressed if needed, and we proceed to generate a new candidate point.

Updating linearization errors. When a serious step is declared, the model ψ_ℓ has to be properly revised to make sure that a correct lower approximation for h_{k+1} is obtained. In the given setting, a special care is required for the following reason: since $f(x^{k+1}) > f(x^k)$ is perfectly possible, we may have $h_{k+1}(\cdot) \leq h_k(\cdot)$, where the inequality can be strict for some points. This means that a lower approximation for h_k may no longer be valid for h_{k+1}. The correct update of linearization errors which does the job is given by the following formulas, see [30, Lemma 3.2] for a proof.

We update the linearization errors for $i \in \mathcal{B}_{\ell+1}^{oracle}$ according to the relations

$$\begin{aligned} {e_f}^{k+1,i} &= {e_f}^{k,i} + f(x^{k+1}) - f(x^k) + \langle g_f^i, x^k - x^{k+1} \rangle, \\ {e_c}^{k+1,i} &= {e_c}^{k,i} + c(x^{k+1}) - c(x^k) + \langle g_c^i, x^k - x^{k+1} \rangle, \end{aligned} \tag{11}$$

and the aggregate errors for $i \in \mathcal{B}_{\ell+1}^{agg}$ according to the relation

$$\hat{\varepsilon}_i^{k+1} = \hat{\varepsilon}_i^k + c^+(x^{k+1}) - c^+(x^k) + \left(f(x^{k+1}) - f(x^k) \right)^+ + \langle \hat{g}^i, x^k - x^{k+1} \rangle. \tag{12}$$

2.2 Filter strategy

Let parameters $\alpha_f, \alpha_c \in (0, 1)$ be given. These parameters are used to define the forbidden region by shifting the optimality and feasibility values by a certain margin, as described next.

At kth iteration, the filter F_k is formed by the union of pairs

$$(\tilde{f}_i, \tilde{c}_i^+) := \Big(f(x^i) - \alpha_f c^+(x^i), \alpha_c c^+(x^i)\Big), \quad i < k,$$

where indices correspond to (some of) past serious iterations, such that no pair in F_k is dominated (in the Pareto sense) by any other pair.

Given the last serious iterate x^k, at the beginning of the kth iteration the pair

$$(\tilde{f}, \tilde{c}^+) := \Big(f(x^k) - \alpha_f c^+(x^k),\ \alpha_c c^+(x^k)\Big)$$

is *temporarily* introduced into the filter, see Step 1 of Algorithm 1 below. Together with pairs in F_k, this pair defines the current forbidden region $\bar{\mathcal{F}}_k$. Essentially, the bundle subroutine (by generating candidate points) aims at producing a point that is not forbidden. This point will be the new serious iterate x^{k+1}. There are some subtleties to our filter strategy, however. Those subtleties stem from the need to relate the usual filter objects to the natural measure of optimality of a candidate y^ℓ associated with the bundle technique. Specifically, this measure of optimality (recall Lemma 1) is given by

$$\delta_\ell := h_k(x^k) - \left(\psi_\ell(y^\ell) + \frac{1}{2}\mu_\ell \|y^\ell - x^k\|^2\right) = \hat{\varepsilon}_\ell^k + \frac{1}{2\mu_\ell}\|\hat{g}^\ell\|^2. \tag{13}$$

If "infeasibility is bigger than nonoptimality" (in the sense that $c^+(x^k) > m_2\delta_\ell$, where $m_2 \in (0, 1)$, see relation (14) of Algorithm 1), then simply having y^ℓ outside the forbidden region is enough to accept this point as the next serious iterate. But in the situation where the candidate point is "more feasible than optimal" (in the sense that (14) in Algorithm 1 does not hold), an additional (almost-)descent condition on f (in the sense that $f(y^\ell) \le f(x^k) + c^+(x^k) - m_1\delta_\ell$, where $m_1 \in (0, 1)$, see relation (15) in Algorithm 1), is required to declare a serious step. This additional condition seems to be similar in spirit to some other filter methods, e.g., [5].

After having found a new serious iterate, the temporary pair enters the new filter F_{k+1} only if this iteration *did not* produce a decrease in f. Such iterations are called c^+*-iterations*. Serious iterations which did reduce the objective function are referred to as f*-iterations*; see Step 8 of Algorithm 1. Whenever the filter has changed, it is cleaned: every pair dominated by the newly added pair is removed, and the algorithm proceeds.

As usual in filter methods, the algorithm also needs an auxiliary restoration phase (Step 6 of Algorithm 1) to recover from certain situations. We emphasize that situations which require restoration phase in our setting are different in nature from those in filter methods for standard smooth nonlinear programming. In particular, they have nothing

to do with infeasibility of subproblems (our subproblems are always feasible). Those differences are actually natural, having in mind that smooth and nonsmooth cases require significantly different tools. In our setting, the restoration phase is activated in the following case. If we already accumulated enough null steps to satisfy the bundle descent condition for h_k (condition (16) in Algorithm 1), but y^ℓ is still not acceptable, then we compute the next serious iterate by generating a point which is less infeasible than any point in the current filter. This can be done by making a few iterations of an unconstrained bundle method applied to the unconstrained problem of minimizing $c^+(x)$ (possibly even one iteration, if we start from a point with the smallest infeasibility). We emphasize that while apparently indispensable for convergence analysis, the restoration step was actually never needed in our computational experience (reported in Sect. 4).

2.3 Statement of the algorithm

We are now in position to formally state our method.

Algorithm 1 Constrained Proximal Bundle Filter Method (CPBFM)

Step 0. `Initialization.`
Choose parameters $m_1, m_2 \in (0,1)$, filter parameters $\alpha_f, \alpha_c \in (0,1)$, a stopping tolerance $tol \geq 0$, and a maximum bundle size $|\mathcal{B}|_{\max} \geq 2$.
Choose $x^0 \in \mathbb{R}^n$. Set $y^0 := x^0$, and compute (f_0, c_0, g_f^0, g_c^0). Set $k = 0$, $\ell = 1$, ${e_f}^{0,0} := 0$, ${e_c}^{0,0} := 0$ and define the starting bundles $\mathcal{B}_1^{oracle} := \{({e_f}^{0,0}, {e_c}^{0,0}, f_0, c_0, g_f^0, g_c^0)\}$ and $\mathcal{B}_1^{agg} := \emptyset$.
Set $F_0 = \emptyset$, $\mathcal{F}_0 = \emptyset$.

Step 1. `Filter Regions.` Define the temporary pair, the current filter and forbidden region, respectively, as follows:

$$
\begin{aligned}
(\tilde{f}, \tilde{c}^+) &:= (f(x^k) - \alpha_f c^+(x^k),\ \alpha_c c^+(x^k)),\\
\bar{F}_k &:= F_k \cup \{(\tilde{f}, \tilde{c}^+)\},\\
\bar{\mathcal{F}}_k &:= \mathcal{F}_k \bigcup \{x \in \mathbb{R}^n \mid f(x) \geq \tilde{f}, c^+(x) \geq \tilde{c}^+\}.
\end{aligned}
$$

Step 2. `Quadratic Programming Subproblem.`
Having ψ_ℓ defined by (9) and (10), choose $\mu_\ell > 0$ and compute y^ℓ, the solution to (6). Compute the bundle-related quantities

$$
\hat{g}^\ell = \mu_\ell(x^k - y^\ell), \quad \hat{\varepsilon}_\ell^k = c^+(x^k) - \psi_\ell(y^\ell) - \frac{1}{\mu_\ell}\|\hat{g}^\ell\|^2,
$$

$$
\delta_\ell = \hat{\varepsilon}_\ell^k + \frac{1}{2\mu_\ell}\|\hat{g}^\ell\|^2.
$$

Compute the oracle data $(f_\ell, c_\ell, g_f^\ell, g_c^\ell)$ at y^ℓ and define the linearization errors $({e_f}^{k,\ell}, {e_c}^{k,\ell})$ by using (7) written with $i = \ell$.

Step 3. `Stopping test.`
If $\delta_\ell \le tol$, stop.

Step 4. `Filter tests.` If $y^\ell \notin \bar{\mathcal{F}}_k$, and either

$$c^+(x^k) > m_2\delta_\ell \tag{14}$$

or

$$c^+(x^k) \le m_2\delta_\ell \quad \text{and} \quad f(y^\ell) \le f(x^k) + c^+(x^k) - m_1\delta_\ell\,, \tag{15}$$

declare *iterate* y^ℓ *accepted* (serious step) and go to `Bundle Management.`

Step 5. `Bundle descent (restoration) test.` If

$$h_k(y^\ell) \le c^+(x^k) - m_1\delta_\ell, \tag{16}$$

then go to `Restoration Step`. Otherwise, declare a *null step* and go to `Bundle Management`.

Step 6. `Restoration Step.` Compute y^ℓ such that

$$c^+(y^\ell) < \min\left\{\tilde{c}_j^+ \mid (\tilde{f}_j,\, \tilde{c}_j^+) \in \bar{F}_k\right\}.$$

Declare the *iterate* y^ℓ *accepted* (serious step).

Step 7. `Bundle Management.`
Set $\mathcal{B}_{\ell+1}^{oracle} := \mathcal{B}_\ell^{oracle}$ and $\mathcal{B}_{\ell+1}^{agg} := \mathcal{B}_\ell^{agg}$.
If the bundle has reached the maximum bundle size, i.e.,
if $|\mathcal{B}_{\ell+1}^{oracle} \cup \mathcal{B}_{\ell+1}^{agg}| = |\mathcal{B}|_{\max}$, then delete at least two elements from $\mathcal{B}_{\ell+1}^{oracle} \cup \mathcal{B}_{\ell+1}^{agg}$ and insert the aggregate couple $(\hat{\varepsilon}_\ell^k, \hat{g}^\ell)$ in $\mathcal{B}_{\ell+1}^{agg}$.
Append $(e_f{}^{k,\ell}, e_c{}^{k,\ell}, f_\ell, c_\ell, g_f^\ell, g_c^\ell)$ to $\mathcal{B}_{\ell+1}^{oracle}$.
In case of *null step* (iterate y^ℓ was not accepted), set $\ell = \ell + 1$ and go to `Step 2`.

Step 8. `Model adjustment and Filter update (after a serious step).`
Define the next stability center: $\left(x^{k+1}, f(x^{k+1}), c(x^{k+1})\right) := \left(y^\ell, f_\ell, c_\ell\right)$.
Update the linearization errors for $i \in \mathcal{B}_{\ell+1}^{oracle}$ according to (11).
Update the aggregate errors for $i \in \mathcal{B}_{\ell+1}^{agg}$ according to (12).
If $f(x^{k+1}) < f(x^k)$ then

$$F_{k+1} = F_k, \quad \mathcal{F}_{k+1} = \mathcal{F}_k \qquad (f\text{-iteration}),$$

else

$$F_{k+1} = \bar{F}_k, \quad \mathcal{F}_{k+1} = \bar{\mathcal{F}}_k \qquad (c^+\text{-iteration}).$$

In the latter case, remove from the filter every pair dominated by the newly added pair.
Set $k = k + 1$, $\ell = \ell + 1$ and go to Step 1.

For convergence, the proximal parameters would be required to satisfy

$$0 < \underline{\mu} \leq \mu_\ell \leq \overline{\mu} < +\infty,$$

and be nondecreasing on consecutive null steps. Those requirements are somewhat stronger than in standard unconstrained bundle methods (see [3]), but are not significantly different from the computational point of view.

3 Convergence analysis

First note that if $\delta_\ell = 0$ after Step 2 of Algorithm 1, then at Step 3 the algorithm stops. In this case, x^k is solution to (1). Indeed, $\delta_\ell = 0$ implies that $\hat{\varepsilon}_\ell^k = 0$ and $\hat{g}^\ell = 0$. By Lemma 1, it then holds that $0 \in \partial h_k(x^k)$, which means that x^k solves (1), by Theorem 1.

We assume, from now on, that $\delta_\ell > 0$ for all iterations ℓ and that the stopping tolerance is set to zero. After showing in Proposition 2 below that our method is well defined, this would mean that an infinite sequence $\{y^\ell\}$ is generated.

We start with some relations which follow directly from the construction of the algorithm.

Proposition 1 *For Algorithm 1, the following statements hold*:

(i) *Given $k \in \mathbb{N}$, if a serious iterate x^{k+1} is generated then* at least one *of the following two conditions holds*:

$$c^+(x^{k+1}) < \alpha_c \, c^+(x^k),$$
$$f(x^{k+1}) < f(x^k) - \alpha_f \, c^+(x^k).$$

(ii) *If $f(y^\ell) < \tilde{f}_j$ and $c^+(y^\ell) \leq \tilde{c}_j^+$ for some $(\tilde{f}_j, \tilde{c}_j^+) \in \bar{F}_k$, then $y^\ell \notin \bar{\mathcal{F}}_k$.*

(iii) *Given $k \in \mathbb{N}$, $\tilde{c}_j^+ > 0$ for all $j \in \mathbb{N}$ such that $(\tilde{f}_j, \tilde{c}_j^+) \in F_k$.*

(iv) *If $c^+(y^\ell) < \tilde{c}_j^+$ for all $(\tilde{f}_j, \tilde{c}_j^+) \in \bar{F}_k$, then $y^\ell \notin \bar{\mathcal{F}}_k$.*

(v) *Given any $j \in \mathbb{N}$, if there exists a serious iterate x^{j+p}, $p \geq 1$, then $x^{j+p} \notin \mathcal{F}_{j+1}$.*

Proof (i) If the serious iterate was declared by Step 4, then the assertion is a consequence of the fact that $x^{k+1} \notin \bar{\mathcal{F}}_k$. If it was declared by Step 6 then it should hold, in particular, that $c^+(x^{k+1}) < \tilde{c}^+ = \alpha_c \, c^+(x^k)$.

(ii) Take any other pair $(\tilde{f}_i, \tilde{c}_i^+) \in \bar{F}_k$. Since no pair in the filter dominates any other pair, we have that $f(y^\ell) < \tilde{f}_j < \tilde{f}_i$ or $c^+(y^\ell) \leq \tilde{c}_j^+ < \tilde{c}_i^+$. It follows that $y^\ell \notin \bar{\mathcal{F}}_k$.

(iii) The pair $(\tilde{f}, \tilde{c}^+)$ is included in the filter if and only if this iteration is a (serious step) c^+-iteration. If $c^+(x^k) = 0$, then the first relation in item (i) is not possible.

Hence, the second relation holds. Therefore, $f(x^{k+1}) < f(x^k)$. In particular, the iteration k is an f-iteration. It follows that an inclusion into the filter can occur only when $c^+(x^k) > 0$, in which case $\tilde{c}^+ > 0$, implying the assertion.

(iv) The assertion is obvious, because y^ℓ with this property is not dominated by any pair in $\bar{F}_k$.

(v) By construction, $x^{j+p} \notin \bar{\mathcal{F}}_{j+p-1}$ (this is explicit if x^{j+p} is declared a serious iterate by Step 4; if it is declared by Step 6, then the same fact follows from item (iv)).

As is easily seen, the forbidden region cannot contract: $\mathcal{F}_{j+1} \subset \mathcal{F}_{j+p}$. Also, the forbidden region is contained in the temporary forbidden region of the previous iteration: $\mathcal{F}_{j+p} \subset \bar{\mathcal{F}}_{j+p-1}$. It follows that $x^{j+p} \notin \mathcal{F}_{j+1}$. □

Some ideas of the proofs below are based on the following considerations. By [30, Theorem 4.5], it is known that if x^k is not a solution to (1) then, after a finite number of null steps, y^ℓ would satisfy the descent condition (16) unless, of course, the filter tests of Step 4 are satisfied first. In fact, we shall prove that in some situations (16) implies the filter tests of Step 4, and, thus, the filter tests are satisfied after a finite number of null steps and we do not enter the restoration phase. Furthermore, if the method does not generate the next serious step x^{k+1}, this means that, over an infinite sequence of null steps associated to the *fixed* function h_k, we do not satisfy the filter conditions of Step 4 and do not enter the restoration phase of Step 6. The latter, in particular, implies that the descent condition of Step 5 is never achieved. This can only happen if x^k is a solution to (1), see Theorem 2 below.

For future reference, note that, by definition of the improvement function, condition (16) is equivalent to the combination of the following two relations:

$$f(y^\ell) - f(x^k) \le c^+(x^k) - m_1\delta_\ell\,, \tag{17}$$

and

$$c(y^\ell) \le c^+(x^k) - m_1\delta_\ell\,. \tag{18}$$

We next show that our algorithm is well-defined.

Proposition 2 *Algorithm 1 is well-defined.*

Proof The method is just solving a sequence of well-posed quadratic programs (6), except for the possibility of entering the restoration phase (Step 6).

The restoration phase is well-posed whenever $\tilde{c}^+ > 0$ (i.e., $c^+(x^k) > 0$), because $\tilde{c}_j^+ > 0$ for all $j \in \mathbb{N}$ such that $(\tilde{f}_j, \tilde{c}_j^+) \in F_k$, by Proposition 1(iii).

We next show that if $c^+(x^k) = 0$ then the filter tests of Step 4 are satisfied no later than condition (16) of Step 5. Consequently, we do not enter the restoration phase of Step 6.

Assume (16) [equivalently, (17) and (18)]. Then (18) implies that $c(y^\ell) \le -m_1\delta_\ell < 0$, and thus,

$$c^+(y^\ell) = 0 = \alpha_c c^+(x^k).$$

By (17), we also have that

$$f(y^\ell) \le f(x^k) - m_1\delta_\ell < f(x^k) = f(x^k) - \alpha_f c^+(x^k).$$

By Proposition 1 (ii), used with

$$(\tilde{f}_j, \tilde{c}_j^+) = (\tilde{f}, \tilde{c}^+) = \big(f(x^k) - \alpha_f c^+(x^k),\ \alpha_c c^+(x^k)\big) \in \bar{F}_k,$$

we conclude that $y^\ell \notin \bar{\mathcal{F}}_k$.

Furthermore, since $c^+(x^k) = 0$, the relation

$$c^+(x^k) \le m_2\delta_\ell$$

holds trivially. And, as shown above,

$$f(y^\ell) \le f(x^k) - m_1\delta_\ell = f(x^k) + c^+(x^k) - m_1\delta_\ell,$$

which gives (15). Hence, y^ℓ is accepted by the filter tests in Step 4 and we cannot enter the restoration phase of Step 6. □

For each k, let $\ell(k)$ be the index such that $x^{k+1} = y^{\ell(k)}$, i.e., $y^{\ell(k)}$ has been declared the $(k+1)$-st serious iterate. As customary in bundle methods, we consider two cases: either the serious step sequence $\{x^k\}$ is finite or it is infinite.

Theorem 2 *Suppose Algorithm 1 with $tol = 0$ generates a finite sequence of serious steps followed by infinitely many null steps. Let x^k be the last serious iterate. If*

$$\bar{\mu} \ge \mu_{\ell+1} \ge \mu_\ell \quad \forall\ \ell \ge \ell(k-1), \tag{19}$$

then x^k is a solution to (1).

Proof We are in the situation where the method enters an infinite loop of null steps without ever generating a new serious step. This can only happen if, for $\ell > \ell(k-1)$, iterates y^ℓ are never accepted by the filter (Step 4) and the restoration phase (Step 6) is never visited. The fact that we never enter the restoration phase, in particular, means that the descent test (16) is never satisfied for the *fixed* function h_k. In this case, the convergence analysis of [30, Theorem 4.5] applies to claim that x^k is a solution. □

We next show that if x^k is not a solution, then Algorithm 1 always generates the next serious iterate.

Proposition 3 *Consider Algorithm 1 with $tol = 0$. At any given iteration k, if x^k is not a solution to (1) and if parameters μ_ℓ are chosen for the subsequent (null) iterations according to (19), then the next serious iterate x^{k+1} is generated.*

Proof If x^k is not a solution then [30, Theorem 4.5] guarantees that the descent condition (16) of Step 5 would be satisfied after a finite number of null steps, unless filter

tests of Step 4 are satisfied first (note that we cannot enter the restoration step before (16) is satisfied). If the filter tests hold first, the assertion follows.

As exhibited in the proof of Proposition 2 above, condition (16) can be satisfied before the filter tests only when x^k is infeasible. In the latter case, we go to the restoration phase of Step 6, and Proposition 2 shows that it is well-posed in the case of infeasible x^k. So, the next serious iterate is generated. □

To show that an infinite sequence $\{x^k\}$ of serious steps is minimizing, we consider separately the cases of feasible and infeasible accumulation points.

Proposition 4 *Suppose Algorithm 1 with* $tol = 0$ *generates an infinite sequence of serious steps. Consider a subsequence of serious iterates* $\{x^{k_i}\}$ *converging to* $\bar{x}$. *Assume that* $0 < \mu_{\ell(k_i)} \leq \bar{\mu} < +\infty$ *for all* i.

If $\bar{x}$ *is feasible but is not a solution to* (1), *then the corresponding subsequence* $\{\delta_{\ell(k_i)}\}$ *is bounded away from zero:*

$$\exists \gamma > 0 \text{ such that } \delta_{\ell(k_i)} \geq \gamma \quad \forall\, i. \tag{20}$$

Moreover, there exists an index i_0 *such that*

$$f(x^{k_i}) - f(x^{k_i+1}) \geq m_1 \gamma/2 \quad \forall\, i \geq i_0. \tag{21}$$

In particular, all iterations indexed by k_i *with* $i \geq i_0$ *are* f*-iterations.*

Proof Suppose, for contradiction purposes, that (20) does not hold. Then there exists an infinite set $\mathcal{K} \subset \{k_i \mid i \in \mathbb{N}\}$ such that $\delta_{\ell(k)} \overset{\mathcal{K}}{\to} 0$. By (13) and the boundedness assumption on $\mu_{\ell(k_i)}$, we conclude that

$$\hat{\varepsilon}^k_{\ell(k)} \overset{\mathcal{K}}{\to} 0 \quad \text{and} \quad \hat{g}^{\ell(k)} \overset{\mathcal{K}}{\to} 0.$$

By item (ii) in Lemma 1 and the definition of ε-subgradient, for any $y \in \mathbb{R}^n$ and $k \in \mathcal{K}$, it holds that

$$h_k(x^k + y) \geq h_k(x^k) + \langle \hat{g}^{\ell(k)}, y \rangle - \hat{\varepsilon}^k_{\ell(k)}. \tag{22}$$

By the continuity of the functions involved, for a fixed but arbitrary $y \in \mathbb{R}^n$, we have that

$$\begin{aligned} \lim_{k\to\infty, k\in\mathcal{K}} h_k(x^k + y) &= \lim_{k\to\infty, k\in\mathcal{K}} \max\left\{ f(x^k + y) - f(x^k), c(x^k + y) \right\} \\ &= \max\{ f(\bar{x} + y) - f(\bar{x}), c(\bar{x} + y) \} \\ &= h_{\bar{x}}(\bar{x} + y). \end{aligned}$$

In particular, choosing $y = 0$ yields the relation $h_k(x^k) \overset{\mathcal{K}}{\to} h_{\bar{x}}(\bar{x}) = c^+(\bar{x}) = 0$, because $\bar{x}$ is feasible.

Hence, passing to the limit in (22) we obtain that, for any $y \in \mathbb{R}^n$,

$$h_{\bar{x}}(\bar{x} + y) \geq h_{\bar{x}}(\bar{x}) = c^+(\bar{x}) = 0,$$

which means that $\bar{x}$ is a minimum of $h_{\bar{x}}(\cdot)$ with zero optimal value. Using the equivalence between items (ii) and (i) in Theorem 1, we obtain a contradiction with the hypothesis that $\bar{x}$ is not a solution to (1). Hence, (20) holds.

To prove (21), we show that for all iteration indices sufficiently large, the method does not enter the restoration phase and the filter test is satisfied with condition (15). Once this claim is established, (15) immediately gives the desired result, because

$$f(x^{k_i}) - f(y^{\ell(k_i)}) \geq m_1 \delta_{\ell(k_i)} - c^+(x^{k_i}) \geq m_1 \gamma/2,$$

by (20) and by the fact that $c^+(x^{k_i}) \to c^+(\bar{x}) = 0$.

To prove the claim, we start by noting that when i is sufficiently large, the filter test with option (14) cannot be satisfied, because $c^+(x^{k_i}) \to c^+(\bar{x}) = 0$, while $\{\delta_{\ell(k_i)}\}$ is bounded away from zero. Therefore, the first condition in (15) eventually always holds.

It remains to show that if at some iteration $\ell > \ell(k_i - 1)$, with i sufficiently large, the point y^ℓ satisfies the descent test (16), then $y^\ell \notin \bar{\mathcal{F}}_{k_i}$ and (15) holds (this would mean that we never enter the restoration phase, as the filter test is satisfied before, yielding $x^{k_i+1} = y^\ell$).

Observe that the second condition in (15) is the same as (17), which is a part of (16). Thus, we only have to prove that $y^\ell \notin \bar{\mathcal{F}}_{k_i}$ whenever y^ℓ satisfies (16) [equivalently, (17) and (18)].

Observe that by (20) and because $c^+(x^{k_i}) \to 0$, we have, for sufficiently large i, that

$$c^+(x^{k_i}) - m_1 \delta_\ell < -\alpha_f c^+(x^{k_i})$$

and

$$c^+(x^{k_i}) - m_1 \delta_\ell < \alpha_c c^+(x^{k_i}).$$

By (18), we obtain that

$$c(y^\ell) \leq c^+(x^{k_i}) - m_1 \delta_\ell < \alpha_c c^+(x^{k_i}).$$

Hence,

$$c^+(y^\ell) \leq \alpha_c c^+(x^{k_i}). \tag{23}$$

Similarly, by (17),

$$f(y^\ell) \leq f(x^{k_i}) + c^+(x^{k_i}) - m_1 \delta_\ell < f(x^{k_i}) - \alpha_f c^+(x^{k_i}).$$

Together with (23), the latter relation implies that $y^\ell \notin \bar{\mathcal{F}}_{k_i}$ by Proposition 1(ii). The proof is complete. □

We now address the case of the sequence of serious steps accumulating at an infeasible point.

Proposition 5 *Suppose Algorithm 1 with $tol = 0$ generates an infinite sequence of serious steps. Consider a subsequence of serious iterates $\{x^{k_i}\}$ converging to $\bar{x}$.*

If $\bar{x}$ is infeasible, then there exists an index i_0 such that each iteration k_i with $i \geq i_0$ is an f-iteration.

Proof For contradiction purposes, suppose that there exists an infinite set $\mathcal{K} \subset \{k_i \mid i \in \mathbb{N}\}$ such that all iterations in $\mathcal{K}$ are c^+-iterations. By the continuity of c^+ and f, we have that

$$c^+(x^k) \overset{\mathcal{K}}{\to} c^+(\bar{x}) \quad \text{and} \quad f(x^k) \overset{\mathcal{K}}{\to} f(\bar{x}).$$

Hence, since $\bar{x}$ is infeasible, we have the following relations:

$$c^+(x^k) - \alpha_c c^+(x^j) \overset{j,k\in\mathcal{K}}{\longrightarrow} (1-\alpha_c)c^+(\bar{x}) > 0,$$

and

$$f(x^k) - f(x^j) + \alpha_f c^+(x^j) \overset{j,k\in\mathcal{K}}{\longrightarrow} \alpha_f c^+(\bar{x}) > 0.$$

It follows that for sufficiently large $j, k \in \mathcal{K}$, say with $j < k$,

$$c^+(x^k) > \alpha_c c^+(x^j) \quad \text{and} \quad f(x^k) > f(x^j) - \alpha_f c^+(x^j).$$

The latter relations mean that $x^k \in \bar{\mathcal{F}}_j$. Furthermore, since $j \in \mathcal{K}$, this index corresponds to a c^+-iteration, defining the next filter by $\mathcal{F}_{j+1} = \bar{\mathcal{F}}_j$. As a result, $x^k \in \mathcal{F}_{j+1}$, which contradicts Proposition 1 (v), written with $p = k - j$. □

The following convergence result assumes boundedness of the sequence of iterates and affirms optimality of at least one of its accumulation points. Convergence results of this nature are typical for filter methods [7,10]. Boundedness of the iterates (sometimes passed in the form of including a bounded polyhedral set into the problem constraints and then inserting it into subproblems of the algorithm) is also a common assumption for constrained bundle methods [5,17,18,26]. It would be interesting to remove the boundedness assumption, for which the ideas of [2] could be useful.

Theorem 3 *Suppose Algorithm 1 with $tol = 0$ and $\{\mu_\ell\}$ bounded above generates an infinite sequence of serious steps. If the sequence $\{x^k\}$ is bounded, then it has an accumulation point which is a solution to (1).*

Proof Suppose first that there exists an infinite number of c^+-iterations. Taking a convergent subsequence $\{x^{k_i}\}$ along those indices, we obtain an accumulation point $\bar{x}$. Since all iterations indexed by $\{k_i\}$ are c^+-iterations, by Proposition 5, $\bar{x}$ must be feasible. Then, by Proposition 4, either $\bar{x}$ is a solution to (1) or all iterations k_i for i sufficiently large must be f-iterations. Since the latter contradicts the current assumption, we conclude that in this case $\bar{x}$ is a solution to (1).

Now assume that the number of c^+-iterations is finite, i.e., all iterations indexed by $k \geq k_0$ are f-iterations. Then the sequence $\{f(x^k)\}$ is monotone (for $k \geq k_0$). Since it is bounded (by boundedness of $\{x^k\}$ and continuity of f), it converges. Hence,

$$f(x^k) - f(x^{k+1}) \to 0. \tag{24}$$

Suppose there exists an infinite set $\mathcal{K} \subset \mathbb{N}$ such that the second relation in Proposition 1 (i) holds for all $k \in \mathcal{K}$. In that case,

$$\alpha_f c^+(x^k) \leq f(x^k) - f(x^{k+1}), \quad k \in \mathcal{K},$$

and (24) implies that $c^+(x^k) \overset{\mathcal{K}}{\to} 0$.

If there is no set $\mathcal{K}$ with the above property, then there exists $k_1 \in \mathbb{N}$ such that the first relation in Proposition 1(i) holds for all $k > k_1$. In that case, it easily follows that $c^+(x^k) \to 0$ (at a linear rate).

In either case, $\{x^k\}$ has a feasible accumulation point $\bar{x}$. It can now be seen that $\bar{x}$ must be a solution to (1). Indeed, if it is not, then the relation (21) of Proposition 4 holds, in contradiction with (24). □

4 Computational experiments

For preliminary validation of our approach, we wrote a FORTRAN implementation of Algorithm 1, and analyzed its performance on a battery of test problems that we describe below.

Test problems. We used the following set of test problems:

- Constrained `MAXQUAD`, `LOCAT`, `MINSUM`, `ROSEN`, and `HILBERT`.
- Problems 10, 11, 12, 22, 100, 113, 227, 228, 264, 284 and 285 from [12], referred to as `HKnnn` in the sequel.
- Twenty randomly generated problems (see [18]) of the form

$$\begin{array}{ll} \text{minimize} & \langle a, x \rangle \\ \text{subject to} & Ax \leq b, \\ & \|Q_i x - q_i\|_2 \leq \rho_i \quad i = 1, \ldots, m, \\ & \|x\|_\infty \leq 10\,, \end{array}$$

where $a \in \mathbb{R}^n$, A is an $m_l \times n$ matrix, $b \in \mathbb{R}^{m_l}$ and, for $i = 1, \ldots, m$, $q_i \in \mathbb{R}^k$ and Q_i are $k \times n$ matrices. To be able to compare with solvers that accept linear

constraints only, ten instances are linear programming problems denoted by LINj, for $j = 1, \ldots, 10$. The remaining ten instances contain nonlinear constraints and are denoted by RNDj, $j = 1, \ldots, 10$.

The first set of problems above is standard in nonsmooth convex optimization; see [30, Sect. 5] for a detailed description. To this standard collection, we added those problems from [12] which we identified as being convex. Note that problems in [12] are smooth. We created nonsmooth problems by defining the constraint using the max-operation. The randomly generated problems are taken from [18], and we refer to the numerical section in [18] for full details about the random generator and its input parameters.

Table 1 shows some relevant data for the first 16 problems, such as their dimension, optimal value, and the chosen starting points. The column entitled $c(x)$ has values *lin* or *cvx* to indicate whether the respective test problem has only linear or general convex constraints. For the first four problems we used both a feasible and an infeasible starting point. The fifth problem, HILBERT, is a feasibility problem for which we used two different infeasible starting points. Starting points for HKnnn problems are those given in the Fortran code of test problems for nonlinear programming available at http://www.uni-bayreuth.de/departments/math/~kschittkowski/.

Table 2 reports relevant data for the 20 randomly generated problems LINj and RNDj. For all these problems the starting point is $x_i = 0.1$ for all $i = 1, \ldots, n$.

Solvers. Since all the problems in Tables 1 and 2 have known optimal values, the exact improvement function $h_{\bar{x}}$ is available. For comparison purposes, we use the following solvers:

Table 1 Some problem data

Name	n	$c(x)$	$f(\bar{x})$	Feasible x^0	Infeasible x^0
MAXQUAD	10	lin	−0.368166	$x_i = 0$	$x_i = 10$
LOCAT	4	cvx	23.88676767	(15, 22, 26, 11)	$x_i = 10$
MINSUM	6	lin	68.82956	(0, 0, 0, 0, 3, 0)	$x_i = 10$
ROSEN	4	cvx	−44	$x_i = 0$	(−1, 2, −3, −4)
HILBERT	50	cvx	0	–	$x_i = 0$ and $x_i = 10$
HK010	2	cvx	−1	–	(−10, 10)
HK011	2	cvx	−8.4984642231	–	(4.9, 0.1)
HK012	2	cvx	−30	(0, 0)	–
HK022	2	cvx	1	–	(2, 2)
HK100	7	cvx	680.6300572	(1, 2, 0, 4, 0, 1, 1)	–
HK113	10	cvx	24.3062090641	(2, 3, 5, 5, 1, 2, 7, 3, 6, 10)	–
HK227	2	cvx	1	(0.5, 0.5)	–
HK228	2	cvx	−3	(0, 0)	–
HK264	4	cvx	−0.44	(0, 0, 0, 0)	–
HK284	15	cvx	−1840	$x_i = 0$	–
HK285	15	cvx	−8,252	$x_i = 0$	–

Table 2 Problem data for randomly generated problems

Name	n	$c(x)$	m_l	m	k	$f(\bar{x})$
LIN01	3	lin	2	0	0	2343.10
LIN02	5	lin	3	0	0	2983.86
LIN03	10	lin	3	0	0	−5191.87
LIN04	10	lin	6	0	0	−12820.62
LIN05	10	lin	10	0	0	7580.54
LIN06	15	lin	3	0	0	4353.87
LIN07	15	lin	10	0	0	−1612.96
LIN08	15	lin	15	0	0	−3872.07
LIN09	30	lin	5	0	0	4003.20
LIN10	30	lin	10	0	0	−3410.63
RND01	2	cvx	0	1	1	−26.50
RND02	3	cvx	3	2	1	−48774.07
RND03	10	cvx	0	2	2	79.69
RND04	10	cvx	1	3	2	1923.58
RND05	15	cvx	5	2	2	−68746.10
RND06	15	cvx	3	8	4	53562.60
RND07	15	cvx	2	6	6	−31665.67
RND08	20	cvx	5	5	2	28177.93
RND09	30	cvx	3	10	2	18231.20
RND10	30	cvx	2	17	7	4117.29

- CLEV and NLEV, two constrained variants of the level bundle method in [18].
- PBUN, PNEW, and PVAR, corresponding, respectively, to the bundle, Newton, and variable metric nonsmooth methods described in [23]; see also [21] and [22]. The corresponding source codes are available at [24].
- N1CV2 (applied to minimize $h_{\bar{x}}$), the proximal bundle method from [19]; see http://www-rocq.inria.fr/estime/modulopt/optimization-routines/n1cv2.html.
- ICPBM, the Infeasible Constrained Proximal Bundle Method from [30].

Since the set of solvers PBUN, PNEW, and PVAR is designed specifically for linearly constrained problems, these methods are only applicable to MAXQUAD, MINSUM, and LINj, $j = 1, \ldots, 10$. Solvers CLEV and NLEV have the option of handling linear and nonlinear constraints separately. For the sake of comparison of general-purpose solvers, in our implementation all the constraints are treated as general nonlinear convex constraints.

For solving the quadratic programming subproblems, CLEV and NLEV use QL0001, developed by K. Schittkowski in 1987 on the basis of algorithm ZQPCVX in [28]. Solver PBUN uses the dual space range method [20]. Finally, N1CV2, ICPBM and Algorithm 1 use the method described in [15].

Parameters. For solvers CLEV, NLEV, PBUN, PNEW, and PVAR, parameters were set to the default values suggested by the respective codes. We also set a maximum of 100 oracle calls and a tolerance threshold for optimality equal to 10^{-4}.

Parameters of Algorithm 1 are set as follows: $m_1 = 0.1$, $m_2 = 0.5$, $\alpha_f = 0.0001$, $\alpha_c = 0.9999$.

Optimality is declared when

$$\hat{\varepsilon}_\ell^k \le 10^{-4} \quad \text{and} \quad \|\hat{g}^\ell\|^2 \le 10^{-8},$$

corresponding to taking $tol = (1 + \frac{1}{2\mu_\ell}10^{-4})10^{-4}$ in Step 3 of Algorithm 1. The above split stopping criterion, independent of μ_ℓ, is generally preferable to the one based on δ_ℓ.

The proximal parameter μ_ℓ in (6) is only changed at serious steps ($\mu_\ell = \mu_{k(\ell)}$ at null steps). The modification is performed by the safeguarded version of the *reversal quasi-Newton* scalar update in N1CV2; see [3, Sect. 9.3.3]. More precisely, recalling that $x^{k+1} = y^{\ell(k)}$, we set $\underline{\mu} := 0.01$, $\overline{\mu} := 1$, and define

$$\mu_{k+1} := \max\left(\underline{\mu}, \min\left(\mu_{k+1}^{\text{N1CV2}}, \overline{\mu}\right)\right),$$

where

$$\mu_0^{\text{N1CV2}} := \frac{5\|g_{h_0}(x^0)\|^2}{\|h_0(x^0)\|},$$
$$\mu_{k+1}^{\text{N1CV2}} := \frac{\|g_{h_k}^{\ell(k)} - g_{h_k}^{\ell(k-1)}\|^2}{\langle g_{h_k}^{\ell(k)} - g_{h_k}^{\ell(k-1)}, x^{k+1} - x^k\rangle}.$$

With this choice of μ_ℓ, none of the runs of Algorithm 1 ever entered the restoration phase of Step 6.

Solvers ICPBM and N1CV2 have essentially identical settings, with the following exceptions. The safeguards $\underline{\mu}$ and $\overline{\mu}$ were set so that to optimize performance of each solver. In addition, N1CV2 modifies μ_ℓ at every iteration (not only at serious steps), according to the curve-search described in [19].

Results. Since we are considering the setting when the objective and constraint functions are given by an oracle, effectiveness of an algorithm is not computed in terms of time, but in terms of the number of oracle calls made. More precisely, having set the optimality tolerance to 10^{-4} in all the algorithms, we compare their accuracy by using the expression

$$\text{RELACC} := \frac{-\log_{10}\left(\left|\frac{f(x^{best}) - f(\bar{x})}{f(\bar{x})}\right|\right)}{\#\text{OR}_{calls}},$$

where x^{best} is the best point found by a given algorithm and $\#\text{OR}_{calls}$ is the number of oracle calls required to compute x^{best}. When $f(\bar{x}) = 0$, we replace the denominator in the logarithm by 1. The value of RELACC can be understood as the number of exact digits gained per oracle call used. The solver with highest values is considered the most successful.

Our numerical results are reported in Tables 3 and 4, where Algorithm 1 is referred to as CPBFM, for Constrained Proximal Bundle Filter Method. For test problems with more than one starting point, we give the average value of RELACC. We note that for all the algorithms the accuracy obtained was similar both for feasible and infeasible starting points. Furthermore, in all the runs, the final value obtained for the constraint violation was less than 10^{-4}.

Table 3 reports relative accuracy for the eight solvers: CLEVEL, NLEVEL, PBUN, PNEW, PVAR, N1CV2, ICPBM and CPBFM, applied to the 12 linearly constrained problems in the battery.

The summary of results in Table 3 puts in evidence the importance of incorporating into the method all the information available for a given problem. Not surprisingly, solvers PBUN, PNEW, and PVAR, which are specifically designed to treat separately affine and box constraints (by inserting these constraints directly into subproblems) are most efficient. Similarly, solver N1CV2, which uses information about the optimal value, is also efficient. Solvers implemented in general-purpose mode, i.e., CLEV, NLEV and ICPBM, CPBFM, are comparable to each other but show much lower accuracy values than solvers using additional information.

Table 4 reports the RELACC values obtained when running solvers CLEVEL, NLEVEL, N1CV2, ICPBM, and CPBFM on all the test problems given in Tables 1 and 2.

In our opinion, the results obtained show reasonably good performance of CPBFM. In terms of accuracy, all approaches counting with similar information, i.e., CLEV, NLEV, ICPBM, and CFPBM are overall comparable, perhaps with a slight preference for CPBFM. We note that in terms of computing times, solvers CLEV and NLEV stalled on some problems when the dimensions increased. For NLEV we also observed some failures (which appear as 0.00 in Table 4). In average CPBFM succeeds in obtaining a reasonably high accuracy, at the price of less than 3.3 times the number of oracle calls required by N1CV2 to solve the "ideal" unconstrained problem of minimizing $h_{\bar{x}}$.

Table 3 RELACC values for linearly constrained problems

	CLEVEL	NLEVEL	PBUN	PNEW	PVAR	n1cv2	ICPBM	CPBFM
MAXQUAD	0.09	0.09	0.60	0.00	0.10	0.07	0.06	0.15
MINSUM	0.08	0.11	0.83	0.33	0.83	0.07	0.12	0.06
Lin01	0.25	0.86	5.33	5.33	4.00	4.00	0.21	0.09
Lin02	0.20	0.11	3.00	2.00	2.60	1.25	0.13	0.07
Lin03	0.28	0.09	1.30	1.00	1.50	3.20	1.20	0.80
Lin04	0.19	0.00	1.50	1.00	1.20	0.93	0.10	0.07
Lin05	0.24	0.57	1.40	1.40	1.20	0.64	0.10	0.08
Lin06	0.17	0.00	0.73	0.60	1.07	1.08	0.16	0.05
Lin07	0.17	0.05	0.87	0.67	0.87	1.88	0.17	0.08
Lin08	0.22	0.04	0.87	0.53	1.00	1.00	0.22	0.05
Lin09	0.19	0.07	0.43	0.53	0.33	1.36	0.16	0.06
Lin10	0.20	0.63	0.43	0.33	0.50	1.45	0.17	0.11
Mean	0.19	0.22	1.44	1.14	1.27	1.41	0.23	0.14

Table 4 Summary of RELACC values

	CLEVEL	NLEVEL	n1cv2	ICPBM	CPBFM
HILBERT	1.78	1.78	0.50	1.60	1.60
HK010	0.06	0.08	0.11	0.13	0.30
HK011	0.30	0.29	0.36	0.09	0.54
HK012	0.08	0.00	0.28	0.30	0.33
HK022	0.33	0.19	0.33	0.60	0.26
HK100	0.08	0.00	4.00	0.06	0.06
HK113	0.07	0.09	2.29	0.04	0.08
HK227	0.18	0.20	0.23	0.14	0.38
HK228	0.10	0.17	2.29	0.46	1.88
HK264	0.03	0.10	0.07	0.04	0.14
HK284	0.03	0.00	0.14	0.03	0.22
HK285	0.07	0.00	0.08	0.05	0.15
LOCAT	0.02	0.45	0.33	0.20	0.35
MAXQUAD	0.09	0.09	0.07	0.06	0.15
ROSEN	0.15	0.19	0.16	0.09	0.12
MINSUM	0.08	0.11	0.07	0.12	0.06
Lin01	0.25	0.86	4.00	0.21	0.09
Lin02	0.20	0.11	1.25	0.13	0.07
Lin03	0.28	0.09	3.20	1.20	0.80
Lin04	0.19	0.00	0.93	0.10	0.07
Lin05	0.24	0.57	0.64	0.10	0.08
Lin06	0.17	0.00	1.08	0.16	0.05
Lin07	0.17	0.05	1.88	0.17	0.08
Lin08	0.22	0.04	1.00	0.22	0.05
Lin09	0.19	0.07	1.36	0.16	0.06
Lin10	0.20	0.63	1.45	0.17	0.11
Cvx01	0.18	0.36	0.38	0.41	0.16
Cvx02	0.10	0.43	0.24	0.10	0.09
Cvx03	0.09	0.04	0.11	0.16	0.17
Cvx04	0.08	0.22	0.33	0.09	0.08
Cvx05	0.08	0.12	0.04	0.08	0.07
Cvx06	0.05	0.16	0.02	0.08	0.05
Cvx07	0.10	0.03	0.04	0.07	0.03
Cvx08	0.09	0.11	0.04	0.10	0.04
Cvx09	0.04	0.12	0.02	0.07	0.08
Cvx10	0.05	0.06	0.05	0.08	0.10
Mean	0.18	0.22	0.82	0.22	0.25

We finish by mentioning that for CPBFM and ICPBM, "suitable" settings for the proximal parameter μ_ℓ are not clear so far. From our preliminary numerical results, we observed that allowing this parameter to vary abruptly from one iteration to the

next made both algorithms slow down (sometimes, dramatically). In our runs, we controlled this variation by choosing μ and $\overline{\mu}$ "close enough". The study of computationally good and theoretically sound choices of the proximal parameter for solving constrained problems deserves further investigation, possibly along the lines of the update in [19]; see also [29]. Similarly, tuning and adjusting filter strategy specifically for the nonsmooth case requires further investigation and improvements.

References

1. Auslender, A.: Numerical methods for nondifferentiable convex optimization. Math. Program. Study **30**, 102–126 (1987)
2. Auslender, A.: How to deal with the unbounded in optimization: theory and algorithms. Math. Program. **79**, 3–18 (1997)
3. Bonnans, J.F., Gilbert, J.-Ch., Lemaréchal, C., Sagastizábal C.: Numerical Optimization. Theoretical and Practical Aspects. Universitext. Springer, Berlin (2003)
4. Fletcher, R., Gould, N., Leyffer, S., Toint, P., Wächter, A.: Global convergence of trust-region and SQP-filter algorithms for general nonlinear programming. SIAM J. Optim. **13**, 635–659 (2002)
5. Fletcher, R., Leyffer, S.: A bundle filter method for nonsmooth nonlinear optimization. Numerical Analysis Report NA/195. Department of Mathematics, The University of Dundee, Scotland (1999)
6. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program **91**, 239–269 (2002)
7. Fletcher, R., Leyffer, S., Toint, P.L.: On the global convergence of a filter-SQP algorithm. SIAM J. Optim. **13**, 44–59 (2002)
8. Frangioni, A.: Solving semidefinite quadratic problems within nonsmooth optimization algorithms. Comput. Oper. Res. **23**, 1099–1118 (1996)
9. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **13**, 117–156 (2002)
10. Gonzaga, C.C., Karas, E.W., Vanti, M.: A globally convergent filter method for nonlinear programming. SIAM J. Optim. **14**, 646–669 (2003)
11. Hiriart-Urruty, J.-B., Lemaréchal C.: Convex Analysis and Minimization Algorithms. Number 305–306 in Grund. der Math. Wiss. Springer, Heidelberg (1993)
12. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, vol. 187. Springer, Berlin (1981)
13. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics, vol. 1133. Springer, Berlin (1985)
14. Kiwiel, K.C.: An exact penalty function algorithm for nonsmooth convex constrained minimization problems. IMA J. Numer. Anal. **5**, 111–119 (1985)
15. Kiwiel, K.C.: A method for solving certain quadratic programming problems arising in nonsmooth optimization. IMA J. Numer. Anal. **6**, 137–152 (1986)
16. Kiwiel, K.C.: A constraint linearization method for nondifferentiable convex minimization. Numer. Math. **51**, 395–414 (1987)
17. Kiwiel, K.C.: Exact penalty functions in proximal bundle methods for constrained convex nondifferentiable minimization. Math. Program. **52**, 285–302 (1991)
18. Lemaréchal, C., Nemirovskii, A., Nesterov, Yu.: New variants of bundle methods. Math. Program. **69**, 111–148 (1995)
19. Lemaréchal, C., Sagastizábal, C.: Variable metric bundle methods: from conceptual to implementable forms. Math Program **76**, 393–410 (1997)
20. Lukšan L: Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minimax approximation. Kybernetika **20**, 445–457 (1984)
21. Lukšan, L., Vlček, J.: A bundle-Newton method for nonsmooth unconstrained minimization. Math. Program. 83(3, Ser. A):373–391 (1998)
22. Lukšan, L., Vlček, J.: Globally convergent variable metric method for convex nonsmooth unconstrained minimization. J. Optim. Theory Appl. **102**(3), 593–613 (1999)
23. Lukšan, L., Vlček, J.: NDA: Algorithms for nondifferentiable optimization. Research Report V-797. Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (2000)

24. Lukšan, L., Vlček, J.: Algorithm 811, NDA, http://www.netlib.org/toms/811. ACM Trans. Math. Softw. **27**(2), 193–213 (2001)
25. Mangasarian, O.L.: Nonlinear Programming. McGraw-Hill, New York, (1969)
26. Mifflin, R.: An algorithm for constrained optimization with semismooth functions. Math. Oper. Res. **2**, 191–207 (1977)
27. Mifflin, R.: A modification and extension of Lemarechal's algorithm for nonsmooth minimization. Math. Program. Study **17**, 77–90 (1982)
28. Powell, M.J.D.: On the quadratic programming algorithm of Goldfarb and Idnani. Math. Program. Study **25**, 46–61 (1985)
29. Rey, P.A., Sagastizábal, C.: Dynamical adjustment of the prox-parameter in variable metric bundle methods. Optimization **51**, 423–447 (2002)
30. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**, 146–169 (2005)