Uncontrolled inexact information within bundle methods

Jérôme Malick, × Welington de Oliveira, Sofia Zaourar-Michel

Noname manuscript No.
(will be inserted by the editor)

Uncontrolled inexact information within bundle methods

Jérôme Malick · Welington de Oliveira · Sofia Zaourar

Received: date / Accepted: date

Abstract We consider convex nonsmooth optimization problems where additional information with uncontrolled accuracy is readily available. It is often the case when the objective function is itself the output of an optimization solver, as for large-scale energy optimization problems tackled by decomposition. In this paper, we study how to incorporate the uncontrolled linearizations into (proximal and level) bundle algorithms in view of generating better iterates and possibly accelerating the methods. We provide the convergence analysis of the algorithms using uncontrolled linearizations, and we present numerical illustrations showing they indeed speed up resolution of two stochastic optimization problems coming from energy optimization (two-stage linear problems and chance-constrained problems in reservoir management).

Keywords Nonsmooth optimization, bundle methods, inexact oracle, energy optimization, two-stage stochastic problems, chance-constrained problems

Mathematics Subject Classification (2000) 65K05 · 49J52 · 49M27 · 90C15 · 90C25 · 90C27

J. Malick
CNRS, LJK, Grenoble, France
E-mail: jerome.malick@inria.fr

W. de Oliveira
UERJ, Rio de Janeiro, Brazil
BCAM, Bilbao, Basque Country - Spain
E-mail: welington@ime.uerj.br

S. Zaourar
INRIA, UJF, Grenoble, France
E-mail: sofia.zaourar@inria.fr

1 Introduction: context, problem, and contributions

1.1 Nonsmooth minimization with an (inexact) oracle

We consider nonsmooth optimization problems of the form

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

with a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$; and a (nonempty) polyhedral set $X \subset \mathbb{R}^n$, and we assume that the infimum is finite ($f_* > -\infty$). Typically, the nonsmoothness of f comes after a maximization, i.e. when f itself is the result of an inner optimization problem

$$f(x) = \sup_{u\in U} h(u,x) \tag{2}$$

where $h(u,\cdot)$ are convex for each u lying in a set U. Such nonsmooth objective functions appear in Lagrangian relaxation (see e.g. [Lem01]), in stochastic optimization with recourse (see e.g. [SDR09]), or in Benders decomposition (see e.g. [Geo72]).

For a fixed accuracy $\eta \geq 0$, a so-called (lower) η-oracle of f provides, for a point $x \in X$ as an input, an approximate value and an approximate linearization

$$\begin{cases} f_x \in \mathbb{R} \quad \text{such that} \quad f(x) - \eta \leq f_x \leq f(x), \\ g_x \in \mathbb{R}^n \quad \text{such that} \quad f_x + g_x^\top(\cdot - x) \leq f(\cdot). \end{cases} \tag{3}$$

If the oracle error is null ($\eta = 0$), the oracle returns the exact value $f_x = f(x)$ and a subgradient $g_x \in \partial f(x)$. For some problems, as in large-scale stochastic optimization or in combinatiorial optimization, computing exact information on f is expensive, or even out-of-reach, whereas computing some inexact information ($\eta > 0$) is still possible. For example, when f is given by (2), any $\bar{u} \in U$ gives an inexact value and an approximate linearization of f at a given $x \in X$. Indeed, the convexity of $h(u,\cdot)$ yields

$$h(\bar{u},x) + g^\top(z-x) \leq h(\bar{u},z) \leq f(z), \quad \text{for any } g \in \partial_x h(\bar{u},x).$$

So we have inexact information on f by taking

$$f_x = h(\bar{u},x) \quad \text{and} \quad g_x = g \in \partial_x h(\bar{u},x). \tag{4}$$

In this case, an η-oracle maximizes $h(\cdot,x)$ over U up to the tolerance η, i.e., computes $\bar{u} \in U$ satisfying $f(x) - \eta \leq h(\bar{u},x) \leq f(x)$ so that (4) gives the η-information (3).

Among the nonsmooth optimization methods to solve problems (1) with f known by an oracle (3), are the bundle-type methods: the Kelley method [Kel60, HUL93], proximal bundle methods [HUL93], level bundle methods [LNN95], generalized bundle methods [Fra02], and doubly stabilized bundle methods [dS15]. Initially devellopped for exact oracles ($\eta = 0$), these methods have been extended to handle inexact oracles ($\eta > 0$) and to solve (1) up to an accuracy of η. Complete convergence analysis of these methods exists; roughly speaking, under some assumptions, the iterates x_k are an η-minimizing sequence

$$f_* \leq \liminf f(x_k) \leq f_* + \eta. \tag{5}$$

We refer to [Hin01] and [Sol03] for first articles, [ZPR00] for an inexact version of the Kelley method, [Fáb00] for an inexact level method, [Kiw06] and [dOSL14] for inexact proximal bundle methods, and [dOS14] for inexact level methods with vanishing errors.

1.2 Inexact oracle... and more

For some optimization problems as above with an η-oracle, there is in fact additional uncontrolled information on f, which is already available or cheap to get.

A typical example is in combinatorial optimization when f has the form (2), with a discrete set U and with a Lagrangian function h (see e.g. [Lem01]). In this case, exact or approximate resolution schemes produce "good" feasible points $\bar{u} \in U$, that give, in turn, linearizations of f by (4) – but with uncontrolled accuracy, so that this cannot be used for an oracle with fixed accuracy η. For instance, when (2) is solved by a branch-and-bound algorithm, feasible solutions are generated during the exploration of the branch-and-bound tree, but only the final one, the optimal solution, is used by the oracle to generate (3). The (uncontrolled) information (4) produced by the intermediate feasible solutions is not used, whereas it is available for free and possibly fine (since nearly optimal solutions are usually obtained soon in the branch-and-bound process). It is the same situation when we have cheap heuristics computing solutions that are "good" in practice (sometimes with probabilistic guarantees) but without the (deterministic) guarantee required for an η-oracle. We will consider in section 4 an energy optimization problem with such an efficient specific heuristic; other examples include p-median problems [BTV06] and unit-commitment problems (see e.g. the recent review [TvAFL15]).

Another type of example of cheap uncontrolled information appears in two-stage stochastic linear problems (see e.g. [SDR09], and applications to energy problems in [ZPR00] and [dOSP$^+$10]). In this case, the function has a form (2) with separable terms corresponding to linear maximization subproblems

$$f(x) = c^\top x + \sum_{i=1}^{N} \pi_i f_i(x) \qquad \text{with} \qquad f_i(x) = \sup_{W^\top u \leq q} \; (h_i - Tx)^\top u, \tag{6}$$

for given N, π_i, h_i, T, W and q (details to come in section 4). Computing exact information on f requires to solve the N linear optimization subproblems, which is costly when N is large. Solving only a fraction of these subproblems (say 10%) still gives inexact uncontrolled information on f. Indeed if we compute $\bar{u}_i$ an optimal solution giving $f_i(x)$, then we can also use it to under-approximate other terms $f_j(x)$ (since the feasible sets are the same, we have $(h_j - Tx)^\top \bar{u}_i \leq f_j(x)$). Thus, for a given fraction of solved problems, we have an inexact linearization but with an unknown accuracy.

We formalize the situation where we can compute controlled information together with some uncontrolled inexact information by assuming that we have

$$\begin{array}{c}\text{an oracle with accuracy bounded by } \eta \geq 0, \text{ and} \\ \text{a "cutting-plane generator" adding linearizations with uncontrolled accuracy.}\end{array} \tag{7}$$

This abstract cutting-plane generator should be seen as an external module, having the previous bundle of linearizaions as an input, and adding other linearizations without calling the η-oracle. There is no other requirement on the generator: it can use information already available, call heuristics, or even run optimization algorithms. For example, in our numerical experiments, the cutting-plane generators will add inexact (uncontrolled) linearizations produced during a fixed number of iterations of a standard bundle method using heuristics.

Note that the situation (7) does not fit in the context of ”on-demand accuracy oracles” of [dOS14] where the oracle both requires and provides more information. Note also that the cutting-plane generator is different from the multi-cuts techniques used to accelerate cutting-plane methods in operation research (see e.g. [DL05] in ”column generation”, [MW81] for the Benders decomposition of mixed-integer programming, and [RS03] in stochastic programming). Contrary to our cutting-plane generator, these techniques usually add several ”controlled” cuts. In this context, our approach can be seen as an uncontrolled multi-cut technique.

In the two situations mentioned in this section (Lagrangian relaxations of combinatorial optimization problems, and decompositions of stochastic optimization problems), obtaining uncontrolled bundle information and calling the cutting-plane generator are often of neglectable computational cost compared to the cost of calling the (controlled) oracle. A wise practitioner can therefore be tempted to use the uncontrolled bundle information inside of his bundle method. The goal of this paper is to serve as an incentive to follow this meaningful practical intuition, as it establishes that incorporating uncontrolled bundle information can help in practice and is also consistent in theory.

1.3 Using uncontrolled linearizations in bundle methods

Assume that we are at iteration k of a bundle method solving (1), and that we have a family of linearizations

$$\bar{f}_i(\cdot) := f_{x_i} + g_{x_i}^\top(\cdot - x_i) \quad \big(\, \leq f(\cdot)\,\big) \tag{8}$$

associated to points $\{x_i\} \subset X$. In this paper, we consider that some of these linearizations (indexed by J_k^η) were given by the oracle, (so they are inexact up to the oracle error $f(x_i) - \eta \leq f_{x_i} \leq f(x_i)$ for all $i \in J_k^\eta$), and that the others (indexed by J_k^{u}) were created by the cutting-plane generator (so we do not known and do not control their inexactness). Bundle methods use available linearizations to create the so-called cutting-plane model of f, which is

$$\check{f}_k(\cdot) := \max_{i \in J_k^\eta \cup J_k^{\mathrm{u}}} \bar{f}_i(\cdot) \quad (\, \leq f(\cdot)\,). \tag{9}$$

This model is used to compute the next iterate x_{k+1} by solving a convex quadratic programming problem. In proximal bundle algorithms (see e.g. [HUL93]), x_{k+1} is the proximal point of $\check{f}_k$ given a ”prox-parameter” $t_k > 0$ and the “stability center” $\hat{x}_k$;

the quadratic optimization problem is the following:

$$\min_{x\in X} \check{f}_k(x) + \frac{1}{2t_k}\|x-\hat{x}_k\|^2 \quad \Longleftrightarrow \quad \begin{cases} \min_{x,r} \; r + \frac{1}{2t_k}\|x-\hat{x}_k\|^2 \\ \text{s.t.} \quad \bar{f}_i(x) \le r, \quad \forall\, i \in J_k^{\eta} \cup J_k^{\mathrm{u}} \\ \qquad\; x \in X,\; r \in \mathbb{R}. \end{cases} \tag{10}$$

In level bundle algorithms (see e.g. [LNN95]), x_{k+1} is the projection of the current stability center $\hat{x}_k$ onto the level set of "level parameter" f_k^{lev}

$$\mathbb{X}_k := \left\{x \in X : \check{f}_k(x) \le f_k^{\mathrm{lev}}\right\} = \left\{x \in X : \bar{f}_i(x) \le f_k^{\mathrm{lev}} \text{ for all } i \in J_k^{\eta} \cup J_k^{\mathrm{u}}\right\}; \tag{11}$$

the quadratic optimization problem is the following:

$$\min_{x\in \mathbb{X}_k} \frac{1}{2}\|x-\hat{x}_k\|^2 \quad \Longleftrightarrow \quad \begin{cases} \min_{x} \; \frac{1}{2}\|x-\hat{x}_k\|^2 \\ \text{s.t.} \quad \bar{f}_i(x) \le f_k^{\mathrm{lev}} \quad \forall\, i \in J_k^{\eta} \cup J_k^{\mathrm{u}} \\ \qquad\; x \in X. \end{cases} \tag{12}$$

In both cases, it is clear that using more information gives a more precise model, so would possibly lead to computation of better iterates since the model (9) using all the information (max on both J_k^{η} and J_k^{u}) is obviously always above the model of f that would restrict the max to J_k^{η} only. Admittedly, in practice using the complete model (9) rather than ignoring uncontrolled information makes quadratic programming problems (10) and (12) larger and then more difficult to solve. This is partly compensated by the ever-growing performance of (specific or even general-purpose) linear-quadratic programming solvers. Anyway, this drawback does not really hold in the case of expensive oracles – which is the situation we consider in this paper. Thus, there is a clear practical interest to consider as much information as possible when solving (1) with bundle methods: richer information can accelerate numerical methods at a neglectable cost, so that the overall computing time is lower than using only the controlled information. This will be illustrated in section 4.

There is nevertheless a theoretical argument against using the uncontrolled information in the model. Up to our understanding, the convergence results of bundle methods do not extend in a straightforward way for handling general models (9). Standard proofs of convergence use indeed that iterates are computed using a cutting-plane model with "controlled" linearizations, produced by an oracle with bounded or vanishing accuracy ($\eta \to 0$); see e.g. [Fáb00], [Kiw06], and [dOS14]. In our situation, the call of the cutting-plane generator makes us lose control on the construction of the cutting-plane model and therefore on the next iterate.

The only analysis which is generic enough to cover uncontrolled linearizations is the recent article [dOSL14] on the convergence of various forms of proximal bundle methods. So we start with considering, in section 2, a proximal bundle algorithm using the cutting-plane generator which is a trivial extension of the standard inexact proximal method of [Kiw06] and whose analysis is an instantiation of the generic analysis of [dOSL14]. Numerical experiments show that this proximal algorithm does accelerate the convergence with the help of uncontrolled linearizations added by a cheap cutting-plane generator. However this algorithm may not fully capture the uncontrolled information: the prox-parameter t_k in (10) ties the next iterate to the stability center $\hat{x}_k$, so that the step can be small even when the model is reasonably rich due to the added uncontrolled linearizations. This feature is inherent to proximal algorithms.

In contrast, level bundle method would better benefit from additional uncontrolled inexact information: richer cutting-plane models would tend to generate useful lower bounds, and then the level set $\mathbb{X}_k$ would better approximate the solution set, and the next iterate would better approximate a solution. However, to our knowledge, there is no level bundle method which we could build on to introduce uncontrolled information: the level counterpart of [Kiw06] able to deal with bounded accuracy η-oracle in the general case has not been developed yet, because of the difficulty of setting up a noise attenuation without direct control on the step (operated by t_k in proximal methods). In particular, our situation does not fit in the recent analysis of [dOS14] that features oracles with varying accuracy but controlling the error (and driving it to zero), nor in [vAdO14] that assumes the oracle to have uniformly bounded errors on a compact feasible set X.

1.4 Contributions, structure, and notation

This paper presents two inexact bundle algorithms (proximal and level) incorporating (already available or cheap to compute) uncontrolled bundle information. We formalize the additional information as produced by an external module (the cutting-plane generator of (7)) producing inexact linearizations without known or bounded accuracy. In section 2, we consider a proximal bundle algorithm using this cutting-plane generator, which is simple (in the sense that is a trivial extension of the standard inexact proximal method of [Kiw06]) and has a simple analysis (in the sense that it is an instantiation of the generic analysis of proximal methods of [dOSL14]). We introduce in section 3 a new inexact level bundle algorithm using the cutting-plane generator which is an extension of the limited-memory proximal level algorithm [BKL95] with an implicit noise attenuation step. This is the first level algorithm able to handle inexact oracles without assuming compactness of the feasible set X or a vanishing error; this is the main technical contribution of this paper. Finally, in section 4, we present and discuss computational illustrations on stochastic optimization problems coming from energy optimization: two-stage linear problems (arising in the planning of hydro-electric power generation, see [ZPR00] and [dOSP$^+$10]) and joint chance-constrained optimization problems (arising in cascaded reservoir management, see e.g. [vAdO14]). For these problems, we show that the methods save computational time in using both controlled and uncontrolled information.

Before moving to these developments, we finish this introduction by recalling some notation and terminology of bundle algorithms.

– *Aggregate linearizations.* We will see that the optimality conditions of the quadratic problems (10) and (12) introduce the "aggregated subgradients" $\hat{g}_k \in \partial \check{f}_k(x_{k+1}) + N_X(x_{k+1})$, which will have a role in the stopping tests. They define in turn the so-called "aggregate linearizations", denoted with the convenient notation "$-k$" borrowed from [dOSL14],

$$\bar{f}_{-k}(\cdot) := \check{f}_k(x_{k+1}) + \hat{g}_k^\top(\cdot - x_{k+1}) \,. \tag{13}$$

It can be proved (see e.g. [dOS14, Prop. 3.2]) that $\bar{f}_{-k}$ is indeed a linearization

$$\bar{f}_{-k}(x) \le \check{f}_k(x) \le f(x) \qquad \text{for all } x \in X \,. \tag{14}$$

We also define the "aggregate linearization error" by

$$\hat{e}_k := f_{\hat{x}_k} - \bar{f}_{-k}(\hat{x}_k). \tag{15}$$

– *Bundle compression.* The linearizations used in $\check{f}_k$ are possibly numerous and imprecise. It is interesting to be able to work with a limited memory and to somehow extract the useful part from all linearizations. In bundle algorithm terminology, this is called "bundle compression", which is a desirable property in general [HUL93], and thus even more in our context where it would make sense to compress uncontrolled bundle information. In theory we can compress a lot in the algorithms presented in this paper: as usual for bundle methods, the current controlled linearization and the aggregate linearization are sufficient to guarantee convergence.
– *Descent test.* The two bundle methods presented in this paper have a descent step which is the technical point bringing convergence without compactness of X. The stability center $\hat{x}_k$ is updated if the observed decrease is at least a fraction of the predicted decrease

$$v_k = f_{\hat{x}_k} - \check{f}_k(x_{k+1}). \tag{16}$$

More specifically, we use the following descent test

$$f_{x_{k+1}} \le f_{\hat{x}_k} - \kappa_f v_k \,, \quad \text{with} \quad \kappa_f \in (0,1) \,. \tag{17}$$

2 Proximal bundle method using uncontrolled information

This section explains how the usual inexact bundle method extends easily to deal with uncontrolled bundle information: Algorithm 1 below is a version of the inexact proximal algorithm of [Kiw06] using the cutting-plane generator to incorporate uncontrolled bundle information.

At iteration k of this algorithm, optimality conditions of the quadratic proximal subproblem (10) can be written with the help of the (simplicial) Lagrange multipliers α_i associated to the constraints $\bar{f}_i(x) \le r$, as

$$-x + \hat{x}_k - t_k \sum_{i \in J_k^\eta \cup J_k^{\mathrm{u}}} \alpha_i g_i \;\in\; N_X(x).$$

The unique solution x_{k+1} can thus be written as a "subgradient step" along $\hat{g}_k \in \partial \check{f}_k(x_{k+1}) + N_X(x_{k+1})$ with specified stepsize t_k

$$x_{k+1} = \hat{x}_k - t_k \hat{g}_k \,. \tag{18}$$

Combined with (13), this yields that the aggregate linearization error $\hat{e}_k$ defined in (15) and the predicted decrease v_k of (16) are connected by $v_k = e_k + t_k\|\hat{g}_k\|^2$.

Excessive inexactness is handled in a standard way: we employ the "noise attenuation" rule proposed by [Kiw06], consisting in increasing sharply t_k whenever $\hat{e}_k$ is overly negative. More precisely, if $\hat{e}_k < -\kappa_{\text{att}} t_k \|\hat{g}_k\|^2$, we set $t_k = 10t_k$ and solve again (10) to obtain another iterate. Otherwise (i.e., $\hat{e}_k \geq -\kappa_{\text{att}} t_k \|\hat{g}_k\|^2$), the algorithm performs like a classical proximal bundle method. We implement this using an extra binary variable `na` indicating noise attenuation. After noise attenuation, t_k does not decrease until a new descent step is performed (see line 20). Though it deals with the coarse information ($J_k^{\mathrm{u}} \neq \emptyset$), the convergence of the algorithm still fits into the generic bundle scheme analysis of [dOSL14].

Algorithm 1 Usual inexact proximal bundle method using cutting plane generator

1: Choose $x_1 \in X$, and set $\hat{x}_1 \leftarrow x_1$
2: Choose stopping tolerances, $\text{tol}_e \geq 0$ and $\text{tol}_g \geq 0$
3: Select $\kappa_f, \kappa_{\text{att}} \in (0,1)$ and $t_1 \geq \bar{t} > 0$
4: $(f_{x_1}, g_{x_1}) \leftarrow \eta$-oracle$(x_1)$, set $\hat{g}_1 \leftarrow g_{x_1}$ and $\hat{e}_1 \leftarrow 0$
5: $J_1^{\eta} \leftarrow \{1\}$, $J_0^{\mathrm{u}} \leftarrow \emptyset$ and `na` $\leftarrow 0$
6: **for** $k = 1, 2, \ldots$ **do**
7: $\quad J_k^{\mathrm{u}} \leftarrow$cutting-plane-generator$(\hat{x}_k, J_k^{\eta}, J_{k-1}^{\mathrm{u}})$ ▷ introduction of uncontrolled linearizations
8: $\quad$ Solve (10) to get x_{k+1} and compute $\hat{g}_k$
9: $\quad$ Set $\hat{e}_k \leftarrow v_k - t_k\|\hat{g}_k\|^2$
10: $\quad$ **if** $\hat{e}_k + \hat{g}_k^{\top}\hat{x}_k \leq \text{tol}_e$ and $\|\hat{g}_k\| \leq \text{tol}_g$ **then** ▷ stopping test
11: $\quad\quad$ **return** $\hat{x}_k$ and $f_{\hat{x}_k}$
12: $\quad$ **end if**
13: $\quad$ **if** $\hat{e}_k < -\kappa_{\text{att}} t_k \|\hat{g}_k\|^2$ **then** ▷ (noise) attenuation
14: $\quad\quad$ `na` $\leftarrow 1$, $t_k \leftarrow 10t_k$, and go back to line 8
15: $\quad$ **end if**
16: $\quad (f_{x_{k+1}}, g_{x_{k+1}}) \leftarrow \eta$-oracle$(x_{k+1})$ ▷ call η-oracle
17: $\quad$ **if** $f_{x_{k+1}} \leq f_{\hat{x}_k} - \kappa_f v_k$ **then**
18: $\quad\quad \hat{x}_{k+1} \leftarrow x_{k+1}$, `na` $\leftarrow 0$ and choose $t_{k+1} \geq \bar{t}$ ▷ descent step
19: $\quad$ **else**
20: $\quad\quad \hat{x}_{k+1} \leftarrow \hat{x}_k$ and update t_k: $\begin{cases} t_{k+1} \in [\bar{t}, t_k] & \text{if } \texttt{na} = 0 \\ t_{k+1} = t_k & \text{if } \texttt{na} = 1 \end{cases}$
21: $\quad$ **end if**
22: $\quad$ Choose $J_{k+1}^{\eta} \supset \{k+1, -k\}$ ▷ bundle compression
23: **end for**

Theorem 1 (Convergence of inexact proximal bundle) *Set the tolerances to zero in Algorithm 1. Then the sequences testing optimality $\{\hat{e}_k + \hat{g}_k^{\top}\hat{x}_k\}$ and $\{\hat{g}_k\}$ become "nonpositive", in the sense that there exists a subsequence (indexed by $\mathscr{I}$) such that:*

$$\limsup_{k \in \mathscr{I}} \hat{e}_k + \hat{g}_k^{\top}\hat{x}_k \leq 0 \;\; \textit{and} \;\; \lim_{k \in \mathscr{I}} \|\hat{g}_k\| = 0\,.$$

Furthermore, the iterates $\{\hat{x}_k\}$ generate an η-minimizing sequence, i.e. (22) *holds. Thus Algorithm 1 terminates after finitely many steps with an approximate solution if the tolerances* tol_g, *and* tol_e *are strictly positive.*

Proof The algorithm fits into the algorithmic pattern 4.2 of [dOSL14], and roughly speaking the convergence comes from the use of the η-oracle at descent steps. More specifically, we apply the generic convergence result of Theorems 6.11 and 4.4 of [dOSL14]; let us check their assumptions one by one:

– The oracle error is uniformly bounded by η (for iterates x_j with $j \in J_k^\eta$), and thus satisfies (6.8) of [dOSL14].
– The cutting-plane model (9) satisfies $\check{f}_k \leq f$, i.e., equation (4.10) in [dOSL14].
– We have (3.8) of [dOSL14] by setting $\ell_k = f_{\hat{x}_k}$.
– The prox-parameter updating rule is of the type (6.14) of [dOSL14].
– Equation (6.11) in [dOSL14] holds trivially for $f_{\hat{x}_k} - f_{x_{k+1}}$ as effective decrease.
– We have (6.16) of [dOSL14] (specifically, with $\alpha_k = 0$ and $\beta_k = \kappa_{\rm att}$ in there).

Thus Algorithm 1 satisfies all the assumptions (6.15) and (6.16) of [dOSL14, Theorem 6.11]. This opens the way to apply [dOSL14, Theorem 4.4], which in turn states that having a subsequence $\mathscr{I}$ such that $\limsup_{k\in\mathscr{I}}(\hat{e}_k + \hat{g}_k^\top \hat{x}_k) \leq 0$ and $\lim_{k\in\mathscr{I}} \hat{g}_k = 0$ gives the convergence up to η, which is the desired conclusion. □

We report numerical illustrations of this algorithm in section 3.2. They show that using uncontrolled linearizations within this algorithm leads to less iterations and lower CPU time than using only controlled linearizations. However we see on (18) that the algorithm may not exploit completely the added uncontrolled linearizations: x_{k+1} is tied to $\hat{x}_k$ by the explicit prox-parameter t_k, which could prevent the algorithm from making big steps in case of rich cutting-plane model. Such behavior would not appear with level bundle method.

3 Level method using uncontrolled information

This section presents a level bundle algorithm dealing with an η-oracle and a cutting-plane generator introducing uncontrolled linearizations, as in (7). When disregarding the cutting-plane generator, this algorithm turns out to be the first level method able to deal with inexact η-oracles in general; in this way, it can be seen as the level counterpart of the proximal bundle method of [Kiw06]. The algorithm is presented in section 3.1, its convergence is stated in section 3.2 and analyzed in section 3.3. Its numerical behaviour is illustrated in section 4.

3.1 An inexact proximal-descent level bundle method

To avoid any compactness assumption, we consider a proximal-descent version of level bundle method, inspired from the one of [BKL95]. At iteration k of this algorithm, optimality conditions of the projection problem (12) can be written, with the help of the Lagrange multipliers $\alpha_i \geq 0$ associated to the constraints $\bar{f}_i(x) \leq f_k^{\rm lev}$, as

$$-x + \hat{x}_k - \sum_{i\in J_k^\eta \cup J_k^{\rm u}} \alpha_i\, g_i \;\in\; N_X(x).$$

Introducing the "stepsize"

$$\mu_k := \sum_{i\in J_k^\eta \cup J_k^{\mathrm{u}}} \alpha_i,$$

we observe that x_{k+1}, the unique solution of the above optimality conditions, can be written as the "subgradient step" along a direction $\hat{g}_k \in \partial \check{f}_k(x_{k+1}) + N_X(x_{k+1})$

$$x_{k+1} = \hat{x}_k - \mu_k \hat{g}_k \qquad \text{such that} \quad \mu_k(\check{f}_k(x_{k+1}) - f_k^{\mathrm{lev}}) = 0. \tag{19}$$

The (inexact) upper bound is given by the η-oracle at the stability center ($f_k^{\mathrm{up}} = f_{\hat{x}_k}$). When $\mu_k > 0$, the predicted decrease (16) then corresponds to the level depth

$$v_k = f_k^{\mathrm{up}} - f_k^{\mathrm{lev}},$$

and the aggregate linearization error is related to it, as

$$\hat{e}_k = v_k - \mu_k \|\hat{g}_k\|^2. \tag{20}$$

To see this, notice from (19) that $\mu_k > 0$ ensures that $\check{f}_k(x_{k+1}) = f_k^{\mathrm{lev}}$ and, therefore

$$\hat{e}_k = f_{\hat{x}_k} - (\check{f}_k(x_{k+1}) + \hat{g}_k^\top(\hat{x}_k - x_{k+1})) = f_{\hat{x}_k} - f_k^{\mathrm{lev}} - \mu_k\|\hat{g}_k\|^2 = v_k - \mu_k\|\hat{g}_k\|^2.$$

We emphasize that we do not control the stepsize μ_k in level bundle methods, in contrast with proximal bundle methods where we can choose the prox-parameter t_k giving the stepsize. This poses a technical difficulty for handling excessive inexactness within level methods. In Algorithm 1, as in other inexact proximal bundle methods, t_k is increased when the noise is excessively large compared to $\hat{g}_k$ (see line 13 in Algorithm 1); this can not be done directly in an inexact level method. So we propose in Algorithm 2 an *implicit* noise attenuation rule, combined with the level attenuation rule. The idea is simple: we do not allow the depth v_k to decrease if the noise is excessive (see line 21). We will prove in the key proposition 2 that this simple idea makes μ_k to go to infinity in presence of noise, such that either a new descent step is generated, or the algorithm terminates.

In practice, the projection onto $\mathbb{X}_k$ (problem (12)) is solved by a quadratic programming solver (at line 18 of Algorithm 2). If $\mathbb{X}_k$ is nonempty, the solver provides x_{k+1} and μ_k, from which we deduce $\hat{g}_k$ by (19). If $\mathbb{X}_k$ is empty, the solver raises a flag of infeasibility and we exploit this information by updating the lower bound for the optimal value f_*: observe indeed that when $\mathbb{X}_k$ is empty, there holds

$$f_k^{\mathrm{lev}} < \check{f}_k(x) \le f(x) \quad \text{for all } x \in X,$$

so that we can set $f_k^{\mathrm{low}} = f_k^{\mathrm{lev}}$ (see line 15). At each iteration of Algorithm 2, we thus have a lower bound f_k^{low} and an inexact upper bound f_k^{up} such that

$$f_k^{\mathrm{low}} \le f_* \le f_k^{\mathrm{up}} + \eta. \tag{21}$$

Algorithm 2 New inexact proximal level method using cutting-plane generator

1: Choose $x_1 \in X$, $\nu_1 > 0$, and set $\hat{x}_1 \leftarrow x_1$
2: Choose stopping tolerances $\mathrm{tol}_\Delta \geq 0$, $\mathrm{tol}_e \geq 0$ and $\mathrm{tol}_g \geq 0$
3: Select $\kappa_l, \kappa_f, \kappa_{\mathrm{att}} \in (0,1)$
4: Choose a threshold $\mu_{\mathrm{large}} > 0$
5: $(f_{x_1}, g_{x_1}) \leftarrow \eta$-oracle$(x_1)$, set $\hat{g}_1 \leftarrow g_{x_1}$ and $\hat{e}_1 \leftarrow 0$
6: Set $f_1^{\mathrm{up}} \leftarrow f_{x_1}$, $f_1^{\mathrm{low}} \leftarrow -\infty$, $\Delta_1 \leftarrow +\infty$, $J_1^\eta \leftarrow \{1\}$, $J_0^{\mathrm{u}} \leftarrow \emptyset$
7: **for** $k = 1, 2, \ldots$ **do**
8: $\quad J_k^{\mathrm{u}} \leftarrow$cutting-plane-generator$(\hat{x}_k, J_k^\eta, J_{k-1}^{\mathrm{u}})$ ▷ introduction of uncontrolled linearizations
9: $\quad$ Update $f_k^{\mathrm{lev}} \leftarrow f_k^{\mathrm{up}} - \nu_k$ and $\mathbb{X}_k \leftarrow \left\{x \in X : \check{f}_k(x) \leq f_k^{\mathrm{lev}}\right\}$
10: $\quad$ **if** $\Delta_k \leq \mathrm{tol}_\Delta$ or ($\hat{e}_k \leq \mathrm{tol}_e$ and $\|\hat{g}_k\| \leq \mathrm{tol}_g$) **then** ▷ stopping test
11: $\qquad$ **return** $\hat{x}_k$ and $f_{\hat{x}_k} = f_k^{\mathrm{up}}$
12: $\quad$ **end if**
13: $\quad$ Run a quadratic optimization software on problem (12)
14: $\quad$ **if** $\mathbb{X}_k$ is empty **then**
15: $\qquad f_k^{\mathrm{low}} \leftarrow f_k^{\mathrm{lev}}$, $\Delta_k \leftarrow f_k^{\mathrm{up}} - f_k^{\mathrm{low}}$, $\nu_k \leftarrow \min\{\nu_k, \kappa_l \Delta_k\}$ ▷ lower bound
16: $\qquad$ Go back to line 9
17: $\quad$ **else**
18: $\qquad$ Get x_{k+1} and μ_k, and compute $\hat{g}_k$ using (19)
19: $\qquad \hat{e}_k \leftarrow \nu_k - \mu_k \|\hat{g}_k\|^2$
20: $\quad$ **end if**
21: $\quad$ **if** $\mu_k > \mu_{\mathrm{large}}$ and $\hat{e}_k \geq -\kappa_{\mathrm{att}} \mu_k \|\hat{g}_k\|^2$ **then** ▷ (level+noise) attenuation
22: $\qquad \nu_k \leftarrow \frac{\nu_k}{2}$, and go back to line 9
23: $\quad$ **end if**
24: $\quad (f_{x_{k+1}}, g_{x_{k+1}}) \leftarrow \eta$-oracle$(x_{k+1})$ ▷ call η-oracle
25: $\quad$ **if** $f_{x_{k+1}} \leq f_{\hat{x}_k} - \kappa_f \nu_k$ **then**
26: $\qquad \hat{x}_{k+1} \leftarrow x_{k+1}$, $f_{k+1}^{\mathrm{up}} \leftarrow f_{x_{k+1}}$ and $f_{k+1}^{\mathrm{low}} \leftarrow f_k^{\mathrm{low}}$ ▷ descent step
27: $\qquad \Delta_{k+1} \leftarrow f_{k+1}^{\mathrm{up}} - f_{k+1}^{\mathrm{low}}$ and $\nu_{k+1} \leftarrow \min\{\nu_k, \kappa_l \Delta_{k+1}\}$
28: $\quad$ **else**
29: $\qquad \hat{x}_{k+1} \leftarrow \hat{x}_k$, $\Delta_{k+1} \leftarrow \Delta_k$, $\nu_{k+1} \leftarrow \nu_k$, $f_{k+1}^{\mathrm{up}} \leftarrow f_k^{\mathrm{up}}$ and $f_{k+1}^{\mathrm{low}} \leftarrow f_k^{\mathrm{low}}$
30: $\quad$ **end if**
31: $\quad$ Choose $J_{k+1}^\eta \supset \{k+1, -k\}$ ▷ bundle compression
32: **end for**

3.2 Convergence result

We have the following theorem stating the convergence of Algorithm 2, which is of the same vein as Theorem 1 for Algorithm 1.

Theorem 2 (Convergence of inexact proximal level) *Set the tolerances to zero in Algorithm 2. Then the sequences testing optimality $\{\Delta_k = f_k^{\mathrm{up}} - f_k^{\mathrm{low}}\}$, $\{\hat{e}_k\}$ and $\{\hat{g}_k\}$ become "nonpositive", in the sense that*

- *either the sequence $\{\Delta_k\}$ tends to be nonpositive:* $\lim \Delta_k \leq 0$,
- *or there exists a subsequence (indexed by $\mathscr{I}$) such that:* $\liminf_{k \in \mathscr{I}} \hat{e}_k \leq 0$ *and* $\lim_{k \in \mathscr{I}} \|\hat{g}_k\| = 0$.

Furthermore, the iterates $\{\hat{x}_k\}$ generate an η-minimizing sequence, i.e.

$$f_* \leq \liminf f(\hat{x}_k) \leq f_* + \eta. \tag{22}$$

Thus Algorithm 2 terminates after finitely many steps with an approximate solution if the tolerances tol_Δ, tol_g, *and* tol_e *are strictly positive.*

The next section is devoted to the proof of this theorem. We will say that the algorithm converges up to η when (22) holds. Note that there are two ways to stop the algorithm (see line 10): the usual criterion based on the gap $\Delta_k = f_k^{\text{up}} - f_k^{\text{low}}$

$$\lim \Delta_k \leq 0 \qquad \Longrightarrow \text{ convergence up to } \eta \tag{23}$$

and a second one inspired from [BKL95] based on the aggregated error and subgradients to deal with unbounded feasible sets. The next two lemmas explain these two stopping tests and their consistency.

Lemma 1 (Nonpositivity of Δ_k and convergence) *If* $\lim \Delta_k \leq 0$, *then the sequence* $\{\hat{x}_k\}$ *satisfies*

$$f_* - \eta \leq \lim f_{\hat{x}_k} \leq f_* \leq \liminf f(\hat{x}_k) \leq f_* + \eta \,. \tag{24}$$

Furthermore, if at some iteration k we have $\Delta_k \leq 0$, *then we have in fact*

$$f_* - \eta \leq f_{\hat{x}_k} \leq f_* \leq f(\hat{x}_k) \leq f_* + \eta \,. \tag{25}$$

Proof Note first that the η-oracle properties imply that, for all k,

$$f_* - \eta \leq f(\hat{x}_k) - \eta \leq f_{\hat{x}_k} \,, \tag{26}$$

so that $f_k^{\text{up}} = f_{\hat{x}_k}$ satisfies (21). We see that $\{f_k^{\text{up}} = f_{\hat{x}_k}\}$ is nonincreasing (line 26), $\{f_k^{\text{low}}\}$ is nondecreasing (line 15), and so $\{\Delta_k\}$ is nonincreasing (line 27). The nonincreasing sequence $\{f_{\hat{x}_k}\}$ is bounded from below thus converges and $\lim f_{\hat{x}_k} \geq f_* - \eta$. Similarly the nondecreasing $\{f_k^{\text{low}}\}$ is bounded from above by f_*, thus it also converges and $\lim f_k^{\text{low}} \leq f_*$. Writing $\lim \Delta_k \leq 0$ as $\lim f_{\hat{x}_k} - \lim f_k^{\text{low}} \leq 0$ we obtain

$$f_* - \eta \leq \lim f_{\hat{x}_k} \leq f_*. \tag{27}$$

Now passing to the limit-inf in (26) and adding η, we also have

$$f_* \leq \liminf f(\hat{x}_k) \leq \lim f_{\hat{x}_k} + \eta \leq f_* + \eta.$$

Combining this inequalities with (27) gives the announced inequalities (24).

The argument leading to the second inequality (25) is the same as above. For a fixed k, (26) and $\Delta_k \leq 0$ give $f_* - \eta \leq f_{\hat{x}_k} \leq f_*$, and adding η to (26) yields

$$f_* \leq f(\hat{x}_k) \leq f_{\hat{x}_k} + \eta \leq f_* + \eta \,.$$

Combining the inequalities gives (25). □

Lemma 2 (Vanishing aggregate errors and convergence) *For the sequences* $\{\hat{x}_k\}$, $\{\hat{e}_k\}$ *and* $\{\hat{g}_k\}$ *generated by Algorithm 2, we have, for all* $x \in X$,

$$f(\hat{x}_k) \leq f(x) + \hat{e}_k + \eta - \hat{g}_k^\top (x - \hat{x}_k). \tag{28}$$

Assume that $\{\hat{x}_k\}$ *is bounded and there exists a subsequence indexed by* $\mathscr{I}$ *such that*

$$\liminf_{k \in \mathscr{I}} \hat{e}_k \leq 0 \quad \textit{and} \quad \lim_{k \in \mathscr{I}} \|\hat{g}_k\| = 0. \tag{29}$$

Then the algorithm converges up to η.

Proof Fix $x \in X$. The inequality (28) comes from (13) as follows:

$$\begin{aligned} f(x) &\geq \bar{f}_{-k}(x) \\ &= \check{f}_k(x_{k+1}) + \hat{g}_k^\top (x - x_{k+1}) \\ &= \check{f}_k(x_{k+1}) + \hat{g}_k^\top (\hat{x}_k - x_{k+1}) + \hat{g}_k^\top (x - \hat{x}_k) \\ &= \bar{f}_{-k}(\hat{x}_k) + \hat{g}_k^\top (x - \hat{x}_k) \\ &= f_{\hat{x}_k} - (f_{\hat{x}_k} - \bar{f}_{-k}(\hat{x}_k)) + \hat{g}_k^\top (x - \hat{x}_k) \\ &= f_{\hat{x}_k} - \hat{e}_k + \hat{g}_k^\top (x - \hat{x}_k) \\ &\geq f(\hat{x}_k) - \eta - \hat{e}_k + \hat{g}_k^\top (x - \hat{x}_k). \end{aligned}$$

We also get the upper bound

$$f_* \leq f(\hat{x}_k) \leq f(x) + \hat{e}_k + \eta + \|\hat{g}_k\| \|x - \hat{x}_k\|.$$

Passing to the lim inf, (29) together with the boundedness of $\{\hat{x}_k\}$ yields

$$f_* \leq \liminf_{k \in \mathscr{I}} f(\hat{x}_k) \leq f(x) + \eta.$$

Taking the infimum over $x \in X$ gives (22). □

3.3 Convergence proof

To prove Theorem 2, we adapt the usual rationale of convergence proof of bundle methods, by considering the two cases of infinitely many and finitely many descent steps (line 26). We show that in both cases one of the two stopping tests is active, which guarantees in turn that the algorithm converges up η (by (23) and Lemma 2). The technical challenge is to handle, first, a fixed inexactness in a level method and, second, the uncontrolled cutting-plane model. We note that this proof of convergence differs from the one of [BKL95].

We start with a remark about the level depth v_k. Looking at lines 15, 22 and 27, we see that $\{v_k\}$ is nonincreasing, and that if $v_k \geq 0$ then $\hat{e}_k \geq -\mu_k \|\hat{g}_k\|^2$. We also notice that v_k can be negative only if so is Δ_k, and then (23) holds. Therefore, we consider that $v_k \geq 0$ in the remainder of the section.

We will also need the index set $\mathscr{A}$ of the iterations requiring a noise attenuation (line 22). The following lemma studies the situation of infinitely many of such attenuations. The following proposition treats the first case of infinitely many descent steps.

Lemma 3 (Infinitely many attenuations) *If $\mathscr{A}$ contains infinitely many indices, then* (29) *holds with $\mathscr{I} = \mathscr{A}$. If the sequence $\{\hat{x}_k\}$ is furthermore bounded, then the algorithm converges up to η.*

Proof Recall that $v_k = \hat{e}_k + \mu_k \|\hat{g}_k\|^2$ by (20). If $k \in \mathscr{A}$, then we have

$$v_k = \hat{e}_k + \mu_k \|\hat{g}_k\|^2 \geq (1 - \kappa_{\text{att}}) \mu_k \|\hat{g}_k\|^2 \geq (1 - \kappa_{\text{att}}) \mu_{\text{large}} \|\hat{g}_k\|^2 \geq 0.$$

If the set $\mathscr{A}$ is infinite, then we have $v_k \to 0$, and therefore $\|\hat{g}_k\| \to 0$ by the above inequality. By (20), this yields that $\hat{e}_k \to 0$ and then we have (29) with $\mathscr{I} = \mathscr{A}$. As a result, if the sequence $\{\hat{x}_k\}$ is bounded, we can invoke Lemma 2 and get that $\{\hat{x}_k\}$ is η-minimizing. □

Proposition 1 (Infinitely many descent steps) *Suppose there are infinitely many descent steps (line 26). Then the algorithm converges up to η.*

Proof Let us index the descent steps by ℓ. More precisely $k(\ell)$ denotes the ℓ^{th} descent iteration, and $j(\ell) = k(\ell+1) - 1$ the last iteration before the $(\ell+1)^{\text{th}}$. Note that $\hat{x}_{k(\ell)}$ is the ℓ^{th} (different) stability center, and that $\hat{x}_{k(\ell)} = \hat{x}_{j(\ell)}$. The descent test (17) gives the inequality

$$f_{x_{k(\ell)}} - f_{x_{k(\ell+1)}} \geq \kappa_f v_{j(\ell)} \geq 0\,.$$

Summing over ℓ we get

$$f_{x_{k(0)}} - \lim_{\ell} f_{x_{k(\ell+1)}} \geq \kappa_f \sum_{\ell=0}^{\infty} v_{j(\ell)}\,.$$

Since $\lim_\ell f_{x_{k(\ell+1)}} \geq f_* - \eta > -\infty$, we get that the series converges and then

$$\lim_{\ell} v_{j(\ell)} = 0. \tag{30}$$

By monotonicity of v_k, we thus have $\lim_k v_k = 0$. Let us distinguish now three cases:

(i)$\mathscr{A}$ finite (ii)$\mathscr{A}$ infinite and $\{\hat{x}_k\}_k$ bounded (iii)$\mathscr{A}$ infinite and $\{\hat{x}_k\}_k$ unbounded

In the case (i), for k large enough, we have $v_k = \kappa_l \Delta_k$, and then $\lim_k \Delta_k = 0$. Thus, (23) holds and the proof is over. In the case (ii), we can use Lemma 3 which gives (29) and that $\{\hat{x}_k\}$ is η-minimizing. So let us focus on the case (iii), and let us prove by contradiction that $\{\hat{x}_k\}$ is still η-minimizing.

Suppose that there exists $\varepsilon > 0$ such that $f(\hat{x}_k) > f_* + \eta + \varepsilon$ for all k large enough. This yields that there exists $\tilde{x} \in X$ such that $f(\hat{x}_{k(\ell)}) \geq f(\tilde{x}) + \eta + \varepsilon/2$ for all large ℓ. Then (28) applied to $k = j(\ell)$ gives

$$\hat{g}_{j(\ell)}^{\top}(x - \hat{x}_{k(\ell)}) \leq f(x) + \eta - f(\hat{x}_{k(\ell)}) + \hat{e}_{j(\ell)} \quad \text{for all}\, x \in X\,,$$

which yields

$$\hat{g}_{j(\ell)}^{\top}(\tilde{x} - \hat{x}_{k(\ell)}) \leq \hat{e}_{j(\ell)} - \varepsilon/2\,.$$

Using this inequality and (20), we develop

$$\begin{aligned}
\|\hat{x}_{k(\ell+1)} - \tilde{x}\|^2 &= \|\hat{x}_{k(\ell)} - \mu_{j(\ell)}\hat{g}_{j(\ell)} - \tilde{x}\|^2 \\
&= \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + \|\mu_{j(\ell)}\hat{g}_{j(\ell)}\|^2 + 2\mu_{j(\ell)}\hat{g}_{j(\ell)}^{\top}(\tilde{x} - \hat{x}_{k(\ell)}) \\
&= \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + \mu_{j(\ell)}[\mu_{j(\ell)}\|\hat{g}_{j(\ell)}\|^2 + 2\hat{g}_{j(\ell)}^{\top}(\tilde{x} - \hat{x}_{k(\ell)})] \\
&\leq \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + \mu_{j(\ell)}[\mu_{j(\ell)}\|\hat{g}_{j(\ell)}\|^2 + 2\hat{e}_{j(\ell)} - \varepsilon] \\
&\leq \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + 2\mu_{j(\ell)}[v_{j(\ell)} - \varepsilon/2]\,.
\end{aligned}$$

As $\lim_\ell v_{j(\ell)} = 0$ by (30), we have for all ℓ large enough $v_{j(\ell)} \leq \varepsilon/2$ and then

$$\|\hat{x}_{k(\ell+1)} - \tilde{x}\|^2 \leq \|\hat{x}_{k(\ell)} - \tilde{x}\|^2$$

which contradicts the fact that $\{\hat{x}_k\}$ is unbounded. Hence, (22) must hold. □

We consider now the second case of finitely many descent steps. We start with a lemma stating that null iterates get further away from the last stability center.

Lemma 4 (After a last descent step) *If $\hat{x}_k = \hat{x}_{k-1} = \hat{x}$, $f_k^{\text{lev}} \leq f_{k-1}^{\text{lev}}$ and $v_k = v_{k-1}$, then we have*

$$\|x_{k+1} - \hat{x}\|^2 \geq \|x_k - \hat{x}\|^2 + \frac{(1-\kappa_f)^2}{\|g_{x_k}\|^2} v_k^2.$$

Proof The bundle management of line 31 incorporates two pieces in the model $\check{f}_k$: the k-th linearization $\bar{f}_k$ and the aggregate linearization $\bar{f}_{-k}$. Both bring some information, as follows. First, since $\bar{f}_{-(k-1)} \leq \check{f}_k$ and $f_k^{\text{lev}} \leq f_{k-1}^{\text{lev}}$, we have that the level set $\mathbb{X}_k$ is included in the "aggregate level set" $\mathbb{X}_{-(k-1)} := \{x \in X : \bar{f}_{-(k-1)}(x) \leq f_{k-1}^{\text{lev}}\}$, and therefore that $x_{k+1} \in \mathbb{X}_{-(k-1)}$. It can be proved (see e.g. [dOS14, Prop. 3.2]) that the aggregate level-set produces the same iterate that $\mathbb{X}_{k-1}$; in other words,

$$x_k = P_{\mathbb{X}_{k-1}}(\hat{x}) = P_{\mathbb{X}_{-(k-1)}}(\hat{x}) \quad \text{and} \quad (\hat{x} - x_k)^\top (x - x_k) \leq 0 \text{ for all } x \in \mathbb{X}_{-(k-1)}\,. \tag{31}$$

Thus, we have $(\hat{x} - x_k)^\top (x_{k+1} - x_k) \leq 0$ and developing $\|x_{k+1} - \hat{x}\|^2 = \|x_{k+1} - x_k + (x_k - \hat{x})\|^2$, the inequality gives

$$\|x_{k+1} - \hat{x}\|^2 \geq \|x_k - \hat{x}\|^2 + \|x_k - x_{k+1}\|^2. \tag{32}$$

Now since $\bar{f}_k \leq \check{f}_k$ and $x_{k+1} \in \mathbb{X}_k$, we have $f_{x_k} + g_{x_k}^\top (x_{k+1} - x_k) \leq f_k^{\text{lev}}$, which gives

$$f_{x_k} - f_k^{\text{lev}} \leq \|g_{x_k}\| \|x_{k+1} - x_k\|\,. \tag{33}$$

Iteration k is not a descent iteration: the converse of line 25 reads $f_{x_k} \geq f_{\hat{x}} - \kappa_f v_{k-1}$. Recalling that $f_k^{\text{lev}} = f_{\hat{x}} - v_k$ and $v_k = v_{k-1}$, this yields $f_{x_k} - f_k^{\text{lev}} \geq (1-\kappa_f) v_k$. Together with (33), this gives

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

which ends the proof with (32). □

Proposition 2 (Finitely many descent steps) *Suppose that Algorithm 2 generates only finitely many descent steps. Then the algorithm converges up to η.*

Proof Let us consider first two easy cases. If $\lim \Delta_k \leq 0$ then (23) holds, and the proof is over. If $\mathscr{A}$ has infinitely many indices, we can conclude with Lemma 3 together with the fact that the sequence $\{\hat{x}_k\}$ is constant for k large enough.

Let us focus on the case where there exists $\bar{\Delta} > 0$ such that $\Delta_k \geq \bar{\Delta}$ for all k, and there is eventually no noise attenuation ($\mathscr{A}$ has finitely many indices). For k large enough, the stability center is fixed (denoted $\hat{x}$) and the depth is also fixed (at $\bar{v} > 0$).

We claim that the sequence $\|x_{k+1} - \hat{x}\|$ is not bounded. For sake of a contradiction, suppose that it is bounded. Then the η-subgradients are bounded (by a constant Λ) by [HUL93, Prop. XI.4.1.2]. Apply Lemma 4; since the v_k and the f_k^{lev} are fixed, the sequence $\{\|x_{k+1} - \hat{x}\|\}_k$ increases by a constant factor $(1-\kappa_f)^2 \bar{v}^2 / \Lambda^2$ at each iteration. This contradicts the boundedness.

We claim now that $\mu_k \to \infty$. In view of a contradiction, suppose that $\{\mu_k\}$ is bounded: let $\bar{\mu} > 0$ be such that $\mu_k \le \bar{\mu}$ for all k large enough. Using (20) we have that

$$\mu_k v_k = \mu_k \hat{e}_k + \mu_k^2 \|\hat{g}_k\|^2 \ge -\mu_k \eta + \mu_k^2 \|\hat{g}_k\|^2 \ge -\bar{\mu}\eta + \|x_{k+1} - \hat{x}\|^2 .$$

As $\{v_k\}$ is nonincreasing, we have that $\bar{\mu} v_0 \ge \mu_k v_k \ge -\bar{\mu}\eta + \|x_{k+1} - \hat{x}\|^2$, contradicting that $\|x_{k+1} - \hat{x}\|^2 \to \infty$. Hence, $\mu_k \to \infty$.

Since there is eventually no noise attenuation, we have (see line 21)

$$\hat{e}_k < -\kappa_{\mathrm{att}} \mu_k \|\hat{g}_k\|^2 < 0 \qquad \text{for all } k \text{ large enough.}$$

By definition of $\hat{e}_k$ in (15), we have that $\hat{e}_k \ge f(\hat{x}_k) - \eta - \bar{f}_{-k}(\hat{x}_k) \ge -\eta$, from (3) and (14). This yields $\|\hat{g}_k\|^2 \le \eta/(\kappa_{\mathrm{att}}\mu_k)$. Since $\mu_k \to \infty$, we get that $\hat{g}_k \to 0$. Hence, (29) holds with $\mathscr{I}$ being all the large indices. Since the sequence $\{\hat{x}_k\}$ is finite (thus bounded), we can conclude with Lemma 2. □

Remark 1 (More sophisticated versions) We emphasize that the important point of the above proofs was to control the linearization error at descent steps. As a consequence, we could add a test in the algorithm to stop the oracle whenever we detect that the descent test will be false. This version of the algorithm would be proved to be convergent with the exact same proof.

We could also cover the case of ”upper oracles” in the terminology of [dOSL14]. The algorithm could indeed deal with controllable linearizations overestimating the function by no more than a constant $\eta^g > 0$. The same convergence proof would result in a convergence to an $(\eta + \eta^g)$-solution. □

4 Numerical illustration on energy optimization

We illustrate the efficiency of our approach on two classes of energy optimization problems: two-stage stochastic programming problems (with publicly available data sets) and chance-constrained optimization problems arising from cascaded reservoir management (with real-life data). Each following subsection treats one family of problems for which we consider an exact oracle and a cutting-plane generator incorporating uncontrolled linearizations. Our goal here is not to obtain the best computational results for these problems, but to show that using the uncontrolled bundle information can speed-up computations.

Specifically, we compare Algorithm 1 and Algorithm 2 using cutting-plane generators to their basic versions not using any additional uncontrolled linearizations ($J_k^{\mathrm{u}} = \emptyset$ for all k). We have implemented these algorithms in MATLAB (using the Gurobi solver for LP and QP problems); we name them as follows

- u-P: Algorithm 1, the proximal bundle using uncontrolled information,
- P: Algorithm 1 with $J_k^{\mathrm{u}} = \emptyset$, the standard proximal bundle algorithm,
- u-L: Algorithm 2, the level bundle using uncontrolled information,
- L: Algorithm 2 with $J_k^{\mathrm{u}} = \emptyset$, the (new) level algorithm.

Notice that the comparison between level and proximal bundle algorithms are (surprisingly) rare; an exception is [dS15]. In particular, in section 5.1.4 of [dS15], tests are reported with tuning parameters of proximal and level bundle methods. Here we set the parameters of the algorithms according to these tests: for both algorithms, we take $\kappa_f = 0.1$ and $\kappa_{\text{att}} = 0.99$; for Algorithm 1, we take $t_1 = 10$, $\bar{t} = 10^{-6}$, and the update rule of Section 5.1.2 of [dS15] (with $a = 2$) for t_k ; for Algorithm 2, we take $\kappa_l = 0.2$ and $\mu_{\text{large}} = 5$.

Since the controlled oracle is exact ($\eta = 0$), the four methods converge to the exact solution. The algorithms are compared by measuring the number of calls to the exact oracle and the total CPU time to reach the stopping test. We use the relative stopping tolerance

$$\text{tol}_e = \text{tol}_\Delta = 10^{-5}(1+f(\hat{x}_k)) \qquad \text{and} \qquad \text{tol}_g = 10^{-4}(1+f(\hat{x}_k)).$$

These experiments were performed on a computer with Intel(R) Core(TM), i3-3110M CPU 2.40, 4G (RAM), under Windows 8, 64Bits.

We also compare the speed and robustness of the algorithms globally on all the problems by using performance profiles [DM02]. For each algorithm, we plot the proportion of problems that it solved within a factor of the time required by the best algorithm. More precisely, if we denote by $t_A(p)$ the time spent by algorithm A to solve problem p and $t^*(p)$ the best time for solving problem p, then the proportion of problems solved by A within a factor τ is

$$\theta_A(\tau) = \frac{\text{number of problems } p \text{ such that } t_A(p) \leq \tau t^*(p)}{\text{total number of problems}}.$$

4.1 Two-stage stochastic linear optimization problems

Problem and instances description. Two-stage stochastic linear problems arise in the planning of hydro-electric power generation; see e.g. [ZPR00] and [dOSP⁺10] for applications to the New Zealand and Brazilian electricity system. The problem can be formulated as (1) with

$$X = \{x \in \mathbb{R}^n_+ : Ax = b\} \quad \text{and} \quad f(x) = c^\top x + \sum_{i=1}^{N} \pi_i f_i(x)$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m_1 \times n}$ and $b \in \mathbb{R}^{m_1}$ are such that the set X is bounded. Also,

$$f_i(x) := \min_{y \in \mathbb{R}^{n_2}_+} q^\top y \ \text{ s.t. } \ Tx + Wy = h_i \tag{34}$$

is the so-called recourse function associated with the i-th scenario $h_i \in \mathbb{R}^{m_2}$ (which has a probability $\pi_i > 0$). In these problems, the vectors h_i are the only uncertainty parameters and are normally distributed. The dual linear problem of (34) is

$$f_i(x) = \sup_{W^\top u \leq q} (h_i - Tx)^\top u. \tag{35}$$

We use the set of two-stage stochastic linear test-problems that have been used by several authors including [SDR09, Deá06, dOSS11]). The set is available online on the webpage of István Deák[1]. The data set consists in 7 families of problems of different sizes; we call them F1 to F7. A family of problems is given by the data (c, A, b, q, T, W) along with a generator of appropriate scenarios, which takes as an input the number of scenarios N, and returns (π_i, h_i) for $i = 1, \dots, N$. For each family, we have 7 problems corresponding to $N \in \{100, 200, 500, 800, 1000, 1200, 1500\}$.

Oracles and cutting-plane generator. Computing exact information on f requires solving the N linear optimization subproblems (34)-(35). Solving only a fraction of these subproblems still gives inexact information on f: the optimal solution $\bar{u}_i$ giving $f_i(x)$ can also be used to under-approximate other terms $f_j(x)$ (since the dual feasible sets are the same, we have $(d_j - T_i x)^\top \bar{u}_i \leq f_j(x)$). Thus we are in the situation (7) with

- an exact oracle providing the value $f(x)$ and a subgradient $g \in \partial f(x)$ ($\eta = 0$) by solving exactly the N subproblems (35);
- an uncontrolled oracle by solving 10% of the subproblems (35) and taking a feasible solution of the remaining subproblems. This oracle is about 90% times faster than the fine one, but we do not know its accuracy.
- a cutting-plane generator consisting in running several iterations of a bundle method using only the uncontrolled oracle (with the same stopping test and a maximum of 100 iterations).

Numerical results. Table 1 presents the performances of the four algorithms on the 49 test-problems. It reports the number of (exact) oracle calls and CPU time (in minutes) required to reach convergence. Each entry is the average over the seven instances of the family, except for the last line which is the grand total over the 49 instances. This table shows that adding uncontrolled linearization does speed up significantly the two algorithms: we observe 25% less oracle calls and 10% less CPU time between L and u-L, and 39.4% and 28.6% between P and u-P.

	# oracle calls				CPU time (min)			
N	L	P	u-L	u-P	L	P	u-L	u-P
F1	19	25	13	11	1.2	1.7	1.0	0.8
F2	25	39	19	25	2.9	4.6	2.5	3.2
F3	37	56	23	30	2.5	3.7	1.8	2.2
F4	39	62	31	35	3.3	5.1	2.9	3.0
F5	38	63	36	39	4.5	7.1	4.5	4.7
F6	57	81	37	51	4.8	6.2	4.2	5.4
F7	59	68	47	49	6.7	7.9	6.1	6.6
Total	1928	2768	1440	1678	3.0h	4.2h	2.7h	3.0h

Table 1 Comparison of the four algorithms with respect to the number of oracle calls and the global CPU time to get convergence. Each entry is the average over the seven instances of the family, except for the last line which is the grand total over the 49 instances.

The decrease of oracle calls is more important than the one of CPU time because of the additional time taken by the calls of the cutting-plane generator and by solving

[1] `http://www.uni-corvinus.hu/index.php?id=26637`

larger quadratic subproblems. We still observe a decrease of CPU time in all our instances, even if the uncontrolled linearizations may have a poor accuracy. We also note that the decrease is more important for proximal algorithm than the level one. This is due to the fact that the level algorithm without uncontrolled information (L) does already well: we see that the CPU times of L are comparable to the ones of u-P.

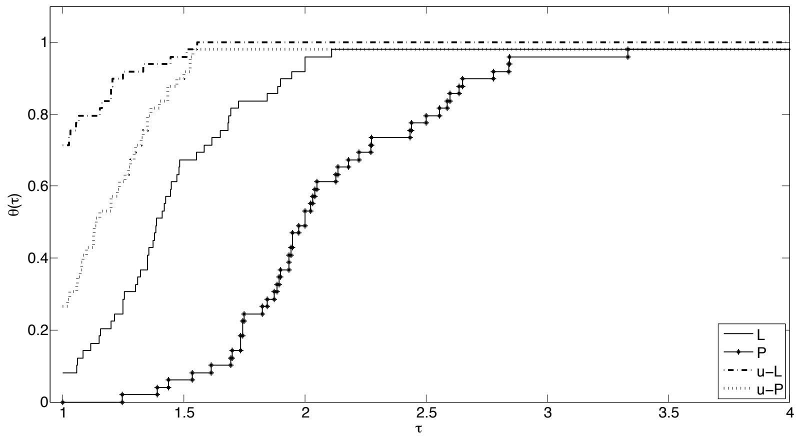

Fig. 1 Performance profiles of the four methods over the 49 instances

For these problems, the best method in terms of both oracle calls and CPU time is u-L, the level method using the uncontrolled cutting-plane generator. Figure 1 confirms this by showing the performance profile of all solvers with respect to oracle calls (the plot for CPU time is similar). Since its curve is always higher, u-L clearly dominates the other methods in terms of speed and robustness. The value at $\tau = 1$ indicates that u-L is the best to solve around 70% of the 49 problems; it also solves all the problems within a factor $\tau \approx 1.5$ of the best method.

We finish with a remark about the influence of the accuracy of the uncontrolled oracle. We note indeed that we can adjust the accuracy of the uncontrolled oracle in this situation by changing the percentage of subproblems (35) solved. In the reported experiments, we choose to solve 10% of the subproblems because we found out that it provides a good compromise between performance of the overall algorithm and computational burden of the external module. We mention here that, during preliminary tests, we observed that solving fewer subproblems (e.g. 1% of all subproblems) tends to increase of the total number of exact oracle calls and to give higher CPU times to solve the overall problem. On the other hand, we also observed that solving more subproblems (e.g. 20% of all subproblems) tends to decrease of the number of exact oracle calls, but with higher total CPU costs (since the uncontrollable oracle becomes more expensive). In general, the efficient choice of the percentage of problems solved depends on the problem's data, such as variance of the random vectors and number of considered scenarios.

4.2 Chance-constraint optimization problems

Problem and instances description. Joint chance-constrained optimization problems appear in cascaded reservoir management in presence of probabilistic guarantees that volumes in the reservoirs remain within bounds see e.g. [vAdO14]. With a target probability $p \in (0,1)$, these constraints can be expressed as $\mathrm{P}[g(y) \geq \xi] \geq p$ where $\xi \in \mathbb{R}^n$ represents the random vector of water inflows (of associate probability measure P) and $g : \mathbb{R}^m \to \mathbb{R}^n$ is an affine mapping. The associated optimization problem can be written (see more precisely [vAHMZ14, Eq.(15)]) as

$$\min_{y \in Y, v \in V} q^\top y \quad \text{s.t.} \quad g(y) \geq v\,, \tag{36}$$

where Y is a bounded polyhedron and V the set of points satisfying the probability constraint. When considering finitely many scenarios $\{\xi^1, \ldots, \xi^N\}$ with associated probability $\{\pi_1, \ldots, \pi_N\}$ (see e.g. [SDR09, Chap. 4], [DM13, Chap. 6], and [vABdOS15]), V can be expressed as the following feasibility set

$$V = \left\{ v \in \mathbb{R}^n : \exists\, z \in \{0,1\}^N,\ \pi^\top z \leq 1-p,\ (1-z_i)\xi^i \leq v - \underline{b} z_i,\ i = 1, \ldots, N \right\}$$

where $\underline{b} \in \mathbb{R}^n$ is defined component-wise by $\underline{b}_j := \min_{1 \leq i \leq N} \xi^i_j$. Then the dual problem has the form (1) with

$$X = \mathbb{R}^n_+ \quad \text{and} \quad f(x) := -(h(x) + d(x))$$

where $h(x)$ is the optimal value of a mere linear programming problem (since Y is a polyedron and g is affine)

$$h(x) := \min_{y \in Y} q^\top y - x^\top g(y)$$

and $d(x)$ is the optimal value of a (large-scale) mixed-binary linear problem

$$d(x) := \begin{cases} \min\limits_{v \in \mathbb{R}^m,\ z \in \{0,1\}^N} & x^\top v \\ \text{s.t.} & \xi^i(1-z_i) \leq v - \underline{b} z_i,\ i = 1, \ldots, N \\ & \pi^\top z \leq 1-p\,. \end{cases} \tag{37}$$

Here we use the instances described in [vAHMZ14] and [vAdO14] constructed from real-life data on the French hydro-valley Isère (provided to us by EDF, the French Electricity Board). For $N \in \{50, 100, 150, 200, 250\}$ and $p \in \{80\%, 90\%\}$, three different scenario samples are randomly generated, and as a result, we get thirty different associated instances.

Oracles and cutting-plane generator. The bulk of the work of an exact oracle for f is to solve the mixed-binary linear optimization problem (37) to optimality, which is expensive as N grows. On the other hand, we have an easy way to produce feasible solutions, as follows. To any binary point $\tilde{z} \in \{0,1\}^N$ satisfying $\pi^\top \tilde{z} \leq 1-p$ we associate the vector $\tilde{v} \in \mathbb{R}^n$ such that

$$\tilde{v}_j := \max_{i \in \{l:\, z_l = 0\}} \xi^i_j \ \text{ for all } \ j = 1, \ldots, N\,. \tag{38}$$

Observe then that the pair $(\tilde{v}, \tilde{z})$ is feasible for (37). Accordingly, $d_x := x^\top \tilde{v}$ is an upper approximation for $d(x)$, which in turn provides a (cheap but imprecise) approximation for $f(x)$

$$f_x := -(h(x) + d_x) \leq f(x).$$

The recent work [vABdOS15] proposes a fast heuristic (denoted *Heuristic h1* therein) to compute a good candidate $\tilde{z}$ (and therefore $\tilde{v}$ as above) to approximate a solution of problem (37). Thus we are in the situation (7) with

- an exact oracle providing the value $f(x)$ and a subgradient $g \in \partial f(x)$ ($\eta = 0$) by solving exactly the subproblem (37) with Gurobi;
- an uncontrolled oracle using the heuristic of [vABdOS15] and (38);
- a cutting-plane generator consisting in running several iterations of a bundle method using only the uncontrolled oracle (with the same stopping test and a maximum of 100 iterations).

Numerical results. Table 2 reports the number of (exact) oracle calls and CPU time (in minutes) required to reach convergence, for the four algorithms over the 30 test-problems. Each entry is the average over the instances with same N and p, except for the last line which is the grand total over the 30 instances.

		# oracle calls				CPU time (min)			
N	p	L	P	u-L	u-P	L	P	u-L	u-P
50	0.8	18	32	13	13	0.6	0.9	0.4	0.4
50	0.9	18	18	11	9	0.4	0.4	0.3	0.2
100	0.8	19	24	12	15	2.9	3.5	1.7	1.6
100	0.9	19	19	11	21	1.3	1.1	0.7	1.2
150	0.8	19	24	11	20	12.8	12.0	6.2	7.9
150	0.9	18	24	8	18	2.4	3.6	1.2	2.1
200	0.8	20	19	13	23	24.2	20.2	12.1	12.6
200	0.9	19	22	10	6	5.6	5.7	3.2	1.4
250	0.8	18	32	15	15	48.9	45.6	29.9	28.0
250	0.9	19	36	12	40	17.0	26.1	5.3	20.5
Total		558	751	346	543	5.8h	6.0h	3.0h	3.8h

Table 2 Comparison of the four algorithms with respect to oracle calls and global CPU time to get convergence. Each entry is the average over the three instances, except for the last line which is the grand total over the 30 instances.

The figures show that introducing uncontrolled linearizations reduces both the number of oracle calls and the CPU time for both proximal and level algorithms. This improvement is even more significant than for the two-stage problems: the reduction of CPU times is of 47% for u-L and 36% for u-P.

We also see that u-L is more efficient u-P, both in CPU time and number of oracle calls. In fact u-L makes a better use of uncontrolled information added by the cutting-plane generator: L and P are comparable in terms of CPU time whereas u-L is faster than u-P (by more than 20%). The performance profiles of Figure 2 confirm that u-L is the fastest and most robust among the four methods.

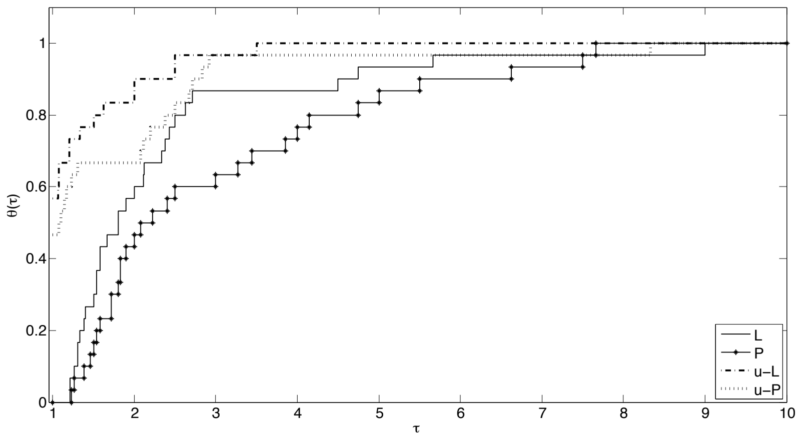

Fig. 2 Performance profiles of the four methods on the 30 instances.

5 Conclusions

This paper analyzes two bundle algorithms (a proximal one and a level one) handling cheap uncontrolled inexact linearizations, incorporated by an abstract cutting-plane generator. Beside the formalisation and the emphasis on uncontrolled bundle information, the main technical contribution of this paper is the challenging convergence analysis of the level algorithm. This algorithm extends [BKL95] to handle inexact η-oracles and to use the general uncontrolled cutting-plane model generator. The key feature of this algorithm is a novel noise attenuation rule, that can be seen as an implicit version of the one of [Kiw06]. Numerical experiments on two energy optimization problems show that including cheap uncontrolled information can decrease the CPU time to reach optimality, and that the level algorithm, fully exploiting the additional information, works particularly well on these problems.

To our knowledge, this paper is the first one to consider cheap uncontrolled inexact information within bundle methods and to show the interest to use it. A recent preprint [vAFO15] builds on this line of research in a context of Benders decomposition. Note finally that we consider here an extreme case of a cutting-plane generator with no control at all on the linearizations. More sophisticated and subtle ways to incorporate cheap information should be possible, as for exemple using "adaptative oracles". Such inexact oracles would interact with the bundle algorithm (though accuracy parameters as in [dOS14] but not only) and would be able to choose between several available approximation schemes with increasing accuracy and increasing cost (as for example the three specific heuristics of [BTV06] for a combinatorial problem). A general study of smart and communicating oracles goes beyond the scope of the paper and deserves special research and developments.

Acknowledgment

We thank Antonio Frangioni (Univ. of Pisa, Italy) for insightful discussions on a first version of this article and Wim van Ackooij (EDF, France) for providing us with the real-life data set used in section 4.2. The first author gratefully acknowledges the support of the grant "ANR GeoLMI" and the CNRS Mastodons project "gargantua/titan". The second author gratefully acknowledges the support provided by Severo Ochoa Program SEV-2013-0323 and Basque Government BERC Program 2014-2017.

References

[BKL95] U. Brannlund, K. 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.

[BTV06] C. Beltran, C. Tadonki, and J.Ph. Vial, *Solving the p-median problem with a semi-lagrangian relaxation*, Computational Optimization and Applications **35** (2006), no. 2.

[Deá06] I. Deák, *Two-stage stochastic problems with correlated normal variables: computational experiences*, Annals OR **142** (2006), no. 1, 79–97.

[DL05] J. Desrosiers and M. Lubbecke, *A primer in column generation*, Column Generation (G. Desaulniers, J. Desrosiers, and M.. Solomon, eds.), Springer US, 2005, pp. 1–32 (English).

[DM02] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming **91** (2002), 201–213.

[DM13] D. Dentcheva and G. Martinez, *Regularization methods for optimization problems with probabilistic constraints*, Math. Programming (series A) **138** (2013), no. 1-2, 223–251.

[dOS14] W. de Oliveira and C. Sagastizábal, *Level bundle methods for oracles with on-demand accuracy*, Optimization Methods and Software **29** (2014), no. 6, 1180–1209.

[dOSL14] W. de Oliveira, C. Sagastizábal, and C. Lemaréchal, *Convex proximal bundle methods in depth: a unified analysis for inexact oracles*, Mathematical Programming **148** (2014), no. 1-2, 241–277 (English).

[dOSP+10] W. de Oliveira, C. Sagastizábal, D. Penna, M. Maceira, and J. Damázio, *Optimal scenario tree reduction for stochastic streamflows in power generation planning problems*, Optimization Methods and Software **25** (2010), no. 6, 917–936.

[dOSS11] W. de Oliveira, C. Sagastizábal, and S. Scheimberg, *Inexact bundle methods for two-stage stochastic programming*, SIAM Journal on Optimization **21** (2011), no. 2, 517–544.

[dS15] W. de Oliveira and M. Solodov, *A doubly stabilized bundle method for nonsmooth convex optimization*, Mathematical Programming (2015), 1–35.

[Fáb00] C. Fábián, *Bundle-type methods for inexact data*, Central European Journal of Operations Research **8** (2000), 35–55.

[Fra02] A. Frangioni, *Generalized bundle methods*, SIAM Journal on Optimization **13** (2002).

[Geo72] A.M. Geoffrion, *Generalized Benders decomposition*, Journal of Optimization Theory and Applications **10** (1972), no. 4, 237–260.

[Hin01] M. 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).

[Kel60] J. E. Kelley, *The cutting plane method for solving convex programs*, J. Soc. Indust. Appl. Math. **8** (1960), 703–712.

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

[Lem01] C. Lemaréchal, *Lagrangian relaxation*, Computational Combinatorial Optimization (M. Jünger and D. Naddef, eds.), Springer Verlag, Heidelberg, 2001, pp. 112–156.

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

[MW81] T. L. Magnanti and R. T. Wong, *Accelerating benders decomposition: Algorithmic enhancement and model selection criteria*, Operations Research **29** (1981), no. 3, 464–484.

[RS03] A. Ruszczyński and A. Shapiro, *Stochastic programming*, Handbooks in Operations Research and Management Science, vol. 10, Elsevier, Amsterdam, 2003.

[SDR09] A. Shapiro, D. Dentcheva, and A. 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] M.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.

[TvAFL15] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra, *Large-scale unit commitment under uncertainty*, 4OR (2015), 1–57 (English).

[vABdOS15] W. van Ackooij, V. Berge, W. de Oliveira, and C. Sagastizábal, *Probabilistic optimization via approximate p-efficient points and bundle methods*, Tech. report, 2015, Optimization Online report number 4927.

[vAdO14] W. van Ackooij and W. de Oliveira, *Level bundle methods for constrained convex optimization with various oracles*, Computation Optimization and Applications **57** (2014), no. 3, 555–597.

[vAFO15] W. van Ackooij, A. Frangioni, and W. Oliveira, *Inexact stabilized benders decomposition approaches to chance-constrained problems with finite support*, Submitted; Available as preprint TR-15-01 of Universita di Pisa Dipartimento di Informatica (2015).

[vAHMZ14] W. van Ackooij, R. Henrion, A. Moller, and R. Zorgati, *Joint chance constrained programming for hydro reservoir management*, Optimization and Engineering **15** (2014), no. 2, 509–531.

[ZPR00] G. Zakeri, A. Philpott, and D. Ryan, *Inexact cuts in benders decomposition*, SIAM Journal on Optimization **10** (2000), no. 3, 643–657.