Raport Badawczy

Research Report

RB/36/2009

Bundle methods for convex minimization with partially inexact oracles

K.C. Kiwiel

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 3810100

fax: (+48) (22) 3810105

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Krzysztof C. Kiwiel

Warszawa 2009

Computational Optimization and Applications manuscript No.
(will be inserted by the editor)

Bundle Methods for Convex Minimization with Partially Inexact Oracles

K.C. Kiwiel

Received: March 18, 2009; revised November 11, 2009 / Accepted: date

Abstract Recently the proximal bundle method for minimizing a convex function has been extended to an inexact oracle that delivers function and subgradient values of unknown accuracy. We adapt this method to a partially inexact oracle that becomes exact only when an objective target level for a descent step is met. In Lagrangian relaxation, such oracles may save work by evaluating the dual function approximately on most iterations, without compromising the strong convergence properties of exact bundle methods. We also show that the recent method of Gaudioso et al. for finite min–max problems fits our partially inexact framework, we correct and strengthen its convergence results and give useful modifications. Numerical illustrations on standard instances of the generalized assignment problem (GAP) are included.

Keywords Nondifferentiable optimization · Convex programming · Proximal bundle methods · Approximate subgradients · Finite min-max

Mathematics Subject Classification (2000) 65K05 · 90C25 · 90C27

1 Introduction

We consider the convex constrained minimization problem

$$f_* := \inf\{f(u) : u \in C\}, \tag{1.1}$$

where C is a "simple" closed convex set (typically a polyhedron) in the Euclidean space $\mathbb{R}^n$ with inner product $\langle \cdot, \cdot \rangle$ and norm $|\cdot|$, and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function.

We are interested in bundle methods, which at each trial point in C call an *oracle* to produce a *linearization* of f, given by a tuple in $\mathbb{R} \times \mathbb{R}^n$. At the current iteration k of such a method, the oracle has been called at trial points $u^1, \ldots, u^k$ in C, and has returned the corresponding tuples $\{(f_u^j, g^j)\}_{j=1}^k$ in $\mathbb{R} \times \mathbb{R}^n$. For an *exact* oracle,

K.C. Kiwiel
Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01–447 Warsaw, Poland, E-mail: kiwiel@ibspan.waw.pl

$f_u^j = f(u^j)$ and $g^j \in \partial f(u^j)$ denote the exact objective value and a subgradient at u^j. An *inexact* oracle may return $f_u^j = f(u^j) - \varepsilon_f^j$ and $g^j \in \partial_{\varepsilon_f^j + \varepsilon_g^j} f(u^j)$ with *errors* $\varepsilon_f^j + \varepsilon_g^j \geq 0$, where $\partial_\varepsilon f(u) := \{g : f(\cdot) \geq f(u) - \varepsilon + \langle g, \cdot - u\rangle\}$ is the ε-subdifferential of f at u; in other words, it delivers the linearization

$$f_j(\cdot) := f_u^j + \langle g^j, \cdot - u^j\rangle \leq f(\cdot) + \varepsilon_g^j \quad \text{with} \quad f_j(u^j) = f_u^j = f(u^j) - \varepsilon_f^j. \tag{1.2}$$

The errors are unknown, but bounded by some (unknown) constants $\varepsilon_f^{\max}$ and $\varepsilon_g^{\max}$:

$$\varepsilon_f^j \leq \varepsilon_f^{\max} \quad \text{and} \quad \varepsilon_g^j \leq \varepsilon_g^{\max} \quad \text{for all } j. \tag{1.3}$$

For instance, in many applications f is a max-type function of the form

$$f(u) := \sup\{F_z(u) : z \in Z\}, \tag{1.4}$$

where each $F_z : \mathbb{R}^n \to \mathbb{R}$ is convex and Z is an arbitrary set. If, for $u = u^j$, the oracle finds a possibly inexact maximizer $z^j \in Z$ of (1.4), sets $f_u^j := F_{z^j}(u^j)$ and takes g^j as any subgradient of F_{z^j} at u^j, then (1.2) holds with $\varepsilon_f^j = f(u^j) - F_{z^j}(u^j)$, $\varepsilon_g^j = 0$.

In [13] we extended the proximal bundle methods of [9] and [8, §XV.3] to the inexact setting of (1.2); see [14–16, 18] for further developments and [17, 18] for numerical tests. In general terms, such methods maintain:

- a closed convex *model* $\check{f}_k \leq f + \check{\varepsilon}_g^k$ with $\check{\varepsilon}_g^k \leq \varepsilon_g^{\max}$, for instance (see (1.2)–(1.3))

$$\check{f}_k = \max_{j \in J_k} f_j \quad \text{with} \quad J_k \subset \{1, \ldots, k\} \quad \text{and} \quad \check{\varepsilon}_g^k = \max_{j \in J_k} \varepsilon_g^j;$$

- a *stability center* $\hat{u}^k = u^{k'}$ for some $k' \leq k$ that has the value $f_{\hat{u}}^k = f_u^{k'}$, and
- a *proximity stepsize* $t_k > 0$ that controls the distance from $\hat{u}^k$ to the next trial point

$$u^{k+1} := \arg\min \left\{ \check{f}_k(u) + \tfrac{1}{2t_k}|u - \hat{u}^k|^2 : u \in C \right\}. \tag{1.5}$$

After the oracle called at u^{k+1} produces f_u^{k+1}, a *descent* step to $\hat{u}^{k+1} := u^{k+1}$ is taken if the objective reduction is at least a given fraction $\kappa \in (0,1)$ of the *predicted decrease*

$$v_k := f_{\hat{u}}^k - \check{f}_k(u^{k+1}), \tag{1.6}$$

i.e., if

$$f_u^{k+1} \leq f_{\hat{u}}^k - \kappa v_k. \tag{1.7}$$

Otherwise, a *null* step $\hat{u}^{k+1} := \hat{u}^k$ occurs; then the new linearization f_{k+1} is used to produce a better model $\check{f}_{k+1} \geq f_{k+1}$ for the next iteration. This summarizes exact bundle. The inexact extension of [13] is based on the observation that having the majorization

$$v_k \geq |u^{k+1} - \hat{u}^k|^2/2t_k \tag{1.8}$$

suffices for convergence. Hence, if necessary, t_k is increased and u^{k+1} is recomputed to decrease $\check{f}_k(u^{k+1})$ until (1.8) holds. As for convergence, [13] showed that the *asymptotic objective value* $f_{\hat{u}}^\infty := \lim_k f_{\hat{u}}^k$ estimates the optimal value f_* of (1.1):

$f_{\hat{u}}^{\infty} \in [f_* - \varepsilon_f^{\max}, f_* + \varepsilon_g^{\max}]$. Next, for $\varepsilon_g^{\max} = 0$, [14] observed that in fact the asymptotic accuracy depends only on the errors that occur at descent steps. Specifically, let $\ell(k) - 1$ index the last descent iteration prior to k, and denote by

$$\varepsilon_f^{\infty} := \overline{\lim_{k\to\infty}} \varepsilon_f^{\ell(k)} \tag{1.9}$$

the *asymptotic oracle error* at descent steps; then $f_{\hat{u}}^{\infty} \in [f_* - \varepsilon_f^{\infty}, f_*]$ (see [14, §4.2]).

First, in this paper we extend the analysis of [14] to the case $\varepsilon_g^{\max} > 0$. We show that $f_{\hat{u}}^{\infty} \in [f_* - \varepsilon_f^{\infty}, f_* + \varepsilon_g^{\infty}]$ with $\varepsilon_g^{\infty} \le \underline{\lim}_{k\in K} \check{\varepsilon}_g^k$ for some subset $K \subset \mathbb{N}$. This improves the recent result of [4, §5]; in our notation, it replaces ε_g^{∞} by $\overline{\lim}_{k\in K} \check{\varepsilon}_g^k$ and assumes additionally that f is coercive and $C = \mathbb{R}^n$.

Second, this paper shows how to ensure a null asymptotic error for a partially inexact oracle. We call the oracle *partially inexact* if, given the point u^{k+1} and a *target objective level* λ_k, it delivers a tuple (f_u^{k+1}, g^{k+1}) such that (1.2) holds with $\varepsilon_g^j \equiv 0$ for $j = k+1$, and additionally

$$f_u^{k+1} = f(u^{k+1}) \quad \text{and} \quad \varepsilon_f^{k+1} = 0 \quad \text{if } f_u^{k+1} \le \lambda_k. \tag{1.10}$$

Then, in view of the descent criterion (1.7), setting λ_k to the "natural" target level

$$\lambda_v^k := f_{\hat{u}}^k - \kappa v_k \tag{1.11}$$

for $k \ge 1$ ensures that each descent step is exact. The initial oracle call at u^1, corresponding to $k = 0$, can be handled in two ways. First, we may require exact evaluation by setting $\lambda_0 = \infty$. Second, to save the oracle work, we may accept any error by setting $\lambda_0 = -\infty$; in this case, if the initial inaccuracy is detected later via the majorization (1.8) failing for some k, we may reset v_k to $-\infty$, so that $\lambda_k = \infty$ and an exact descent step occurs. We will show that the asymptotic error is zero for both initial choices.

As the simplest useful example consider the *finite min-max* framework of (1.4) with a finite set Z. Suppose at iteration k the oracle examines the elements z of Z (in any order). If $F_z(u^{k+1}) > \lambda_k$ occurs for some z, the oracle sets $z^{k+1} = z$, otherwise it takes $z^{k+1} \in \operatorname{Argmax}_{z\in Z} F_z(u^{k+1})$, setting f_u^{k+1} and g^{k+1} as before; then (1.10) holds.

We add that in Lagrangian relaxation of integer programming problems (see, e.g., [2, 19] and references therein), the finite min-max framework covers typical oracles which employ branch and bound, possibly preceded by heuristics. Thus, whenever an exact oracle is available, a partially inexact oracle can be obtained by inserting the stopping criterion $F_z(u^{k+1}) > \lambda_k$ for each incumbent z, and to save work, cheaper heuristics can be run first before switching to branch and bound.

Inexact null steps have already been used in [7, §4.2] and [14, §4.2], but these references need exact initialization ($\lambda_0 = \infty$) to get $\varepsilon_f^{\infty} = 0$ in the partially inexact case. Relative to exact bundle, our main message is that inexact null steps do not impair convergence when the oracle is locally bounded (see (3.2) and Remark 4.2(1)). In effect, the existing bundle implementations can be extended to the partially inexact case simply by adding a few lines in their codes that determine the level λ_k sent to the oracle (see §4). Similar modifications can be employed whenever "standard" exact bundle is used as a procedure within a more complex algorithm. For example, in this

way one can extend the method of centers of [21] and the dynamic bundle of [1] (by invoking our Lemma 3.3 in their analysis of null steps).

Our third major contribution concerns the recent paper of [5], which introduced a different bundle method (GGM for short) for the finite min-max framework above. The main difference is that the GGM method replaces the descent criterion (1.7) by

$$f_u^{k+1} \le \check{f}_k(u^{k+1}) + \gamma_k \tag{1.12}$$

for a carefully chosen *gap* $\gamma_k > 0$; accordingly, the oracle level λ_k in (1.10) is set to

$$\lambda_\gamma^k := \check{f}_k(u^{k+1}) + \gamma_k. \tag{1.13}$$

We extend the GGM method to our partially inexact oracle framework, and strengthen its convergence results; in fact, the analysis of [5] has two major flaws (see Rem. 6.8). We also give a modification of the GGM method that does not reset the bundle to a singleton whenever the gap γ_k is decreased (such bundle reductions can hurt practical performance). Further, relating the GGM method to our "standard" proximal bundle method (PBM for short), we show that after at least one descent step, the GGM level λ_γ^k of (1.13) equals the PBM level λ_v^k of (1.11) with κ replaced by the coefficient $\kappa_k := 1 - \gamma_k/v_k \in [\kappa_{GGM}, 1)$, where $\kappa_{GGM} \in (0,1)$ is determined by the initial GGM parameters. Thus, the GGM method may be viewed as a PBM variant with a variable coefficient κ_k, where the lower bound $\kappa_k \ge \kappa_{GGM}$ suffices for analyzing descent steps. This also explains a serious weakness of the GGM method discovered in our experiments, when for κ_k close to 1, each descent step is followed by many null steps with tiny improvements of successive models. Our simple cure is to set λ_k to

$$\hat{\lambda}_\gamma^k := \max\{\lambda_\gamma^k, \lambda_v^k\}, \tag{1.14}$$

which gives another PBM variant with the descent coefficient $\hat{\kappa}_k := \min\{\kappa_k, \kappa\}$, where $\kappa \in (0,1)$ prevents $\hat{\kappa}_k$ from getting "too close" to 1 (see Lem. 7.1). With this modification, the performance of the GGM method in our experiments became comparable with that of our "standard" PBM (see §8).

Finally, we illustrate practical differences between the PBM and GGM variants in Lagrangian relaxation of the generalized assignment problem (GAP); for basic references on the GAP, see, e.g., [20,22–24]. We add that all the numerical results of [6] for the GGM method on the GAP were affected by some code bugs [1].

The paper is organized as follows. In §2 we present a streamlined version of our general inexact PBM, which simplifies some constructs of [13,14]. Section 3 gives a self-contained convergence analysis of our inexact PBM (although we could omit some results based on [13,14], most of them are needed anyway for our partially inexact PBM and GGM). In §4 we discuss using a partially inexact oracle within our PBM. Our extension of the GGM method is presented in §5, its convergence is analyzed in §6, and a modified version is given in §7. We conclude in §8 with numerical illustrations on the standard GAP instances used in [6].

[1] M. Gaudioso, private communication, October 15, 2008.

2 The inexact proximal bundle method

Before stating our method, we summarize below basic properties of subproblem (1.5), in a way simpler than in [13,14].

2.1 Aggregate linearizations and predicted decrease

We regard (1.1) as an unconstrained problem $f_* = \inf f_C$ with the *essential objective*

$$f_C := f + i_C, \tag{2.1}$$

where i_C is the *indicator* function of the set C ($i_C(u) = 0$ if $u \in C$, ∞ otherwise). Following the structure in (2.1), we rewrite the trial point finding subproblem (1.5) as

$$u^{k+1} := \arg\min \left\{ \phi_k(\cdot) := \check{f}_k(\cdot) + i_C(\cdot) + \tfrac{1}{2t_k}|\cdot - \hat{u}^k|^2 \right\}. \tag{2.2}$$

Recall from §1 that $\check{f}_k \le f + \check{\varepsilon}_g^k$ is closed convex, $\check{\varepsilon}_g^k \le \varepsilon_g^{\max}$, $t_k > 0$ and $\hat{u}^k \in C$ above; further, the stability center $\hat{u}^k = u^{\ell(k)}$ obtained at the last descent iteration $\ell(k) - 1$ prior to k has the value $f_{\hat{u}}^k = f_u^{\ell(k)}$, so that the oracle property (1.2) yields

$$f_{\hat{u}}^k = f(\hat{u}^k) - \varepsilon_f^{\ell(k)}. \tag{2.3}$$

We now use a standard optimality condition for subproblem (2.2) to derive aggregate linearizations (i.e., affine minorants) of the subproblem functions at u^{k+1}, and an optimality estimate; see (2.11)–(2.12), where f_C^* is the convex conjugate of f_C.

Lemma 2.1 (1) *There exist subgradients $\hat{g}^k$ and v^k such that*

$$\hat{g}^k \in \partial \check{f}_k(u^{k+1}), \quad v^k \in \partial i_C(u^{k+1}) \quad \textit{and} \quad \hat{g}^k + v^k = (\hat{u}^k - u^{k+1})/t_k. \tag{2.4}$$

(2) *These subgradients determine the following three* aggregate linearizations *of the functions $\check{f}_k$ and f, i_C, $\check{f}_C^* := \check{f}_k + i_C$ and $f_C := f + i_C$, respectively*:

$$\bar{f}_k(\cdot) := \check{f}_k(u^{k+1}) + \langle \hat{g}^k, \cdot - u^{k+1} \rangle \le \check{f}_k(\cdot) \le f(\cdot) + \check{\varepsilon}_g^k, \tag{2.5}$$

$$\bar{i}_C^k(\cdot) := i_C(u^{k+1}) + \langle v^k, \cdot - u^{k+1} \rangle \le i_C(\cdot), \tag{2.6}$$

$$\bar{f}_C^k(\cdot) := \bar{f}_k(\cdot) + \bar{i}_C^k(\cdot) \le \check{f}_C^*(\cdot) \le f_C(\cdot) + \check{\varepsilon}_g^k. \tag{2.7}$$

(3) *For the* aggregate subgradient *and the* aggregate linearization error *given by*

$$p^k := \hat{g}^k + v^k = (\hat{u}^k - u^{k+1})/t_k \quad \textit{and} \quad \varepsilon_k := f_{\hat{u}}^k - \bar{f}_C^k(\hat{u}^k), \tag{2.8}$$

and the optimality measure

$$V_k := \max\{|p^k|, \varepsilon_k + \langle p^k, \hat{u}^k \rangle\}, \tag{2.9}$$

we have

$$\bar{f}_C^k(\cdot) = \check{f}_k(u^{k+1}) + \langle p^k, \cdot - u^{k+1} \rangle, \tag{2.10}$$

$$f_{\hat{u}}^k - \varepsilon_k + \langle p^k, \cdot - \hat{u}^k \rangle = \bar{f}_C^k(\cdot) \le f_C(\cdot) + \varepsilon_p^k \quad \textit{with} \tag{2.11a}$$

$$\varepsilon_p^k := f_C^*(p^k) + \bar{f}_C^k(0) \le \check{\varepsilon}_g^k, \tag{2.11b}$$

$$f_{\hat{u}}^k \le f_C(u) + \varepsilon_p^k + V_k(1 + |u|) \quad \textit{for all } u. \tag{2.12}$$

Proof (1) Use the optimality condition $0 \in \partial\phi_k(u^{k+1})$ for subproblem (2.2).

(2) The inequalities in (2.5)–(2.6) stem from (2.4) as subgradient inequalities, and from our assumption that $f_k^j \le f + \check{\varepsilon}_g^k$; note that $i_C(u^{k+1}) = 0$ in (2.6). Adding (2.5) and (2.6) gives (2.7).

(3) The first equalities in (2.5)–(2.8) yield (2.10). Since $\overline{f}_C^k$ is affine, we have $f_{\hat{u}}^k - \varepsilon_k + \langle p^k, \cdot - \hat{u}^k\rangle = \overline{f}_C^k \le f_C + \check{\varepsilon}_g^k$ by (2.7) and (2.8), and $\overline{f}_C^k \le f_C + \varepsilon$ iff

$$\varepsilon \ge \sup_\cdot[\overline{f}_C^k(\cdot) - f_C(\cdot)] = \overline{f}_C^k(0) + \sup_\cdot[\langle p^k, \cdot\rangle - f_C(\cdot)] = \overline{f}_C^k(0) + f_C^*(p^k);$$

this gives (2.11). Finally, since $|a||b| + c \le \max\{|a|, c\}(1 + |b|)$ for any scalars a, b, c, and thus in (2.11), for $a = |p^k|$ and $b = |u|$, by the Cauchy-Schwarz inequality,

$$-\langle p^k, u\rangle + \varepsilon_k + \langle p^k, \hat{u}^k\rangle \le |p^k||u| + \varepsilon_k + \langle p^k, \hat{u}^k\rangle \le \max\{|p^k|, \varepsilon_k + \langle p^k, \hat{u}^k\rangle\}(1 + |u|),$$

we obtain (2.12) from the definition of V_k in (2.9). □

To ensure that the optimality measure V_k vanishes asymptotically, it is crucial to bound V_k by the predicted decrease v_k, since bundling and descent steps drive v_k to 0.

Lemma 2.2 (1) *In the notation of* (2.8), *the predicted decrease* v_k *of* (1.6) *satisfies*

$$v_k = t_k|p^k|^2 + \varepsilon_k. \tag{2.13}$$

(2) *The optimality measure* V_k *of* (2.9) *satisfies* $V_k \le \max\{|p^k|, \varepsilon_k\}(1 + |\hat{u}^k|)$.

(3) *We have the equivalences*

$$v_k \ge -\varepsilon_k \;\Leftrightarrow\; t_k|p^k|^2/2 \ge -\varepsilon_k \;\Leftrightarrow\; v_k \ge t_k|p^k|^2/2 \;\Leftrightarrow\; v_k \ge |u^{k+1} - \hat{u}^k|^2/2t_k.$$

Moreover, $v_k \ge \varepsilon_k$. *Finally, for* $\varepsilon_f^{\ell(k)}$ *in* (2.3), *we have* $-\varepsilon_k \le \varepsilon_f^{\ell(k)} + \varepsilon_g^{\max}$ *and*

$$v_k \ge \max\{t_k|p^k|^2/2, |\varepsilon_k|\} \qquad \text{if} \quad v_k \ge -\varepsilon_k, \tag{2.14}$$

$$V_k \le \max\{(2v_k/t_k)^{1/2}, v_k\}(1 + |\hat{u}^k|) \qquad \text{if} \quad v_k \ge -\varepsilon_k, \tag{2.15}$$

$$V_k < [2(\varepsilon_f^{\ell(k)} + \varepsilon_g^{\max})/t_k]^{1/2}(1 + |\hat{u}^k|) \qquad \text{if} \quad v_k < -\varepsilon_k. \tag{2.16}$$

Proof (1) By (2.10) and (2.8), $\overline{f}_C^k(\hat{u}^k) = f_k^j(u^{k+1}) + t_k|p^k|^2$. Rewrite (1.6), using (2.8).

(2) Using the Cauchy-Schwarz inequality in (2.9), we have

$$V_k \le \max\{|p^k|, \varepsilon_k + |p^k||\hat{u}^k|\} \le \max\{|p^k|, \varepsilon_k\} + |p^k||\hat{u}^k| \le \max\{|p^k|, \varepsilon_k\}(1 + |\hat{u}^k|).$$

(3) The equivalences follow from (2.13); in particular, $v_k \ge \varepsilon_k$. Next, by (2.11) with $\cdot = \hat{u}^k$ and $\check{\varepsilon}_g^k \le \varepsilon_g^{\max}$, and by (2.3) with $f_C(\hat{u}^k) = f(\hat{u}^k)$ ($\hat{u}^k \in C$), we have

$$-\varepsilon_k \le f_C(\hat{u}^k) - f_{\hat{u}}^k + \varepsilon_p^k \le \varepsilon_f^{\ell(k)} + \check{\varepsilon}_g^k \le \varepsilon_f^{\ell(k)} + \varepsilon_g^{\max}.$$

Finally, to obtain the bounds (2.14)–(2.16), use the equivalences together with the facts that $v_k \ge \varepsilon_k$, $-\varepsilon_k \le \varepsilon_f^{\ell(k)} + \varepsilon_g^{\max}$ and the bound on V_k from assertion (2). For instance, $v_k < -\varepsilon_k$ yields $0 \le t_k|p^k|^2/2 < -\varepsilon_k$ and $|p^k| < (-2\varepsilon_k/t_k)^{1/2}$ for (2.16). □

2.2 The method

We now have the necessary ingredients to state our method in detail.

Algorithm 2.3 (inexact proximal bundle method)

Step 0 (*Initialization*). Select $u^1 \in C$, a *descent parameter* $\kappa \in (0,1)$, a *stepsize bound* $t_{\min} > 0$ and a *stepsize* $t_1 \geq t_{\min}$. Call the oracle at u^1 to obtain f_u^1 and g^1 of (1.2), and set $\check{f}_1 := f_1$. Set $\hat{u}^1 := u^1$, $f_{\hat{u}}^1 := f_u^1$, $i_t^1 := 0$, $\ell(1) = 1$ and $k := 1$.

Step 1 (*Trial point finding*). Find the solution u^{k+1} of subproblem (2.2). Set v_k by (1.6), $p^k := (\hat{u}^k - u^{k+1})/t_k$, $\varepsilon_k := v_k - t_k|p^k|^2$ and V_k by (2.9).

Step 2 (*Stopping criterion*). If $V_k = 0$, stop.

Step 3 (*Stepsize correction*). If $v_k < -\varepsilon_k$, set $t_k := 10t_k$, $i_t^k := k$ and return to Step 1.

Step 4 (*Oracle call*). Call the oracle at u^{k+1} to obtain f_u^{k+1} and g^{k+1} of (1.2).

Step 5 (*Descent test*). If the descent test (1.7) holds, set $\hat{u}^{k+1} := u^{k+1}$, $f_{\hat{u}}^{k+1} := f_u^{k+1}$, $i_t^{k+1} := 0$ and $\ell(k+1) := k+1$ (*descent step*); otherwise, set $\hat{u}^{k+1} := \hat{u}^k$, $f_{\hat{u}}^{k+1} := f_{\hat{u}}^k$, $t_t^{k+1} := t_t^k$ and $\ell(k+1) := \ell(k)$ (*null step*).

Step 6 (*Stepsize updating*). If $\ell(k+1) = k+1$ (i.e., after a descent step), select $t_{k+1} \geq t_{\min}$; otherwise, either set $t_{k+1} := t_k$, or choose $t_{k+1} \in [t_{\min}, t_k]$ if $i_t^{k+1} = 0$.

Step 7 (*Model selection*). Choose a closed convex model $\check{f}_{k+1} : \mathbb{R}^n \to \mathbb{R}$ such that

$$\max\{\bar{f}_k, f_{k+1}\} \leq \check{f}_{k+1} \leq f + \varepsilon_g^{\max}. \tag{2.17}$$

Step 8 (*Loop*). Increase k by 1 and go to Step 1.

A few comments on the method are in order.

Remark 2.4 (1) When $\check{f}_k$ and C are polyhedral, Step 1 may use the QP method of [10], which can efficiently solve sequences of related subproblems (2.2).

(2) The stopping criterion of Step 2 is justified by the optimality estimate (2.12): $V_k = 0$ yields $f_{\hat{u}}^k - \varepsilon_p^k \leq \inf f_C = f_*$; thus, by (2.3), the point $\hat{u}^k$ is ε-*optimal* for $\varepsilon = \varepsilon_f^{\ell(k)} + \varepsilon_p^k$, i.e., $f(\hat{u}^k) \leq f_* + \varepsilon$. Step 2 may stop if $V_k \leq \tau_V$ for a tolerance $\tau_V > 0$. Section 3 below will show that this stopping criterion will be met, unless $f_{\hat{u}}^k \to -\infty$. More practicable stopping criteria are discussed in [13, §4.2].

(3) When $\varepsilon_g^{\max} = 0$, then after an exact descent step ($\varepsilon_f^{\ell(k)} = 0$), we have $v_k \geq \varepsilon_k \geq 0$ by Lemma 2.2(3), and Step 3 is redundant. When inexactness is discovered via the test $v_k < -\varepsilon_k$, the stepsize t_k is increased to produce a "safe" v_k for the descent test or confirm that the stability center $\hat{u}^k$ is already ε-optimal.

(4) At Step 4, we have $u^{k+1} \in C$ and $v_k > 0$ (by (2.15), since $V_k > 0$ after Step 2); hence Step 5 produces $\hat{u}^{k+1} \in C$ and $f_{\hat{u}}^{k+1} \leq f_{\hat{u}}^k$.

(5) Whenever t_k is increased at Step 3, the *stepsize indicator* $i_t^k \neq 0$ prevents Step 6 from decreasing t_k after null steps until the next descent step occurs (cf. Step 5). Otherwise, decreasing t_k at Step 6 aims at collecting more local information about the objective f at null steps. Step 6 may use the procedure of [9, §2] for updating the proximity weight $\rho_k := 1/t_k$, with obvious modifications.

(6) Step 7 may choose the simplest model $\check{f}_{k+1} := \max\{\bar{f}_k, f_{k+1}\}$. Our general requirement (2.17) accomodates more efficient choices based on aggregation or selection (see, e.g, [9], [14, §4.4]), and the nonpolyhedral SDP models of [7,14].

3 Convergence

Our analysis splits into several cases.

3.1 The case of an infinite cycle due to oracle errors

For our purposes, it suffices to give a very simple result on cycles between Steps 1 and 3; see [13, Lem. 2.3] for more sophisticated results. In this case, the algorithm drives $t_k \to \infty$ and $V_k \to 0$ for a fixed index k. Hence, for somewhat cleaner notation, in this context Step 3 may be replaced by the following.

Step 3_0 (*Stepsize correction*). If $v_k < -\varepsilon_k$, set $t_{k+1} := 10t_k$, $i_t^{k+1} := k$, $\hat{u}^{k+1} := \hat{u}^k$, $f_{\hat{u}}^{k+1} := f_{\hat{u}}^k$, $\ell(k+1) := \ell(k)$, $\check{f}_{k+1} := \check{f}_k$, increase k by 1 and go to Step 1.

Lemma 3.1 *If an infinite cycle between Steps* 1 *and* 3_0 *occurs, starting at iteration* $\bar{k}$, *then* $V_k \to 0$, $f_{\hat{u}}^{\bar{k}} \le f_* + \check{\varepsilon}_g^{\bar{k}}$ *and* $f(\hat{u}^{\bar{k}}) \le f_* + \varepsilon_f^{\ell(\bar{k})} + \check{\varepsilon}_g^{\bar{k}}$.

Proof For $k \ge \bar{k}$, we have $\hat{u}^k = \hat{u}^{\bar{k}}$, $f_{\hat{u}}^k = f_{\hat{u}}^{\bar{k}}$, $\ell(k) = \ell(\bar{k})$ and $\check{\varepsilon}_g^k = \check{\varepsilon}_g^{\bar{k}}$. Then, at Step 3_0, (2.16) with $t_k \uparrow \infty$ gives $V_k \to 0$. Hence (2.12) with $\varepsilon_p^k \le \check{\varepsilon}_g^k$ yields $f_{\hat{u}}^{\bar{k}} - \check{\varepsilon}_g^{\bar{k}} \le \inf f_C = f_*$, and the conclusion follows from (2.3). □

In view of Lemma 3.1, from now on we assume (unless stated otherwise) that the algorithm neither terminates nor cycles infinitely between Steps 1 and 3.

3.2 The case of finitely many descent steps

We now consider the case where only finitely many descent steps occur. After the last descent step, only null steps occur and the sequence $\{t_k\}$ becomes eventually monotone, since once Step 3 increases t_k, Step 6 cannot decrease t_k; thus the limit $t_\infty := \lim_k t_k$ exists. We first deal with the case of $t_\infty = \infty$.

Lemma 3.2 *Suppose there exists* $\bar{k}$ *such that only null steps occur for all* $k \ge \bar{k}$, *and* $t_\infty := \lim_k t_k = \infty$. *Let* $K := \{k \ge \bar{k} : t_{k+1} > t_k\}$. *Then* $V_k \overset{K}{\to} 0$ *at Step* 3.

Proof At iteration $k \in K$, before Step 3 increases t_k for the last time, we have the bound (2.16) with constant $\varepsilon_f^{\ell(k)} = \varepsilon_f^{\ell(\bar{k})}$; hence, $t_k \to \infty$ gives $V_k \overset{K}{\to} 0$. □

For the remaining case of $t_\infty < \infty$, we now give a fairly abstract result which shows that the *approximation errors*

$$\check{\gamma}_k := f_u^{k+1} - \check{f}_k(u^{k+1}) \tag{3.1}$$

vanish asymptotically, *independently* of the particular form (1.7) of our descent criterion. Further, instead of assuming that $\varepsilon_f^j \le \varepsilon_f^{\max}$ in (1.3) as in [13], we suppose that the oracle is *locally bounded* on C in the sense that

the sequence $\{g^k\}$ is bounded whenever the sequence $\{u^k\} \subset C$ is bounded. (3.2)

Note that the former condition implies the latter, since for $\varepsilon = \varepsilon_f^{\max} + \varepsilon_g^{\max}$, the mapping $\partial_\varepsilon f$ is locally bounded (see, e.g., [8, §XI.4.1]).

Lemma 3.3 *Suppose there exists $\bar{k}$ such that for all $k \geq \bar{k}$, we have $\hat{u}^k = \hat{u}^{\bar{k}}$ and $t_{\min} \leq t_{k+1} \leq t_k$. Further, assume that the oracle is locally bounded in the sense of* (3.2). *Then the approximation errors of* (3.1) *satisfy* $\overline{\lim}_k \check{\gamma}_k \leq 0$. *Moreover, if the descent criterion* (1.7) *fails for all $k \geq \bar{k}$, then $V_k \to 0$.*

Proof First, using partial linearizations $\bar{\phi}_k$ of the objectives ϕ_k of consecutive subproblems (2.2), we show that their optimal values $\phi_k(u^{k+1})$ are nondecreasing and bounded above.

Fix $k \geq \bar{k}$. By (2.10), (2.2) and (2.8), we have $\bar{f}_C^k(u^{k+1}) = \check{f}_k(u^{k+1})$ and

$$u^{k+1} = \arg\min \left\{ \bar{\phi}_k(\cdot) := \bar{f}_C^k(\cdot) + \tfrac{1}{2t_k}|\cdot - \hat{u}^k|^2 \right\} \tag{3.3}$$

from $\nabla \bar{\phi}_k(u^{k+1}) = 0$. Since $\bar{\phi}_k$ is quadratic and $\bar{\phi}_k(u^{k+1}) = \phi_k(u^{k+1})$, by Taylor's expansion

$$\bar{\phi}_k(\cdot) = \phi_k(u^{k+1}) + \tfrac{1}{2t_k}|\cdot - u^{k+1}|^2. \tag{3.4}$$

Next, since $\bar{f}_C^k(\hat{u}^k) \leq f(\hat{u}^k) + \varepsilon_g^{\max}$ by (2.7) with $\hat{u}^k \in C$, relations (3.4) and (3.3) yield

$$\phi_k(u^{k+1}) + \tfrac{1}{2t_k}|u^{k+1} - \hat{u}^k|^2 = \bar{\phi}_k(\hat{u}^k) \leq f(\hat{u}^k) + \varepsilon_g^{\max}. \tag{3.5}$$

Now, the minorizations $\bar{f}_k \leq \check{f}_{k+1}$ of (2.17) and $\bar{i}_C^k \leq i_C$ of (2.6) give $\bar{f}_C^k := \bar{f}_k + \bar{i}_C^k \leq \check{f}_{k+1} + i_C$; since we also have $\hat{u}^{k+1} = \hat{u}^k$ and $t_{k+1} \leq t_k$ by assumption, the objectives of (3.3) and the next subproblem (2.2) satisfy $\phi_k \leq \phi_{k+1}$. Hence by (3.4),

$$\phi_k(u^{k+1}) + \tfrac{1}{2t_k}|u^{k+2} - u^{k+1}|^2 = \bar{\phi}_k(u^{k+2}) \leq \phi_{k+1}(u^{k+2}). \tag{3.6}$$

Thus the nondecreasing sequence $\{\phi_k(u^{k+1})\}_{k \geq \bar{k}}$, being bounded above by (3.5) with $\hat{u}^k = \hat{u}^{\bar{k}}$ for $k \geq \bar{k}$, must have a limit, say $\phi_\infty \leq f(\hat{u}^{\bar{k}}) + \varepsilon_g^{\max}$. Moreover, since the stepsizes satisfy $t_k \leq t_{\bar{k}}$ for $k \geq \bar{k}$, we deduce from the bounds (3.5)–(3.6) that

$$\phi_k(u^{k+1}) \uparrow \phi_\infty, \quad u^{k+2} - u^{k+1} \to 0, \tag{3.7}$$

and the sequence $\{u^{k+1}\}$ is bounded. Then the sequence $\{g^k\}$ is bounded as well, since by our assumption the oracle is locally bounded in the sense of (3.2).

We now show that the approximation error $\check{\gamma}_k$ of (3.1) vanishes. Using the form (1.2) of f_{k+1}, the minorization $f_{k+1} \leq \check{f}_{k+1}$ of (2.17), the Cauchy-Schwarz inequality, and the optimal values of subproblems (2.2) with $\hat{u}^k = \hat{u}^{\bar{k}}$ for $k \geq \bar{k}$, we estimate

$$\begin{aligned}
\check{\gamma}_k &:= f_u^{k+1} - \check{f}_k(u^{k+1}) = f_{k+1}(u^{k+2}) - \check{f}_k(u^{k+1}) + \langle g^{k+1}, u^{k+1} - u^{k+2} \rangle \\
&\leq \check{f}_{k+1}(u^{k+2}) - \check{f}_k(u^{k+1}) + |g^{k+1}||u^{k+1} - u^{k+2}| \\
&= \phi_{k+1}(u^{k+2}) - \phi_k(u^{k+1}) + \Delta_k + |g^{k+1}||u^{k+1} - u^{k+2}|,
\end{aligned} \tag{3.8}$$

where $\Delta_k := |u^{k+1} - \hat{u}^{\bar{k}}|^2/2t_k - |u^{k+2} - \hat{u}^{\bar{k}}|^2/2t_{k+1}$. We have $\Delta_k \to 0$, since $t_{\min} \leq t_{k+1} \leq t_k$ for $k \geq \bar{k}$ by our assumption, $|u^{k+1} - \hat{u}^{\bar{k}}|^2$ is bounded, $u^{k+2} - u^{k+1} \to 0$ by (3.7), and thus

$$|u^{k+2} - \hat{u}^{\bar{k}}|^2 - |u^{k+1} - \hat{u}^{\bar{k}}|^2 = 2\langle u^{k+2} - u^{k+1}, u^{k+1} - \hat{u}^{\bar{k}} \rangle + |u^{k+2} - u^{k+1}|^2 \to 0.$$

Hence, using (3.7) and the boundedness of $\{g_{\underline{k}}^{k+1}\}$ in (3.8) yields $\overline{\lim}_k \check{\gamma}_k \le 0$.

Next, if the descent test (1.7) fails for $k \ge \bar{k}$, then $f_u^{k+1} > f_{\hat{u}}^k - \kappa v_k$ gives

$$\check{\gamma}_k = [f_u^{k+1} - f_{\hat{u}}^k] + [f_{\hat{u}}^k - \check{f}_k(u^{k+1})] > -\kappa v_k + v_k = (1-\kappa)v_k \ge 0, \tag{3.9}$$

where $\kappa < 1$ by Step 0; we conclude that $\check{\gamma}_k \to 0$ and $v_k \to 0$. Finally, since $v_k \to 0$, $t_k \ge t_{\min}$ and $\hat{u}^k = \hat{u}^{\bar{k}}$ for $k \ge \bar{k}$ by our assumption, we have $V_k \to 0$ by (2.15). □

We may now finish the case of infinitely many consecutive null steps.

Lemma 3.4 *Suppose there exists $\bar{k}$ such that only null steps occur for all $k \ge \bar{k}$, and the oracle is locally bounded in the sense of* (3.2). *Let $K := \{k : t_{k+1} > t_k\}$ if $t_k \to \infty$, $K := \mathbb{N}$ otherwise. Then $V_k \overset{K}{\to} 0$.*

Proof Steps 3, 5 and 6 ensure that the sequence $\{t_k\}$ is monotone for large k. We have $V_k \overset{K}{\to} 0$ from either Lemma 3.2 if $t_\infty = \infty$, or Lemma 3.3 if $t_\infty < \infty$. □

3.3 The case of infinitely many descent steps

Although our result for infinitely many descent steps does not involve the oracle errors explicitly, note that we might have $f_{\hat{u}}^\infty = -\infty$ if the objective errors ε_f^k were unbounded, giving "false" descent to $-\infty$, even when $f_* > -\infty$.

Lemma 3.5 *Suppose the set $\mathcal{D}$ of descent iterations is infinite and $f_{\hat{u}}^\infty := \lim_k f_{\hat{u}}^k > -\infty$. Then $\underline{\lim}_{k\in\mathcal{D}} V_k = 0$. Moreover, if $\{\hat{u}^k\}$ is bounded, then $V_k \overset{\mathcal{D}}{\to} 0$.*

Proof We have $0 \le \kappa v_k \le f_{\hat{u}}^k - f_{\hat{u}}^{k+1}$ if $k \in \mathcal{D}$, $f_{\hat{u}}^{k+1} = f_{\hat{u}}^k$ otherwise (see Step 5). Thus $\sum_{k\in\mathcal{D}} \kappa v_k \le f_{\hat{u}}^1 - f_{\hat{u}}^\infty < \infty$ gives $v_k \overset{\mathcal{D}}{\to} 0$ and hence $\varepsilon_k, t_k|p^k|^2 \overset{\mathcal{D}}{\to} 0$ by (2.14) and $|p^k| \overset{\mathcal{D}}{\to} 0$, using $t_k \ge t_{\min}$ (cf. Step 6). For $k \in \mathcal{D}$, $\hat{u}^{k+1} - \hat{u}^k = -t_k p^k$ by (2.8), so that

$$|\hat{u}^{k+1}|^2 - |\hat{u}^k|^2 = t_k\{t_k|p^k|^2 - 2\langle p^k, \hat{u}^k\rangle\}.$$

Sum up and use the facts that $\hat{u}^{k+1} = \hat{u}^k$ if $k \notin \mathcal{D}$, $\sum_{k\in\mathcal{D}} t_k \ge \sum_{k\in\mathcal{D}} t_{\min} = \infty$ to get

$$\overline{\lim_{k\in\mathcal{D}}}\{t_k|p^k|^2 - 2\langle p^k, \hat{u}^k\rangle\} \ge 0$$

(since otherwise $|\hat{u}^k|^2 \to -\infty$, which is impossible). Combining this with $t_k|p^k|^2 \overset{\mathcal{D}}{\to} 0$ gives $\underline{\lim}_{k\in\mathcal{D}} \langle p^k, \hat{u}^k\rangle \le 0$. Since also $\varepsilon_k, |p^k| \overset{\mathcal{D}}{\to} 0$, we have $\underline{\lim}_{k\in\mathcal{D}} V_k = 0$ by (2.9).

If $\{\hat{u}^k\}$ is bounded, using $\varepsilon_k, |p^k| \overset{\mathcal{D}}{\to} 0$ in Lemma 2.2(2) gives $V_k \overset{\mathcal{D}}{\to} 0$. □

3.4 Synthesis

Our principal result on the asymptotic objective value $f_{\hat{u}}^\infty := \lim_k f_{\hat{u}}^k$ follows.

Theorem 3.6 *Suppose Algorithm* 2.3 *neither terminates nor loops infinitely between Steps* 1 *and* 3 *(so that $k \to \infty$), the oracle is locally bounded in the sense of* (3.2) *with $\sup_k \varepsilon_g^k < \infty$, and its asymptotic error ε_f^∞ of* (1.9) *is finite if infinitely many descent steps occur. Let $\varepsilon_g^\infty := \underline{\lim}_{k\in K} \varepsilon_g^k$ (cf.* (2.11)), *where $K := \mathbb{N}$ if $f_{\hat{u}}^\infty := \lim_k f_{\hat{u}}^k = -\infty$; otherwise, let K be such that $V_k \overset{K}{\to} 0$ (such K exists by Lemmas* 3.4 *and* 3.5). *Then:*

(1) $f_* \le \underline{\lim}_k f(\hat{u}^k) \le \overline{\lim}_k f(\hat{u}^k) = f_{\hat{u}}^{\infty} + \varepsilon_f^{\infty}$, *where* f_* *is the optimal value of* (1.1).
(2) *We have* $f_{\hat{u}}^k \downarrow f_{\hat{u}}^{\infty} \le f_* + \varepsilon_g^{\infty}$, *and additionally* $V_k \xrightarrow{K} 0$ *if* $f_* > -\infty$.

Proof (1) For all k, we have $\hat{u}^k \in C$ and $f_* := \inf_C f \le f(\hat{u}^k) = f_{\hat{u}}^k + \varepsilon_f^{\ell(k)}$ by (2.3). Pass to the limit, with $f_{\hat{u}}^k$ converging to $f_{\hat{u}}^{\infty}$, and $\varepsilon_f^{\infty} < \infty$ in (1.9) by our assumption.

(2) By (1), if $f_{\hat{u}}^{\infty} = -\infty$, then $f_* = -\infty$. Hence, suppose that $f_* > -\infty$. Then $f_{\hat{u}}^{\infty} \ge f_* - \varepsilon_f^{\infty} > -\infty$ by (1), so Lemmas 3.4 and 3.5 guarantee the existence of K such that $V_k \xrightarrow{K} 0$. Pass to the limit in (2.12) to obtain $f_{\hat{u}}^{\infty} \le \inf f_C + \varepsilon_g^{\infty} = f_* + \varepsilon_g^{\infty}$. □

Remark 3.7 The bound $f_{\hat{u}}^{\infty} \le f_* + \varepsilon_g^{\infty}$ of Theorem 3.6 employs $\varepsilon_g^{\infty} := \underline{\lim}_{k \in K} \varepsilon_p^k$, where by (2.11b), $\varepsilon_p^k \le \check{\varepsilon}_g^k$ for *any* $\check{\varepsilon}_g^k$ such that $\check{f}_k \le f + \check{\varepsilon}_g^k$. In particular, if $\check{f}_k := \max_{j \in J_k} f_j$ and $\check{\varepsilon}_g^k := \max_{j \in J_k} \varepsilon_g^j$, then $\underline{\check{\varepsilon}}_g^{\infty} := \underline{\lim}_{k \in K} \check{\varepsilon}_g^k$ and $\overline{\check{\varepsilon}}_g^{\infty} := \overline{\lim}_{k \in K} \check{\varepsilon}_g^k$ satisfy $\varepsilon_g^{\infty} \le \underline{\check{\varepsilon}}_g^{\infty} \le \overline{\check{\varepsilon}}_g^{\infty}$. In this case, our (weaker) bound $f_{\hat{u}}^{\infty} \le f_* + \overline{\check{\varepsilon}}_g^{\infty}$ corresponds to the result of [4, §5], which assumes additionally that f is coercive and $C = \mathbb{R}^n$.

4 Using a partially inexact oracle

We now discuss using a partially inexact oracle (with $\varepsilon_g^j \equiv 0$) that satisfies the additional requirement of (1.10); our aim is to get $\varepsilon_f^{\infty} = \varepsilon_g^{\infty} = 0$ in Theorem 3.6.

Our modification of Algorithm 2.3 employs the counter n_D of exact *descent* steps, which is incremented at Step 5 at each descent step. As for initialization of n_D and the level λ_0 before the first oracle call at Step 0, we have two options:

- *exact initialization*: set $\lambda_0 := \infty$ and $n_D := 1$, or
- *inexact initialization*: set $\lambda_0 := -\infty$ and $n_D := 0$.

Note that $\varepsilon_f^1 = 0$ if $\lambda_0 = \infty$ in (1.10). At Step 0 choose a *model optimality* tolerance $\tau_m > 0$ (e.g., $\tau_m = \infty$). Steps 2 and 3 are replaced by

Step 2' (*Stopping criterion*). If $V_k = 0$ and $n_D > 0$, stop. Otherwise, if $V_k \le \tau_m$ and $n_D = 0$, set $v_k := -\infty$, $\lambda_k := \infty$ and go to Step 4.

Step 3' (*Inaccuracy detection*). If $v_k < -\varepsilon_k$, set $v_k := -\infty$. Set λ_k to λ_v^k of (1.11).

These steps are motivated as follows. It will be seen below that $f_{\hat{u}}^k = f(\hat{u}^k)$ if $n_D > 0$. Hence, if Step 2' stops, $\hat{u}^k$ is optimal by (2.12). However, if $n_D = 0$, we may have $f_{\hat{u}}^k < f_*$. Thus infinitely many null steps that drive V_k to 0 (cf. Lemma 3.3) could occur, but once $V_k \le \tau_m$ occurs (here $\tau_m > 0$ is crucial), this potential loop is broken, $\lambda_k = \infty$ forces the oracle at Step 4 to deliver the exact value $f_u^{k+1} = f(u^{k+1})$ by (1.10), whereas $v_k = -\infty$ forces Step 5 to make a *correcting* step to $\hat{u}^{k+1} := u^{k+1}$, which increments n_D as stated above. Similarly for Step 3': If initial oracle inaccuracy is detected ($\varepsilon_f^1 > 0$; cf. Remark 2.4(3)), the oracle is called with $\lambda_k = \infty$ and a correcting step occurs, which increments n_D. Further, having $\lambda_k = \lambda_v^k$ ensures that all descent steps occuring at Step 5 are *exact* with $\varepsilon_f^{k+1} = 0$ (cf. (1.7), (1.10) and (1.11)).

Thus, regarding exact initialization as a correcting step for $k = 0$, we have two cases:

(1) a correcting step occurs for some k, say k_c; then $\varepsilon_f^{\ell(k)} = 0$ for all $k > k_c$;
(2) inexact initialization followed by null steps only; then $n_D = 0$ forever.

In both cases, the convergence results of §3 apply with $\varepsilon_f^\infty = \varepsilon_g^\infty = 0$. Indeed, case (1) is obvious, whereas case (2) cannot occur under the assumptions of Lemma 3.3 (otherwise $V_k \to 0$ eventually gives $V_k \le \tau_m$ at Step 2' and a correcting step occurs).

In fact, our algorithm inherits the usual strong convergence properties of exact bundle methods. Instead of requiring that $\inf_k t_k \ge t_{\min} > 0$ as before, we consider more general stepsize conditions below.

Theorem 4.1 *Suppose that the oracle is partially inexact, and Algorithm* 2.3 *employs Steps* 2' *and* 3'. *Let* $U_* := \operatorname{Argmin}_C f$ *denote the (possibly empty) solution set of problem* (1.1). *Then we have the following statements.*

(1) *If there is* $\bar{k}$ *such that only null steps occur for all* $k \ge \bar{k}$, *the oracle is locally bounded in the sense of* (3.2), *and* $t_k \downarrow t_\infty > 0$, *then* $\hat{u}^k \in U_*$ *and* $V_k \to 0$.

(2) *Assuming that infinitely many descent steps occur, suppose that* $\sum_{k\in K} t_k = \infty$ *for* $K := \{k : f(\hat{u}^{k+1}) < f(\hat{u}^k)\}$. *Then* $f(\hat{u}^k) \downarrow f_*$. *Moreover, we have the following.*

(a) *Let* $\tilde{\gamma}_k := f(\hat{u}^{k+1}) - f_k(\hat{u}^{k+1})$ *for* $k \in K$. *If* $U_* \ne \emptyset$ *and* $\sum_{k\in K} t_k\tilde{\gamma}_k < \infty$ (*e.g.*, $\sup_{k\in K} t_k < \infty$), *then* $\{\hat{u}^k\}$ *converges to a solution* $\hat{u}^\infty \in U_*$, *and* $V_k \overset{K}{\to} 0$ *if* $\inf_{k\in K} t_k > 0$.

(b) *If* $U_* = \emptyset$, *then* $|\hat{u}^k| \to \infty$.

Proof Statement (1) follows from Lemma 3.3 and Theorem 3.6, with $\varepsilon_f^\infty = \varepsilon_g^\infty = 0$ as shown above. For (2), the proof of [11, Thm. 4.4] yields all claims except the final one on V_k in (a). For this claim, since $\hat{u}^k \to \hat{u}^\infty$ implies boundedness of $\{\hat{u}^k\}$, use the proof of Lemma 3.5 with $\mathcal{G} = K$ and $t_{\min}$ replaced by $\inf_{k\in K} t_k > 0$. □

Remark 4.2 (1) The local boundedness condition (3.2) holds in the min-max setting of (1.4) if $\partial F_z(\cdot)$ is locally bounded on C, uniformly w.r.t. $z \in Z$; e.g., when Z is finite.

(2) In tune with Remark 2.4(3), Step 3' could replace the test $v_k < -\varepsilon_k$ by the stronger test $\varepsilon_k < 0$; however, the former test is more robust w.r.t. roundoff errors.

(3) By Theorem 4.1, if $U_* \ne \emptyset$ and $0 < t_{\min} \le t_k \le t_{\max} < \infty$ for all k, then $\hat{u}^k \to \hat{u}^\infty \in U_*$. Hence for exact initialization, the efficiency estimates of [12] hold by their proofs. For inexact initialization, we may replace the test $V_k \le \tau_m$ of Step 2' by $w_k := t_k|p^k|^2/2 + \varepsilon_k \le \tau_m$. This replacement is valid because $0 \le w_k \le v_k$ when $v_k \ge -\varepsilon_k$ by Lemma 2.2(3), whereas $v_k \to 0$ in the proof of Lemma 3.3, so if n_D stayed null, we would eventually get $w_k \le \tau_m$, a contradiction. Then the analysis of [12] provides an upper bound on k_c, the iteration number of a correcting step with $w_{k_c} \le \tau_m$ or $v_{k_c} < -\varepsilon_{k_c}$, after which the "usual" estimates apply with k replaced by $k - k_c$.

5 The GGM method

We now state our extension of the GGM method of Gaudioso et al. [5] to the case of a partially inexact oracle (its relations with the original version will be discussed in §6.6). Relative to the PBM variant of §4, it replaces the PBM descent test (1.7) and level (1.11) by the GGM descent test (1.12) and level (1.13).

Algorithm 5.1 (GGM method)

Step 0 (*Initialization*). Select $u^1 \in C$, *optimality tolerances* $\tau_p > 0$ and $\tau_\varepsilon > 0$, an initial *stepsize* $t_1 > 0$ and an initial *gap* $\gamma_1 > 0$ such that

$$\tau_p^2 > \gamma_1 / t_1, \tag{5.1}$$

and a *stepsize decrease* parameter $\sigma > 1$. Set $\lambda_0 := -\infty$, call the oracle at u^1 to obtain f_u^1 and g^1 of (1.2), and set $\check{f}_1 := f_1$. Set $\hat{u}^1 := u^1$, $f_{\hat{u}}^1 := f_u^1$, $\ell(1) = 1$, $n_D := 0$, and $k := 1$.

Step 1 (*Trial point finding*). Find the solution u^{k+1} of subproblem (2.2). Set v_k by (1.6), $p^k := (\hat{u}^k - u^{k+1})/t_k$, $\varepsilon_k := v_k - t_k|p^k|^2$ and $\lambda_k := \lambda_\gamma^k$ for λ_γ^k of (1.13).

Step 2 (*First optimality test*). If $|p^k| > \tau_p$, go to Step 4.

Step 3 (*Second optimality test and stepsize updating*). If $\varepsilon_k \le \tau_\varepsilon$ and $n_D > 0$, stop. Otherwise, if $n_D = 0$, set $\lambda_k := \infty$ and go to Step 4. Else set $t_k := t_k/\sigma$, $\gamma_k := \gamma_k/\sigma$ and go back to Step 1.

Step 4 (*Oracle call*). Call the oracle at u^{k+1} to get f_u^{k+1} and g^{k+1} of (1.2) and (1.10).

Step 5 (*Descent test*). If the following descent test holds

$$f_u^{k+1} \le \lambda_k, \tag{5.2}$$

set $\hat{u}^{k+1} := u^{k+1}$, $f_{\hat{u}}^{k+1} := f_u^{k+1}$, $n_D := n_D + 1$ and $\ell(k+1) := k+1$ (*descent step*); otherwise, set $\hat{u}^{k+1} := \hat{u}^k$, $f_{\hat{u}}^{k+1} := f_{\hat{u}}^k$ and $\ell(k+1) := \ell(k)$ (*null step*).

Step 6 (*Stepsize updating*). If $\ell(k+1) = k+1$ (i.e., after a descent step), select $t_{k+1} \in [t_k, t_1]$ and set $\gamma_{k+1} := t_{k+1}\gamma_1/t_1$; otherwise, set $t_{k+1} := t_k$ and $\gamma_{k+1} := \gamma_k$.

Step 7 (*Model selection*). Choose $\check{f}_{k+1} : \mathbb{R}^n \to \mathbb{R}$ closed convex and such that

$$\max\{\bar{f}_k, f_{k+1}\} \le \check{f}_{k+1} \le f \quad \text{and} \quad f_{\ell(k+1)} \le \check{f}_{k+1}. \tag{5.3}$$

Step 8 (*Loop*). Increase k by 1 and go to Step 1.

Several comments on the method are in order.

Remark 5.2 (1) Step 0 employs inexact initialization. Step 1 chooses λ_k for the oracle condition (1.10) and the descent test (5.2) of Step 5 so that each descent step is exact.

(2) The aim of Algorithm 5.1 is to meet the *approximate optimality* condition

$$|p^k| \le \tau_p, \quad \varepsilon_k \le \tau_\varepsilon \quad \text{and} \quad f_{\hat{u}}^k = f(\hat{u}^k). \tag{5.4}$$

Note that (2.11) and the third part of (5.4) yield $p^k \in \partial_{\varepsilon_k} f_C(\hat{u}^k)$, and then the first two parts give the optimality estimate $f(u) \ge f(\hat{u}^k) - \tau_\varepsilon - \tau_p|u - \hat{u}^k|$ for all $u \in C$.

(3) When the first part of (5.4) fails at Step 2, Step 3 is skipped. If Step 3 does not stop, $n_D = 0$ means the third part of (5.4) could fail, so $\lambda_k = \infty$ forces the oracle to deliver $f_u^{k+1} = f(u^{k+1})$ and Step 5 makes a correcting step to $\hat{u}^{k+1} := u^{k+1}$; since n_D increases, this may happen at most once. If only the middle part of (5.4) fails, the stepsize t_k is decreased to reduce ε_k eventually (as will be seen in §6.1 below). Since t_k and γ_k are decreased by the same factor, the initial condition (5.1) is maintained:

$$\tau_p^2 > \gamma_1/t_1 = \gamma_k/t_k. \tag{5.5}$$

(4) At Step 6, t_k and γ_k can increase only after a descent step, and (5.5) is maintained.

(5) At Step 7, the first part of (5.3) repeats the standard model requirement (2.17); the additional second part is needed to bound ε_k via t_k (see §6.1 below). In fact, the second part of (5.3) may be omitted if Step 3 is modified as follows: Before returning to Step 1, if $f_{\ell(k)} \not\leq \check{f}_k$, choose another closed convex model $\check{f}_k$ such that $f_{\ell(k)} \leq \check{f}_k \leq f$.

6 Convergence of the GGM method

Again, we need to consider several cases.

6.1 Bounding the aggregate linearization error

Consider the descent predicted by the augmented model ϕ_k in subproblem (2.2):

$$w_k := f_{\hat{u}}^k - \phi_k(u^{k+1}) = t_k|p^k|^2/2 + \varepsilon_k, \tag{6.1}$$

where the second equality stems from (1.6), the first relation in (2.8) and (2.13). Further, let

$$G_k := \max\{|g^{\ell(j)}| : 1 \leq j \leq k\}. \tag{6.2}$$

Lemma 6.1 *At any iteration k of Algorithm 5.1, we have the following.*
(1) $\varepsilon_k \leq t_k|g^{\ell(k)}|^2/2$.
(2) $t_k \geq \min\{t_1, 2\tau_\varepsilon/(\sigma G_k)^2\}$ *for G_k given by* (6.2).
(3) *An infinite cycle between Steps* 1 *and* 3 *cannot occur.*

Proof (1) Let $\bar{k} = \ell(k)$. By Steps 0 and 5, $\hat{u}^k = u^{\bar{k}}$ and $f_{\hat{u}}^k = f_u^{\bar{k}}$. First, suppose $k = \bar{k}$. Then, using the bound $\check{f}_k \geq f_k$ (due to Step 0 or the first part of (5.3)) in subproblem (2.2) and the form (1.2) of f_k gives

$$\phi_k(u^{k+1}) \geq \min\left\{f_k(\cdot) + \tfrac{1}{2t_k}|\cdot - \hat{u}^k|^2\right\} = f_{\hat{u}}^k - t_k|g^k|^2/2.$$

Thus $w_{\bar{k}} \leq t_{\bar{k}}|g^{\bar{k}}|^2/2$ by (6.1). Next, suppose $k > \bar{k}$. Then $t_{j+1} \leq t_j$ for $j = \bar{k}, \ldots, k-1$ by Step 3, and hence $\phi_j(u^{j+1}) \leq \phi_{j+1}(u^{j+2})$ by (3.6) and $w_{j+1} \leq w_j$ by (6.1). Two cases may arise. First, if $t_k = t_{\bar{k}}$, the preceding relations give $w_k \leq t_k|g^{\bar{k}}|^2/2$. Second, if Step 3 decreases t_j for some j, then $\check{f}_j \geq f_{\bar{k}}$ (cf. Remark 5.2(5)); hence, replacing $\bar{k}$ by j in the previous argument, and repeating it if more stepsize decreases occur, we again get $w_k \leq t_k|g^{\bar{k}}|^2/2$. Since $\varepsilon_k \leq w_k$ in (6.1), the conclusion follows.

(2) By (1), before Step 3 divides t_k by $\sigma > 1$, we have $\tau_\varepsilon < \varepsilon_k \leq t_k|g^{\ell(k)}|^2/2$.

(3) An infinite cycle would drive t_k to 0, contradicting (2). □

6.2 Relating the gap and predicted descent levels

It turns out that the GGM descent steps may be regarded as descent steps of the PBM variant of §4 for a special choice of κ in the PBM descent test (1.7).

Lemma 6.2 *Consider any iteration k of Algorithm 5.1 after at least one descent step. Then at Step 5 we have the following.*

(1) $f_0^k = f(\hat{u}^k)$, $\varepsilon_k \geq 0$ *and* $v_k > 0$.

(2) *The level λ_γ^k of (1.13) is equal to the level λ_v^k of (1.11) with κ replaced by the descent coefficient $\kappa_k := 1 - \gamma_k/v_k < 1$. Moreover,*

$$\kappa_k \geq 1 - (1 - \kappa_{\mathrm{GGM}})\alpha_k^2 > \max\{\kappa_{\mathrm{GGM}}, 1 - \alpha_k^2\} \geq \kappa_{\mathrm{GGM}}, \tag{6.3}$$

where $\alpha_k := \tau_p/|p^k| < 1$ and

$$\kappa_{\mathrm{GGM}} := 1 - \frac{\gamma_1}{t_1 \tau_p^2} \in (0,1). \tag{6.4}$$

In particular, $f(\hat{u}^{k+1}) \leq f(\hat{u}^k) - \kappa_k v_k \leq f(\hat{u}^k) - \kappa_{\mathrm{GGM}} v_k$ if a descent step occurs.

Proof (1) Denoting the first descent iteration by $\bar{k}$, for $k > \bar{k}$ we have $f_0^k = f(\hat{u}^k)$ (cf. (1.10) at Step 4 and (5.2) at Step 5). Plugging $f_0^k = f_C(\hat{u}^k)$ into (2.11) yields $\varepsilon_k \geq 0$. Then $|p^k| > \tau_p$ (cf. Step 2) gives $v_k = t_k|p^k|^2 + \varepsilon_k \geq t_k|p^k|^2 > 0$ by (2.13).

(2) The first statement follows from (1.6), (1.11) and (1.13). Next, using the relations $v_k \geq t_k|p^k|^2$ and $\gamma_k/t_k = \gamma_1/t_1$ (cf. (5.5)) together with (6.4), we estimate

$$\kappa_k := 1 - \gamma_k/v_k \geq 1 - \frac{\gamma_k}{t_k|p^k|^2} = 1 - \frac{\gamma_1}{t_1\tau_p^2}\alpha_k^2 = 1 - (1 - \kappa_{\mathrm{GGM}})\alpha_k^2,$$

where $\alpha_k < 1$ from $|p^k| > \tau_p$ and $0 < \kappa_{\mathrm{GGM}} < 1$ by (5.1); the conclusion follows. □

Remark 6.3 (1) For practical comparisons of GGM and PBM, note that the lower bound of (6.3) is not tight unless $\varepsilon_k \ll t_k|p^k|^2$. Moreover, even if $\kappa_{\mathrm{GGM}} \ll 1$, κ_k is relatively large unless the first part of the stopping criterion (5.4) is almost met; e.g., $\kappa_k \geq 3/4$ if $|p^k| \geq 2\tau_p$.

(2) Note that Lemma 6.2 starts working after the first descent step or exact initialization. In contrast, for inexact initialization we may have many iterations (possibly with $v_k < 0$) until $|p^k| \leq \tau_p$ and a correcting step occurs. Here the ideas of §4 suggest a simple cure: If Step 1 produces $v_k < -\varepsilon_k$ (or just $\varepsilon_k < 0$, but see Remark 4.2(2)), set $\lambda_k := \infty$ and go to Step 4.

6.3 Analyzing successive null steps

The analysis of null steps is quite simple, thanks to our general Lemma 3.3.

Lemma 6.4 *Suppose the oracle is locally bounded in the sense of (3.2). Then Algorithm 5.1 cannot make infinitely many successive null steps.*

Proof For contradiction, suppose that starting from iteration k', null steps occur for all $k \geq k'$. By Lemma 6.1(2) and Steps 3 and 6, there is $\underline{k} \geq k'$ such that $t_k = t_{\underline{k}}$ and $\gamma_k = \gamma_{\underline{k}}$ for $k \geq \underline{k}$. Then, by Steps 1 through 5, for $k \geq \underline{k}$ each null step means that $\hat{\gamma}_k > \gamma_k = \gamma_{\underline{k}} > 0$ (cf. (1.13), (3.1), (5.2)). However, Lemma 3.3 gives $\overline{\lim}_k \hat{\gamma}_k \leq 0$, a contradiction. □

6.4 Analyzing infinitely many descent steps

Thanks to Lemma 6.2, our proof of Lemma 6.5 below may employ standard descent results from the exact bundle framework.

Lemma 6.5 *Suppose Algorithm* 5.1 *makes infinitely many descent steps. Then*:
(1) $f_0^{\infty} := \lim_k f(\hat{u}^k) = f_*$ *for the optimal value* f_* *of problem* (1.1).
(2) *Problem* (1.1) *has no solution.*
(3) $f_0^{\infty} = f_* = -\infty$ *if* $\sup_k G_k < \infty$ *in* (6.2).

Proof (1) Denoting the first descent iteration by $\bar{k}$, for $k > \bar{k}$ at Step 5 we have $f_0^k = f(\hat{u}^k) \geq f_*$, $p^k \in \partial_{\varepsilon_k} f_C(\hat{u}^k)$ (cf. Remark 5.2(2)) and, by Lemma 6.2, each descent step yields $f(\hat{u}^{k+1}) \leq f(\hat{u}^k) - \kappa_{\mathrm{GGM}} v_k$ with $v_k > 0$, $\kappa_{\mathrm{GGM}} > 0$ and $g^{k+1} \in \partial f(\hat{u}^{k+1})$. Moreover, since $t_k \leq t_1$ by Steps 3 and 6, the proximity weight $\rho_k := 1/t_k$ satisfies $\rho_k \geq 1/t_1$.

Suppose there is $\bar{u} \in C$ such that $f(\hat{u}^k) \geq f(\bar{u})$ for all $k > \bar{k}$. Then, since the preceding paragraph means that we are in the exact bundle framework, by standard arguments (e.g., [9, Lem. 3.1] with obvious translations), we have $v_k \underset{K}{\to} 0$ for $K := \{k > \bar{k} : f_0^{k+1} < f_0^k\}$ and the sequence $\{\hat{u}^k\}$ converges to some $\bar{u} \in C$.

Now, since $g^{k+1} \in \partial f(\hat{u}^{k+1})$ for $k \in K$, ∂f is locally bounded and $\hat{u}^k \to \bar{u}$, we have $\sup_k G_k < \infty$ in (6.2) and $\underline{\lim}_k t_k > 0$ by Lemma 6.1(2). Using this and $\varepsilon_k \geq 0$ in (2.13) with $v_k \underset{K}{\to} 0$, we obtain $p^k \underset{K}{\to} 0$ and $\varepsilon_k \underset{K}{\to} 0$. Thus eventually $|p^k| \leq \tau_p$ and $\varepsilon_k \leq \tau_\varepsilon$ imply termination at Step 3, a contradiction. Therefore, for each $u \in C$ we must have $f(\hat{u}^k) < f(u)$ for some $k > \bar{k}$; since $f_* \leq f(\hat{u}^k)$, we get $f_0^{\infty} = f_*$ from (1.1).

(2) Otherwise, take $\bar{u} \in \operatorname{Arg\,min}_C f$ to obtain a contradiction as in the proof of (1).

(3) For contradiction, suppose $f_0^{\infty} > -\infty$ and $\sup_k G_k < \infty$. By Lemma 6.1(2), $\sup_k G_k < \infty$ yields $\underline{\lim}_k t_k > 0$. Then the proof of Lemma 3.5 gives $\varepsilon_k, t_k |p^k|^2 \underset{K}{\to} 0$ and $|p^k| \underset{K}{\to} 0$ from $\underline{\lim}_k t_k > 0$. Thus eventually $|p^k| \leq \tau_p$ and $\varepsilon_k \leq \tau_\varepsilon$ imply termination at Step 3, a contradiction. □

6.5 Synthesis

We may now give our main result on finite termination of Algoritm 5.1.

Theorem 6.6 *Suppose the oracle is locally bounded in the sense of* (3.2). *Then Algorithm* 5.1 *stops after finitely many iterations under any of the following conditions.*
(1) *Problem* (1.1) *has at least one solution.*
(2) $f_* > -\infty$ *and* $\sup_k G_k < \infty$ *for* f_* *given by* (1.1) *and* G_k *by* (6.2).

Proof For contradiction, suppose the algorithm does not stop. Since neither an infinite cycle between Steps 1 and 3 (cf. Lemma 6.1) nor infinitely many successive null steps (cf. Lemma 6.4) can occur, infinitely many descent steps are made. Then Lemma 6.5 yields both assertions. □

Remark 6.7 Concerning the assumption $\sup_k G_k < \infty$ in Theorem 6.6, consider the level set $C_D := \{u \in C : f(u) \leq f(u^{k_D+1})\}$, where k_D is the iteration number of the first descent step if any, $k_D = 0$ otherwise. Then $\sup_k G_k < \infty$ if f is Lipschitzian on

C_D, or the subgradients delivered by the oracle are uniformly bounded for trial points in C_D. Indeed, it is easy to see that $\hat{u}^k \in C_D$ and $g^{\ell(k)} \in \partial f(\hat{u}^k)$ for $k > k_D$, because each descent step is exact.

6.6 Comparison with the original GGM setting

As noted in §1, the original GGM (OGGM for short) method employs a partially inexact oracle in the finite min-max framework of (1.4) with a finite set Z, whereas our GGM method can work with more general oracles. Further, OGGM works only in the unconstrained case of $C = \mathbb{R}^n$.

The OGGM method can be obtained from Algorithm 5.1 via the following modifications (and by using the *proximity weight* $\rho_k := 1/t_k$). First, replace Step 3 by

Step 3" (*Second optimality test and bundle reset*). If $\varepsilon_k \le \tau_\ell$ and $n_D > 0$, stop. Else if $n_D = 0$, set $n_D := 1$, $\lambda_k := \infty$, call the oracle at $u^{k+1} := \hat{u}^k$ to obtain f_u^{k+1} and g^{k+1}, and set $f_{\hat{u}}^k := f_u^{k+1}$, $g^{\ell(k)} := g^{k+1}$, $f_{\ell(k)} := f_{k+1}$. Reset $\check{f}_k := f_{\ell(k)}$, set $t_k := t_k/\sigma$, $\gamma_k := \gamma_k/\sigma$ and go back to Step 1.

Note that for $n_D = 0$, Step 3" calls the oracle at $\hat{u}^k$, whereas our version takes a correcting step to u^{k+1}, which may be better, being based on accumulated linearizations. Of course, our version could call the oracle at $\hat{u}^k$ as well, without affecting our convergence results. Anyway, this difference is minor, since a correcting step occurs at most once. On the other hand, the reset of $\check{f}_k := f_{\ell(k)}$ at Step 3" can hurt practical performance, since almost all accumulated information is lost. Our convergence results show that such resets are not necessary.

The second modification consists in setting $t_{k+1} := t_1$ and $\gamma_{k+1} := \gamma_1$ after a descent step (this choice is allowed in our Step 6). Note that resetting t_k and γ_k to their initial values need not be best in practice, unless t_1 and γ_1 are chosen carefully.

The third modification consists in choosing $\check{f}_{k+1} := \max\{\check{f}_k, f_{k+1}\}$ at Step 7; then (5.3) holds.

The OGGM convergence result of [5, Thm. 4.1] is a special case of our Theorem 6.6(2), assuming that $f_* > -\infty$ and f is Lipschitzian on $C_D' := \{u \in C : f(u) \le f(u^1)\}$ (the small difference relative to C_D in Remark 6.7 is due to Step 3" calling the oracle at $\hat{u}^k$). Our Theorem 6.6(1) for the OGGM method is new. Finally, we point out below two major flaws in the convergence analysis of [5].

Remark 6.8 (1) The proof of [5, Lem. 4.2] is flawed: in the setting of Lemma 3.3, for $d^k := u^{k+1} - \hat{u}^k$, it claims that $|d^{k+1} - d^k| \to 0$ (cf. (3.7)) implies convergence of $\{d^k\}$; this argument is obviously wrong.

(2) Reference [5, §5] also considers aggregation with $\check{f}_{k+1} := \max\{\overline{f}_k, f_{k+1}, \check{f}_k\}$ and $\overline{f}_k = \max_{j \in J_k} f_j$ for any $J_k \subset \{1, \dots, k\}$. Then the first part of (5.3) holds, but the second part may fail if $\ell(k+1) < k+1$ and $\ell(k+1) \notin J_k$ (e.g., $J_k = \emptyset$). This possible failure is ignored in the convergence analysis of [5, §5] (where the proof of [5, Lem. 4.2] needs $\check{f}_k \ge f_{\ell(k)}$), but it is handled easily in our analysis, since Step 3" employs $\check{f}_k = f_{\ell(k)}$ at resets; cf. Remark 5.2(5).

7 A modified GGM method

As announced in §1, our modified GGM (MGGM for short) method is obtained from Algorithm 5.1 by choosing $\kappa \in (0,1)$ at Step 0, and setting λ_k at Step 1 to the *modified level* $\hat{\lambda}_k$ of (1.14). This modification is justified below.

Lemma 7.1 *Suppose Step* 1 *of Algorithm* 5.1 *sets* λ_k *to the modified level* $\hat{\lambda}_f^k$ *of* (1.14) *for the levels* λ_f^k *of* (1.13) *and* λ_v^k *of* (1.11). *Let*

$$\hat{\kappa}_k := \min\{\kappa_k, \kappa\} \quad \text{and} \quad \hat{\kappa}_{\mathrm{GGM}} := \min\{\kappa_{\mathrm{GGM}}, \kappa\}, \tag{7.1}$$

where $\kappa_k := 1 - \gamma_k/v_k$, $\kappa \in (0,1)$ *and* κ_{GGM} *is given by* (6.4). *Then at Step* 5 *we have the following.*

(1) *If* $v_k \le 0$, *then* $\hat{\lambda}_f^k = \lambda_f^k > \lambda_v^k$.

(2) *If* $\varepsilon_k \ge 0$ (*e.g.*, $\ell(k) > 1$) *and* $\lambda_k = \hat{\lambda}_f^k$, *then* $v_k > 0$ *and* $\hat{\lambda}_f^k = f_0^k - \hat{\kappa}_k v_k$. *In particular,* $f_u^{k+1} \le f_0^k - \hat{\kappa}_k v_k \le f_0^k - \hat{\kappa}_{\mathrm{GGM}} v_k$ *if a descent step occurs.*

Proof (1) For contradiction, suppose $v_k \le 0$ and $\lambda_f^k \le \lambda_v^k$. Then by (1.6), (1.11) and (1.13), we have $0 < \gamma_k \le (1-\kappa)v_k$; since $\kappa < 1$, this gives $v_k > 0$, a contradiction.

(2) Arguing as in the proof of Lemma 6.2, we get $v_k > 0$ and

$$\hat{\lambda}_k = \max\{f_0^k - v_k + \gamma_k, f_{j^k}^k - \kappa v_k\} = f_0^k - \min\{1 - \gamma_k/v_k, \kappa\} v_k.$$

Since $\kappa_k := 1 - \gamma_k/v_k \ge \kappa_{\mathrm{GGM}}$ by (6.3), the conclusion follows from (7.1). □

Thanks to Lemma 7.1(2), $\hat{\kappa}_{\mathrm{GGM}}$ may replace κ_{GGM} in the proof of Lemma 6.5(1). In effect, the convergence results of §6 hold for the MGGM method as well.

8 Numerical illustrations

We conclude this paper with a brief comparison of the PBM and GGM variants in practice, for the motivating application of [6]: Lagrangian relaxation of the GAP.

8.1 The generalized assignment problem

In the GAP the objective is to find a maximum profit assignment of n jobs to m agents such that each job is assigned to precisely one agent subject to capacity restrictions on the agents. The standard formulation is the following

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \tag{8.1a}$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ij} x_{ij} \le c_i, \quad i \in M := \{1, \ldots, m\}, \tag{8.1b}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j \in N := \{1, \ldots, n\}, \tag{8.1c}$$

$$x_{ij} \in \{0,1\}, \quad i \in M, j \in N, \tag{8.1d}$$

where $p_{ij} \in \mathbb{Z}$ is the profit associated with assigning job j to agent i, $w_{ij} \in \mathbb{Z}_+$ the claim on the capacity of agent i by job j if it is assigned to agent i, $c_i \in \mathbb{Z}_+$ the capacity of agent i, and x_{ij} is a 0–1 variable indicating whether job j is assigned to agent i ($x_{ij} = 1$) or not ($x_{ij} = 0$).

The Lagrangian relaxation obtained by dualizing the semi-assignment constraints (8.1c) with unconstrained multipliers u_j, $j \in N$, gives the dual objective

$$f(u) := \max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} + \sum_{j=1}^{n} u_j \left(1 - \sum_{i=1}^{m} x_{ij}\right), \quad \text{s.t.} \quad (8.1b), (8.1d),$$

to be minimized over $C := \mathbb{R}^n$. This objective f has the additive structure

$$f(u) = \sum_{j=1}^{n} u_j + \sum_{i=1}^{m} z_i(u),$$

where

$$z_i(u) := \max \sum_{j=1}^{n} (p_{ij} - u_j) x_{ij}, \quad \text{s.t.} \quad (8.1b), (8.1d), \tag{8.2}$$

so that f is a max-type function which can be evaluated exactly at any u by solving the m binary knapsack problems in (8.2).

In our experiments, an exact oracle solved the knapsack problems with the branch and bound procedure MT1R of [20, §2.5.2]. As in [6, §5], our partially inexact oracle called the heuristic greedy procedure of [20, §2.4] first, reverting to MT1R only if the sum of $\sum_j u_j$ and the heuristic knapsack values did not exceed the target level for a null step. Next, as in [17, §2.2], our third *relatively inexact* oracle allowed MT1R to exit earlier when, for a given relative accuracy tolerance ε_r, its incumbent value ζ_i satisfied $\bar{\zeta}_i - \zeta_i \le \varepsilon_r |\zeta_i|$, where $\bar{\zeta}_i$ was MT1R's upper bound on $z_i(u)$ in (8.2). For $u = u^{k+1}$, this also gave the upper bound $f_{\text{up}}^{k+1} := \sum_j u_j + \sum_i \bar{\zeta}_i$ on $f(u)$, where ζ_i was replaced by $\bar{\zeta}_i$ when MT1R exited normally with an optimal solution. We also augmented the greedy procedure with MT1R's upper bound, and set $f_{\text{up}}^{k+1} := f(u^{k+1})$ for the exact oracle, so that for each oracle at iteration k, $f_{\text{min}}^{k} := \min_{j=1}^{k} f_{\text{up}}^{j}$ was our best upper bound on the optimal value f_*, with the relative error

$$\text{Rel\_err} := e_k := \left(f_{\text{min}}^{k} - f_*\right) / |f_*|. \tag{8.3}$$

As in [6, §5], for testing we used the 60 "small-size" instances C515-1 through C1060-5, and the 30 "large-size" instances of types A, B, C, D and E from the OR-Library[2]. Here we note that the small-size instances have names of the form Cxyy-z (or Cxxyy-z for $m \ge 10$), where x = m (or xx = m), yy = n, and z numbers the 5 instances from the same class. Similarly, the large-size instances have names of the form Txxyyy, where T is the type, xx = m and yyy = n.

[2] Maintained by J.E. Beasley at http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html.

8.2 Implementation

For testing we used a notebook PC (Pentium M 755 2 GHz, 1.5 GB RAM) under MS Windows XP, and Fortran 77. We developed four codes, called IPBM, PIPBM, GGM and MGGM, which implemented Algorithm 2.3 (inexact PBM), the partially inexact version of §4, Algorithm 5.1 and the modified variant of §7, respectively.

As in [20, §7.2.3] and [6, §4], the starting point had components $u_j^1 := \max_2\{p_{ij} : i \in M, w_{ij} \le c_i\}$, $j \in N$, where $\max_2$ denotes the second maximum.

Step 0 of Algorithm 2.3 set $\kappa := 0.1$, $t_1 := 1/|g^1|$ and $t_{\min} := 10^{-20}t_1$; Step 1 used the QP solver of [10] for (2.2); Step 6 updated t_k as in [9, §2], with [9, Eq. (2.17)] replaced by [13, Eq. (4.5)] for $R = 1$; Step 7 used subgradient selection (see, e.g., [14, §4.4]), storing up to $n+5$ subgradients.

We now explain the (practical) stopping criteria of IPBM and PIPBM.

For a given *gradient tolerance* $\varepsilon_g = 10^{-3}$ and a given relative *optimality tolerance* $\tau_{\text{opt}} = 10^{-5}$, IPBM stops at Step 2 of Algorithm 2.3 when $|p^k| \le \varepsilon_g\sqrt{n}$ and either

$$v_k \le \tau_{\text{opt}}\left(1+|f_a^k|\right) \tag{8.4}$$

and $v_k \ge -\varepsilon_k$, or $|p^k| + \varepsilon_k \le \tau_{\text{opt}}(1+|f_a^k|)$ (cf. [13, §4.2]). Similarly, for PIPBM the conditions $V_k = 0$ and $V_k \le \tau_m$ of Step 2' in §4 are replaced by $|p^k| \le \varepsilon_g\sqrt{n}$ and (8.4).

For GGM, since the parameters from [6, §5] gave very poor accuracy, we used the parameters suggested by M. Gaudioso[3]: $\tau_\varepsilon = 10^{-9}$, $\tau_p = 0.012$, $\gamma_1 = 0.02$, $\rho_1 := 1/t_1 = 0.005$ and $\sigma = 2$.

For MGGM, we set $\kappa := 0.1$, $\tau_\varepsilon := 0.001$ for the small-size instances, $\tau_\varepsilon := 0.01$ for the large-size instances, $\tau_p := \varepsilon_g\sqrt{n}$ with $\varepsilon_g = 10^{-3}$ as for IPBM and PIPBM, $\gamma_1 := 10$, $\rho_1 := 0.05$ and $\sigma := 10$; then, if (5.1) were violated, we set γ_1 to $0.99t_1\tau_p^2$.

8.3 Results

Tables 8.1 through 8.4 give the statistics for the four solvers on the small-size instances. In these tables, k_{av} and k_{mx} are respectively the average and maximum iteration counts for the 5 instances in the corresponding class, d_{av} is the average number of descent steps, e_{mx} is the maximum final relative error of (8.3) (with f_* computed to at least 15 digits in other runs), t_{av} is the average CPU time in seconds, and h_{av} is the average number of null steps due to the greedy heuristic; the final *all* row gives their aggregates over all the 60 instances.

Concerning Table 8.1, note that the low knapsack solution accuracy of $\varepsilon_r = 10^{-2}$ did not suffice for IPBM to obtain relative objective errors (i.e., e_{mx}) below 10^{-5}, but $\varepsilon_r = 10^{-3}$ was enough. We add that IPBM gave identical results for smaller tolerances up to $\varepsilon_r = 0$, i.e., exact knapsack solutions. Thus, in view of our results for the large-size instances (see below), we were quite disappointed to find that, in contrast with [17], our relatively inexact oracle did not provide any speedups against the exact oracle; in other words, our GAP instances seemed to be "easy" for MT1R.

[3] M. Gaudioso, private communication, October 15, 2008.

Table 8.1 IPBM with relatively inexact oracle: small-size instances (5 instances per row)

	relative accuracy $\varepsilon_r = 10^{-3}$					relative accuracy $\varepsilon_r = 10^{-2}$				
Class	k_{av}	k_{mx}	d_{av}	e_{mx}	t_{av}	k_{av}	k_{mx}	d_{av}	e_{mx}	t_{av}
C515	49	65	26	1.2E-15	0.00	49	65	26	1.2E-15	0.00
C520	50	71	30	8.1E-16	0.00	49	71	29	8.1E-16	0.00
C525	44	56	24	7.3E-06	0.00	44	56	23	7.3E-06	0.00
C530	76	127	40	3.3E-06	0.00	76	127	40	3.3E-06	0.00
C824	71	102	39	4.8E-06	0.00	71	102	39	3.9E-05	0.00
C832	87	117	40	8.2E-06	0.01	87	117	40	8.2E-06	0.01
C840	68	101	33	4.0E-06	0.00	70	101	33	4.0E-06	0.01
C848	135	165	51	5.7E-06	0.04	149	181	53	2.3E-06	0.05
C1030	77	98	37	6.4E-06	0.01	77	98	37	6.4E-06	0.01
C1040	92	176	37	5.9E-06	0.01	93	176	37	5.9E-06	0.02
C1050	130	263	47	5.3E-06	0.03	128	241	46	6.4E-06	0.04
C1060	114	133	45	8.7E-06	0.03	118	133	47	7.2E-06	0.04
all	83	263	37	8.7E-06	0.01	84	241	38	3.9E-05	0.01

Table 8.2 PIPBM: small-size instances (5 instances per row)

	Exact oracle					Partially inexact oracle					
Class	k_{av}	k_{mx}	d_{av}	e_{mx}	t_{av}	k_{av}	k_{mx}	d_{av}	h_{av}	e_{mx}	t_{av}
C515	49	65	26	1.2E-15	0.00	46	65	25	3	1.2E-15	0.00
C520	50	71	30	8.1E-16	0.00	52	88	26	2	3.6E-06	0.00
C525	44	56	24	7.3E-06	0.00	42	49	24	2	7.3E-06	0.00
C530	76	127	40	3.3E-06	0.00	80	127	40	1	1.2E-07	0.01
C824	71	102	39	4.8E-06	0.00	68	93	37	2	4.8E-06	0.01
C832	87	117	40	8.2E-06	0.01	87	118	40	1	8.2E-06	0.01
C840	68	101	33	4.0E-06	0.00	70	98	34	1	9.8E-06	0.01
C848	135	165	51	5.7E-06	0.03	130	165	49	1	5.7E-06	0.04
C1030	77	98	37	6.4E-06	0.00	71	87	35	2	8.3E-06	0.01
C1040	92	176	37	5.9E-06	0.01	110	177	42	1	5.0E-06	0.03
C1050	130	263	47	5.3E-06	0.04	121	263	42	2	9.5E-06	0.02
C1060	114	133	45	8.7E-06	0.03	115	142	44	2	7.2E-06	0.02
all	83	263	37	8.7E-06	0.01	83	263	37	2	9.8E-06	0.02

Table 8.3 GGM: small-size instances (5 instances per row)

	Exact oracle					Partially inexact oracle					
Class	k_{av}	k_{mx}	d_{av}	e_{mx}	t_{av}	k_{av}	k_{mx}	d_{av}	h_{av}	e_{mx}	t_{av}
C515	112	140	1	4.7E-15	0.00	154	187	1	100	5.0E-14	0.00
C520	143	182	1	2.0E-14	0.00	178	235	1	119	7.9E-15	0.00
C525	210	238	1	8.5E-15	0.01	273	328	1	193	4.4E-15	0.01
C530	312	357	2	1.3E-14	0.01	403	464	2	227	2.4E-14	0.01
C824	259	328	2	1.1E-14	0.01	374	438	2	259	4.1E-15	0.00
C832	440	535	2	2.1E-13	0.02	582	680	2	354	1.9E-14	0.02
C840	655	820	2	2.0E-15	0.12	900	1135	2	485	1.4E-15	0.14
C848	967	1228	3	4.7E-13	0.37	1337	1706	3	554	7.1E-15	0.52
C1030	389	468	2	1.5E-14	0.01	551	595	2	360	6.1E-15	0.02
C1040	618	709	5	4.4E-15	0.18	909	1029	4	546	4.2E-15	0.18
C1050	760	1082	2	1.1E-14	0.29	1026	1369	2	603	6.2E-15	0.40
C1060	942	1298	3	7.0E-15	0.51	1440	1772	3	982	7.5E-15	0.72
all	484	1298	2	4.7E-13	0.13	677	1772	2	398	5.0E-14	0.17

Table 8.4 MGGM: small-size instances (5 instances per row)

	Exact oracle					Partially inexact oracle					
Class	k_{av}	k_{mx}	d_{av}	e_{mx}	t_{av}	k_{av}	k_{mx}	d_{av}	h_{av}	e_{mx}	t_{av}
C515	38	42	10	2.7E-06	0.00	39	44	10	9	5.2E-16	0.00
C520	43	48	11	3.5E-15	0.00	47	64	11	13	8.1E-16	0.00
C525	61	63	11	6.0E-16	0.00	66	71	11	27	6.0E-16	0.00
C530	71	83	16	1.2E-15	0.00	82	91	17	22	1.3E-06	0.00
C824	70	88	13	1.2E-15	0.01	75	88	12	27	1.4E-15	0.00
C832	102	112	15	1.5E-15	0.01	107	113	14	33	1.9E-15	0.01
C840	133	176	17	9.6E-16	0.01	139	171	15	41	5.6E-07	0.02
C848	164	211	22	3.0E-08	0.03	172	217	23	30	5.2E-07	0.06
C1030	99	117	15	1.6E-15	0.00	105	125	13	34	4.5E-07	0.00
C1040	140	164	18	4.4E-07	0.02	155	180	18	44	2.1E-07	0.04
C1050	143	180	19	2.3E-07	0.03	150	180	20	45	1.9E-15	0.05
C1060	197	240	21	3.2E-07	0.08	206	254	17	65	4.0E-07	0.05
all	105	240	16	2.7E-06	0.02	112	254	15	32	1.3E-06	0.02

As for Table 8.2, its "Exact oracle" entries agree with the $\varepsilon_r = 10^{-3}$ entries in Tab. 8.1, but for PIPBM, the partially inexact oracle did not offer improvements, with the heuristic greedy procedure succeeding very rarely (see the h_{av} values).

Relative to Tab. 8.2, Table 8.3 exhibits quite dismal performance of GGM in terms of iteration counts and running times[4]. GGM made very few descent steps, and a lot of null steps. This is not really suprising, since GGM's descent test $f_u^{k+1} \le f_{\hat{u}}^k - \kappa_k v_k$ is hard to meet when κ_k is close to 1, whereas the average and minimum values of $1 - \kappa_k$ (over all iterations and instances) were 2×10^{-2} and 9×10^{-6}, respectively, for both oracles. At the same time, this helped the greedy heuristic to succeed quite frequently (see the h_{av} values).

Table 8.4 shows that MGGM performed much better than GGM. MGGM was about five times faster in iteration counts and running times, and about as fast as PIPBM, whereas MGGM's values of d_{av} and h_{av} were between those of PIPBM and GGM in Tabs. 8.2 and 8.3.

Tables 8.5 through 8.8 give our results on the large-size instances. In these tables, k is the final iteration number, N_d is the number of descent steps, Rel_err is the final relative error of (8.3), CPU is the CPU time in seconds, and N_h is the number of null steps due to the greedy heuristic; the final *all* row gives the averages of k, N_d, CPU and N_h, and the maximum of Rel_err, over all the 30 instances.

Concerning Table 8.5, note that the knapsack accuracy of $\varepsilon_r = 10^{-3}$ did not suffice for IPBM to obtain Rel_err $\le 10^{-5}$ for all instances, but $\varepsilon_r = 10^{-4}$ was enough. We add that IPBM gave almost identical results for smaller tolerances up to $\varepsilon_r = 0$.

As for Table 8.6, its "Exact oracle" entries agree with the $\varepsilon_r = 10^{-4}$ entries in Tab. 8.5 for all but 13 instances. Here we add that for fair comparisons with GGM and MGGM, PIPBM employed inexact initialization, and hence was not equivalent with IPBM even when their oracles delivered exact answers. For example, on instances A05100, A05200, A10200 and A20100, the starting point was optimal, but PIPBM

[4] Its parameters give very high accuracy in Tab. 8.3, but "normal" accuracy in Tab. 8.7.

Table 8.5 IPBM with relatively inexact oracle: large-size instances

Characteristics			relative accuracy $\varepsilon_r = 10^{-4}$				relative accuracy $\varepsilon_r = 10^{-3}$			
Type	m	n	k	N_h	Rel_err	CPU	k	N_h	Rel_err	CPU
A	5	100	3	0	0.0E+00	0.00	3	0	0.0E+00	0.00
A	5	200	4	0	0.0E+00	0.00	4	0	5.9E-05	0.00
A	10	100	21	10	7.0E-06	0.01	25	7	7.1E-05	0.00
A	10	200	6	0	0.0E+00	0.00	6	0	0.0E+00	0.00
A	20	100	29	0	0.0E+00	0.00	26	0	0.0E+00	0.00
A	20	200	33	14	7.8E-06	0.01	28	14	3.9E-06	0.00
B	5	100	264	89	2.1E-06	0.17	264	89	2.1E-06	0.17
B	5	200	419	107	1.8E-06	0.49	539	46	8.7E-05	0.54
B	10	100	74	34	4.4E-06	0.00	77	34	6.4E-06	0.00
B	10	200	291	63	2.4E-06	0.19	291	63	2.4E-06	0.22
B	20	100	87	39	3.1E-06	0.00	87	39	3.1E-06	0.00
B	20	200	401	139	4.5E-06	0.36	421	121	5.4E-06	0.45
C	5	100	231	84	9.0E-07	0.11	231	84	9.0E-07	0.05
C	5	200	338	93	8.8E-06	0.36	476	98	8.8E-06	0.48
C	10	100	265	93	2.5E-06	0.11	265	93	2.5E-06	0.06
C	10	200	460	99	5.3E-06	0.50	699	149	5.9E-06	0.94
C	20	100	192	58	2.6E-06	0.04	192	58	2.6E-06	0.02
C	20	200	588	112	2.6E-06	0.65	588	112	2.6E-06	0.70
D	5	100	147	43	6.9E-07	0.02	190	35	9.4E-05	0.02
D	5	200	198	46	1.2E-06	0.18	234	24	3.5E-04	0.13
D	10	100	191	53	2.7E-07	0.11	191	53	2.7E-07	0.11
D	10	200	230	45	1.3E-06	0.32	312	27	3.7E-04	0.37
D	20	100	241	54	1.0E-06	0.20	241	54	1.0E-06	0.21
D	20	200	325	61	7.3E-07	0.65	430	47	1.1E-04	1.04
E	5	100	163	45	2.4E-06	0.04	132	37	2.3E-06	0.02
E	5	200	143	39	2.9E-06	0.06	183	26	2.3E-04	0.02
E	10	100	220	57	2.7E-07	0.01	324	85	1.2E-06	0.19
E	10	200	919	208	1.5E-06	1.36	1398	31	5.0E-04	1.90
E	20	100	726	183	4.6E-06	0.69	726	183	4.6E-06	0.71
E	20	200	749	129	1.7E-06	1.80	1836	59	2.1E-04	3.97
	all		265	67	8.8E-06	0.28	347	56	5.0E-04	0.41

had to make a single correcting step. For the partially inexact oracle, the heuristic greedy procedure succeeded quite rarely (see the N_h values).

Tables 8.7 and 8.8 support our earlier discussion of Tabs. 8.3 and 8.4, with minor exceptions. Again, relative to PIPBM, GGM obtained Rel_err $\leq 10^{-5}$ for all instances at much larger iteration counts and running times, and made relatively few descent steps; yet, except for type A instances, the greedy heuristic did not succeed as frequently as in Tab. 8.3. In contrast, MGGM was comparable with PIPBM in the final accuracy, iteration counts and CPU times. Note that switching from the exact oracle to the partially inexact oracle did not reduce the average CPU times, but it did reduce the average counts of exact objective evaluations by the *same* percentage of 8% for PIPBM, GGM and MGGM (this count is $k - N_h$ for the partially inexact oracle).

The previous tables are summarized as performance profiles [3] in Figs. 8.1–8.2, which plot the portion of instances $\rho_s(\omega)$ on which a particular solver was not slower than the fastest solver by more than a given ratio ω. Here IPBM used $\varepsilon_r = 10^{-3}$ in Fig. 8.1 and $\varepsilon_r = 10^{-4}$ in Fig. 8.2, whereas PIPBM, GGM and MGGM employed

Table 8.6 PIPBM: large-size instances

Characteristics			Exact oracle				Partially inexact oracle				
Type	m	n	k	N_d	Rel_err	CPU	k	N_d	N_h	Rel_err	CPU
A	5	100	4	1	0.0E+00	0.00	4	1	3	0.0E+00	0.00
A	5	200	5	1	0.0E+00	0.00	5	1	4	2.8E-16	0.00
A	10	100	21	10	6.8E-06	0.01	25	9	7	7.1E-06	0.01
A	10	200	7	1	0.0E+00	0.00	7	1	6	0.0E+00	0.00
A	20	100	31	1	0.0E+00	0.00	26	1	25	2.9E-06	0.00
A	20	200	33	15	8.4E-06	0.00	36	14	18	7.7E-06	0.01
B	5	100	264	89	2.1E-06	0.19	265	76	3	4.0E-06	0.19
B	5	200	419	107	1.8E-06	0.51	475	115	9	1.2E-06	0.67
B	10	100	74	34	4.4E-06	0.00	73	32	1	3.6E-06	0.00
B	10	200	291	63	2.4E-06	0.17	323	68	3	2.4E-06	0.31
B	20	100	87	39	3.1E-06	0.00	73	26	1	1.8E-15	0.01
B	20	200	368	123	5.1E-06	0.36	431	132	2	4.9E-06	0.62
C	5	100	231	84	9.0E-07	0.14	190	66	2	0.0E+00	0.13
C	5	200	338	93	8.8E-06	0.27	378	107	8	8.4E-06	0.39
C	10	100	265	93	2.5E-06	0.05	265	93	1	2.5E-06	0.07
C	10	200	460	99	5.3E-06	0.50	564	136	2	5.5E-06	0.90
C	20	100	192	58	2.6E-06	0.00	186	54	1	7.2E-06	0.05
C	20	200	588	112	2.6E-06	0.72	588	112	1	2.6E-06	0.96
D	5	100	147	43	6.9E-07	0.02	141	41	22	1.2E-07	0.02
D	5	200	207	49	4.7E-07	0.18	176	40	30	4.7E-07	0.13
D	10	100	191	53	2.7E-07	0.11	205	48	14	1.9E-06	0.12
D	10	200	219	49	1.7E-06	0.37	244	53	30	3.1E-07	0.40
D	20	100	241	54	1.0E-06	0.19	263	60	22	7.4E-07	0.30
D	20	200	325	61	7.3E-07	0.66	302	60	28	8.6E-07	0.74
E	5	100	163	45	2.4E-06	0.03	140	43	17	2.3E-06	0.01
E	5	200	159	45	2.2E-15	0.09	136	35	18	2.7E-06	0.04
E	10	100	218	59	5.4E-07	0.00	206	57	11	5.0E-08	0.05
E	10	200	911	228	1.1E-06	1.46	866	218	14	9.7E-07	1.48
E	20	100	726	183	4.6E-06	0.72	761	186	4	4.3E-06	0.90
E	20	200	737	149	1.7E-06	1.80	229	56	15	1.5E-06	0.35
	all		264	68	8.8E-06	0.29	253	65	11	8.4E-06	0.30

the partially inexact oracle; further, zero CPU times were replaced by 0.01 due to the poor resolution of our timer.

The partially inexact oracle may be more useful when we impose a small limit, say $\bar{N}_{ex}$, on the number of exact objective evaluations, e.g., at each branch and bound node as discussed in [6, §5]. Tables 8.9 and 8.10 give results for $\bar{N}_{ex} = 10$ and 20 respectively, where e_{av} are the averages of the relative errors of (8.3). Note that MGGM was slightly better than PIPBM except for $\bar{N}_{ex} = 20$ and the partially inexact oracle, where PIPBM reduced both e_{av} and e_{mx} by more than 50% w.r.t. the exact oracle. GGM gained less from the partially inexact oracle in reducing e_{mx}.

Acknowledgements I would like to thank Paul Tseng for extensive comments on an earlier version of this paper, and two anonymous referees for helpful suggestions.

Table 8.7 GGM: large-size instances

Characteristics			Exact oracle				Partially inexact oracle				
Type	m	n	k	N_d	Rel_err	CPU	k	N_d	N_b	Rel_err	CPU
A	5	100	196	2	0.0E+00	0.02	221	2	216	2.7E-16	0.02
A	5	200	307	2	0.0E+00	0.09	254	2	252	9.8E-16	0.07
A	10	100	309	2	1.0E-15	0.15	357	2	350	1.3E-15	0.09
A	10	200	537	2	0.0E+00	0.31	605	2	600	0.0E+00	0.30
A	20	100	487	2	0.0E+00	0.29	530	2	520	2.6E-15	0.17
A	20	200	766	2	1.4E-14	0.72	1016	4	976	3.3E-15	1.04
B	5	100	1705	6	3.7E-16	1.00	1994	7	479	1.2E-15	1.03
B	5	200	3465	5	9.7E-07	6.39	4700	8	975	9.8E-07	9.76
B	10	100	738	3	3.2E-16	0.18	1183	2	732	2.7E-15	0.40
B	10	200	4352	6	2.2E-06	5.29	5118	6	1171	2.2E-06	6.78
B	20	100	1394	3	1.4E-15	0.41	2318	2	1251	9.8E-16	1.34
B	20	200	3388	5	1.9E-06	3.49	4979	6	2449	1.6E-06	5.77
C	5	100	1666	6	1.1E-15	0.96	2049	4	505	1.2E-16	1.19
C	5	200	3625	12	8.4E-06	6.96	4134	10	878	9.0E-06	7.80
C	10	100	1985	5	1.1E-15	0.87	2388	5	549	4.9E-16	1.10
C	10	200	4423	7	2.8E-06	6.27	4851	9	941	3.9E-06	7.80
C	20	100	1825	4	4.4E-15	0.68	2049	4	792	2.4E-15	1.22
C	20	200	5274	7	1.1E-06	8.09	7383	8	1840	1.1E-06	12.37
D	5	100	2619	7	4.6E-07	1.87	3286	7	897	4.5E-07	2.44
D	5	200	4249	8	1.5E-08	13.32	5735	8	1733	2.0E-08	15.76
D	10	100	2766	6	8.9E-07	2.72	3761	6	1092	9.1E-07	3.63
D	10	200	6634	12	2.5E-07	21.55	8143	10	2035	2.8E-07	22.92
D	20	100	2817	10	1.1E-07	3.01	3734	9	1252	1.1E-07	3.83
D	20	200	7118	5	9.2E-08	21.82	9252	7	2288	1.0E-07	27.83
E	5	100	3005	13	3.5E-11	2.21	3192	12	382	2.9E-09	2.44
E	5	200	8229	17	1.2E-15	18.71	8523	18	775	4.5E-15	19.27
E	10	100	4507	11	1.0E-06	2.54	5137	12	435	9.7E-07	3.51
E	10	200	13717	16	9.8E-07	56.99	12642	19	822	9.4E-07	48.45
E	20	100	6051	13	3.5E-06	4.59	6367	14	487	3.6E-06	5.77
E	20	200	18862	17	1.6E-06	61.36	20363	18	855	1.6E-06	73.54
	all		3901	7	8.4E-06	8.43	4542	8	951	9.0E-06	9.59

References

1. Belloni, A., Sagastizábal, C.: Dynamic bundle methods. Math. Program. **120**, 289–311 (2009)
2. Briant, O., Lemaréchal, C., Meurdesoif, Ph., Michel, S., Perrot, N., Vanderbeck, F.: Comparison of bundle and classical column generation. Math. Program. **113**, 299–344 (2007)
3. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
4. Emiel, G., Sagastizábal, C.: Incremental-like bundle methods with application to energy planning. Comput. Optim. Appl. ? (2009). DOI 10.1007/s10589-009-9288-8
5. Gaudioso, M., Giallombardo, G., Miglionico, G.: An incremental method for solving finite min-max problems. Math. Oper. Res. **31**, 173–187 (2006)
6. Gaudioso, M., Giallombardo, G., Miglionico, G.: On solving the Lagrangian dual of integer programs via an incremental approach. Comput. Optim. Appl. **44**, 117–138 (2009)
7. Helmberg, C., Kiwiel, K.C.: A spectral bundle method with bounds. Math. Program. **93**, 173–194 (2002)
8. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer, Berlin (1993)
9. Kiwiel, K.C.: Proximity control in bundle methods for convex nondifferentiable minimization. Math. Program. **46**, 105–122 (1990)

Table 8.8 MGGM: large-size instances

Characteristics			Exact oracle				Partially inexact oracle				
Type	m	n	k	N_d	ReLerr	CPU	k	N_d	N_h	ReLerr	CPU
A	5	100	40	1	0.0E+00	0.00	40	1	36	2.7E-16	0.00
A	5	200	58	2	0.0E+00	0.03	54	3	51	4.2E-16	0.01
A	10	100	84	7	3.0E-15	0.05	84	8	59	6.8E-06	0.00
A	10	200	97	5	0.0E+00	0.01	135	3	129	0.0E+00	0.09
A	20	100	98	5	0.0E+00	0.00	123	5	118	4.7E-15	0.00
A	20	200	157	9	2.2E-06	0.09	171	10	152	1.6E-06	0.14
B	5	100	127	28	1.4E-07	0.02	125	27	15	8.7E-16	0.00
B	5	200	194	29	2.3E-06	0.12	204	31	26	1.7E-07	0.07
B	10	100	129	16	8.1E-16	0.00	149	17	43	3.3E-06	0.06
B	10	200	314	34	3.4E-06	0.30	319	34	27	2.7E-06	0.32
B	20	100	230	20	1.4E-15	0.08	250	18	46	4.0E-06	0.10
B	20	200	529	29	3.1E-06	0.45	501	27	62	5.4E-06	0.56
C	5	100	141	23	1.3E-15	0.00	144	25	25	0.0E+00	0.00
C	5	200	170	29	4.0E-07	0.04	176	25	29	8.5E-07	0.09
C	10	100	179	27	6.4E-06	0.01	192	27	20	4.7E-15	0.03
C	10	200	299	33	3.8E-06	0.14	315	32	33	3.6E-06	0.23
C	20	100	254	26	6.1E-06	0.02	280	35	29	4.2E-06	0.12
C	20	200	485	37	4.3E-06	0.59	469	37	39	3.9E-06	0.67
D	5	100	209	44	8.7E-07	0.01	205	38	13	5.3E-07	0.02
D	5	200	280	42	5.4E-07	0.35	313	41	25	6.6E-07	0.34
D	10	100	296	49	1.3E-06	0.25	304	51	12	1.6E-06	0.30
D	10	200	442	52	7.2E-07	0.74	425	54	21	6.8E-07	0.78
D	20	100	400	53	1.6E-06	0.45	410	57	10	1.3E-06	0.57
D	20	200	613	55	8.9E-07	1.48	655	58	19	7.9E-07	2.02
E	5	100	200	60	1.2E-06	0.09	181	61	1	6.1E-07	0.05
E	5	200	591	103	2.3E-07	0.59	611	103	2	3.4E-07	0.70
E	10	100	357	75	6.1E-07	0.18	357	75	1	6.1E-07	0.18
E	10	200	466	76	4.5E-07	0.69	491	70	1	3.4E-07	0.90
E	20	100	434	72	3.3E-06	0.34	434	72	1	3.3E-06	0.43
E	20	200	780	81	1.2E-06	1.76	867	85	1	1.1E-06	2.47
	all		288	37	6.4E-06	0.30	299	38	35	6.8E-06	0.38

Table 8.9 At most $\bar{N}_{ex} = 10$ exact evaluations, large-size instances (30 instances per row)

	Exact oracle			Partially inexact oracle			
Code	e_{av}	e_{mx}	t_{av}	k_{av}	e_{av}	e_{mx}	t_{av}
PIPBM	8.3E-02	3.2E-01	0.00	12	5.3E-02	2.7E-01	0.00
GGM	3.9E-02	3.4E-01	0.00	489	4.1E-02	2.3E-01	0.24
MGGM	5.5E-02	2.4E-01	0.00	38	4.1E-02	2.3E-01	0.01

Table 8.10 At most $\bar{N}_{ex} = 20$ exact evaluations, large-size instances (30 instances per row)

	Exact oracle			Partially inexact oracle			
Code	e_{av}	e_{mx}	t_{av}	k_{av}	e_{av}	e_{mx}	t_{av}
PIPBM	2.2E-02	1.2E-01	0.00	27	9.6E-03	5.3E-02	0.01
GGM	8.3E-02	3.0E-01	0.00	552	4.1E-02	2.3E-01	0.28
MGGM	1.9E-02	8.3E-02	0.00	50	1.2E-02	7.5E-02	0.02

Fig. 8.1 Performance profile for small-size instances

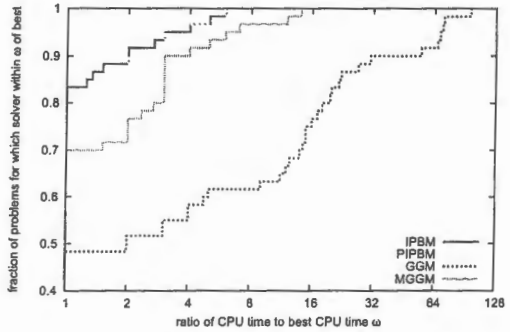
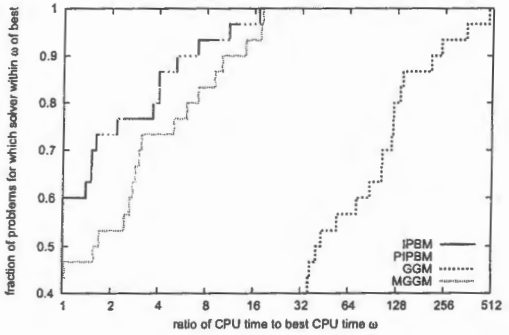


Fig. 8.2 Performance profile for large-size instances



10. Kiwiel, K.C.: A Cholesky dual method for proximal piecewise linear programming. Numer. Math. **68**, 325–340 (1994)
11. Kiwiel, K.C.: A projection-proximal bundle method for convex nondifferentiable minimization. In: M. Théra, R. Tichatschke (eds.) Ill-posed Variational Problems and Regularization Techniques, Lecture Notes in Economics and Mathematical Systems **477**, pp. 137–150. Springer-Verlag, Berlin (1999)
12. Kiwiel, K.C.: Efficiency of proximal bundle methods. J. Optim. Theory Appl. **104**, 589–603 (2000)
13. Kiwiel, K.C.: A proximal bundle method with approximate subgradient linearizations. SIAM J. Optim. **16**, 1007–1023 (2006)
14. Kiwiel, K.C.: A proximal-projection bundle method for Lagrangian relaxation, including semidefinite programming. SIAM J. Optim. **17**, 1015–1034 (2006)

15. Kiwiel, K.C.: A method of centers with approximate subgradient linearizations for nonsmooth convex optimization. SIAM J. Optim. **18**, 1467–1489 (2007)
16. Kiwiel, K.C.: An alternating linearization bundle method for convex optimization and nonlinear multicommodity flow problems. Math. Program. **?** (2009). DOI 10.1007/s10107-009-0327-0
17. Kiwiel, K.C.: An inexact bundle approach to cutting-stock problems. INFORMS J. Comput. **?** (2009). DOI 10.1287/ijoc.1090.0326
18. Kiwiel, K.C., Lemaréchal, C.: An inexact bundle variant suited to column generation. Math. Program. **118**, 177–206 (2009)
19. Lemaréchal, C.: Lagrangian relaxation. In: M. Jünger, D. Naddef (eds.) Computational Combinatorial Optimization, Lecture Notes in Computer Science 2241, pp. 112–156. Springer-Verlag, Berlin (2001)
20. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, New York (1990)
21. Sagastizábal, C., Solodov, M.V.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM J. Optim. **16**, 146–169 (2005)
22. Savelsbergh, M.W.P.: A branch-and-price algorithm for the generalized assignment problem. Oper. Res. **45**, 831–841 (1997)
23. Yagiura, M., Ibaraki, T., Glover, F.: An ejection chain approach for the generalized assignment problem. INFORMS J. Comput. **16**, 133–151 (2006)
24. Yagiura, M., Ibaraki, T., Glover, F.: A path relinking approach with ejection chains for the generalized assignment problem. European J. Oper. Res. **169**, 548–569 (2006)