Level Bundle Methods for Oracles with On Demand Accuracy

Article *in* Optimization Methods and Software · April 2014

DOI: 10.1080/10556788.2013.871282

2 authors:

Welington de Oliveira
Rio de Janeiro State University
22 PUBLICATIONS **180** CITATIONS
SEE PROFILE

Claudia Sagastizabal
National Institute for Research in Computer Sci…
122 PUBLICATIONS **2,143** CITATIONS
SEE PROFILE

LEVEL BUNDLE METHODS FOR ORACLES WITH ON-DEMAND ACCURACY

WELINGTON OLIVEIRA[1] AND CLAUDIA SAGASTIZÁBAL[2]

Abstract. For nonsmooth convex optimization, we consider level bundle methods built using an oracle that computes values for the objective function and a subgradient at any given feasible point. For the problems of interest, the exact oracle information is computable, but difficult to obtain. In order to save computational effort the oracle can provide estimations with an accuracy that depends on two additional parameters, informed to the oracle together with the evaluation point. The first of such parameters is a descent target while the second one is a bound for inexactness. If the oracle can reach the target with its function estimation, then the corresponding error is bounded by the second parameter. Otherwise, if the oracle detects that the target cannot be met, the function and subgradient estimations can be rough and have an unknown accuracy. The considered methods drive the inexactness parameter to zero, thus ensuring that an exact solution to the optimization problem is asymptotically found. The approach is comprehensive and covers known exact and inexact level methods as well as some novel variants, that can handle inaccuracy in a variable manner. In particular, when the feasible set is also compact, some of the new on-demand accuracy methods have the same rate of convergence of exact level variants known in the literature. A numerical benchmark on a battery of two-stage stochastic linear programs assesses the interest of the approach, substantially faster than the L-shaped method, without any accuracy loss.
Keywords: bundle methods, inexact oracles, nonsmooth optimization, level methods.

1. Introduction

We are interested in solving problems of the form

$$f_* := \inf_{x\in\mathcal{X}} f(x)\,, \tag{1}$$

for f a lower semicontinuous convex function and a convex closed set $\emptyset \neq \mathcal{X} \subset \Re^n$, typically polyhedral. We consider a setting such that for any given point x it is possible to compute exact values for $f(x)$ and a subgradient $g(x) \in \partial f(x)$, yet it is not desirable to do exact calculations too often. This is a common situation in large-scale problems: if each f/g evaluation involves heavy computations, it might be beneficial to have sometimes the flexibility of computing such values with some error. We refer to oracles making such f/g calculations as having "on-demand" accuracy because, depending on the input parameters, they have the ability of computing either exact values or just estimates for the function and subgradient.

For dealing with such on-demand oracles, we are interested in a special type of bundle methods, called level methods, introduced in [LNN95] and further studied by [Kiw95, BKL95, Fáb00, BTN05, Ric11]. The original level method [LNN95], defined for compact feasible sets in (1), lacks for the important bundle compression mechanism, that keeps bounded the amount of information maintained along iterations. Also, each iteration involves solving two optimization problems; first a linear program (to compute certain level parameter) and then a projection problem which is a quadratic program (to define a new iterate). Unlike [LNN95], which simply projects the last iterate, the proximal level variant [Kiw95] uses as projected point certain *stability center*, defined below. This modification eliminates the linear program solution and allows for bundle compression. The issue of how to handle unbounded feasible sets with the proximal level method was addressed in [BKL95]. Level methods were revisited from a mirror-descent perspective some years

Date: Version from November 25, 2012
1. Instituto Nacional de Matemática Pura e Aplicada, Rio de Janeiro, Brazil, wlo@impa.br
2. On leave from INRIA, France. Visiting researcher at IMPA, Brazil, sagastiz@impa.br .

later in [BTN05]. The resulting algorithm still requires the compactness assumption but replaces the quadratic program by a problem that takes advantage of the feasible set geometry. As a result, convergence rates become almost dimension-independent, making the approach attractive for problems with very large scale.

So far, level methods were developed on the basis of exact calculations, with the exception of [Fáb00] and more recently [Ric11]. The latter method considers inexact solution of the linear and quadratic programs, while the former is the only *level* method able to handle inexact oracle information. Indeed, all bundle methods dealing with approximate f/g-calculations, such as [Hin01, Sol03, Kiw06, ES10, OSS11, OSL12], are of the *proximal* form. Specifically, in any bundle method the oracle information is used to build linearizations for the objective function in (1). The piecewise maximum of linearizations defines a cutting-plane model, denoted by $\check{f}_k$ at iteration k. In turn, the model enters the quadratic program (QP) whose solution gives the next iterate. Proximal bundle methods [Kiw90, LS97] use the model in the QP objective function, while level methods use the model in the QP constraints, both in the non-proximal and proximal variants. Specifically, having a point $\hat{x}_k$ and prox- and level-parameters $\mu_k > 0$ and f_k^{lev}, respectively, the next iterate can be calculated

$$\begin{array}{rcll} \text{either as } x_{k+1} & = & \arg\min\left\{\check{f}_k(x) + \frac{1}{2}\mu_k|x-\hat{x}_k|^2 : x \in \mathcal{X}\right\} & \texttt{(prox-QP)} \\ \text{or as } x_{k+1} & = & \arg\min\left\{\frac{1}{2}|x-\hat{x}_k|^2 : \check{f}_k(x) \le f_k^{lev}\,,\, x \in \mathcal{X}\right\}. & \texttt{(level-QP)} \end{array}$$

Furthermore, in a proximal bundle method, $\hat{x}_k$ is the current stability center, which in some sense is the "best" point generated by the iterative process. By contrast, and as mentioned, in a level method the projection point $\hat{x}_k$ can be either the stability center (for proximal level methods) or just the last iterate (for non-proximal variants).

In theory, for properly chosen prox- and level-parameters and the same point $\hat{x}_k$, it is always possible to generate the same minimizer when solving the prox- or the level-QPs. However, when the cutting-plane model uses inexact linearizations, as in this work, we observed empirically that proximal bundle methods are more sensitive to oracle errors than the level variants, non-proximal and proximal. A possible explanation could be that the prox-parameter update relies on f/g-values at stability centers while the level-parameter changes on the basis of upper and lower *estimates* for the optimal value in (1). We focus our study on level bundle methods which, in addition to updating the level-parameter f_k^{lev} with a simple formula that remains efficient in the presence of oracle errors, are versatile regarding the choice of the projection point $\hat{x}_k$. We adopt an abstract setting that is general enough to extend the three level methods [LNN95], [Kiw95] and [BKL95], incorporating the ideas of [BTN05] when possible.

In addition to considering new forms of bundle methods, this work also contributes to the others main ingredient composing a nonsmooth optimization (NSO) solver for (1). This is the oracle information; our analysis considers novel cases and covers two particular on-demand accuracy oracles that are already known: the *asymptotically exact* oracles from [ZPR00] and [Fáb00], [FS07]; and the *partially inexact* oracle in [Kiw09]. More precisely, for stochastic programming problems, [ZPR00] solves subproblems with increasing accuracy, eventually in an exact manner. Similarly for the level decomposition [FS07] and [ZFEM09], based on [Fáb00], a non-proximal level method which can handle inexact data from the oracle, as long as the information is asymptotically exact. The numerical comparison in Section 6 shows that, for the considered test-cases, our methodology compares favorably to these approaches, thanks to the versatility of the on-demand setting. More precisely, asymptotically exact oracles are cheap just at the beginning of the iterative process, but become increasingly computationally expensive as iterations go by. Instead, we can save computational time by considering oracles that are cheap at most iterations, without losing the important property of eventually finding an exact solution to problem (1). We refer to such oracles as being *partly asymptotically exact*: exact information is required eventually only for "good" iterates, that yield progress towards the goal of solving (1).

Partly asymptotically exact oracles bear some resemblance with the partly inexact oracle in [Kiw09], which uses as NSO solver a proximal bundle method. In [Kiw09], the oracle must provide exact information when a descent criterion is met. We consider a less demanding setting, in which, instead of exact oracle computations for some points, exactness is required only *asymptotically*.

The numerical comparison in Section 6 shows the interest of designing algorithms able to handle more versatile oracles, as in our approach. For the considered battery of test problems and without any loss in accuracy, our *partly asymptotically exact* variants defined below are far superior to the exact methods [SW69], [BGLS06], [LNN95], [Kiw95], [BKL95]. Furthermore, such variants are systematically faster than the inexact methods [Kiw09], [ZPR00], [Fáb00].

This work is organized as follows. Section 2 specifies the conditions for an oracle to have on-demand accuracy, and shows how to build such oracles for two-stage stochastic linear programs. Section 3 analyzes proximal and non-proximal level methods for oracle with on-demand accuracy, including complexity estimates, for the case when the feasible set in (1) is compact. The compactness assumption is removed for the proximal level variants considered in Section 4. The exact proximal level method in [BTN05] for compact feasible sets is generalized to handle on-demand accuracy oracles in Section 5. Despite the oracle inaccuracy, we show that the complexity estimates for our on-demand accuracy method and the exact variant [BTN05] are comparable. Some encouraging numerical experiments are presented in Section 6, devoted to the applications of our methods to a battery of two-stage stochastic linear programs found in the literature. The final Section 7 contains concluding comments and remarks.

Regarding notation, we mostly follow [Kiw06] and [OSS11]. We endow the Euclidean space $\Re^n$ with an abstract inner product $\langle\cdot,\cdot\rangle$. For $x\,, g \in \Re^n$ the corresponding induced primal and dual norms are respectively given by

$$|x| = \sqrt{\langle x, x\rangle} \;\text{ and }\; \|g\| = \max\{\langle g, x\rangle : \text{ for all } x \in \Re^n \text{ such that } \; |x| \le 1\}\,.$$

2. Oracles with on-demand accuracy

We assume the f/g calculations can be more or less exact, depending on certain parameters informed to the oracle together with the evaluation point. More precisely, given $x \in \mathcal{X}$, together with a descent target $\gamma_x \in \Re \cup \{+\infty\}$ and an error bound $\epsilon_x \ge 0$, we suppose the *oracle with on-demand accuracy* provides

$$\left[\begin{array}{ll} \text{as function information} & f_x = f(x) - \eta(\gamma_x) \text{ with } \eta(\gamma_x) \ge 0, \\ \text{as subgradient information} & g_x \in \partial_{\eta(\gamma_x)} f(x)\,. \\ \text{Moreover, whenever } f_x \le \gamma_x, & \text{the relation } \eta(\gamma_x) \le \epsilon_x \text{ holds.} \end{array}\right. \tag{2}$$

In the oracle definition, the actual error $\eta(\gamma_x)$ is unknown. However, the conditions on the function inaccuracy are such that the exact function value satisfies the relation

$$f(x) \in \big[f_x, f_x + \epsilon_x\big] \quad \text{whenever} \quad f_x \le \gamma_x\,.$$

Therefore, inaccuracy is controlled only at those function values considered good enough with respect to the descent target. When the algorithm evaluates the function at "bad" points, which do not yield progress in terms of minimizing the objective and, hence, do not reach the target, the f/g-estimates can be rough.

The versatility of our inexact oracle becomes clear from the following particular instances:

(Partly Inexact): Setting the error bound $\epsilon_x = 0$ amounts to the partially inexact oracle in [Kiw09].

(Exact): If, in addition to supposing that the error bound is null, we had, $\gamma_x = +\infty$ then the oracle (2) would be an exact oracle.

(Inex): If $\gamma_x = +\infty$ but $\epsilon_x > 0$ (possibly unknown), then (2) is the inexact oracle considered in [Hin01], [Sol03], [Kiw06] and [OSS11].

(Asymptotically Exact): Still supposing $\gamma_x = +\infty$, if $\epsilon_x \to 0$ along the iterative process, then (2) subsumes the asymptotically exact oracles from [ZPR00], [Fáb00], and [ES10].

(Partly Asymptotically Exact): A new instance comes up when we set $\gamma_x < +\infty$ and $\epsilon_x \to 0$.

The computational effort of the options in this list diverse; the last instance is the less demanding one. We will show that all the variants eventually guarantee that an exact solution to (1) is found.

Our convergence results require the on-demand accuracy oracle to satisfy a boundedness assumption, as follows:

Assumption 2.1. *For the oracle error in* (2)*, we assume there exists a positive constant* $\bar{\eta}$ *such that*

$$\eta(\gamma_x) \leq \bar{\eta} \;\; \textit{for all} \;\; x \in \mathcal{X}\,.$$

When the feasible set in (1) is bounded, we let $\Lambda > 0$ denote a Lipschitz constant for the objective function over the set $\mathcal{X}$, computed with respect to the norm $|\cdot|$. By [HUL93, Prop.XI.4.1.2] the approximate subdifferentials $\partial_{\eta(\gamma_x)} f(\cdot)$ are locally bounded. For this reason, when Assumption 2.1 holds, by compactness of $\mathcal{X}$, the Lipschitz constant is a bound for the approximate subdifferentials, uniform for all $x \in \mathcal{X}$. In particular, the approximate subgradients g_x computed by (2) are uniformly bounded:

$$\|g_x\| \leq \Lambda \quad \text{for all} \quad x \in \mathcal{X}\,. \tag{3}$$

Assumption 2.1 is reasonable and not too restrictive, as illustrated by the examples below.

Example 2.2 (On-demand accuracy oracles for Minimax problems)**.** *Suppose that*

$$f(x) := \sup_{u \in U \subset \Re^m} h(u,x)$$

with $h(u,\cdot) : \Re^n \to \Re$ *convex for each* u*. Such functions appear for example in combinatorial problems and Lagrangian relaxation,* [BGLS06]*. Therefore, producing an approximate maximizer of* $h(\cdot,x)$ *does produce the required information* (2)*: for a given* $\tilde{x}$*, any given* $\tilde{u} \in U$ *gives an approximation to* $f(\tilde{x})$ *and* $g \in \partial f(\tilde{x})$ *by taking* $f_{\tilde{x}} := h(\tilde{u},\tilde{x})$ *and* $g_{\tilde{x}}$ *some subgradient of* $h(\cdot,\tilde{x})$ *at* $\tilde{u}$*, i.e.,* $g_{\tilde{x}} \in \partial_x h(\tilde{u},\tilde{x})$*. Then, convexity of* $h(\tilde{u},\cdot)$ *and definition of* f *give*

$$h(\tilde{u},\tilde{x}) + \langle g_{\tilde{x}}, x - \tilde{x}\rangle \leq h(\tilde{u},x) \leq f(x) \;\; \textit{for all} \;\; x \in \Re^n\,.$$

Therefore, it follows that $f_{\tilde{x}} + \langle g_{\tilde{x}}, x - \tilde{x}\rangle \leq f(x)$ *and*

$$f(\tilde{x}) + \langle g_{\tilde{x}}, x - \tilde{x}\rangle - \tilde{\eta} \leq f(x) \;\; \textit{for all} \;\; x \in \Re^n$$

with $\tilde{\eta} := f(\tilde{x}) - f_{\tilde{x}} \geq 0$*. Thus,* $g_{\tilde{x}} \in \partial_{\tilde{\eta}} f(\tilde{x})$ *and the first two lines in* (2) *are established. To obtain the third line, suppose that it is given an optimality tolerance* $\epsilon_{\tilde{x}}$ *for solving the subproblem* $\sup_{u\in U} h(u,\tilde{x})$*. Besides, let* $\gamma_{\tilde{x}} \in \Re$ *be a target satisfying* $f(\tilde{x}) \leq \gamma_{\tilde{x}}$*. Then, the condition*

$$f_{\tilde{x}} \leq \gamma_{\tilde{x}} \quad \textit{implies} \quad \eta(\gamma_{\tilde{x}}) \leq \epsilon_{\tilde{x}}$$

can be obtained if $\tilde{u} \in U$ *is chosen to be a* $\epsilon_{\tilde{x}}$*-solution to* $\sup_{u\in U} h(u,\tilde{x})$*. If for instance* $f(\tilde{x}) > \gamma_{\tilde{x}}$*, then* $\tilde{u} \in U$ *can be the first iterate* u *of the optimization process (for the subproblem) such that* $f_{\tilde{x}} > \gamma_{\tilde{x}}$ *is verified.*

Finally, notice that (3) *holds for instance if both sets* $\mathcal{X}$ *and* U *are bounded, and if* $h(\cdot,x)$ *is concave for any given* $x \in \mathcal{X}$*.*

Example 2.3 (On-demand accuracy oracles for stochastic programming)**.** We now show how to build, for two-stage stochastic programming problems, an inexact oracle in the form (2) satisfying Assumption 2.1.

We consider two-stage stochastic linear programming problems with fixed recourse and deterministic second-stage cost. In such a context, the discretization of uncertainty into N scenarios yields a problem like (1), with

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

where $c \in \Re^n$, the matrix $A \in \Re^{m_1 \times n}$ and the vector $b \in \Re^{m_1}$ are such that the set $\mathcal{X}$ is compact. Also,

$$Q_i(x) := \min_{y \in \Re^{n_2}_+} \langle q, y\rangle \;\; \text{s.t.} \;\; T_i x + W y = h_i$$

is the recourse function corresponding to the i-th scenario (h_i, T_i), with probability $p_i > 0$ for $h_i \in \Re^{m_2}$ and $T_i \in \Re^{m_2 \times n}$.

Letting $W \in \Re^{m_2 \times n_2}$ and $q \in \Re^{n_2}$, the recourse function can be written in its dual form, as follows:

$$Q_i(x) := \max_{u \in \Re^{m_2}} \langle u, h_i - T_i x \rangle \quad \text{s.t.} \quad W^\top u \leq q\,. \tag{4}$$

By [SDR09, Props. 2.1 and 2.2], the expected recourse function $Q(x) := \sum_{i=1}^N p_i Q_i(x)$ is convex with subgradient

$$-\sum_{i=1}^N p_i T_i^\top u_i \in \partial Q(x)$$

where u_i is a solution (if any) of problem (4), for $i = 1, \ldots, N$. Moreover, when N is finite and $Q_i(x) < \infty$ for each $i = 1, \ldots, N$ and at least one $x \in \mathcal{X}$, then by [SDR09, Prop. 2.3] f is polyhedral, and thus, locally Lipschitz continuous.

For these problems, the effort for evaluating the function f and a subgradient amounts to solving N linear programs. In particular, when the number of scenarios is large, making the exact f/g-evaluation becomes computationally heavy. To speed up calculations, for some specific points one can compute the oracle information only approximately, by skipping the exact solution of (4) for all, or almost all, scenarios (h_i, T_i), replacing the missing information by some sound value.

This is the purpose of the strategy described below, defining an oracle with on-demand accuracy that satisfies (2).

Inexact Oracle 2.4 (Dual Feasibility strategy)**.**

Step 0 (initialization). A feasible point $x \in \mathcal{X}$, together with a descent parameter γ_x and an error bound $\epsilon_x \geq 0$, are given. Also a set $\mathcal{D}_{LP}$ with finitely many feasible points for (4) may be available. If such is the case, go to Step 1. Otherwise, if $\mathcal{D}_{LP} = \emptyset$, go to Step 2.

Step 1 (fast approximation). For $i = 1, \ldots, N$, compute $u_i := \arg\max_{u \in \mathcal{D}_{LP}} \langle u, h_i - T_i x \rangle$. Set $scen = 0$ and go to Step 3.

Step 2 (full ϵ_x-calculations). For $i = 1, \cdots, N$, find u_i feasible for (4) such that

$$Q_i(x) - \langle u_i, h_i - T_i x \rangle \leq \epsilon_x\,, \tag{5}$$

and add u_i to $\mathcal{D}_{LP}$. Set $scen = N$.

Step 3 (evaluation). Define the approximations $f_x := \langle c, x \rangle + \sum_{i=1}^N p_i \langle u_i, h_i - T_i x \rangle$ and $g_x := c - \sum_{i=1}^N p_i T_i^\top u_i$. If $scen = N$, return.

Step 4 (comparison with target). If $f_x > \gamma_x$, return.

Otherwise, set $scen = scen + 1$, $i = scen$, and find u_i satisfying (5). Add u_i to $\mathcal{D}_{LP}$, and go back to Step 3. □

The (finite) dual set $\mathcal{D}_{LP}$ can be as small or large as desired, keeping in mind that larger sets $\mathcal{D}_{LP}$ yield more accurate values for f_x/g_x.

By construction, the Dual Feasibility Strategy is an oracle with on-demand accuracy satisfying (2). When the procedure returns from Step 3, the descent target may or may not be attained. If at Step 4 $f_x \leq \gamma_x$, the inequality (5) ensures that $\eta(\gamma_x) \leq \epsilon_x$ because adding and subtracting $\langle c, x \rangle$ in (5) yields $0 \leq \eta(\gamma_x) = f(x) - f_x \leq \epsilon_x$. Since $\mathcal{X}$ is bounded, [OSS11, Prop. 4.1] shows that the evaluation error $\eta(\gamma_x)$ is bounded for all $x \in \mathcal{X}$ and, hence, Assumption 2.1 holds.

3. Level bundle method with on-demand accuracy

The level bundle method with on-demand accuracy given in this section is based on [LNN95], [Kiw95], and [Fáb00], incorporating additional steps to deal with inaccurate evaluations that become asymptotically exact only at some iterates.

We consider a unified framework for taking into account the non-proximal and proximal variants. As such, the point $\hat{x}_k$ in the level-QP is referred to as a *projection point*, to emphasize its potential difference from the stability center employed by the proximal level method.

3.1. **Theoretical development.** The method generates a sequence of feasible iterates $\{x_k\} \subset \mathcal{X}$. The corresponding f_{x_k}/g_{x_k}-information is given by an oracle with on-demand accuracy as in (2). With such information, the method creates *approximate linearizations*

$$f_{x_k} + \langle g_{x_k}, \cdot - x_k \rangle \leq f(\cdot)\,, \tag{6}$$

where the inequality follows from having $\eta(\gamma_{x_k}) = f(x_k) - f_{x_k} \geq 0$ in (2). At iteration k, a polyhedral *inexact cutting-plane model* of f

$$\check{f}_k(\cdot) := \max_{j \in J_k} \{f_{x_j} + \langle g_{x_j}, \cdot - x_k \rangle\} \quad \text{with} \quad J_k \subset \{1, \ldots, k\} \tag{7}$$

is available, and a lower bound for the optimal value f_* can be obtained by solving the linear programming problem:

$$f_k^{low} := \min_{x \in \mathcal{X}} \check{f}_k(x)\,. \tag{8}$$

Since $x_k \in \mathcal{X}$ for all k, we have that $f(x_k)$ is an upper bound for f_*. However, when $\epsilon_x > 0$ in (2) the exact value $f(x_k)$ is not available. Instead, a natural candidate for an upper bound of f_* at iteration k is

$$f_k^{up} := f_{x_k^{rec}} + \epsilon_{x_k^{rec}}, \quad \text{for} \quad x_k^{rec} := \arg\min\{f_{x_i} + \epsilon_{x_i} : i \leq k; \text{ such that } f_{x_i} \leq \gamma_{x_i}\}\,, \tag{9}$$

that is, the smallest approximate upper bound computed so far for f_*. It follows from (2) that

$$f(x_j) \in [f_{x_j},\, f_{x_j} + \epsilon_{x_j}] \quad \text{for all} \quad j \in \{i \leq k;\, f_{x_i} \leq \gamma_{x_i}\}\,.$$

The difference between the upper and lower bounds can be used to stop the method, as shown by the following result.

Proposition 3.1. *Given the lower and upper bounds in* (8) *and* (9)*, the* optimality gap $\Delta_k := f_k^{up} - f_k^{low}$ *is nonnegative. Moreover, the minimand* x_k^{rec} *from* (9) *satisfies the relations*

$$0 \leq f(x_k^{rec}) - f_* \leq \Delta_k\,.$$

As a result, if $\Delta_k \to 0$ *and the sequence of records has a cluster point* x_**, then* x_* *solves problem* (1).

Proof. Because the error is nonnegative in (2), the inequality in (6) holds and the cutting-plane model (7) approximates the function f from below. The desired result is straightforward from the following relations:

$$f_k^{up} - \Delta_k = f_k^{low} = \min_{x \in \mathcal{X}} \check{f}_k(x) \leq \min_{x \in \mathcal{X}} f(x) = f_* \leq f(x_k^{rec}) \leq f_{x_k^{rec}} + \epsilon_{x_k^{rec}} = f_k^{up}\,. \quad \square$$

Since the optimality gap Δ_k is nonnegative, the value

$$f_k^{lev} := f_k^{up} - (1 - \kappa_l)\Delta_k = f_k^{low} + \kappa_l \Delta_k \tag{10}$$

lies between the lower and upper bounds, for any parameter $\kappa_l \in (0, 1)$. Such value, called *level parameter*, is used to define the *level set*:

$$\mathbb{X}_k := \{x \in \mathcal{X} :\ \check{f}_k(x) \leq f_k^{lev}\}\,, \tag{11}$$

which is polyhedral if $\mathcal{X}$ is a polyhedron. Denoting by $\hat{x}_k \in \mathcal{X}$ the *projection point* for the k-th iteration, the next iterate is obtained by projecting $\hat{x}_k$ onto the level set above:

$$x_{k+1} := \arg\min_{x \in \mathbb{X}_k} \frac{1}{2}|x - \hat{x}_k|^2\,, \quad \text{or} \quad x_{k+1} = P_{\mathbb{X}_k}(\hat{x}_k) \ \text{ for short.} \tag{12}$$

The level-QP described in the introduction coincides with (12). A more general setting, exploiting the geometry of the feasible set, is considered in Section 5.

The method can use either definition for the projection point $\hat{x}_k$, according on a given parameter $\delta_{\texttt{Prox}} \in \{0, 1\}$. Specifically, we consider:

– *a non-proximal variant*, when $\delta_{\texttt{Prox}} = 0$, for which $\hat{x}_k = x_k$ at each iteration k; and
– *a proximal variant*, when $\delta_{\texttt{Prox}} = 1$, for which the projection point is a stability center, given by some past iterate.

To state some properties of the minimizer x_{k+1} in (12) we use the notation $N_{\mathcal{X}}$ for the Convex Analysis normal cone of a set $\mathcal{X}$.

Proposition 3.2. *The point x_{k+1} solves (12) if and only if $x_{k+1} \in \mathcal{X}$, $\check{f}_k(x_{k+1}) \le f_k^{lev}$ and there exist a subgradient $\hat{g}_k \in \partial \check{f}_k(x_{k+1}) + N_{\mathcal{X}}(x_{k+1})$ and a stepsize $\tau_k \ge 0$, such that*

$$x_{k+1} = \hat{x}_k - \tau_k \hat{g}_k \quad \text{and} \quad \tau_k(\check{f}_k(x_{k+1}) - f_k^{lev}) = 0\,. \tag{13}$$

In addition, the aggregate linearization

$$\bar{f}_{-k}(\cdot) := \check{f}_k(x_{k+1}) + \langle \hat{g}_k, \cdot - x_{k+1}\rangle \ \text{ satisfies } \bar{f}_{-k}(x) \le \check{f}_k(x) \le f(x) \text{ for all } x \in \mathcal{X}. \tag{14}$$

Besides, for the aggregate level set $\mathbb{X}_{-k} := \{x \in \mathcal{X} : \bar{f}_{-k}(x) \le f_k^{lev}\}$, *we have that*

$$P_{\mathbb{X}_k}(\hat{x}_k) = P_{\mathbb{X}_{-k}}(\hat{x}_k)\,. \tag{15}$$

Proof. The first claim results from the KKT conditions for problem (12). The subgradient inequality gives the second claim. To see (15), consider the iterate x_{k+1} obtained by projecting $\hat{x}_k$ onto $\mathbb{X}_k$, i.e., $x_{k+1} = P_{\mathbb{X}_k}(\hat{x}_k)$.
Suppose first that $\check{f}_k(x_{k+1}) = f_k^{lev}$. Then by definition (14)

$$\begin{array}{rcl} \mathbb{X}_{-k} & = & \{x \in \mathcal{X} : \check{f}_k(x_{k+1}) + \langle \hat{g}_k, x - x_{k+1}\rangle \le f_k^{lev}\} \\ & = & \{x \in \mathcal{X} : \langle \hat{g}_k, x - x_{k+1}\rangle \le 0\} \\ & = & \{x \in \mathcal{X} : \frac{1}{\tau_k}\langle \hat{x}_k - x_{k+1}, x - x_{k+1}\rangle \le 0\}\,. \end{array}$$

Hence, $\langle \hat{x}_k - x_{k+1}, x - x_{k+1}\rangle \le 0$ for all $x \in \mathbb{X}_{-k}$, which is equivalent to the identity $x_{k+1} = P_{\mathbb{X}_{-k}}(\hat{x}_k)$.
When $\check{f}_k(x_{k+1}) < f_k^{lev}$, (13) implies that $x_{k+1} = \hat{x}_k$ and the relation $\bar{f}_{-k} \le \check{f}_k$ gives once more that $\hat{x}_k = P_{\mathbb{X}_{-k}}(\hat{x}_k)$, as claimed. □

The interest of the aggregate level set is that $\mathbb{X}_{-k}$ condenses all the past information relevant for defining x_{k+1}. This property is crucial for the bundle compression mechanism to be put in place. Namely, the property makes it possible to change the feasible set $\mathbb{X}_k$ in (1), possibly with many constraints, by any smaller set containing $\mathbb{X}_{-k}$. In this manner, for the proximal variant only, the size of the bundle J_k in (7) can be kept controlled, thus making (12) easier to solve.

The algorithmic framework for the proximal and non-proximal variants is stated below.

Algorithm 3.3. Level Method with On-demand Accuracy

Step 0: *(initialization). Choose the center parameter $\delta_{\texttt{Prox}} \in \{0\,,1\}$, a level parameter $\kappa_l \in (0,1)$, and a stopping tolerance $\delta_{\texttt{Tol}} > 0$. Given $x_1 \in \mathcal{X}$, set $\hat{x}_0 = x_1$, choose an error bound $\epsilon_{x_1} \ge 0$ and take the descent target $\gamma_{x_1} = +\infty$. Call the oracle (2) to compute the corresponding f_{x_1}/g_{x_1}-information. Define the upper bound $f_1^{up} = f_{x_1} + \epsilon_{x_1}$. Let $J_1 = \{1\}$, $k = 1$, $\ell = 0$, and $k(\ell) = 1$. Solve (8) to obtain f_1^{low}.*

Step 1: *(stopping criterion). For $\Delta_k = f_k^{up} - f_k^{low}$, stop if $\Delta_k \le \delta_{\texttt{Tol}}$.*

Step 2: *(choice of projection point). If $\Delta_k \le (1-\kappa_l)\Delta_{k(\ell)}$, select $\hat{x}_k \in \{x_j : j \in J_k\}$, setting $\ell = \ell + 1$ and $k(\ell) = k$. Otherwise, take*

$$\hat{x}_k = \delta_{\texttt{Prox}}\hat{x}_{k-1} + (1 - \delta_{\texttt{Prox}})x_k\,.$$

Step 3: *(QP solution). Update the level value f_k^{lev} by (10) and the level set $\mathbb{X}_k$ by (11). If $\mathbb{X}_k = \emptyset$, set $f_k^{low} = f_k^{lev}$ and go back to Step 1. Otherwise, let x_{k+1} be the solution to (12).*

Step 4: *(on-demand oracle parameters and call). Select a new error bound $\epsilon_{x_{k+1}} \ge 0$ and a new descent target $\gamma_{x_{k+1}} \in \Re \cup \{\infty\}$.*
Call the oracle (2) to compute the corresponding $f_{x_{k+1}}/g_{x_{k+1}}$-information.
If $f_{x_{k+1}} \le \gamma_{x_{k+1}}$, define the upper bound as $f_{k+1}^{up} = \min\{f_k^{up}, f_{x_{k+1}} + \epsilon_{x_{k+1}}\}$.
Otherwise, set $f_{k+1}^{up} = f_k^{up}$.

Step 5: *(bundle management)*
For $\delta_{\texttt{Prox}} = 1$, choose $J_{k+1} \supset \{k+1\}$ if $k(\ell) = k$; otherwise, $J_{k+1} \supset \{k+1, -k\}$ where the index $-k$ denotes the aggregate linearization (14).
For $\delta_{\texttt{Prox}} = 0$, choose $J_{k+1} \supset \{k+1\}$ if $k(\ell) = k$; otherwise, $J_{k+1} = J_k \cup \{k+1\}$.

Step 6: *(lower bound and loop)*
For $\delta_{\texttt{Prox}} = 1$, set $f_{k+1}^{low} = f_k^{low}$.
For $\delta_{\texttt{Prox}} = 0$, find the optimal value f_{LP} of (8) with k replaced by $k+1$, and set $f_{k+1}^{low} = \max\{f_{LP}, f_k^{low}\}$.
Set $k = k+1$ and go back to Step 1. □

Algorithm 3.3 starts with an infinite descent target for the purposes of bounding from above the optimal value f_*, an essential issue when defining level sets.

In Step 2, the "arbitrary" choice for a new projection point (when the optimality gap decreases enough) can take the iterate associated with f_k^{up}, if such point was kept in the bundle. Iterations such that Step 2 defines a new projection point are called *critical*; they are separated by *cycles*, or groups of consecutive non-critical iterations following a critical one, denoted by

$$K^\ell := \{k(\ell), \ldots, k(\ell+1) - 1\}, \quad \text{for } \ell \geq 0\,. \tag{16}$$

By construction, the optimality gap strictly decreases at critical iterations:

$$(1-\kappa_l)\Delta_{k(\ell)} \geq \Delta_{k(\ell+1)}, \quad \text{for all } \ell \geq 0\,. \tag{17}$$

Since each cycle is finite (see Propositions 3.6 and 3.8 below), when the stopping tolerance is positive, eventually the stopping test in Step 1 is triggered and Proposition 3.1 ensures that the record x_k^{rec} from (9) yields a $\delta_{\texttt{Tol}}$-solution to problem (1).

Step 4 always updates the upper bound f_k^{up} to the value corresponding to the new iterate when there is enough descent. However, descent is not enough for the iteration to be declared critical with the proximal variant. Specifically, in this variant the projection point in Step 2 is changed only when *both* the objective value and the gap were reduced, enabling the bundle compression in Step 5. Namely, with the proximal variant, the bundle can be reduced down to only two elements, corresponding to the last iterate and the aggregate linearizations. By contrast, and similarly to [LNN95] and [Fáb00], with the non-proximal variant ($\delta_{\texttt{Prox}} = 0$) the aggregation/compression mechanism is not possible. Even though in this variant the bundle size can still be reduced at critical iterations, there is no aggregation or memory of the past linearizations. Only the last iterate information can be kept with this variant, as if re-initializing the algorithm, setting $x_1 = x_{k+1}$.

Regarding the linear program (8), Algorithm 3.3 in its non-proximal variant solves (8) at each iteration to define a lower bound f_k^{low}. After cleaning the bundle, the safeguard $f_k^{low} = \max\{f_{LP}, f_{k-1}^{low}\}$ at Step 6 ensures the monotonicity of the sequence of lower bounds, which in turn ensures non-emptiness of the level sets. As a result, the sequence of optimality gaps is monotone too, and the sets

$$\{x \in \mathcal{X} :\ \check{f}_k(x) \leq f_k^{low}\} \text{ are nonempty for all } k \in K^\ell\,. \tag{18}$$

This relation is particularly important to prove Lemma 3.5 below.

On the other hand, Algorithm 3.3 in its proximal variant solves program (8) only once, at Step 0. In this case, to check if the level set is empty at Step 3, we can either solve a linear programming problem or simply try solving (12) and let the QP solver return with an error flag. If the level set is empty, then $f_k^{lev} < \check{f}_k(x)$ for all $x \in \mathcal{X}$, which implies by (6) that f_k^{lev} is a lower bound for f_*. Besides, to make $\mathbb{X}_k$ nonempty we must increase the level f_k^{lev}. This is the purpose of the update replacing f_k^{low} by f_k^{lev} in Step 3. Since this update reduces the optimality gap in an amount equal to $(1-\kappa_l)\Delta_k$, Step 2 will declare the iteration critical. Therefore, the stability center will be changed and, most importantly,

$$\text{the sequence } \{f_j^{lev}\}_{j \in K^\ell} \text{ will be non-increasing for each fixed } \ell \geq 0\,, \tag{19}$$

a property essential for Lemma 3.7, given below.

The specific method fitting the general framework of Algorithm 3.3 depends on the definitions of accuracy and descent target at Step 4. Table 1 below gives several alternative instances, resulting in different methods that can be further declined in their proximal and non-proximal variants. Some of the instances make use of two additional parameters, $\kappa_\epsilon\,, \kappa_f \in (0,1)$, dealing respectively with the error bound and the descent target. The last two columns in Table 1 contain explicit

constants λ and c for each instance, which are useful in the complexity analysis given in Theorem 3.9 below.

TABLE 1. On-demand oracle: possibly degrees of inaccuracy for Algorithm 3.3

Instances	Parameters	$\epsilon_{x_{k+1}}$	$\gamma_{x_{k+1}}$	$\lambda > 0$	c
`Ex`	$\kappa_l \in (0,1)$	0	$+\infty$	$1-\kappa_l$	1
`PI1`	$\kappa_l \in (0,1)$	0	f_k^{up}	$1-\kappa_l$	1
`PI2`	$\kappa_l \in (0,1)$, $0 < \kappa_f < (1-\kappa_l)^2$	0	$f_k^{up} - \kappa_f \Delta_k$	$1-\kappa_l - \dfrac{\kappa_f}{1-\kappa_l}$	2
`AE`	$\kappa_l \in (0,1)$, $0 < \kappa_\epsilon < (1-\kappa_l)^2$	$\kappa_\epsilon \Delta_k$	$+\infty$	$1-\kappa_l - \dfrac{\kappa_\epsilon}{1-\kappa_l}$	2
`PAE`	$\kappa_l \in (0,1)$, $0 < \kappa_f + \kappa_\epsilon < (1-\kappa_l)^2$	$\kappa_\epsilon \Delta_k$	$f_k^{up} - (\kappa_f + \kappa_\epsilon)\Delta_k$	$1-\kappa_l - \dfrac{\kappa_f + \kappa_\epsilon}{1-\kappa_l}$	2

The short instance names `Ex, PI1, PI2, AE, PAE` correspond, respectively, to the Exact, Partly Inexact in two variants, Asymptotically Exact, and Partly Asymptotically Exact oracles described in Section 2.

Depending on the two options for the proximal parameter, the instances in Table 1 provide ten different configurations for Algorithm 3.3. In particular, when $\delta_{\texttt{Prox}} = 0$, Algorithm 3.3 with instance `Ex` is the level method introduced in [LNN95] while $\delta_{\texttt{Prox}} = 1$ corresponds to the proximal level method given in [Kiw95]. Instance `AE` with $\delta_{\texttt{Prox}} = 0$ fits in the framework of the inexact level method [Fáb00]. The remaining seven configurations provided by Algorithm 3.3 are, in the authors' knowledge, new variants of level bundle methods.

3.2. **Convergence analysis.** Throughout this section we suppose the feasible set in (1) is compact with diameter D, and denote by $\Lambda > 0$ a Lipschitz constant for the objective function over the set $\mathcal{X}$, recalling that both values depend on the choice of the norm $|\cdot|$. We also suppose the oracle satisfies Assumption 2.1 so that (3) is satisfied. In what follows, both constants D and Λ can be unknown, since they are not used by the algorithm but only to give upper bounds for the length of cycles.

We start with a general result that is independent of the choice of the parameter $\delta_{\texttt{Prox}}$ in Algorithm 3.3.

Lemma 3.4. *Suppose the feasible set $\mathcal{X}$ in (1) is compact and the oracle (2) satisfies Assumption 2.1. Let Λ be a Lipschitz constant for f in (1) and λ be the instance-dependent constant in Table 1. At the k-th iteration of Algorithm 3.3 the following relation holds:*

$$|x_{j+1} - x_j| \geq \lambda \frac{\Delta_k}{\Lambda} \text{ for any past iteration in the cycle such that } j = k(\ell)+1, \ldots, k\,.$$

Moreover, for instances `Ex` or `PI1` in Table 1 the relation

$$|x_{j+1} - x_j| = |x_{k(\ell)+1} - \hat{x}_{k(\ell)}| \geq \lambda \frac{\Delta_k}{\Lambda} \quad \text{holds if } \; j = k(\ell)\,.$$

Proof. At Step 5 of iteration j, the current iterate enters the bundle, so $j \in J_j$. Therefore, by (11) combined with (12), $f_{x_j} + \langle g_{x_j}, x_{j+1} - x_j \rangle \leq f_j^{lev}$ and, hence,

$$f_{x_j} - f_j^{lev} \leq \langle g_{x_j}, x_j - x_{j+1} \rangle \leq \|g_{x_j}\| |x_{j+1} - x_j| \leq \Lambda |x_{j+1} - x_j|\,, \tag{20}$$

because (3) is satisfied. So the desired result would hold if $f_{x_j} - f_j^{lev} \geq \lambda \Delta_k$. To prove this inequality, consider the index set $\mathcal{D}_k := \{i \leq k; f_{x_i} \leq \gamma_{x_i}\}$ and recall that $f_j^{up} = \min_{i \in \mathcal{D}_j} \{f_{x_i} + \epsilon_{x_i}\}$ and $f_j^{lev} = f_j^{up} - (1-\kappa_l)\Delta_j$, by construction.

We split our analysis for the various instances in Table 1, letting $j \leq k$.

– Instances `Ex` and `PI1` are such that $f_{x_j} \geq f_j^{up}$ for all j, so

$$f_{x_j} - f_j^{lev} \geq (1-\kappa_l)\Delta_j = \lambda \Delta_j \geq \lambda \Delta_k\,,$$

where the equality comes from Table 1 and the rightmost inequality uses that $j \leq k$. Since $\hat{x}_{k(\ell)} \in \{x_j : j \in J_{k(\ell)}\}$ for both proximal and non-proximal variants, take $j = k(\ell)$ in the above development to see that $|x_{j+1} - x_j| = |x_{k(\ell)+1} - \hat{x}_{k(\ell)}| \geq \lambda \frac{\Delta_k}{\Lambda}$.

– Instances `PI2` and `AE` are particular cases of instance `PAE`, considered below.

– Instance `PAE` sets $\gamma_{x_j} = f^{up}_{j-1} - (\kappa_f + \kappa_\epsilon)\Delta_{j-1} = f^{up}_{j-1} - \kappa_f \Delta_{j-1} - \epsilon_{x_j}$, so

$$\begin{array}{rcll} f_{x_j} & \geq & f^{up}_j - \epsilon_{x_j} & \text{if} \quad j \in \mathcal{D}_j \\ f_{x_j} & \geq & f^{up}_{j-1} - \kappa_f \Delta_{j-1} - \epsilon_{x_j} & \text{if} \quad j \notin \mathcal{D}_j \,. \end{array}$$

Therefore, $f_{x_j} \geq f^{up}_j - \kappa_f \Delta_{j-1} - \epsilon_{x_j} = f^{up}_j - (\kappa_f + \kappa_\epsilon)\Delta_{j-1}$ for all j, because $f^{up}_{j-1} \geq f^{up}_j$. Furthermore,

$$f_{x_j} - f^{lev}_j \geq f^{up}_j - (\kappa_f + \kappa_\epsilon)\Delta_{j-1} - (f^{up}_j - (1-\kappa_l)\Delta_j) \geq (1-\kappa_l)\Delta_j - (\kappa_f + \kappa_\epsilon)\Delta_{j-1} \,.$$

Since $k \in K^\ell$ and $k(\ell) < j \leq k$, the gap satisfies $\Delta_j > (1-\kappa_l)\Delta_{k(\ell)} \geq (1-\kappa_l)\Delta_{j-1}$. Accordingly, the last right hand side in the inequality above gives

$$(1-\kappa_l)\Delta_j - \frac{(\kappa_f + \kappa_\epsilon)}{1-\kappa_l}(1-\kappa_l)\Delta_{j-1} > \left[(1-\kappa_l) - \frac{(\kappa_f + \kappa_\epsilon)}{1-\kappa_l}\right]\Delta_j = \lambda\Delta_j \geq \lambda\Delta_k \,,$$

where once more the last inequality comes from having $j \leq k$.

By (20), our claim is proved and so is the desired result for all instances in Table 1. □

For our subsequent analysis, we consider separately the non-proximal and proximal as of Algorithm 3.3.

3.2.1. *Non-Proximal variant.* To estimate the maximum number of iterations in a cycle, until a new critical iteration $k(\ell+1)$ is done, we start with the following preliminary result.

Lemma 3.5. *For the non-proximal variant of Algorithm 3.3 ($\delta_{\texttt{Prox}} = 0$) the following holds:*

$$\bigcap_{j=k(\ell)}^{k} \mathbb{X}_j \neq \emptyset \quad \textit{for all} \;\; k \in K^\ell \;\; \textit{and each fixed} \;\; \ell \geq 0 \,.$$

Proof. The intervals $[f^{low}_k, f^{up}_k] \subseteq [f^{low}_{k(\ell)}, f^{up}_{k(\ell)}]$ have respective lengths Δ_k and $\Delta_{k(\ell)}$. Moreover, the point $f^{low}_{k(\ell)} + \kappa_l \Delta_{k(\ell)}$ splits the second interval into two sub-intervals in a manner such that right sub-interval, that is $[f^{low}_{k(\ell)} + \kappa_l\Delta_{k(\ell)}, f^{up}_{k(\ell)}]$, has length $(1-\kappa_l)\Delta_{k(\ell)}$. Since $k \in K^\ell$, the inequality $(1-\kappa_l)\Delta_{k(\ell)} < \Delta_k$ holds, and the interval $[f^{low}_k, f^{up}_k] \not\subset [f^{low}_{k(\ell)} + \kappa_l\Delta_{k(\ell)}, f^{up}_{k(\ell)}]$. As $f^{up}_{k(\ell)} \geq f^{up}_k$ for all $k \in K^\ell$, it follows that $f^{low}_k \leq f^{low}_{k(\ell)} + \kappa_l\Delta_{k(\ell)} = f^{lev}_{k(\ell)}$ by (10). Moreover, by monotonicity of the sequence $\{\Delta_k\}$, it follows that $(1-\kappa_l)\Delta_j \leq (1-\kappa_l)\Delta_{k(\ell)} < \Delta_k$ for all $k(\ell) \leq j \leq k$. By repeating the reasoning using j instead of $k(\ell)$, we can show that $f^{low}_k \leq f^{lev}_j$ for all $k(\ell) \leq j \leq k$. Because $J_j \subset J_k$ at Step 5 (the bundle information is maintained along the same cycle for the non-proximal variant), it follows that $\check{f}_j \leq \check{f}_k$ for such indices j. By (18), there exists a point $\check{x}_k \in \mathcal{X}$ such that for all $k(\ell) \leq j \leq k$

$$\check{x}_k \in \{x \in \mathcal{X} : \check{f}_k(x) \leq f^{low}_k\} \subseteq \{x \in \mathcal{X} : \check{f}_j(x) \leq f^{low}_k\} \subseteq \{x \in \mathcal{X} : \check{f}_j(x) \leq f^{lev}_j\} = \mathbb{X}_j \,,$$

and the result follows. □

Lemma 3.5 relies strongly on the bundle management strategy at Step 5, which for the non-proximal variant only increases the bundle size at non-critical iterations. If some of past linearizations in the same cycle were eliminated (by using aggregation and compression), the crucial relation $\check{f}_j \leq \check{f}_k$ for all $k(\ell) \leq j \leq k \in K^\ell$ might no longer hold, yielding an empty intersection of level sets in the cycle. We now show that iterates get closer to a specific point in (18), belonging to intersection.

Proposition 3.6. *Suppose the feasible set $\mathcal{X}$ in (1) is compact and the oracle (2) satisfies Assumption 2.1. Let Λ be a Lipschitz constant for f in (1) and λ be the instance-dependent constant*

in Table 1. Consider the non-proximal variant of Algorithm 3.3 ($\delta_{\texttt{Prox}} = 0$). For any iteration $k \in K^\ell$ with $\Delta_k > \delta_{\texttt{Tol}}$ the number of iterations between $k(\ell)$ and k is bounded, as follows:

$$k - k(\ell) + 1 \leq \begin{cases} \left(\dfrac{D\Lambda}{\lambda\Delta_k}\right)^2 & \text{for instances } \texttt{Ex} \text{ or } \texttt{PI1}\,, \\ \left(\dfrac{D\Lambda}{\lambda\Delta_k}\right)^2 + 1 & \text{for instances } \texttt{PI2}, \texttt{AE}, \text{ or } \texttt{PAE}\,. \end{cases}$$

Proof. Since $\Delta_k > \delta_{\texttt{Tol}}$ for all $k \in K^\ell$, Algorithm 3.3 does not stop. Because the iterate x_{j+1} is the projection of $x_j\,(= \hat{x}_j)$ onto the convex set $\mathbb{X}_j$,

$$|x_{j+1} - x_j|^2 + |x_{j+1} - x|^2 \leq |x - x_j|^2 \quad \text{for all} \quad x \in \mathbb{X}_j\,.$$

By Lemma 3.5 there exists a point $\check{x}_k$ in the intersection of level sets. In particular,

$$|x_{j+1} - \check{x}_k|^2 \leq |\check{x}_k - x_j|^2 - |x_{j+1} - x_j|^2 \quad \text{for } k(\ell) \leq j \leq k.$$

Together with Lemma 3.4 we see that

$$|x_{j+1} - \check{x}_k|^2 \leq |\check{x}_k - x_j|^2 - \left(\lambda\frac{\Delta_k}{\Lambda}\right)^2 \quad \text{for } k(\ell) < j \leq k\,,$$

recalling that the relation is also true for $j = k(\ell)$ if instances `Ex` or `PI1` are considered. Thus, at each iteration j the new iterate x_{j+1} gets closer to $\check{x}_k$ by a factor of at least $(\lambda\Delta_k/\Lambda)^2$. Because the feasible set is supposed compact, and, hence $|x_{k(\ell)} - \check{x}_k| \leq D$. The result follows. □

We now proceed to a similar analysis when in Algorithm 3.3 the center parameter $\delta_{\texttt{Prox}}$ is set to one.

3.2.2. *Proximal variant.* Since with this variant, Step 5 may compress the bundle set J_k, Lemma 3.5 is not applicable. However, the fact that in this variant the stability center remains fixed at non-critical iterations somehow replaces the need of a point $\check{x}_k$ in the intersection of level sets. Actually, instead of showing that each iterate gets closer to $\check{x}_k$, we will show that iterates in a cycle get further away from the current (fixed) stability center $\hat{x}_k$.

Lemma 3.7. *Consider the proximal variant of Algorithm 3.3 ($\delta_{\texttt{Prox}} = 1$). For any $k \in K^\ell$ with $k > k(\ell)$ and $\ell \geq 0$ the stability center remains fixed ($\hat{x}_k = x_{k(\ell)}$) and the relation*

$$|x_{k+1} - \hat{x}_k|^2 \geq |x_k - \hat{x}_k|^2 + |x_k - x_{k+1}|^2$$

holds.

Proof. By Step 2 and $\delta_{\texttt{Prox}} = 1$, the stability center is fixed for all $k \in K^\ell$; so $\hat{x}_k = \hat{x}_{k-1}$ when $k > k(\ell)$. Consider Proposition 3.2 written at iteration $k-1$: the aggregate index $-k$ therein is replaced by $-(k-1)$ and the level set $\mathbb{X}_{-(k-1)}$ uses the aggregate linearization $\bar{f}_{-(k-1)}$. This means that $x_k = P_{\mathbb{X}_{-(k-1)}}(\hat{x}_{k-1})$ and, hence,

$$\langle \hat{x}_{k-1} - x_k, x - x_k\rangle \leq 0 \quad \text{for all} \quad x \in \mathbb{X}_{-(k-1)}\,. \tag{21}$$

The bundle management strategy in Step 5 incorporates the aggregate index into the bundle, so $-(k-1) \in J_k$. As a result, $\check{f}_k \geq \bar{f}_{-(k-1)}$ and $\mathbb{X}_k \subset \mathbb{X}_{-(k-1)}$, because (19) holds. As $x_{k+1} \in \mathbb{X}_k$, then $x_{k+1} \in \mathbb{X}_{-(k-1)}$, and by (21) (using that $\hat{x}_{k-1} = \hat{x}_k$), we have that $\langle \hat{x}_k - x_k, x_{k+1} - x_k\rangle \leq 0$. The result follows by developing squares in the identity $|x_{k+1} - \hat{x}_k|^2 = |x_{k+1} - x_k + (x_k - \hat{x}_k)|^2$. □

For Lemma 3.7 to hold it is enough that $J_k = \{-(k-1)\}$ after a null step. The bundle includes the current index too ($k \in J_k$) so that in Lemma 3.4 the difference $|x_{k+1} - x_k|$ is strictly positive, thus allowing each iterate in the cycle to move further away from the fixed stability center.

Proposition 3.8. *Suppose the feasible set $\mathcal{X}$ in (1) is compact and the oracle (2) satisfies Assumption 2.1. Let Λ be a Lipschitz constant for f in (1) and λ be the instance-dependent constant in Table 1. Consider the proximal variant of Algorithm 3.3 ($\delta_{\texttt{Prox}} = 1$). For any iteration $k \in K^\ell$ with $\Delta_k > \delta_{\texttt{Tol}}$ the bound for the number of iterations between $k(\ell)$ and k is the same as in Proposition 3.6.*

Proof. As $\Delta_k > \delta_{\texttt{Tol}}$ for all $k \in K^\ell$, Algorithm 3.3 does not stop, and $\hat{x}_k = \hat{x}_{k(\ell)} \in \mathcal{X}$. A recursive application of Lemma 3.7 implies that

$$\begin{aligned} |x_{k+1} - \hat{x}_k|^2 &\geq |x_k - \hat{x}_k|^2 + |x_{k+1} - x_k|^2 \\ &= |x_k - \hat{x}_{k-1}|^2 + |x_{k+1} - x_k|^2 \\ &\geq |x_{k-1} - \hat{x}_{k-1}|^2 + |x_k - x_{k-1}|^2 + |x_{k+1} - x_k|^2 \\ &\vdots \\ &\geq |x_{k(\ell)+1} - \hat{x}_{k(\ell)}|^2 + \sum_{j=k(\ell)+1}^{k} |x_{j+1} - x_j|^2, \text{ for } k \in K^\ell\,, \ell \geq 0\,. \end{aligned}$$

Together with Lemma 3.4, this yields the bounds

$$D^2 \geq \left(\lambda \frac{\Delta_k}{\Lambda}\right)^2 + \sum_{j=k(\ell)+1}^{k} \left(\lambda \frac{\Delta_k}{\Lambda}\right)^2 = \left(\lambda \frac{\Delta_k}{\Lambda}\right)^2 (k - k(\ell) + 1)$$

for instances `Ex` or `PI1`; and

$$D^2 \geq \left(\lambda \frac{\Delta_k}{\Lambda}\right)^2 (k - k(\ell))$$

for the remaining instances. The result follows. □

Both the level and the proximal level methods in their exact variants are known to satisfy the stopping condition after a finite number of iterations, [LNN95], [Kiw95]. These methods correspond, respectively, to the non-proximal and proximal variants of instance `Ex` of Algorithm 3.3. More generally, all instances of Algorithm 3.3 have finite termination because Propositions 3.6 and 3.8 hold. For the sake of completeness we finish our analysis by bounding the number of iterations for each instance in Table 1.

3.2.3. *Unified Complexity Estimates for all the instances and both variants.* We now show how to refine the complexity result for both variants when the initial error bound, sent to the on-demand accuracy oracle (2) for the first f/g evaluation, is suitably chosen.

Theorem 3.9. *Suppose the feasible set $\mathcal{X}$ in (1) is compact and the oracle (2) satisfies Assumption 2.1. Let Λ be a Lipschitz constant for f in (1) and λ and c be the instance-dependent constants in Table 1. Consider Algorithm 3.3 and suppose that at Step 0 the initial oracle error ϵ_{x_1} is chosen so that*

$$\epsilon_{x_1} \leq \Lambda(D - \max_{x \in \mathcal{X}} |x - x_1|)\,. \tag{22}$$

Then, to reach an optimality gap smaller than $\delta_{Tol} > 0$ it is enough to perform at most

$$\left(\frac{\Lambda D}{\delta_{Tol}}\right)^2 \frac{c}{(1 - (1 - \kappa_l)^2)\lambda^2}$$

iterations, with both the proximal and non-proximal variants of any instance in Table 1.

Proof. We first show that when (22) holds the number of iterations in a cycle is bounded above by $c\left(\frac{D\Lambda}{\lambda \Delta_k}\right)^2$. To prove this claim, by Propositions 3.6 and 3.8 together with the values for c in Table 1, we only need to show that

$$1 \leq \left(\frac{D\Lambda}{\lambda \Delta_k}\right)^2.$$

Since with our assumptions (3) holds, the gap can be bounded as follows:

$$\Delta_k \leq \Delta_1 = f_1^{up} - f_1^{low} = f_1^{up} - \min_{x \in \mathcal{X}} [f_{x_1} + \langle g_{x_1}, x - x_1 \rangle] \leq f_1^{up} - f_{x_1} + \max_{x \in \mathcal{X}} |x - x_1| \Lambda\,.$$

Together with the initial definition for the upper bound, $f_1^{up} = f_{x_1} + \epsilon_{x_1}$, the choice (22) for the oracle error implies that $\Delta_k \leq D\Lambda$. The claim is straightforward, because in Table 1 all settings ensure that $\lambda \in (0, 1)$.

We now consider those iterations prior to triggering the stopping test in Step 1: $K(\delta_{\texttt{Tol}}) := \{k : \Delta_k \geq \delta_{\texttt{Tol}}\}$ and denote by $k_{\delta_{\texttt{Tol}}}$ the largest index in the set. Suppose that there are m cycles in $K(\delta_{\texttt{Tol}})$, i.e., $K(\delta_{\texttt{Tol}}) \subset \bigcup_{\ell=0}^{m} K^\ell$, for K^ℓ given in (16). Then, the inequality $\Delta_{k(\ell)} > \delta_{\texttt{Tol}}$ holds for all $\ell = 0, \ldots, m$. By (17)

$$(1-\kappa_l)^{m-\ell}\Delta_k \geq \delta_{\texttt{Tol}}, \quad \text{for all } k \in K^\ell \cap K(\delta_{\texttt{Tol}}) \text{ and } \ell = 0, \ldots, m\,,$$

hence, by Proposition 3.6 and 3.8,

$$|K^\ell \cap K(\delta_{\texttt{Tol}})| \leq c\left(\frac{D\Lambda}{\lambda\Delta_k}\right)^2 \leq c\left(\frac{D\Lambda(1-\kappa_l)^{m-\ell}}{\lambda\delta_{\texttt{Tol}}}\right)^2, \quad \text{for all } \ell = 0\ldots, m\,.$$

Summing the above inequality over ℓ yields that

$$k_{\delta_{\texttt{Tol}}} = \sum_{\ell=0}^{m} |K^\ell \cap K(\delta_{\texttt{Tol}})| \leq c\sum_{\ell=0}^{m}\left(\frac{D\Lambda(1-\kappa_l)^{m-\ell}}{\lambda\delta_{\texttt{Tol}}}\right)^2 \leq \left(\frac{D\Lambda}{\delta_{\texttt{Tol}}}\right)^2 \frac{c}{(1-(1-\kappa_l)^2)\lambda^2}\,.$$

Therefore, the last iteration number $k_{\delta_{\texttt{Tol}}}$ for which $\Delta_{k_{\delta_{\texttt{Tol}}}} \geq \delta_{\texttt{Tol}}$ cannot be larger than the stated bound, and the result follows. □

Like usually when giving complexity estimates, the smart choice of the first oracle error, satisfying (22), does need knowing explicitly the Lipschitz constant Λ and the diameter D. Nevertheless, the upper bound in Theorem 3.9 is the same for both choices of the center parameter, $\delta_{\texttt{Prox}}$. There is a dependence on the constants λ and c, which in turn depend on the choice of the descent target $\gamma_{x_{k+1}}$ and the upper bound $\epsilon_{x_{k+1}}$ for the oracle error. However, since instances `Ex` and `PI1` have the same constant c in Table 1, the complexity estimate for instance `PI1`, which uses an inexact oracle, coincides with the one obtained for instance `Ex`, which uses an exact oracle. For the remaining inexact instances (`PI2`, `AE`, `PAE`), the more inaccurate is the oracle, the more iterations are required for Algorithm 3.3 to terminate, as expected.

4. Removing the compactness assumption

The exact proximal level method [BKL95] can handle unbounded feasible sets $\mathcal{X}$ in (1). Instead of relying on compactness, using (3) and the diameter D, convergence of the methods in this section relate the *estimated decrease*

$$v_k := f_k^{up} - f_k^{lev} \tag{23}$$

to Lagrange multipliers $t_j, j \in J_k$, associated to the constraint $\check{f}_k(x) \leq f_k^{lev}$ which defines the level sets. In this variant the method converges, but there are no complexity results estimating the maximum number of iterations needed for the method to stop.

We now consider how the proximal level method given in [BKL95] can be extended to deal with oracles with on-demand accuracy.

4.1. **Mathematical development.** As in the previous section, $\hat{x}_k \in \{x_k\}$ is a stability center and the next iterate is obtained by solving the projection problem (12). However, and differently from the proximal variant of Algorithm 3.3, now the stability center is updated when

$$f_{x_{k+1}} \leq f_k^{up} - \kappa_f v_k\,, \quad \text{with} \quad \kappa_f \in (0,1)\,, \tag{24}$$

that is, when the iterate provides sufficient descent. Accordingly, each iteration results either
- in a *descent step*, when (24) holds, so $\hat{x}_k$ is moved to x_{k+1} and f_k^{up} is moved to $f_{x_{k+1}} + \epsilon_{x_{k+1}}$; or
- in a *null step*, when (24) does not hold, and both the stability center and the upper bound are maintained.

When using inexact oracles as in (2), to have a monotone sequence of upper bounds the error parameter sent to the oracle depends on the estimated decrease, as follows: $\epsilon_{x_{k+1}} < \kappa_f v_k$. With this choice, when the descent test (24) is satisfied, the relation

$$f_{k+1}^{up} = f_{x_{k+1}} + \epsilon_{x_{k+1}} < f_{x_{k+1}} + \kappa_f v_k \leq f_k^{up}$$

holds. At both descent or null steps, the bundle of information is updated to some new set J_{k+1} that can be compressed, and the level f_k^{lev} is updated by some suitable rule, for example by (10).

As for the lower bound, it is updated only if the level set $\mathbb{X}_k$ in (11) is empty. Since for the initial choice $f_1^{low} = -\infty$ unboundedness of the feasible set may yield only nonempty level sets, the lower bound may never be updated. In this setting, if eventually there are no more descent steps, the sequence $\{f_k^{up}\}$ will stabilize, and the stopping test $\Delta_k \leq \delta_{\texttt{Tol}}$, based on decreasing of the optimality gap may never be triggered. For this reason, the inexact version makes use of a stronger stopping criterion, based on certain optimality certificate defined below.

Proposition 4.1. *Consider the aggregate linearization and subgradient $\bar{f}_{-k}$ and $\hat{g}_k$ given in Proposition 3.2 and suppose linearizations are defined using an oracle with on-demand accuracy like* (2)*. The following holds.*
(i) The aggregate linearization error

$$\hat{e}_k := f_k^{up} - \bar{f}_{-k}(\hat{x}_k) \tag{25}$$

is nonnegative and satisfies

$$\bar{f}_{-k}(x) = f_k^{up} - \hat{e}_k + \langle \hat{g}_k, x - \hat{x}_k \rangle \leq \check{f}_k(x) \leq f(x) \quad \textit{for all } \; x \in \mathcal{X}\,; \tag{26}$$

(ii) If the parameter τ_k in Proposition 3.2 is positive, the expected decrease (23) *satisfies*

$$v_k = \hat{e}_k + \tau_k \|\hat{g}_k\|^2\,. \tag{27}$$

(iii) The optimality certificate

$$V_k := \min\Big\{\Delta_k, \max\Big(\hat{e}_k, \|\hat{g}_k\|\Big)\Big\} \tag{28}$$

is nonnegative and such that, if the relation $\liminf \max\{V_k,\, |\hat{x}_k - \bar{x}|\} = 0$ *holds for some $\bar{x} \in \mathcal{X}$, then $\bar{x}$ is a solution to problem* (1).

Proof. By Proposition 3.2, $\bar{f}_{-k}$ is bounded above by f and $f_k^{up} = f_{\hat{x}_k} + \epsilon_{\hat{x}_k} \geq f(\hat{x}_k)$ for an oracle satisfying (2). In particular, $\hat{e}_k \geq 0$, and (26) follows from (25) combined with (14).

Whenever $\tau_k > 0$, (13) ensures that $\check{f}_k(x_{k+1}) = f_k^{lev}$ and, hence,

$$\begin{aligned} \hat{e}_k &= f_k^{up} - \bar{f}_{-k}(\hat{x}_k) \\ &= f_k^{up} - (\check{f}_k(x_{k+1}) + \langle \hat{g}_k, \hat{x}_k - x_{k+1} \rangle) \\ &= f_k^{up} - f_k^{lev} - \tau_k \|\hat{g}_k\|^2\,. \end{aligned}$$

The second item then follows from (23).

To show the last assertions, first note that in (28) the optimality certificate is the minimum of two terms that are nonnegative. Also, by (26), for all $x \in \mathcal{X}$ it holds that $f_k^{up} \leq f(x) + \hat{e}_k - \langle \hat{g}_k, x - \hat{x}_k \rangle$. Since, in addition, $f_k^{up} \geq f(\hat{x}_k)$ by (2) and (9), we see that

$$f(\hat{x}_k) \leq f(x) + \max\Big(\hat{e}_k, \|\hat{g}_k\|\Big)(1 + |x - \bar{x}| + |\hat{x}_k - \bar{x}|)\,. \tag{29}$$

When $\liminf \max\{V_k, |\hat{x}_k - \bar{x}|\} = 0$ for an infinite subset of iterations, say K, both V_k and $|\hat{x}_k - \bar{x}| \to 0$ as $k \in K$ goes to infinity. In particular, by (28), either Δ_k or $\max\Big(\hat{e}_k, \|\hat{g}_k\|\Big) \to 0$ as $k \to \infty$ in K. We analyze each one of the cases.

– If $\Delta_k \to 0$ for k in K, by monotonicity the whole sequence $\{\Delta_k\}$ goes to zero. Since the descent test (24) provides a sequence $\{\hat{x}_k\}$ of descent steps that is a subsequence of records $\{x_k^{rec}\}$ yielding the minimum in (9), the cluster point $\bar{x}$ of $\{\hat{x}_k\}$ is also a cluster point of $\{x_k^{rec}\}$. Hence, Proposition 3.1 gives the result.

– Otherwise, $\max\Big(\hat{e}_k, \|\hat{g}_k\|\Big) \to 0$ for a K-subsequence. Since $\hat{x}_k \to \bar{x}$ in particular for $K \ni k \to \infty$, by lower semicontinuity of f, $f(\bar{x}) \leq \liminf_{k \in K} f(\hat{x}_k)$. Hence, passing to the limit in (29), we obtain that $f(\bar{x}) \leq f(x)$ for any feasible point $x \in \mathcal{X}$, as stated.

□

We are now in a position to give the algorithmic scheme with full details.

Algorithm 4.2. Generalized Level Method with On-demand Accuracy

Step 0: *(initialization). Choose an Armijo-like parameter* $\kappa_f \in (0,1)$*, a level parameter* $\kappa_l \in (0,1)$*, an upper bound* $\tau_{\max} > 0$ *for the Lagrange multiplier in* (13)*, and a stopping tolerance* $\delta_{\texttt{Tol}} > 0$*. Given* $x_1 \in \mathcal{X}$*, set* $\hat{x}_1 = x_1$*, choose an error bound* $\epsilon_{x_1} \geq 0$ *and take the descent target* $\gamma_{x_1} = +\infty$*. Call the oracle* (2) *to compute the corresponding* f_{x_1}/g_{x_1}*-information. For the upper bound* $f_1^{up} = f_{x_1} + \epsilon_{x_1}$*, and a lower bound given by any value satisfying* $f_1^{low} \leq f_*$*, define the gap* $\Delta_1 = f_1^{up} - f_1^{low}$*. Set* $v_1 = (1-\kappa_l)\Delta_1$ *if* $f^{low} > -\infty$*, or choose* $v_1 > 0$ *otherwise. Let* $J_1 = \{1\}$*,* $k = 1$*,* $\ell = 0$*, and* $k(\ell) = 1$*.*

Step 1: *(level management and QP solution). Update the level value* $f_k^{lev} = f_k^{up} - v_k$ *and the level set* $\mathbb{X}_k$ *by* (11)*.*

Step 2: *(next iterate) If* $\mathbb{X}_k = \emptyset$*, set* $f_k^{low} = f_k^{lev}$*,* $\Delta_k = f_k^{up} - f_k^{low}$*,* $v_k = (1-\kappa_l)\Delta_k$ *and go back to Step 1. Otherwise, let* x_{k+1} *be the solution to* (12)*, with corresponding multipliers* t_j *with* $j \in J_k$*. Set* $\tau_k = \sum_{j \in J_k} t_j$*.*

Step 3: *(stopping criterion). Stop if* $V_k \leq \delta_{\texttt{Tol}}$*.*

Step 4: *(multiplier noise attenuation) If* $\tau_k > \tau_{\max}$*, set* $v_k = v_k/2$ *and go back to Step 1.*

Step 5: *(on-demand oracle parameters and call). Select a new error bound* $\epsilon_{x_{k+1}} \geq 0$ *and a new descent target* $\gamma_{x_{k+1}} \in \Re \cup \{\infty\}$*.*
Call the oracle (2) *to compute the corresponding* $f_{x_{k+1}}/g_{x_{k+1}}$*-information.*

Step 6: *(descent test). Set* $f_{k+1}^{low} = f_k^{low}$*.*
If (24) *holds, declare a descent step; set* $\ell = \ell + 1$*,* $k(\ell) = k+1$ *and update the stability center* $\hat{x}_{k+1} = x_{k+1}$*. Set* $f_{k+1}^{up} = f_{x_{k+1}} + \epsilon_{x_{k+1}}$*,* $\Delta_{k+1} = f_{k+1}^{up} - f_{k+1}^{low}$ *and* $v_{k+1} = \min\{v_k, (1-\kappa_l)\Delta_{k+1}\}$*.*
Otherwise, declare a null step; set $\hat{x}_{k+1} = \hat{x}_k$*,* $f_{k+1}^{up} = f_k^{up}$*,* $\Delta_{k+1} = \Delta_k$*, and* $v_{k+1} = v_k$*.*

Step 7: *(bundle management)*
Choose $J_{k+1} \supset \{k+1\}$ *if there was a descent step (*$k(\ell) = k+1$*).*
Otherwise, in case of null step, choose $J_{k+1} \supset \{k+1, -k\}$ *where the index* $-k$ *denotes the aggregate linearization* (14)*.*

Set $k = k+1$ *and go back to Step 1.* □

Steps 2 and 6 ensure that the sequences $\{\Delta_k\}$ and $\{v_k\}$ are monotonically nonincreasing. Moreover, $0 \leq v_k \leq \Delta_k$ for all iterations k, by construction.

When $v_k = (1-\kappa_l)\Delta_k$, the level parameter updating at Step 1 is equivalent to (10). A distinctive feature of this variant is that, even without updating neither f_k^{low} nor f_k^{up}, the level parameter f_k^{lev} can still change, thanks to the attenuation of noise in Step 4.

As mentioned in [BKL95], the upper bound $\tau_{\max}$ for τ_k is necessary to ensure global convergence. In fact, without a finite $\tau_{\max}$ the sequence $\{v_k\}$ may be constant and a new descent step might never be declared. By (27), a large v_k yields a large $\hat{e}_k$ and $\hat{g}_k$, and the stopping test would not be satisfied because in the unbounded case the optimality gap Δ_k might be infinite. As an example of this situation, consider minimizing the function $f(x) = 1/x$ over $\mathcal{X} = [1, +\infty)$, by taking $f_1^{low} = -\infty$ and $v_1 > 0$, say $v_1 = 0.5$. Since $\mathbb{X}_k \neq \emptyset$ for all k, then $\Delta_k = \infty$. In addition, if $\tau_{\max}$ was infinite then $v_k = v_1$ for all k and the algorithm would generate finite number of descent steps, but it would never stop.

As with Algorithm 3.3, the specific method fitting the general framework of Algorithm 4.2 depends on the definitions of accuracy and descent target at Step 5. Like Table 1, some alternatives in Table 2 make use of an additional parameter κ_ϵ, introduced to cope with the oracle error.

The level method given in [BKL95] corresponds to instance `Ex`. The remaining instances are new variants of level methods capable to handle unbounded feasible sets. Finally, the condition $\kappa_\epsilon < \kappa_f$, useful for Proposition 4.6, also makes the sequence $\{f_k^{up}\}$ monotone, an important matter for instances `AE'` and `PAE'`.

4.2. **Convergence Analysis.** This section follows [BKL95], with suitable modifications regarding the inexact oracle setting. We suppose that the tolerance $\delta_{\texttt{Tol}} = 0$, and that the method does not stop. As a replacement of compactness of the feasible set, we require the function to be bounded from below.

Assumption 4.3. *There exists* $\tilde{x} \in \mathcal{X}$ *such that* $f(\hat{x}_k) \geq f(\tilde{x})$ *for all* k*.*

TABLE 2. On-demand oracle: possibly degrees of inaccuracy for Algorithm 4.2

Instances	Parameters	$\epsilon_{x_{k+1}}$	$\gamma_{x_{k+1}}$
`Ex`	$\kappa_l, \kappa_f \in (0,1)$	0	$+\infty$
`PI1`	$\kappa_l, \kappa_f \in (0,1)$	0	f_k^{up}
`PI2'`	$\kappa_l, \kappa_f \in (0,1)$	0	$f_k^{up} - \kappa_f v_k$
`AE'`	$\kappa_l, \kappa_f \in (0,1)$, $\kappa_\epsilon \in (0, \kappa_f)$	$\kappa_\epsilon v_k$	$+\infty$
`PAE'`	$\kappa_l, \kappa_f \in (0,1)$, $\kappa_\epsilon \in (0, \kappa_f)$	$\kappa_\epsilon v_k$	$f_k^{up} - \kappa_f v_k$

The above condition is satisfied in particular if problem (1) has a minimizer $\tilde{x}$. Since in Algorithm 4.2 and $\hat{x}_k \in \mathcal{X}$ by construction, $f_k^{up} = f_{\hat{x}_k} + \epsilon_{\hat{x}_k} \geq f(\hat{x}_k)$ by (2), thus

$$\text{if Assumption 4.3 holds then } f_k^{up} \geq f(\tilde{x}) \quad \text{ for all } k.$$

We start by estimating the distance between two consecutive iterates and showing that the stopping test in Step 3 eventually holds.

Lemma 4.4. *Consider Algorithm 4.2, for the instances considered in Table 2. If at the k-th iteration $v_k = v_{k-1}$ then*

$$|x_{k+1} - x_k| \geq (1 - \kappa_f) \frac{v_k}{\|g_{x_k}\|} \,. \tag{30}$$

Proof. By (11) and (12), the inequality $f_{x_k} + \langle g_{x_k}, x_{k+1} - x_k \rangle \leq f_k^{lev}$ holds and, hence,

$$f_{x_k} - f_k^{lev} \leq \|g_{x_k}\| |x_{k+1} - x_k| \,.$$

Recalling that $f_k^{lev} = f_k^{up} - v_k$ by construction, we now show that whenever $v_k = v_{k-1}$, for each instance in Table 2 the relation $f_{x_k} - f_k^{lev} \geq (1 - \kappa_f) v_k$ holds.

– Instances `Ex`, `PI1`, and `PI2'` are such that $f_{x_k} = f_k^{up}$ if the iterate gave descent ($x_k = \hat{x}_k$). Otherwise, if the step was declared null, $f_{x_k} > f_{k-1}^{up} - \kappa_f v_{k-1}$. In both cases, $f_{k-1}^{up} \geq f_k^{up}$, our assumption implies that $f_{x_k} \geq f_k^{up} - \kappa_f v_k$, and the claim is satisfied.

– Instances `AE'` and `PAE'` set $f_k^{up} = f_{x_k} + \epsilon_{x_k}$ if the iterate gave descent ($x_k = \hat{x}_k$). Since by Table 2, $f_{x_k} + \epsilon_{x_k} = f_{x_k} + \kappa_\epsilon v_{k-1} < f_{x_k} + \kappa_f v_{k-1}$ in this case $f_{x_k} \geq f_k^{up} - \kappa_f v_{k-1}$. Otherwise, when x_k is declared a null step, $f_{x_k} > f_{k-1}^{up} - \kappa_f v_{k-1}$. Once more, in both cases, the assumption that $v_k = v_{k-1}$ implies that $f_{x_k} \geq f_k^{up} - \kappa_f v_k$, and the desired result follows.

□

If $v_k = \underline{v}$ is a positive constant, the inequality (30) yields that the sequence $\{x_k\}$ is unbounded. However, if the constant is zero, the next result shows that the stopping test in Step 3 of Algorithm 4.2 is eventually satisfied. Before, we define the subset

$$\mathcal{A} \subset \{1, 2, \ldots\}\,, \text{ gathering iterations } k \text{ for which noise attenuation is required.} \tag{31}$$

Lemma 4.5. *If in Algorithm 4.2* $\liminf v_k = 0$ *then* $\lim_k v_k = 0$ *and* $\liminf V_k = 0$.

Proof. When $\liminf v_k = 0$ with $v_k \geq 0$, by monotonicity the whole sequence of estimated decreases converges to zero. In addition to subset $\mathcal{A}$ defined in (31), we consider the subset $\mathcal{K}$, gathering iterations k for which $v_k = (1 - \kappa_l)\Delta_k$. As $v_k \to 0$, then either $v_k \overset{\mathcal{K}}{\to} 0$ or $v_k \overset{\mathcal{A}}{\to} 0$. Suppose that $v_k \overset{\mathcal{K}}{\to} 0$. By Step 2, and possibly Step 6, $v_k = (1 - \kappa_l)\Delta_k$ for all $k \in \mathcal{K}$. Thus, $\Delta_k \overset{\mathcal{K}}{\to} 0$, and Lemma 4.5 holds, because in (28) the max-term is nonnegative, by Proposition 4.1.

Suppose now that $v_k \overset{\mathcal{A}}{\to} 0$. By construction, for all $k \in \mathcal{A}$ Step 4 is accessed, so $\tau_k > \tau_{\max}$ and $f_k^{lev} = f_k^{up} - v_k$. Together with (27), this means that

$$0 \leq \hat{e}_k + \tau_{\max} \|\hat{g}_k\|^2 < \hat{e}_k + \tau_k \|\hat{g}_k\|^2 = v_k \overset{\mathcal{A}}{\to} 0 \,.$$

So now in (28) the max-term goes to 0 for the $\mathcal{A}$-subsequence, yielding the desired result, because the gap Δ_k is nonnegative, by construction. □

If there are infinitely many noise attenuation steps, i.e., the index set $\mathcal{A}$ in (31) is infinity, Step 4 drives the nonincreasing sequence $\{v_k\}$ to zero. Thus, by Lemma 4.5 the optimality certificate eventually holds. Accordingly, to show the convergence of the method, it is enough to proof that $\liminf v_k = 0$ when noise is attenuated a finite number of times. This amounts to considering the two exclusive cases below:

- either there are infinitely many descent steps;
- or the stability center $\hat{x}_k$ remains unchanged after a finite number of iterations.

Proposition 4.6. *(Infinitely many descent steps) Consider Algorithm 4.2, for the instances considered in Table 2 with an oracle satisfying Assumption 2.1. If there are infinitely many descent steps and $f_* > -\infty$, then $\liminf V_k = 0$. If, in addition, Assumption 4.3 holds, the whole sequence $\{\hat{x}_k\}$ converges to an optimal solution to problem* (1)*.*

Proof. Since $f_k^{up} \geq f_* > -\infty$ for all k, $\liminf f_k^{up} > -\infty$. Let $j(\ell) = k(\ell+1) - 1$. The descent test (24) implies that

$$f_{k(\ell)}^{up} - f_{\hat{x}_{k(\ell+1)}} \geq \kappa_f v_{j(\ell)} \,. \tag{32}$$

To show that $\liminf V_k = 0$ we split our analysis for the various instances in Table 2:

– Instances `Ex`, `PI1` and `PI2'` are such that $f_{k(\ell)}^{up} = f_{\hat{x}_{k(\ell)}}$ for all ℓ. Summing the inequality (32) over ℓ yields that

$$\infty > f_{k(0)}^{up} - \lim_\ell f_{k(\ell)}^{up} \geq \sum_{\ell=0}^{\infty} \kappa_f v_{j(\ell)} \geq 0 \,,$$

i.e., $\liminf_k v_k = 0$.

– Instances `AE'` and `PAE'` set $f_{k(\ell)}^{up} = f_{\hat{x}_{k(\ell)}} + \epsilon_{\hat{x}_{k(\ell)}} = f_{\hat{x}_{k(\ell)}} + \kappa_\epsilon v_{k(\ell)-1}$ for all ℓ. Subtracting $\kappa_\epsilon v_{j(\ell)}$ in both sides of (32), and summing over ℓ gives

$$\infty > f_{k(0)}^{up} - \lim_\ell f_{k(\ell)}^{up} \geq \sum_{\ell=0}^{\infty} (\kappa_f - \kappa_\epsilon) v_{j(\ell)} \geq 0 \,.$$

Since by Table 2 $\kappa_\epsilon \in (0, \kappa_f)$, we have once more that $\liminf_k v_k = 0$.

In both instances, Lemma 4.5 provides $\liminf V_k = 0$.

When Assumption 4.3 holds, (26) written with x replaced by $\tilde{x}$ implies that $\langle \hat{g}_k, \tilde{x} - \hat{x}_k \rangle \leq \hat{e}_k$. As $\hat{x}_{k(\ell)} = \hat{x}_{k(\ell+1)-1}$ and $j(\ell) = k(\ell+1)-1$, we conclude that $\langle \hat{g}_{j(l)}, \tilde{x} - \hat{x}_{k(\ell)} \rangle \leq \hat{e}_{j(l)}$. Using this inequality and the descent condition (27) in the following expansion of squares we see that

$$\begin{aligned} |\tilde{x} - \hat{x}_{k(\ell+1)}|^2 &= |\tilde{x} - \hat{x}_{k(\ell)}|^2 + 2\langle \tilde{x} - \hat{x}_{k(\ell)}, \hat{x}_{k(\ell)} - \hat{x}_{k(\ell+1)} \rangle + |\hat{x}_{k(\ell+1)} - \hat{x}_{k(\ell)}|^2 \\ &= |\tilde{x} - \hat{x}_{k(\ell)}|^2 + 2\tau_{j(\ell)} \langle \hat{g}_{j(\ell)}, \tilde{x} - \hat{x}_{k(\ell)} \rangle + \tau_{j(\ell)}^2 \|\hat{g}_{j(\ell)}\|^2 \\ &\leq |\tilde{x} - \hat{x}_{k(\ell)}|^2 + 2\tau_{j(\ell)} (\hat{e}_{j(\ell)} + \tau_{j(\ell)} \|\hat{g}_{j(\ell)}\|^2) \\ &= |\tilde{x} - \hat{x}_{k(\ell)}|^2 + 2\tau_{j(\ell)} v_{j(\ell)} \,. \end{aligned}$$

Therefore, for any integer $n \geq 1$,

$$|\tilde{x} - \hat{x}_{k(\ell+n)}|^2 \leq |\tilde{x} - \hat{x}_{k(\ell)}|^2 + 2\tau_{max} \sum_{i=\ell}^{\infty} v_{j(\ell)} < \infty \,,$$

and the sequence $\{\hat{x}_{k(\ell)}\}_\ell = \{\hat{x}_k\}$ is bounded with cluster point denoted by $\bar{x} \in \mathcal{X}$. A similar reasoning, replacing $\tilde{x}$ by $\bar{x}$, implies that for arbitrary $\varepsilon > 0$ there exists ℓ such that $|\bar{x} - \hat{x}_{k(\ell)}|^2 \leq \varepsilon/2$ and $2\tau_{max} \sum_{i=\ell}^{\infty} v_{j(\ell)} \leq \varepsilon/2$, yielding convergence of the whole sequence, as claimed. By Lemma 4.5, $0 = \liminf V_k = \liminf \max\{V_k, |\hat{x}_k - \bar{x}|\}$, and by Proposition 4.1(iii), $\bar{x}$ is a solution to problem (1). □

The second case, of finitely many descent steps, is considered below.

Proposition 4.7. *(Finitely many descent steps)*
Consider Algorithm 4.2, for the instances considered in Table 2 with an oracle (2) *satisfying Assumption 2.1. Suppose that there are only finitely many noise attenuation steps. If there is a last stability center $\hat{x}$ such that eventually only null steps are done, then* $\liminf V_k = 0$*, and $\hat{x}$ is an optimal solution to problem* (1)*.*

Proof. Let k_1 denote the first iteration triggering the infinite sequence of consecutive null steps, all with stability center $\hat{x}$, for which there are no more noise attenuation steps. By the last assertion in Proposition 4.1(iii), written with $\bar{x} = \hat{x}$, we need to show that $\liminf V_k = 0$. For this, by Lemma 4.5, it is enough to prove that $\liminf v_k = 0$, or by (28), that $\Delta_k \to 0$.

Suppose that, for some iteration $k_2 > k_1$ the gaps satisfy $\Delta_k \geq \underline{\Delta} > 0$ for all $k \geq k_2$. In this case, because there are only finitely many noise attenuation steps, there exists some $\underline{v} > 0$ such that $v_k \geq \underline{v}$. Then for all $k \geq k_2$ it holds that $\hat{x}_k = \hat{x}$, $\Delta_k \geq \underline{\Delta}$ and $v_k = v_{k-1} \geq \underline{v}$. By construction, f_k^{up} remains fixed for $k \geq k_1$, the sequence $\{f_k^{lev}\}_{k\geq k_2}$ is nonincreasing (because, eventually once more, there is not noise attenuation in Step 4). By combining Lemma 3.7 and (30) we see that

$$\begin{array}{rcl} |x_{k+1} - \hat{x}_k|^2 & \geq & |x_k - \hat{x}_k|^2 + |x_{k+1} - x_k|^2 \\ & \geq & |x_k - \hat{x}_k|^2 + (1-\kappa_f)\underline{v}/\|g_{x_k}\| > 0\,. \end{array}$$

By Assumption 2.1 and [HUL93, Prop.XI.4.1.2] the sequence $\{g_{x_k}\}_k$ is locally bounded. So, by (12), $\tau_k^2\|\hat{g}_k\|^2 = |x_{k+1} - \hat{x}_k|^2 = |x_{k+1} - \hat{x}|^2 \to \infty$. Since $\hat{e}_k \geq 0$, it follows by (27) that $v_k \geq \tau_k\|\hat{g}_k\|^2$. Therefore, $\tau_k v_k \geq \tau_k^2\|\hat{g}_k\|^2 \to \infty$. Since the sequence $\{v_k\}$ is nonincreasing, we conclude that $\tau_k \to \infty$. Thus, Step 4 will be accessed eventually, contradicting the existence of an index k_1, the last one of noise attenuation step. Hence, $\{v_k\}$ must tend to zero, and the result is proved. □

We finish this section with the main convergence result.

Theorem 4.8. *Consider Algorithm 4.2, for the instances considered in Table 2 with an oracle* (2) *satisfying Assumption 2.1. If problem* (1) *has a solution, then the sequence $\{\hat{x}_k\}$ converges to a point that solves* (1)*. Otherwise, $|\hat{x}_k| \to \infty$ and* $\liminf V_k = 0$ *if $f_* > \infty$. In both cases, $f_k^{up} \downarrow f_*$.*

Proof. If Assumption 4.3 holds, for example when problem (1) has a solution, Propositions 4.6 and 4.7 show that $\hat{x}_k \to \bar{x}$, a solution to (1). Otherwise, $\{\hat{x}_k\}$ can not have a cluster point. In this case, Algorithm 4.2 performs infinitely many descent steps and $|\hat{x}_k| \to \infty$. If, in addition, $f_* > -\infty$, then Proposition 4.6 shows that $\liminf V_k = 0$.

In both cases, by (28) and the monotonicity of f_k^{up} we have that $f_k^{up} \downarrow f_*$, as stated.

□

In the next section we consider once more that $\mathcal{X}$ in (1) is compact, allowing for more general subproblems (12), to take advantage of geometrical properties of the feasible set.

5. Defining iterates with different stability functions

The level methods presented so far were developed using the simplest possible choice in (12). More precisely, the next iterate is defined by solving a problem of the form

$$\min_{x\in\mathbb{X}} \varphi(x;\hat{x}) \quad \text{where the } \textit{stability function} \quad \varphi(x;\hat{x}) := \frac{1}{2}|x-\hat{x}|^2$$

is just the Euclidean norm and $\hat{x} = \hat{x}_k$. All the complexity results given in Section 3 involve the diameter D of the n-dimensional feasible set. In order to exploit the geometry of this set, and get nearly dimension-independent complexity estimates, more general stability functions were proposed in [BTN05]. Accordingly, we consider abstract stability functions, defined as

$$\varphi(x;\hat{x}) = \omega(x) - \langle \nabla\omega(\hat{x}), x - \hat{x}\rangle$$

for some differentiable strongly convex function with parameter $\rho > 0$ (w.r.t. the norm $|\cdot|$), i.e.,

$$\omega(y) \geq \omega(x) + \langle \nabla\omega(x), y - x\rangle + \frac{\rho}{2}|y-x|^2 \quad \text{for all} \quad y, x \in \mathcal{X}\,.$$

In particular, the inequality

$$\varphi(y;\hat{x}) \geq \varphi(x;\hat{x}) + \langle \nabla\varphi(x;\hat{x}), y-x\rangle + \frac{\rho}{2}|y-x|^2 \tag{33}$$

holds for all $y, x \in \mathcal{X}$.

The interest of this setting is to replace the diameter D in the convergence results by the ω-dependent constant Ω/ρ, with Ω given by

$$\Omega := \max_{y,x\in\mathcal{X}} [\omega(y) - \omega(x) - \langle \nabla\omega(x), y-x\rangle]\,. \tag{34}$$

By suitably choosing the function ω it is possible to fit well the geometry of $\mathcal{X}$ and obtain small values for Ω/ρ (the smaller this ration is, the better). This feature is well illustrated by some examples from [BTN05], recalled below:

Simplex-choice: when $\mathcal{X}$ in (1) is a simplex, a good choice is the regularized entropy $\omega(x) = \sum_{i=1}^n (x_i - \delta/n)\mathtt{ln}(x_i - \delta/n)$, where $\delta \in (0,1)$ is a fixed regularization constant, usually small. The parameter of strong convexity is $\rho = 1/(1+\delta)$ w.r.t. the ℓ_1-norm.

Spectahedron-choice: when (1) is constrained to the set of all symmetric positive definite matrices of order n with trace smaller than one, it is convenient to take the function: $\omega(x) = \mathtt{Tr}[(x - \delta/nI_n)\mathtt{ln}(x-\delta/nI_n)]$, for $\delta \in (0,1)$, I_n the identity matrix of size n, and $x \in \Re^{n\times n}$. With this choice, the parameter of strong convexity is $\rho = 1/2(1+\delta)$, w.r.t. the Frobenius norm.

Ball-choice: if the feasible set is the unit ball in the Euclidean space, taking $\omega(x) = \frac{1}{2}\langle x, x\rangle$ yields $\rho = 1$ w.r.t. the Euclidean norm.

We will show convergence for the proximal variant of Algorithm 3.3 ($\delta_{\mathtt{Prox}} = 1$) when, instead of using (12), iterates are defined by

$$x_{k+1} := \arg\min_{x\in\mathbb{X}_k} \varphi(x;\hat{x}_k)\,. \tag{35}$$

Clearly, for this more general approach to be of any use, subproblems (35) should be easy to solve. Such is indeed the case for the aforementioned simplex, spectahedron and ball choices; we refer to [BTN05] for a further discussion on this matter.

The following result is useful for the method to keep the bundle size bounded along iterations.

Proposition 5.1. *Consider Proposition 3.2 with* (12) *replaced by* (35)*. Then both* (14) *and* (15) *hold.*

Proof. Let x_{k+1} be the iterate obtained by (35). The inequalities in (14) follow from the subgradient inequality. To show (15), first suppose that $\check{f}_k(x_{k+1}) = f_k^{lev}$. By the definition in (14)

$$\begin{aligned}
\mathbb{X}_{-k} &= \{x\in\mathcal{X}:\ \check{f}_k(x_{k+1}) + \langle \hat{g}_k, x - x_{k+1}\rangle \leq f_k^{lev}\}\\
&= \{x\in\mathcal{X}:\ \langle \hat{g}_k, x - x_{k+1}\rangle \leq 0\}\\
&= \{x\in\mathcal{X}:\ -\tfrac{1}{\tau_k}\langle \nabla\varphi(x_{k+1};\hat{x}_k), x - x_{k+1}\rangle \leq 0\}\,.
\end{aligned}$$

Hence, $\langle \nabla\varphi(x_{k+1};\hat{x}_k), x - x_{k+1}\rangle \geq 0$ for all $x \in \mathbb{X}_{-k}$, which is nothing but the first order optimality condition for problem (35), written with $\mathbb{X}_k$ replaced by $\mathbb{X}_{-k}$.

Now suppose that $\check{f}_k(x_{k+1}) < f_k^{lev}$. The KKT conditions for (35) imply that $\tau_k = 0$ with $\nabla\omega(x_{k+1}) = \nabla\omega(\hat{x}_k)$. Since the function ω is strongly convex, it follows that $x_{k+1} = \hat{x}_k$. Moreover, the relation $\bar{f}_{-k} \leq \check{f}_k$ implies that the constraint $\bar{f}_{-k}(x) \leq f_k^{lev}$ is inactive at the solution of problem (35), written with $\mathbb{X}_k$ replaced by $\mathbb{X}_{-k}$. Therefore $\hat{x}_k$ also solves such problem, and the result is proved. □

Given the above result, we now estimate the maximum number of iterations in K^ℓ.

Proposition 5.2. *Suppose the feasible set $\mathcal{X}$ in* (1) *is compact and the oracle* (2) *satisfies Assumption 2.1. Let Λ be a Lipschitz constant for f in* (1) *and λ be the instance-dependent constant in Table 1. Consider the proximal variant of Algorithm 3.3 ($\delta_{\mathtt{Prox}} = 1$), and suppose that in Step 3 the QP problem* (12) *is replaced by the possibly more general convex program* (35)*. The for any iteration $k \in K^\ell$ with $\Delta_k > \delta_{\mathtt{Tol}}$ the number of iterations between $k(\ell)$ and k is bounded by*

$$\frac{2\Omega}{\rho}\left(\frac{\Lambda}{\lambda\Delta_k}\right)^2 + 1\,,$$

where Ω is defined in (34) *and ρ is the parameter of strong convexity of ω with respect to the norm $|\cdot|$.*

Proof. At iteration $k-1$ of the algorithm, consider the aggregate index that enters in the bundle at Step 5. Since $-(k-1) \in J_k$ the next model function satisfies $\check{f}_k \geq \bar{f}_{-(k-1)}$. Together with (19), this means that $\mathbb{X}_k \subset \mathbb{X}_{-(k-1)}$. Since $x_{k+1} \in \mathbb{X}_k$ by construction, we see that $x_{k+1} \in \mathbb{X}_{-(k-1)}$.

By Proposition 5.1, x_k solves problem (35) written with $\mathbb{X}_{k-1}$ replaced by $\mathbb{X}_{-(k-1)}$, so $\left\langle \nabla\varphi(x_k;\hat{x}_{k(\ell)}), x - x_k \right\rangle \geq 0$ for all $x \in \mathbb{X}_{-(k-1)}$. In particular when $x = x_{k+1}$ in the inequality (33), we have that

$$\varphi(x_{k+1};\hat{x}_{k(\ell)}) \geq \varphi(x_k;\hat{x}_{k(\ell)}) + \frac{\rho}{2}|x_{k+1} - x_k|^2 .$$

In view of (34),

$$\Omega \geq \varphi(x_{k+1};\hat{x}_{k(\ell)}) - \varphi(\hat{x}_{k(\ell)};\hat{x}_{k(\ell)}) \geq \frac{\rho}{2}\sum_{j=k(\ell)+1}^{k} |x_{k+1} - x_k|^2 \geq \frac{\rho}{2}\sum_{j=k(\ell)+1}^{k} \left(\lambda\frac{\Delta_k}{\Lambda}\right)^2 ,$$

because $\max_{x\in\mathcal{X}} \varphi(x;\hat{x}_{k(\ell)}) - \min_{x\in\mathcal{X}} \varphi(x;\hat{x}_{k(\ell)}) \leq \Omega$, using Lemma 3.4. □

Regarding the constant λ, our proximal level method for oracles with on-demand accuracy has the same complexity estimate than the level method given in [BTN05] for exact oracles.

The following result gives an upper bound for the maximum number of iterations in a cycle, when the initial oracle error is carefully chosen. Specially, consider

$$\epsilon_{x_1} \leq \frac{1-\lambda}{\lambda} R\Lambda, \quad \text{with} \quad R := \max_{x\in\mathcal{X}} |x - x_1| \quad \text{and } \lambda \text{ given in Table 1} . \tag{36}$$

Lemma 5.3. *Consider the assumptions in Proposition 5.2. If, in addition, the initial oracle error ϵ_{x_1} satisfies* (36)*, then*

$$k - k(\ell) + 1 \leq \frac{4\Omega}{\rho}\left(\frac{\Lambda}{\lambda\Delta_k}\right)^2 \quad \textit{for all} \quad k \in K^\ell . \tag{37}$$

Proof. Consider the relations

$$\Delta_k \leq \Delta_1 = f_1^{up} - f_1^{low} = f_1^{up} - \min_{x\in\mathcal{X}}[f_{x_1} + \langle g_{x_1}, x - x_1\rangle] \leq f_1^{up} - f_{x_1} + \max_{x\in\mathcal{X}} |x - x_1|\Lambda ,$$

where the last inequality is due to (3), the compactness of $\mathcal{X}$ and the Cauchy-Schwarz inequality. Since $f_1^{up} = f_{x_1} + \epsilon_{x_1}$, by (36) we have that

$$\Delta_k \leq \epsilon_{x_1} + R\Lambda \leq \frac{1}{\lambda} R\Lambda .$$

By strong convexity, combining (34) and (36), we see that $\rho R^2/2 \leq \Omega$, so

$$\frac{2\Omega}{\rho}\left(\frac{\Lambda}{\lambda\Delta_k}\right)^2 \geq R^2\left(\frac{\Lambda}{\lambda\Delta_k}\right)^2 \geq R^2\left(\frac{\Lambda\lambda}{\lambda R\Lambda}\right)^2 = 1 ,$$

and Proposition 5.2 yields the desired result. □

Next we give the last complexity result whose proof is analogous the one used to show Theorem 3.9.

Theorem 5.4. *Under the assumptions of Proposition* (5.2)*. If the initial oracle error satisfies* (36)*, then, to reach an optimality gap smaller than $\delta_{Tol} > 0$ it is enough to perform at most*

$$\frac{\Omega}{\rho}\left(\frac{\Lambda}{\delta_{Tol}}\right)^2 \frac{4}{(1-(1-\kappa_l)^2)\lambda^2} \quad \textit{iterations.}$$

□

Condition (36) is useful only for obtaining the estimate above. Even if (36) does not hold, Proposition 5.2 is enough to show that the algorithm terminates after finitely many steps. The smaller the ω-dependent ration Ω/ρ is, the less iterations are required by the method. Depending on the choice of ω, problem (35) may no longer be a quadratic program and some nonlinear programming method may be needed. Nevertheless, the dual problem of (35) has as many variables as the bundle dimension, which can be kept as small as desired.

Finally, note that when $\mathcal{X}$ is not a polyhedron, problem (8) is no longer a linear program. This is not a major issue for Algorithm 3.3 when $\delta_{\texttt{Prox}} = 1$, since this variant needs to solve (8) only once, at Step 0, to compute the lower bound f_1^{low}. In fact, if any lower bound for f_* is available at the beginning of optimization process, there is no need at all of solving (8). This is an advantage of our method over [BTN05], which needs to solve both (8) and (35) at each iteration.

6. Numerical assessment

We now report on results obtained by Algorithm 3.3 with the instances in Table 1 and Algorithm 4.2 with the instances in Table 2, for the inexact oracle using the Dual Feasibility Strategy in Section 2.

We consider ten two-stage stochastic linear programming problems as in (4), available at the homepage `http://web.uni-corvinus.hu/~ideak1/kut_en.htm#`, by I. Deák, named therein as 20x20.1, 20x40.1, 20x60.1, 40x20.1, 40x40.1, 40x60.1, 60x20.1, 60x40.1, 60x60.1, and 100x20.1. The expression n_1x$\,m_2$.1 refers to the size of the optimization problem: there are n_1 first-stage variables and m_2 second-stage constraints. Also, the battery is designed so that there are $m_1 = n_1/2$ first-stage constraints and $n_2 = 3m_2/2$ second-stage variable. As for the right hand side vectors $h \in \Re^{m_2}$, they are normally distributed. All the problems satisfy the assumption of fixed complete recourse, so problem (4) has a solution for all $x \in \mathcal{X}$ and each possible scenario.

For each problem, fifteen instances corresponding to

$$N \in \{10, 20, 30, 40, 50, 80, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$$

scenarios were considered to benchmark nineteen different solvers. Since all solvers reached the optimal values within a relative tolerance fixed to 10^{-5}, accuracy is not an issue, and comparisons are done in terms of CPU time only.

The runs were done using Matlab 7.7.0 (R2008b), on an Intel(R) Core(TM) i3 CPU computer with 2.27 GHz, 4 GB RAM, running under Windows 7 OS. To solve the quadratic programming problems (12) we used MOSEK optimization toolbox for Matlab, `http://www.mosek.com`. For the on-demand accuracy oracle (2), the linear programs (4) were solved with the primal-dual code `lp236a`, available at `http://www.ee.ucla.edu/ee236a/ee236a.html`, by L. Vandenberghe.

6.1. **Main features of the benchmark.**

6.1.1. *Solvers.* In addition to Algorithm 3.3 with Table 1 and Algorithm 4.2 with Table 2, we consider the proximal bundle method [BGLS06], the partly inexact proximal bundle method [Kiw09], the classical L-shaped method [SW69], and the inexact Bender's decomposition [ZPR00], noting that the last two solvers are cutting-plane methods using exact and asymptotically exact oracles, respectively.

For each variant in the benchmark we use the mnemonics CP, μ and ℓ to refer to the NSO algorithm: cutting-plane, proximal, and level method, respectively. The second ingredient of the solver is in the type of oracle employed, denoted using the short form in Table 1. Accordingly, we consider the nineteen solvers below:

CP-`Ex`: the L-shaped method [SW69]);

CP-`AE`: the inexact Bender's decomposition [ZPR00];

μ-`Ex`: the proximal bundle method [BGLS06];

μ-`PI2`: the partly inexact proximal bundle method [Kiw09];

$\ell\delta_{\texttt{Prox}}$-{`Ex`, `PI1`, `PI2`, `AE`, `PAE`}: the level method with on-demand accuracy (Alg. 3.3), where the suffix $\delta_{\texttt{Prox}} \in \{0, 1\}$ informs if the non-proximal or proximal variant is considered, and declined for the oracle instances in Table 1;

$\mathcal{G}\ell$-{`Ex`, `PI1`, `PI2`, `AE`, `PAE`}: the generalized level method with on-demand accuracy (Alg. 4.2) with the instances in Table 2 (we suppressed the quotes on the notation `PI2',AE' and PAE')`.

In this manner, ℓ1-`PI1` (respectively ℓ0-`PI1`) is Algorithm 3.3 in its proximal (respectively non-proximal) variant, for an oracle defined by the Dual Feasibility Strategy using instance `PI1` in Table 1. Likewise, $\mathcal{G}\ell$-`PI2` is Algorithm 4.2 with instance `PI2'` from Table 2.

We mention that:

ℓ0-`Ex` corresponds to the level method given in [LNN95];

ℓ1-`Ex` corresponds to the proximal level method given in [Kiw95];

ℓ0-`AE` corresponds to the inexact level decomposition given in [Fáb00, FS07];

$\mathcal{G}\ell$-`Ex` corresponds to the proximal level given in [BKL95].

Since not all of the solvers allow for bundle compression, no aggregation mechanism was implemented, only inactive linearizations were eliminated when possible.

6.1.2. *Stopping tolerance, nonsmooth parameters and initial point.* Both μ-`Ex` and μ-`PI2` stop when the relative stopping test given in [OSS11, Sec. 5.1.2] is reached for an absolute tolerance equal to 10^{-5}. The remaining algorithms stop when the fraction Δ_k/f_{ub}^k is less than or equal to 10^{-5}.

All methods use as starting point x_1 the solution (on variable x) of the *expected value problem* [BL97, Section 4.2]:

$$EV = \left\{ \begin{array}{lllll} \min & \langle c, x\rangle & + & \langle q, y\rangle & \\ & Ax & & & = b \\ & Tx & + & Wy & = \bar{h} := \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} h_i \\ & x, y & & & \geq 0\,, \end{array} \right.$$

Since W and q are fixed, EV is a lower bound for f (c.f. Example 2.3). Accordingly, we take $f_1^{low} = EV$ for all solvers, except for the μ-family.

6.2. **Assessing performance of the solvers.** We give here condensed tables, complete results are given in the appendix: Table with $CPU\%$ values (defined below), averaged for each solver over all ten problems and each one of the fifteen instances.

Table 3 reports the total CPU running times in hours of each solver for solving all of 150 problems (10 families and 15 instances).

TABLE 3. Total CPU times in hours

CP Ex	CP AE	μ Ex	ℓ1 Ex	$\mathcal{G}\ell$ Ex	ℓ1 PAE	ℓ1 AE	$\mathcal{G}\ell$ AE	ℓ0 Ex	μ PI2	ℓ0 AE	$\mathcal{G}\ell$ PI1	ℓ1 PI1	ℓ1 PI2	$\mathcal{G}\ell$ PI2	$\mathcal{G}\ell$ PAE	ℓ0 PI1	ℓ0 PI2	ℓ0 PAE
7.4	6.3	5.6	5.2	5.1	4.4	4.4	4.2	4.2	3.5	3.5	2.8	2.6	2.5	2.5	2.5	2.4	2.3	2.1

The solvers in Table 3 are arranged in decreasing order of total CPU time. Accordingly, the cutting-plane methods (exact and asymptotically exact) are the slowest ones, while the partly inexact/asymptotically exact level are the fastest.

To determine the impact of inexact oracles in the different methods, we use the measure

$$CPU\% := 100\frac{CPU_{cp-\texttt{Ex}} - CPU_{solv}}{CPU_{cp-\texttt{Ex}}},$$

where $CPU_{cp-\texttt{Ex}}$ and CPU_{solv} are the computational time required to solve a given problem using CP-`Ex` and another solver, respectively. So the best solvers will be those giving the largest values of this measure. Table 4 reports the percent CPU time reduction for each solver, ordered by increasing percents reductions.

TABLE 4. Percentile CPU time reduction ($CPU\%$)

CP AE	μ Ex	ℓ1 Ex	$\mathcal{G}\ell$ Ex	ℓ1 PAE	ℓ1 AE	$\mathcal{G}\ell$ AE	ℓ0 Ex	μ PI2	ℓ0 AE	$\mathcal{G}\ell$ PI1	ℓ1 PI1	ℓ1 PI2	$\mathcal{G}\ell$ PI2	$\mathcal{G}\ell$ PAE	ℓ0 PI1	ℓ0 PI2	ℓ0 PAE
15	24	30	31	41	41	43	43	53	53	62	65	66	66	66	68	69	72

By comparing the performance of the different solvers we note that:

- the only non-bundle solver in Table 4, CP-`AE`, which is the inexact Benders decomposition [ZPR00], is the one with the lowest performance: only 15% of CPU time reduction is observed. The improvement ratio reported in [ZPR00] is higher: 27% for a different battery of test-problems. However, the algorithm progress is likely to systematically stall as iterations go by, due to the well-know "tailing-off" effect of cutting-plane methods.
- In their respectively classes, all the inexact solvers perform better than the exact ones.
- For exact bundle variants, proximal and level methods are quite comparable, since both μ-`Ex` and ℓ1-`Ex` reduce the running times of the L-shaped method in 24% and 30%, respectively. The better performance of ℓ1-`Ex` over μ-`Ex` is due to the initial lower bound $f_1^{low} = EV$, used by the level methods, which is updated along the iterations and allows for a faster satisfaction of the stopping test (for level methods, $\Delta_k / f_k^{up} \leq \delta_{\texttt{Tol}}$). By contrast, the proximal bundle methods do not use lower bounds information; as result, they need more iterations (in practice more null-steps) to check that the last descent-step is a $\delta_{\texttt{Tol}}$-solution.
- As ℓ0 solvers update the lower bound at each iteration by solving an additional linear program (8), satisfaction of stopping test is obtained in less iterations than the ℓ1 solvers. Since the CPU time for solving each additional linear program is negligible when compared to the CPU time required by each oracle call, the ℓ0-family is faster than the ℓ1-family.
- For the inexact oracles, the time reduction of μ-`PI2`, the partly inexact proximal bundle method [Kiw09], is poorer than any other level variant with similar oracle instances. Specially μ-`PI2` reduces the CPU time in 53%, while ℓ1-`PI2` and ℓ0-`PI2` yield a reduction of 66% and 69%, respectively.
- All the level variants using oracles that are partly inexact or partly asymptotically exact gave good results, reducing the running time in more than 41%.
- Versatility pays off: the asymptotically exact level method [Fáb00], corresponding to ℓ0-`AE`, performs less well than its partly asymptotically exact counterpart, ℓ0-`PAE` (53% of time reduction against 72%).

7. Concluding remarks

Starting from the first level method introduced in [LNN95], we have developed a class of level methods combining versatility in the oracle accuracy with the flexibility of choosing stability centers and subproblems that define the next iterate. Our level methods are particularly interesting for nonsmooth convex optimization problems with objective function that is difficult to evaluate and/or with compact feasible set whose geometrical structure can be algorithmically exploited. Although the approach allows for inexact oracle evaluations, all of our variants ensure that an exact solution to the optimization problem is found. Additionally, and differently from [Ric11], neither the knowledge of the diameter of the feasible set nor the Lipschitz constant of f are required for implementation of our methods.

Our setting is quite general: Algorithm 3.3 includes the ideas of [LNN95], [Kiw95], [Fáb00], and [BTN05], while Algorithm 4.2 is based on [BKL95]. Regarding Algorithm 3.3, the partly inexact variants ℓ0-`PI1` and ℓ1-`PI1` have the same complexity result than their exact versions ℓ0-`Ex` and ℓ1-`Ex`, respectively developed in [LNN95] and [Kiw95]. When the chosen metric exploits the geometrical properties of the feasible set, our variants ℓ1-`Ex` and ℓ1-`PI1` have the same complexity than the exact method proposed in [BTN05]. However, unlike [BTN05], neither method in the ℓ1-family needs solving a linear program at each iteration for updating the lower bound.

The $\mathcal{G}\ell$-family was designed to deal with convex optimization problems with unbounded feasible sets. Similarly to the ℓ1-family, our $\mathcal{G}\ell$-methods have the ability of keeping bounded the bundle size, and of updating the algorithm lower bound without solving a linear program.

Our numerical experiments have shown that the versatility of the on-demand accuracy oracles pays off. For a large battery of two stages stochastic programming problems, CPU times were reduced in up to 36%, when compared to the asymptotically exact level method (ℓ0-`AE`) from

[Fáb00], or to the partly inexact proximal bundle method (μ-PI1) given in [Kiw09]. In addition, our variants proved to be up to 72% faster than the exact L-shaped method.

Finally, for those applications with oracles so expensive that even the on-demand accuracy setting is prohibitive, it is possible to define level bundle methods dealing with inexact oracles, as long as inaccuracy remains bounded. This type of oracles has been considered for bundle methods using a proximal QP in [Kiw06] and [OSS11], and for level methods for compact feasible sets in the PhD dissertation of the first author.

References

[BGLS06] J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical optimization. theoretical and practical aspects*, Universitext, Springer-Verlag, Berlin, 2006, Second edition, xiv+490 pp.

[BKL95] Ulf Brannlund, Krzysztof C. Kiwiel, and P. O. Lindberg, *A descent proximal level bundle method for convex nondifferentiable optimization*, Operations Research Letters **17** (1995), no. 3, 121 – 126.

[BL97] John R. Birge and Francois Louveaux, *Introduction to stochastic programming*, Springer Science, New York, 1997.

[BTN05] Aharon Ben-Tal and Arkadi Nemirovski, *Non-euclidean restricted memory level method for large-scale convex optimization*, Math. Program. **102** (2005), 407–456.

[ES10] Grégory Emiel and Claudia Sagastizábal, *Incremental-like bundle methods with application to energy planning*, Computational Optimization and Applications **46** (2010), 305–332.

[Fáb00] Csaba Fábián, *Bundle-type methods for inexact data*, Central European Journal of Operations Research **8 (special issue, T. Csendes and T. Rapcsk, eds.)** (2000), 35–55.

[FS07] Csaba Fábián and Zoltán Szőke, *Solving two-stage stochastic programming problems with level decomposition*, Computational Management Science **4** (2007), 313–353.

[Hin01] Michael Hintermüller, *A proximal bundle method based on approximate subgradients*, Computational Optimization and Applications **20** (2001), 245–266, 10.1023/A:1011259017643.

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms*, Grund. der math. Wiss, no. 305-306, Springer-Verlag, 1993, (two volumes).

[Kiw90] Krzysztof C. Kiwiel, *Proximity control in bundle methods for convex nondiferentiable minimization*, Mathematical Programming **46** (1990), 105–122.

[Kiw95] ———, *Proximal level bubdle methods for convex nondiferentiable optimization, saddle-point problems and variational inequalities*, Math. Program. **69** (1995), no. 1, 89–109.

[Kiw06] ———, *A proximal bundle method with approximate subgradient linearizations*, SIAM Journal on Optimization **16** (2006), no. 4, 1007–1023.

[Kiw09] ———, *Bundle methods for convex minimization with partially inexact oracles*, Tech. report, Optimization Online, 2009.

[LNN95] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov, *New variants of bundle methods*, Math. Program. **69** (1995), no. 1, 111–147.

[LS97] Claude Lemarchal and Claudia Sagastizbal, *Variable metric bundle methods: From conceptual to implementable forms*, Mathematical Programming **76** (1997), 393–410, 10.1007/BF02614390.

[OSL12] Welington Oliveira, Claudia Sagastizábal, and Claude Lemaréchal, *Inaccurate bundle methods in deph: a unified analysis*, In preparation, 2012.

[OSS11] Welington Oliveira, Claudia A. Sagastizbal, and Susana Scheimberg, *Inexact bundle methods for two-stage stochastic programming*, SIAM Journal on Optimization **21** (2011), no. 2, 517–544.

[Ric11] Peter Richtárik, *Approximate level method for nonsmooth convex minimization*, Journal of Optimization Theory and Applications (2011), 1–17.

[SDR09] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński, *Lectures on stochastic programming: Modeling and theory*, MPS-SIAM Series on Optimization, SIAM - Society for Industrial and Applied Mathematics and Mathematical Programming Society, Philadelphia, 2009.

[Sol03] Mikhail V Solodov, *On approximations with finite precision in bundle methods for nonsmooth optimization*, Journal of Optimization Theory and Applications **119** (2003), no. 1, 151–165.

[SW69] R.M. Van Slyke and R.J.-B. Wets, *L-shaped linear programs with applications to optimal control and stochastic programming*, SIAM Journal of Applied Mathematics **17** (1969), 638–663.

[ZFEM09] V. Zverovich, C. Fábián, F. Ellison, and G. Mitra, *A computational study of a solver system for processing two-stage stochastic linear programming problems*, 2009, [Online: Stand 2010-10-30T17:57:37Z].

[ZPR00] Golbon Zakeri, Andrew B. Philpott, and David M. Ryan, *Inexact cuts in Benders decomposition*, SIAM J. Optim. **10** (2000), no. 3, 643–657 (electronic). MR MR1741190 (2001b:90043)

Appendix

TABLE 5. Percent CPU time reduction for each problem.

Problem	μ Ex	CP AE	$\mathcal{G}\ell$ Ex	$\ell 1$ Ex	$\ell 1$ AE	$\ell 1$ PAE	$\ell 0$ Ex	$\mathcal{G}\ell$ AE	μ PI2	$\ell 0$ AE	$\mathcal{G}\ell$ PI1	$\mathcal{G}\ell$ PAE	$\ell 1$ PI1	$\ell 0$ PI1	$\mathcal{G}\ell$ PI2	$\ell 1$ PI2	$\ell 0$ PI2	$\ell 0$ PAE
20x20	-23	13	8	11	17	18	6	21	29	22	42	45	50	44	52	54	50	48
20x40	-24	-1	2	13	-6	2	11	11	28	-1	23	12	38	38	29	40	36	34
20x60	27	14	38	30	39	39	37	46	51	45	63	62	69	62	68	69	65	66
40x20	11	12	23	22	29	29	24	35	44	35	53	54	55	54	58	58	59	60
40x40	11	9	23	19	32	31	25	28	35	32	45	48	53	54	57	56	56	57
40x60	18	11	14	20	32	20	32	39	38	43	45	55	54	53	53	55	56	55
60x20	-35	19	31	29	43	43	40	44	42	52	62	68	66	66	66	67	69	72
60x40	37	24	34	36	50	51	55	47	50	62	54	60	56	67	59	55	65	71
60x60	27	16	47	44	50	50	55	49	45	59	62	63	64	65	61	65	64	68
100x20	-18	15	21	22	29	32	34	39	24	43	57	63	58	61	64	62	64	66
mean	3	13	24	25	31	31	32	36	38	39	51	53	56	56	57	58	58	60