A Bundle Method for Exploiting Additive Structure in Difficult Optimization Problems

Welington de Oliveira* Jonathan Eckstein†

Research Report

May 27, 2015

Abstract

This paper describes a bundle method for (approximately) minimizing complicated nonsmooth convex functions with additive structure, with the primary goal of computing bounds on the solution values of difficult optimization problems such as stochastic integer programs. The method combines features that have appeared in previously proposed bundle methods, but not in the particular configuration we propose. In particular, we use a disaggregate cutting-plane model and approximate lower oracles with on-demand accuracy, and we design the method so that certain trial points can be abandoned without having to invest the full effort of evaluating all their subproblems.

1 Introduction

1.1 Motivation

Consider a difficult optimization problems of the form

$$\begin{array}{ll} \max & \sum_{i=1}^m c^i(p^i) \\ \text{s. t.} & p^i \in P^i, \qquad\qquad \forall i = 1, \ldots, m \\ & \sum_{i=1}^m g^i(p^i) \in K\,, \end{array} \tag{1}$$

where the $K \subset \mathbb{R}^n$ is a nonempty closed convex cone, and, for $i = 1, \ldots, m$, $P^i \subseteq \mathbb{R}^{n^i}$ is an arbitrary nonempty compact (closed and bounded) set, $c^i : \mathbb{R}^{n^i} \to \mathbb{R}$, and $g^i : \mathbb{R}^{n^i} \to \mathbb{R}^n$, with g^i continuous on P^i.

The motivation for the algorithm proposed in this paper comes from a particular special case of (1) arising from large-scale stochastic integer programming with $T \geq 2$ stages and a finite probability space $\Omega = \{1, \ldots, m\}$. There are thus m leaf nodes in the scenario tree, each identified by an integer between 1 and m. For each i, we let $p^i = (p^{i1}, \ldots, p^{iT}) \in \mathbb{R}^{n^i}$, where p^{it} denotes the vector of all primal decision variables determined at the unique stage-t node in the scenario tree that is the ancestor of leaf node i. Furthermore, let s^t denote the number of nodes of the scenario tree at stage t and let $\mathcal{S}^t = \{S^{t1}, \ldots, S^{ts^t}\}$ be a partition of $\{1, \ldots, m\}$ into sets such that i and j are in the same element of the partition when they are indistinguishable at time t. Thus $s^1 = 1$, $\mathcal{S}^1 = \big\{\{1, \ldots, m\}\big\}$, $s_T = m$, and $S^T = \big\{\{1\}, \ldots, \{m\}\big\}$. Let $C^i(p^i)$ denote the objective value realized by the course of action p^i, and π^i denote the probability of leaf node i. Next, letting $n = \sum_{i=1}^m n^i$, define V to be the *nonantipativity* linear subspace of $\mathbb{R}^n$, meaning that if $(z^1, \ldots, z^m) \in V$, then all stage-t decision variables in z^i and z^j must be equal if leaf nodes i and j are indistinguishable at t, that is,

$$V = \big\{(z^1, \ldots, z^m) \in \mathbb{R}^n \;\big|\; z^{it} = z^{jt}\; \forall\, i, j \in S,\, \forall S \in \mathcal{S}^t,\, t = 1, \ldots, T-1\big\}\,. \tag{2}$$

We let the set P^i represent all the within-scenario constraints that apply to the selection of p^i, that is, all constraints on p^i other than nonanticipativity. In particular, we allow P^i to express integrality constraints

*Universidade do Estado do Rio de Janeiro, Brazil
†Rutgers University, Piscataway, NJ, USA

(and hence be nonconvex). In this way, we may express a very general stochastic (and possibly mixed-integer) programming problem as

$$\begin{array}{ll} \max & \sum_{i=1}^m \pi^i C^i(p^i) \\ \text{s.t.} & p^i \in P^i, \qquad\qquad \forall i = 1, \ldots, m \\ & (p^1, \ldots, p^m) \in V\,, \end{array} \tag{3}$$

This problem becomes a special case of (1) if we define K, $c^1, \ldots, c^m$, $g^1, \ldots, g^m$ in a particular way. Specifically, problem (1) is equivalent to (3) if we let $c^i(p^i) = \pi^i C^i(p^i)$, $K = V$, and

$$g^i : p^i \mapsto (0, \ldots, 0, p^i, 0, \ldots, 0). \tag{4}$$

The formulation (3) is natural one for scenario-based decomposition methods, and is essentially the one used in deriving the *progressive hedging* (PH) algorithm for convex stochastic programming [26], which results from applying the alternating direction method of multipliers (ADMM) algorithm [12, 14, 7] to (3) with a particular choice of inner product on $\mathbb{R}^n$. Progressive hedging has been used with some practical success on stochastic programming problems formulated in this manner [17, 30, 16, 31]. However, if any of the sets P^i are nonconvex, the progressive hedging approach is only a heuristic method to current knowledge. In particular, the quality of the solutions it produces can be difficult to assess without additional analysis.

A natural bound on the quality of any feasible solution to (1) may be obtained through standard use of the Lagrangian dual function. Define for $i = 1, \ldots, m$ the function $f^i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ by

$$f^i(x) := \left\{ \begin{array}{ll} \max & c^i(p^i) - \langle x, g^i(p^i) \rangle \\ \text{s.t.} & p^i \in P^i\,. \end{array} \right. \tag{5a}$$

Since f^i is a maximum of affine functions, it must be convex. Since we are assuming that P^i is compact, the maximum in (5a) must be attained for any x and hence f^i is finite-valued everywhere and hence continuous due to its convexity [24, Corollary 10.1.1]. Let $K^* := \{x \in \mathbb{R}^n \mid \langle x, y \rangle \leq 0 \; \forall\, y \in K\}$ denote the polar cone of K, which is necessarily closed, nonempty, and convex, and note that in the case that $K = V$, where V is a linear subspace of $\mathbb{R}^n$, we have $K^* = V^\perp$, the subspace orthogonal to V. Next consider any feasible solution of (1), namely some $(\bar{p}^1, \ldots, \bar{p}^m) \in P^1 \times \cdots \times P^m$ with $\sum_{i=1}^m g^i(\bar{p}^i) \in K$. Then if $x \in K^*$, we have by definition of the polar cone that $\langle x, \sum_{i=1}^m g_i(\bar{p}_i) \rangle \leq 0$ and therefore

$$\sum_{i=1}^m c^i(\bar{p}^i) \leq \sum_{i=1}^m c^i(\bar{p}^i) - \left\langle x, \sum_{i=1}^m g^i(\bar{p}^i) \right\rangle = \sum_{i=1}^m \left[c^i(\bar{p}^i) - \langle x, g^i(\bar{p}^i) \rangle \right] \leq \sum_{i=1}^m f^i(x)\,.$$

Here, the second inequality follows from the definition (5a), since $\bar{p}^i \in P^i$ for all i. We conclude that for any $x \in K^*$, the quantity $\sum_{i=1}^m f^i(x)$ provides an upper bound on the optimal value of (1), and are therefore interested in computing the smallest such upper bound,

$$f^{\inf} := \min_{x \in K^*} f(x)\,, \text{ where } f(x) := \sum_{i=1}^m f^i(x). \tag{5b}$$

Here, $\min_{x \in K^*} f(x)$ is the standard *dual problem* of (1) and $f^{\inf}$ is its optimal value. Calculating $f(x)$ should generally be easier than solving (1), because it involves solving m independent lower-dimensional problems, a phenomenon known as *price decomposition*; see for example [1, §11.1.1]. In addition to calculating each individual $f^i(x)$ being easier than solving the original problem (1), it may be possible to simultaneously carry out more than one such calculation using parallel computing.

This paper presents a solution algorithm for problems of the form (5) in the situation that the individual functions (5a), despite being easier than the original problem (1), remain time consuming to evaluate, even approximately. Our standing assumption is that the original problem (1) is far too difficult to directly solve to provable optimality with a non-decomposition-based solver, but some source of heuristically-generated feasible solutions $(p^1, \ldots, p^m)$ is available, and we would like to compute a bound on their quality. We further assume that it is possible to approximately evaluate each of the functions f^i up to a specified accuracy in a manageable amount of time depending on the accuracy desired. Specifically, for $\varepsilon \geq 0$, we define an *ε-solution* to an optimization problem to be a feasible solution whose objective value is within ε of

the optimal value. In particular, we assume that it is possible for any $x \in K^*$ and $\varepsilon > 0$ to find ε-solutions p_x^i to the problems (5a), that is,

$$p_x^i \in P^i \quad : \quad c^i(p_x^i) - \langle x, g^i(p_x^i)\rangle \geq f^i(x) - \varepsilon. \tag{6}$$

However, the computational effort required may be substantial and will depend on the choice of ε. In particular, we have in mind the case that the sets P^i include integer constraints, and that finding points p_x^i as specified in (6) will require invoking a moderately lengthy run of a commercial-quality MIP solver such as GuRoBi or CPLEX for each i. In this context, our goal is to find a setting of the dual variables $x \in K^*$ which approximates the optimal dual value $f^{\inf}$ to some specified accuracy.

Besides the effort required to approximately evaluate each component f^i of the objective, our problem presents several more difficulties: (i) the number of subproblems m may be large, adding further to the difficulty of evaluating the (dual) objective $f(x)$ for a given point x; (ii) while convex and continuous due to the convexity and continuity of the individual functions f^i, the function f is generally nonsmooth. In fact, Lemma 2.1 below shows that if p_x^i solves (5a), then $-g^i(p_x^i)$ is a subgradient of f^i at the point x, that is, $-g^i(p_x^i) \in \partial f^i(x)$. An immediate consequence of this lemma is that f^i must be nonsmooth at x if (5a) has more than one solution, and thus that f must be nonsmooth at x whenever the optimal solution to (5a) is nonunique for even a single index i.

1.2 Algorithm features and related literature

The only favorable properties of f are its convexity, continuity, and additive structure. Our algorithm will take advantage of these properties, while using only approximate evaluations of each f_i and its subgradients, and trying to avoid evaluating all m subproblems (5a) at certain "unpromising" trial points. The algorithm is in the bundle method family, with a number of features that have essentially all appeared in prior proposed algorithms, but not in the particular combination required by our applications.

The kind of inexact calculation described by (6), which provides inexact information about the value of f^i (and also, as we will see below in Lemma 2.1, its subgradients), is called an *inexact oracle* in the bundle method literature. Essentially, the algorithm we propose is that of [8], but employing inexact oracles in the manner suggested in [4]. In recent years, numerous nonsmooth optimization methods have been proposed to handle inexact oracles, under varying assumptions. The inexact proximal bundle methods proposed in [18] and [28] employ inexact oracles whose errors are bounded by a known constant. More general inexact oracles are addressed in [19, 6]. Oracles with vanishing errors are studied in [9, 33, 11, 8].

The specific kind of oracle we use in this paper follows [8] and is based on (6). We assume that we have a separate oracle for each subproblem i, as follows:

$$\begin{pmatrix} x \\ \varepsilon^i \end{pmatrix} \longrightarrow \boxed{\textbf{Oracle}} \longrightarrow \begin{cases} f_x^i \in [f^i(x) - \varepsilon^i, f^i(x)] \\ g_x^i \in \mathbb{R}^n : f^i(y) \geq f_x^i + \langle g_x^i, y - x\rangle \;\forall\, y \in \mathbb{R}^n. \end{cases} \tag{7}$$

The arguments to the oracle are a given vector $x \in K^*$ and an error tolerance specification $\varepsilon^i \geq 0$. The output of the oracle is a scalar f_x^i which underestimates $f^i(x)$ by no more than ε^i, and a vector g_x^i such that the affine function $y \mapsto f_x^i + \langle g_x^i, y - x\rangle$ always lies below $f^i(y)$. As shown in Lemma 2.1 below, this is precisely the kind of information about f^i provided by the inexact optimization calculation (6). Note also that if $\varepsilon^i = 0$, then $f_x^i = f^i(x)$ and g_x^i is a subgradient of f^i at x.

In the nomenclature introduced in [4], (7) is called an *oracle with on-demand accuracy*, because its maximum error is controlled by an input parameter and may in principle be as small as desired. In [4], a family of level bundle methods is proposed for handling such oracles. Such on-demand accuracy ideas were extended to proximal bundle methods in [5], which presented a unified convergence analysis for these and other proximal bundle methods; see also [23] for a survey of both level and proximal bundle methods using inexact function evaluations. Here, we propose a proximal bundle method, but combining the inexact-oracle techniques of the level bundle method in [4] with the incremental subproblem-evaluation approach of [8].

The bundle-method literature also refers to an oracle of the form (7) as being of *lower type*, because the inexact value provided by the oracle always underestimates the exact value of the function, while the linearization $y \mapsto f_x^i + \langle g_x^i, y - x\rangle$ always approximates $f^i(y)$ from below. This property can be useful in algorithm design. The recent computational study in [32], for solving two-stage stochastic linear programs,

shows that proper use of an oracle of this type can reduce CPU time in up to 79% as compared to classical algorithms for the same application, without sacrificing solution quality. We should note, however, that the bundle application to stochastic programming in [32] is very different from the ones inspiring our proposed algorithm, which involve both discrete variables and a possibly large number of stages. The usefulness of lower oracles with on-demand accuracy is also highlighted in [29], where the authors deal with chance-constrained programming problems.

1.3 Contributions

Rather than proposing an entirely new optimization method, we present a proximal bundle algorithm that combines key features from various algorithms from the prior literature. Specifically, Algorithm 2 given in Section 3 below is based on [8], but includes features from [2, 4, 19] in order to efficiently solve (5). Our proposed method is related to prior bundle methods as follows:

- Similarly to [2, 8], the algorithm employs a disaggregate cutting-plane model for the function f, that is, it maintains separate piecewise-linear approximations to each of the m additive component function f^i. This feature makes it a multi-cut variant of the algorithm given in [19]. In stochastic programming, this strategy is related to to the regularized decomposition of [27] and the level decomposition of [10, 32], but our proposed method differs from such prior work in its cut (linearization) management, which provides the option of making it a limited-memory algorithm. Our choice of a disaggregate function model is influenced by the particular structure of the the function f arising from formulations like (3), as explained in Sections 2.1 and 3.2 below.

- As in [8], some variants of our proposed algorithm might never request exact oracle information. However, other aspects of the algorithm are different from [8].

- In contrast to the methods presented in [4, 10], which use oracles with on-demand accuracy and are in the level bundle family, our algorithm uses on-demand-accuracy oracles but is of the proximal type [1, 20, 13, 2].

- Our algorithm may be able to deduce that certain trial points x are unpromising after considering only a subset of the m subproblems (5a); it will then abandon such points without investing full computational effort in them. Consideration of all m subproblems defining f is necessary only at a subset of iterates called *serious steps* (see the formal definition below in §2). Thus, our algorithm both exploits the additive structure of problem (5) and also puts into practice the key idea from [5]: high-accuracy oracle information is only needed at serious steps.

The core of the convergence analysis of our proposed algorithm is based on the analysis in [5].

2 Preliminary results

Our notation is fairly standard. For any set $X \subseteq \mathbb{R}^n$, we let i_X denote its convex indicator function, that is, $i_X(x) = 0$ when $x \in X$ and $i_X(x) = +\infty$ when $x \notin X$. Given a convex function f and a scalar $\varepsilon \geq 0$, we denote its ε-subdifferential (set of ε-subgradients) at x by $\partial_\varepsilon f(x) = \{g : f(y) \geqslant f(x) + \langle g, y - x\rangle - \varepsilon \ \forall\, y \in \mathbb{R}^n\}$. In the case $\varepsilon = 0$, we have that $\partial_0 f(x)$ is the conventional subdifferential of f at x, $\partial f(x)$.

Lemma 2.1. *Consider any $i \in \{1 \ldots, m\}$, $x \in K^*$ and $\varepsilon^i \geq 0$. Then if p^i_x is a ε^i-solution to subproblem* (5a)*, that is, $p^i_x \in P^i$ and*

$$c^i(p^i_x) - \langle x, g^i(p^i_x)\rangle \geq f^i(x) - \varepsilon^i\,, \tag{8}$$

then

$$f^i_x := c^i(p^i_x) - \langle x, g^i(p^i_x)\rangle \quad \textit{and} \quad g^i_x := -g^i(p^i_x)$$

satisfy the oracle assumptions (7)*. Furthermore, $g^i_x \in \partial_{\varepsilon^i} f^i(x)$.*

Proof. The condition $f_x^i \in [f^i(x) - \varepsilon^i, f^i(x)]$ follows directly from (8) and the definition (5a) of $f^i(x)$ as the maximum possible value of $c^i(p^i) - \langle x, g^i(p^i)\rangle$ over $p^i \in P^i$. Again using this definition and $p^i \in P^i$, we have that for any $y \in \mathbb{R}^n$,

$$\begin{aligned} f^i(y) &\geq c^i(p_x^i) - \langle y, g^i(p_x^i)\rangle = c^i(p_x^i) - \langle x, g^i(p_x^i)\rangle - \langle g^i(p_x^i), y - x\rangle \\ &= f_x^i - \langle g^i(p_x^i), y - x\rangle = f_x^i + \langle g_x^i, y - x\rangle \,. \end{aligned}$$

Thus, we have established the subgradient-like inequality in (7). Finally, for all $y \in \mathbb{R}^n$, we have

$$f^i(y) \geq f_x^i + \langle g_x^i, y - x\rangle = f^i(x) + \big(f_x^i - f^i(x)\big) + \langle g_x^i, y - x\rangle \geq f^i(x) + \langle g_x^i, y - x\rangle - \varepsilon^i \,,$$

which establishes that $g_x^i \in \partial_{\varepsilon^i} f^i(x)$. □

The above result allows us to define a lower oracle with on-demand accuracy in a straightforward manner: given a candidate $x \in K^*$ and tolerances $\varepsilon^i \geq 0$ for $i = 1, \ldots, m$, find an ε^i-solution p_x^i to (5a) and thus determine the pair $(f_x^i, g_x^i) \in \mathbb{R}^{1+n}$ defined in Lemma 2.1 for each $i = 1, \ldots, m$. Thus, we have a lower oracle with on-demand accuracy for each component function f^i. Adding the results of these oracles then provides a lower oracle with on-demand accuracy for the overall objective function $f = \sum_{i=1}^m f^i$:

Lemma 2.2. *Given $x \in K^*$, suppose the $(f_x^i, g_x^i) \in \mathbb{R} \times \mathbb{R}^n$ are as defined in Lemma 2.1 for $i = 1, \ldots, m$. Define $\varepsilon := \sum_{i=1}^m \varepsilon^i$, $f_x := \sum_{i=1}^m f_x^i$ and $g_x := \sum_{i=1}^m g_x^i$. Then*

$$f_x \in [f(x) - \varepsilon, f(x)] \qquad f(y) \geq f_x + \langle g_x, y - x\rangle \;\forall\, y \in \mathbb{R}^n \qquad g_x \in \partial_\varepsilon f(x). \tag{9}$$

Proof. The results follow immediately by respectively summing the inequalities in (7) and the inequality $f^i(y) \geq f^i(x) + \langle g_x^i, y - x\rangle - \varepsilon^i$ over $i = 1, \ldots, m$. □

2.1 Aggregate and disaggregate cutting-plane models

A standard tool in nonsmooth optimization methods is a convex and typically piecewise-linear *model* function which approximates the true objective function f from below, but is relatively easy to optimize. If we have just completed iteration k of our optimization algorithm, having generated trial solution points $x_1, \ldots, x_k$ and corresponding lower-oracle pairs (f_{x_j}, g_{x_j}) as defined in (9), then a natural model function to use is

$$\check{f}_k(x) := \max_{j \in \mathcal{B}_k} \{f_{x_j} + \langle g_{x_j}, x - x_j\rangle\} \,, \tag{10}$$

for some set of indices $\mathcal{B}_k \subseteq \{1, 2, \ldots, k\}$. We call the set of triples $\{(x_j, f_{x_j}, g_{x_j})\}_{j \in \mathcal{B}_k}$ the *information bundle*, and for brevity we will simply refer to $\mathcal{B}_k$ as the *bundle*. The model (10) is valid because each affine function $x \mapsto f_{x_j} + \langle g_{x_j}, x - x_j\rangle$ lies below f, so their pointwise maximum also does. Such a model function $\check{f}_k$ is called an *aggregate* model, since it uses linearizations of the entire function f. A *disaggregate* model for f is obtained by adding similar individual models for each constituent function f^i, as follows: given lower-oracle pairs $(f_{x_j}^i, g_{x_j}^i)$ for each $i = 1, \ldots, m$, we define

$$\check{f}_k^i(x) := \max_{j \in \mathcal{B}_k^i} \{f_{x_j}^i + \langle g_{x_j}^i, x - x_j\rangle\} \;\text{ for some }\; \mathcal{B}_k^i \subseteq \{1, \ldots, k\}, \;\text{ and } f_k^{\mathrm{m}}(x) := \sum_{i=1}^m \check{f}_k^i(x). \tag{11}$$

A disaggregate model is in general more accurate than an aggregate model if they are based on comparable bundle indices, as shown in the following standard result:

Lemma 2.3. *If $\mathcal{B}_k^i \supseteq \mathcal{B}_k$ for $i = 1, \ldots, m$, then $\check{f}_k(x) \leq f_k^{\mathrm{m}}(x) \leq f(x)$ for all $x \in \mathbb{R}^n$.*

Proof. Fix any $x \in \mathbb{R}^n$, and let j' denote some index j which maximizes $\{f_{x_j} + \langle g_{x_j}, x - x_j\rangle\}$ over $\mathcal{B}_k$. Then

$$\begin{aligned} \check{f}_k(x) &= f_{x_{j'}} + \langle g_{x_{j'}}, x - x_{j'}\rangle \\ &= \sum_{i=1}^m f_{x_{j'}}^i + \left\langle \sum_{i=1}^m g_{x_{j'}}^i, x - x_{j'} \right\rangle \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{m} \left(f^i_{x_{j'}} + \langle g^i_{x_{j'}}, x - x_{j'} \rangle \right) \\
&\leq \sum_{i=1}^{m} \max_{j \in \mathcal{B}^i_k} \left\{ f^i_{x_j} + \langle g^i_{x_j}, x - x_j \rangle \right\} \\
&= f^{\mathrm{m}}_k(x),
\end{aligned}
$$

where the inequality follows because $j' \in \mathcal{B}_k \subseteq \mathcal{B}^i_k$. □

Aside from offering greater accuracy, the disaggregate model has another advantage which has already been exploited in [8]: it allows for "partial" oracle evaluations in which information about the value of $f(x)$ at a new trial point x_{k+1} may be obtained without calling all m of the individual subproblem oracles (7). Specifically, for any $L_{k+1} \subseteq \{1, \ldots, m\}$, we have

$$
f_{L_{k+1}} \;:=\; \underbrace{\sum_{i \in L_{k+1}} f^i_{x_{k+1}}}_{|L_{k+1}| \text{ oracle calls}} \;+\; \underbrace{\sum_{\substack{i=1 \\ i \notin L_{k+1}}}^{m} \check{f}^i_k(x_{k+1})}_{\substack{m - |L_{k+1}| \text{ model} \\ \text{approximations}}} \;\leq\; f(x_{k+1})\,. \tag{12}
$$

The inequality above follows because $f^i_{x_{k+1}} \leq f^i(x_{k+1})$ and $\check{f}^i_k(x_{k+1}) \leq f^i(x_{k+1})$ for $i = 1, \ldots, m$, and of course $f(x_{k+1}) = \sum_{i=1}^m f^i(x_{k+1})$.

When our bundle method examines an iterate x_{k+1}, it will sometimes not require the exact value of $f(x_{k+1})$, but will be able to abandon the iterate based only on the information that $f(x_{k+1}) > \gamma_k$, where γ_k is some "target" value. If we can establish this condition using a bound of the form $f_{L_{k+1}}$ as defined in (12), we may be able to economize on oracle calls by calling $|L_{k+1}| < m$ subproblem oracles at x_{k+1}, rather than all m oracles. Algorithm 1 below outlines a class of procedures for attaining this goal by incrementally growing L_{k+1} as needed.

Algorithm 1 Managing oracle calls

Input:
- A point $x_{k+1} \in K^*$
- A target value $\gamma_k \in \mathbb{R}$
- Bounds ε^i_{k+1}, $i = 1, \ldots, m$ for the oracle errors

Initialize $L_{k+1} \leftarrow \emptyset$ and $f_{L_{k+1}} \leftarrow f^{\mathrm{m}}_k(x_{k+1})$
while $|L_{k+1}| < m$ and $f_{L_{k+1}} \leq \gamma_k$ **do**
    Select some nonempty $J \subseteq \{1, \ldots, m\} \backslash L_{k+1}$
    for $i \in J$ **do**
        Call oracle (7) to compute $(f^i_{x_{k+1}}, g^i_{x_{k+1}})$ with accuracy $\varepsilon^i_{k+1} \geq 0$
    end for
    $L_{k+1} \leftarrow L_{k+1} \cup J$
    Compute $f_{L_{k+1}}$ from (12), or $f_{L_{k+1}} \leftarrow f_{L_{k+1}} + \sum_{i \in J} \max\{0, f^i_{x_{k+1}} - \check{f}^i_k(x_{k+1})\}$
end while
Set $f_{x_{k+1}} \leftarrow f_{L_{k+1}}$
Return L_{k+1}, $f_{x_{k+1}}$, and $\{(f^i_{x_{k+1}}, g^i_{x_{k+1}})\}_{i \in L_{k+1}}$

Initially, the algorithm above approximates the value $f(x_{k+1})$ by using the cutting-plane models $\check{f}^i$, $i = 1, \ldots, m$. It then selects a set of subproblems $J \subseteq \{1, \ldots, m\}$ and calls their oracles. It then includes J into the set L_{k+1} and updates its estimate $f_{L_{k+1}}$ of $f(x_{k+1})$. If this calculation is sufficient to establish that $f(x_{k+1}) > \gamma_k$ or all subproblem oracles have been evaluated, it exits. Otherwise, it repeats this procedure with another subset J of subproblem oracles.

In a serial setting, it would be most economical to fix $|J| = 1$ and thus evaluate a single subproblem oracle per iteration within Algorithm 1. In a parallel setting, however, it might be advantageous to choose $|J| > 1$, and perform the oracle calls in line 6 concurrently on different processors or groups of processors.

Note that Algorithm 1 is not specific about how to select the set J of subproblem oracles to invoke at each iteration. Determining the best strategy for selecting J will likely require some experimental study and could well be application-dependent. One can imagine greedy strategies in which one attempts to first select oracles that seem likely to be quick to evaluate, or to yield a large increase in the objective estimate, or some combination of these criteria.

When Algorithm 1 returns with $L_{k+1} < m$, it may still be necessary to have estimates of the values and subgradients of the functions f^i at x_{k+1}, for $i \notin L_{k+1}$. From (12), it is logical to use $\check{f}^i_k(x_{k+1})$ to estimate the value of f^i at x_{k+1}, along with a subgradient from the corresponding linearization. Thus, we define $j(i, k+1)$ to be an index maximizing $f^i_{x_j} + \langle g^i_{x_j}, x_{k+1} - x_j\rangle$ over $j \in \mathcal{B}^i_k$ (breaking ties arbitrarily), and then define the information triple $(x_{k+1}, f^i_{x_{k+1}}, g^i_{x_{k+1}})$ for f^i at x_{k+1} to be given by

$$(f^i_{x_{k+1}}, g^i_{x_{k+1}}) := \begin{cases} \text{as defined in Lemma 2.1} & \text{if } i \in L_{k+1} \\ (\check{f}^i(x_{k+1}), g^i_{x_{j(i,k+1)}}) & \text{if } i \notin L_{k+1}. \end{cases} \tag{13}$$

Regardless of which case holds above, we have a lower linearization of f^i, as formally demonstrated in the following simple lemma:

Lemma 2.4. *Under the definition* (13)*, we have for all* $i = 1, \ldots, m$ *and* $k \geq 0$ *that*

$$\bar{f}^i_{k+1}(x) := f^i_{x_{k+1}} + \langle g^i_{x_{k+1}}, x - x_{k+1}\rangle \leq f^i(x) \;\; \textit{for all} \;\; x \in K^* \,. \tag{14}$$

Proof. If $i \in L_{k+1}$, the above inequality follows directly from the properties of the oracle given in (7). If $i \notin L_{k+1}$, the inequality instead follows because $g_{x(i,k+1)}$ in (13) belongs to $\partial\check{f}^i(x_{k+1})$, the subdifferential of the cutting-plane model $\check{f}^i$ for f^i at x_{k+1}. Thus, using that $\check{f}^i$ lies below f^i, we have

$$f^i_{x_{k+1}} + \langle g^i_{x_{k+1}}, x - x_{k+1}\rangle \leq \check{f}^i(x) \leq f^i(x)$$

for any $x \in K^*$. □

Prior work on inexact bundle methods such as in [19, 28, 6] includes the assumption that the function estimation error $f(x) - f_x$ must be bounded above by a single constant for all $x \in K^*$. Such an assumption may not hold true in our setting due to our use of Algorithm 1: note that the value $f_{x_{k+1}}$ returned by Algorithm 1 can be a highly inaccurate estimate of $f(x_{k+1})$ in the $|L_{k+1}| < m$ case that some subproblem oracles were skipped because it was already established that $f_{L_{k+1}} > \gamma_k$. This phenomenon can occur even when $\varepsilon^i_{k+1} = 0$ for all $i = 1, \ldots, m$, meaning that each individual subproblem oracle invocation is exact. Thanks to the compactness of the P_i and continuity of the functions g^i, however, we are able to bound these errors with the aid of the following result:

Lemma 2.5. *In addition to the assumptions on* (1) *that the sets* P^i *are compact and maps* g^i *are continuous, suppose that* $\{x_k\}$ *and* $\{\varepsilon^i_k\}$, $i = 1, \ldots, m$ *are bounded sequences. Then the sequence of function estimation errors* $\{f(x_k) - f_{x_k}\}$ *and the sequence of subgradient estimates* $\{g_{x_k}\}$ *derived from Algorithm 1 through* (13) *are bounded.*

Proof. In what follows, we abbreviate the notation for the sum $\sum_{\substack{i=1 \\ i\notin L_k}}^m$ as $\sum_{i\notin L_k}$. Consider the following development, with k being an arbitrary iteration index:

$$\begin{aligned} 0 \leq f(x_k) - f_{x_k} = f(x_k) - f_{L_k} &= \sum_{i=1}^m f^i(x_k) - \sum_{i\in L_k} f^i_{x_k} - \sum_{i\notin L_k} \check{f}^i_{k-1}(x_k) \\ &= \sum_{i\in L_k} \left[f^i(x_k) - f^i_{x_k}\right] + \sum_{i\notin L_k} \left[f^i(x_k) - \check{f}^i_{k-1}(x_k)\right] \\ &\leq \sum_{i\in L_k} \varepsilon^i_k + \sum_{i\notin L_k} \left[f^i(x_k) - \check{f}^i_{k-1}(x_k)\right] \\ &= \sum_{i\in L_k} \varepsilon^i_k + \sum_{i\notin L_k} \Big[f^i(x_k) - \big(f^i_{x_{j(i,k)}} + \langle g^i_{x_{j(i,k)}}, x_k - x_{j(i,k)}\rangle\big)\Big]. \end{aligned}$$

Let $\tilde{p}^i_k \in P^i$ be some exact minimizer of (5a) with $x = x_k$, and let $\tilde{g}^i_k = -g^i(\tilde{p}^i_k)$, meaning that $\tilde{g}^i_k \in \partial f^i(x_k)$ by Lemma 2.1. Then, employing the standard subgradient inequality $f^i(x_k) - f^i(x_{j(i,k)}) \leq \langle \tilde{g}^i_k, x_k - x_{j(i,k)} \rangle$, we consider the expression between brackets above:

$$\begin{aligned}
& f^i(x_k) - \big(f^i_{x_{j(i,k)}} + \langle g^i_{x_{j(i,k)}}, x_k - x_{j(i,k)} \rangle\big) \\
= \;& f^i(x_k) - f^i(x_{j(i,k)}) + \big[f^i(x_{j(i,k)}) - f^i_{x_{j(i,k)}}\big] - \langle g^i_{x_{j(i,k)}}, x_k - x_{j(i,k)} \rangle \\
\leq \;& \langle \tilde{g}^i_k, x_k - x_{j(i,k)} \rangle + \varepsilon^i_{j(i,k)} - \langle g^i_{x_{j(i,k)}}, x_k - x_{j(i,k)} \rangle = \varepsilon^i_{j(i,k)} + \langle \tilde{g}^i_k - g^i_{x_{j(i,k)}}, x_k - x_{j(i,k)} \rangle \\
\leq \;& \varepsilon^i_{j(i,k)} + \|\tilde{g}^i_k - g^i_{x_{j(i,k)}}\| \|x_k - x_{j(i,k)}\| \leq \varepsilon^i_{j(i,k)} + \operatorname{diam}\big(g^i(P^i)\big) \|x_k - x_{j(i,k)}\| \,,
\end{aligned}$$

where $\operatorname{diam}(X) = \sup\{\|x - x'\| \mid x, x' \in X\}$ denotes the diameter of the a set X, and $g^i(P^i)$ denotes the image of the set P^i under the mapping g^i. The last inequality above holds because $\tilde{g}_k$ and $g^i_{x_{j(i,k)}}$ are both vectors of the form $-g^i(p^i)$ for some $p^i \in P^i$. Since P^i is compact and g^i is continuous on P^i, we have that $g^i(P^i)$ is bounded and hence that $\operatorname{diam}\big(g^i(P^i)\big)$ is finite. Combining the inequalities above, we have

$$0 \leq f(x_k) - f_{x_k} \leq \sum_{i \in L_k} \varepsilon^i_k + \sum_{i \notin L_k} \varepsilon^i_{j(i,k)} + \sum_{i \notin L_k} \operatorname{diam}\big(g^i(P^i)\big) \|x_k - x_{j(i,k)}\| \,.$$

From these inequalities, we may derive that for all k one has

$$0 \leq f(x_k) - f_{x_k} \leq \sum_{i=1}^{m} \Big(\bar{\varepsilon}^i + \operatorname{diam}\big(g^i(P^i)\big) \operatorname{diam}\big(\{x_j \mid j = 0, 1, \dots\}\big)\Big),$$

where $\bar{\varepsilon}^i := \sup_{k \geq 1}\{\varepsilon^i_k\}$. By hypothesis, $\bar{\varepsilon}^i$ and $\operatorname{diam}\big(\{x_j \mid j = 0, 1, \dots\}\big)$ are finite, and so $f(x_k) - f_{x_k}$ is bounded. The subgradient vectors $\{g_{x_k}\}$ must also be be bounded, because they lie in the Minkowski sum of the bounded sets $-g^i(P^i)$, $i = 1, \dots, m$. □

Clearly, the accuracy of both function and subgradient estimates in Algorithm 1 is related to the oracle errors ε^i_{k+1}, $i = 1, \dots, m$, and target γ_k. While the bounds for the oracle error may be chosen by the user, the target γ_k is automatically adjusted by the proximal bundle algorithm. This subject is addressed below.

3 A disaggregate algorithm for inexact oracles

3.1 Defining new iterates

Following the pattern of most bundle methods, our algorithm will maintain two sequences of iterates, $\{x_k\}$ and $\{\hat{x}_k\}$, both in K^*, that evolve as follows: at iteration k, the algorithm's best estimate of the solution is $\hat{x}_k$, whose function value is approximated by some scalar $f_{\hat{x}_k}$. Given the model f^{m}_k defined in (11), the algorithm then computes a new trial iterate x_{k+1} by solving the following subproblem:

$$x^{k+1} = \left\{ \begin{array}{ll} \arg\min & f^{\mathrm{m}}_k(x) + \frac{1}{2t_k}\|x - \hat{x}_k\|^2, \\ \text{s.t.} & x \in K^* \,, \end{array} \right. \tag{15}$$

where $t_k > 0$ is known as the proximal step. This procedure is just one step of the classical proximal minimization algorithm [22, 25], as applied to the model function f^{m}_k. The predicted corresponding improvement in the objective function is

$$\delta^{\mathrm{m}}_k := f_{\hat{x}_k} - f^{\mathrm{m}}_k(x_{k+1}) \,.$$

Using a parameter $\kappa \in (0, 1)$, the algorithm then sets a target value γ_k for the objective function as follows:

$$\gamma_k := f_{\hat{x}_k} - \kappa\delta^{\mathrm{m}}_k = (1 - \kappa) f_{\hat{x}_k} + \kappa f^{\mathrm{m}}_k(x_{k+1}) \,.$$

Attaining an objective value of γ_k would mean that at least a fraction κ of the predicted improvement is realized. Next, the method calls Algorithm 1 with the target value γ_k, obtaining an estimate $f_{x_{k+1}}$ of the objective value at x_{k+1}. Then

If $f_{x_{k+1}} \leq \gamma_k$,	we declare a *serious step*:	$\hat{x}_{k+1} \leftarrow x_{k+1}$,	$f_{\hat{x}_{k+1}} \leftarrow f_{x_{k+1}}$
Otherwise,	we declare a *null step*:	$\hat{x}_{k+1} \leftarrow \hat{x}_k$,	$f_{\hat{x}_{k+1}} \leftarrow f_{\hat{x}_k}$.

Note that while the algorithm does not change its best estimate of the solution when executing a null step, we will construct the method so that it still gathers information to improve the model function, so that f_{k+1}^{m} will differ from f_k^{m} and thus the next trial iterate x_{k+2} will typically differ from x_{k+1}. Essentially, when the method obtains at least a fraction κ of the model-predicted objective improvement, it moves to the tentative new iterate (a serious step). Otherwise, it keeps its solution estimate in place but attempts to improve its model of the function (a null step). We call $\{\hat{x}_k\}$ the sequence of *serious iterates*.

Serious steps always have $f_{x_{k+1}} \leq \gamma_k$, and therefore Algorithm 1 must evaluate all m subproblem oracles at such steps, making $f_{x_{k+1}}$ an estimate of $f(x_{k+1})$ with the known error bound $\varepsilon_{k+1} := \sum_{i=1}^m \varepsilon_{k+1}^i \geq 0$. At null steps, however, Algorithm 1 will stop evaluating subproblem oracles as soon as it determines that $(f_{L_{k+1}} =)\ f_{x_{k+1}} > \gamma_k$, so its approximation error will in general be unknown and potentially large. However, our analysis will show that such unknown oracle errors at null steps do not impair the quality of our algorithm's solution estimates.

Lemma 3.1. *Consider subproblem* (15). *There exist* $s_k \in \partial f_k^{\mathrm{m}}(x_{k+1})$ *and a vector* p_k *belonging to the normal cone* $N_{K^*}(x_{k+1})$ *of* K^* *at the point* x_{k+1}, *such that*

$$x_{k+1} = \hat{x}_k - t_k \hat{g}_k, \quad \textit{with} \quad \hat{g}_k := s_k + p_k\,. \tag{16}$$

Furthermore, the aggregate linearization

$$\bar{f}_k^{\mathrm{a}}(x) := f_k^{\mathrm{m}}(x_{k+1}) + \langle \hat{g}_k, x - x_{k+1}\rangle \tag{17}$$

satisfies $\bar{f}_k^{\mathrm{a}}(x) \leq f_k^{\mathrm{m}}(x) \leq f(x)$ *for all* $x \in K^*$.

Proof. The first claim follows by applying Theorem 23.8 of [24] to the equivalent problem of minimizing $f_k^{\mathrm{m}}(x) + (1/2t_k)\|x - \hat{x}_k\|^2 + i_{K^*}(x)$ over x, which is equivalent to problem (15). From $s_k \in \partial f_k^{\mathrm{m}}(x_{k+1})$ and $p_k \in N_{K^*}(x_{k+1})$, we have

$$\begin{array}{rcll} f_k^{\mathrm{m}}(x_{k+1}) + \langle s_k, x - x_{k+1}\rangle & \leq & f_k^{\mathrm{m}}(x) & \text{for all } \ x \in \mathbb{R}^n \\ \langle p_k, x - x_{k+1}\rangle & \leq & 0 & \text{for all } \ x \in K^*\,. \end{array}$$

Adding these two inequalities, we conclude that $\bar{f}_k^{\mathrm{a}}(x) \leq f_k^{\mathrm{m}}(x)$ for all $x \in K^*$. The stated result then holds because $f_k^{\mathrm{m}} \leq f$ by Lemma 2.3. □

3.2 Bundle management and the master problem

Given a subproblem i, we have as defined in (14) the following affine underestimators of f^i:

$$\bar{f}_j^i(x) := f_{x_j}^i + \langle g_{x_j}^i, x - x_j\rangle \quad \text{for all } \ j \in \mathcal{B}_k^i\,. \tag{18}$$

Using this notation and recalling (11), one may express the subproblem (15) in the following more concrete manner:

$$\left\{ \begin{array}{lrcll} \min_{x,r} & \sum_{i=1}^m r^i + \frac{1}{2t_k}\|x - \hat{x}_k\|^2 & & & \\ \text{s.t.} & \bar{f}_j^i(x) & \leq & r^i & \forall\, j \in \mathcal{B}_k^i,\ i = 1,\ldots,m \\ & x & \in & K^*, & \\ & r & \in & \mathbb{R}^m. & \end{array} \right. \tag{19}$$

We call this the *master problem*. Because it has a strongly convex objective function, the master problem must have a unique solution. If the cone K^* is polyhedral, then the master problem is a convex quadratic program with linear constraints, and we may expect to be solved relatively efficiently with a variety of standard software packages.

It is clear that the larger one makes the bundles $\mathcal{B}_k^i$, $i = 1,\ldots,m$, the more accurate becomes the disaggregate cutting plane model f_k^{m} in (11). To maximize the accuracy, we may use the simple bundle

management rule of keeping as much information as possible when passing from the model f_k^{m} to f_{k+1}^{m}, as follows:

$$\mathcal{B}_{k+1}^i = \begin{cases} \mathcal{B}_k^i \cup \{k+1\} & \text{if } i \in L_{k+1} \\ \mathcal{B}_k^i & \text{otherwise.} \end{cases} \tag{20}$$

Since this rule constantly increases the number of constraints in problem (19), it makes the task of finding the next iterate x_{k+1} progressively more time consuming. However, the computational burden necessary to solve the convex subproblem (19) might be negligible when compared to the CPU time required by the oracle presented in Algorithm 1, which may require solving up to m complex MIP subproblems. In some situations, however, the time required to solve the master problem might become an algorithmic bottleneck, or a large number of "similar" constraints in (19) might lead to numerical difficulties. In such situations, one may apply a more economical rule that consists of keeping in the bundles $\mathcal{B}_k^i$ only active linearizations $\bar{f}_j^i$. A linearization $\bar{f}_j^i$ is called *active* at iteration k if $\bar{f}_j^i(x_{k+1}) = r_{k+1}^i$, where (x_{k+1}, r_{k+1}) is the unique solution of (19). Letting $\mathcal{B}_k^{i,\mathrm{act}}$ denote the indices in $\mathcal{B}_k^i$ which are active in (19), an alternative bundle management rule is

$$\mathcal{B}_{k+1}^i = \begin{cases} \mathcal{B}_k^{i,\mathrm{act}} \cup \{k+1\} & \text{if } i \in L_{k+1} \\ \mathcal{B}_k^{i,\mathrm{act}} & \text{otherwise.} \end{cases} \tag{21}$$

We will establish convergence of our method for any choice of $\mathcal{B}_{k+1}^i \subseteq \{1, \ldots, k+1\}$ that contains the one in (21), that is, whenever

$$\mathcal{B}_{k+1}^i \supseteq \mathcal{B}_k^{i,\mathrm{act}} \quad \text{and} \quad i \in L_{k+1} \Rightarrow k+1 \in \mathcal{B}_{k+1}^i. \tag{22}$$

Even if one eliminates inactive linearizations at each iteration, it is still conceivable that the number of constraints in (19) may become too large in some cases. Consequently, Section 3.5 below will discuss a variant of the algorithm in which $\mathcal{B}_{k+1}^i$ does not have to contain $\mathcal{B}_k^{i,\mathrm{act}}$, at the cost of having to adjoin an additional aggregate linearization constraint derived from (17) into the subproblem (19). Such strategies are called *bundle compression*. First, however, we will prove convergence of our method using bundle update rules of the form (22) and master problems of the form (19).

We now briefly consider the form of the master problem in the case of the stochastic programming application described in Section 1.1. In this case, $K^* = V^\perp$, where V is the linear subspace defined by (2). The dual variables x will be partitioned in the same manner as the variables $(z^1, \ldots, z^m) \in V$, and we have

$$K^* = V^\perp = \left\{ (x^1, \ldots, x^m) \in \mathbb{R}^n \;\middle|\; \textstyle\sum_{i \in S} x^{it} = 0 \; \forall\, S \in \mathcal{S}^t,\, t = 1, \ldots, T-1 \right\}. \tag{23}$$

Referring to the subproblems (5a) and using the particular form of the g^i in (4), it is clear that subproblem i depends only on the x^i portion of x, and the subgradient with of f^i with respect to the portion of x outside x^i is always zero. Therefore, the constraints $\bar{f}_j^i(x) \le r^i$ in the master problem (33) separate into m groups that have no mutual variables. Thus, the entire problem (33) separates into m noninteracting blocks, except for the constraint $x \in K^*$, which by (23) becomes in this case

$$\sum_{i \in S} x^{it} = 0 \quad \forall\, S \in \mathcal{S}^t,\, t = 1, \ldots, T-1.$$

This structure could allow for various strategies for applying parallel computing to the master problem, should its solution become a bottleneck in the algorithm. Using an aggregate cutting plane model instead of a disaggregate one would produce a smaller master problem, but without this potentially advantageous block structure. This is another reason that a disaggregate cutting-plane model appears most suitable for the applications we have in mind.

3.3 Optimality certificate and algorithm

Before explicitly stating our complete algorithm, the following proposition formulates an optimality certificate depending on the subgradient vectors $\hat{g}_k$ defined in Lemma 3.1:

Proposition 3.2 (**Optimality certificate**). *Let $\delta_k^{\mathrm{m}} = f_{\hat{x}_k} - f_k^{\mathrm{m}}(x_{k+1})$ be the predicted decrease, and $\hat{\varepsilon}_k$ be a bound on the oracle error at the serious iterate $\hat{x}_k$. Moreover, let x_{k+1} and $\hat{g}_k$ be as given in* (15)-(16)*, and define*

$$\hat{e}_k := f_{\hat{x}_k} - \bar{f}_k^{\mathrm{a}}(\hat{x}_k) \quad \textit{and} \quad \phi_k := \hat{e}_k + \langle \hat{x}_k, \hat{g}_k \rangle\,. \tag{24}$$

Then $\hat{e}_k \geq -\hat{\varepsilon}_k$ and δ_k^{m} can be alternatively written as $\delta_k^{\mathrm{m}} = \hat{e}_k + t_k\|\hat{g}_k\|^2$. Moreover, the following inequality holds:

$$f_{\hat{x}_k} \leq f(x) + \phi_k - \langle \hat{g}_k, x\rangle \qquad \forall\, x \in K^*. \tag{25}$$

Further, suppose that there exists an infinitely large index set $\mathcal{I} \subseteq \mathbb{N}$ such that

$$\liminf_{k\in\mathcal{I}} \phi_k \leq 0 \quad \textit{and} \quad \lim_{k\in\mathcal{I}} \hat{g}_k = 0\,. \tag{26}$$

Then, every cluster point (if any) of the subsequence $\{\hat{x}_k\}_{k\in\mathcal{I}}$ is a $\hat{\varepsilon}_{\mathcal{I}}$-solution to problem (5), where we define $\hat{\varepsilon}_{\mathcal{I}} := \liminf_{k\in\mathcal{I}} \hat{\varepsilon}_k$,

Proof. We prove the first claim by noting that

$$\hat{e}_k = f_{\hat{x}_k} - \bar{f}_k^a(\hat{x}_k) = f_{\hat{x}_k} - \big(f_k^{\mathrm{m}}(x_{k+1}) + \langle \hat{g}_k, \hat{x}_k - x_{k+1}\rangle\big) = \delta_k^{\mathrm{m}} - t_k\langle \hat{g}_k, \hat{g}_k\rangle\,,$$

where the second equality follows from (17) and the third one from (16).

Lemma 3.1 ensures that $f(x) \geq \bar{f}_k^a(x)$ for all $x \in K^*$. Then, by (9) and the definition of $\hat{\varepsilon}_k$ we have that $\hat{e}_k = f_{\hat{x}_k} - \bar{f}_k^a(\hat{x}_k) \geq f_{\hat{x}_k} - f(\hat{x}_k) \geq f(\hat{x}_k) - \hat{\varepsilon}_k - f(\hat{x}_k) = -\hat{\varepsilon}_k$, establishing the second claim. Moreover,

$$\begin{aligned} f(x) &\geq \bar{f}_k^a(x) = f_k^{\mathrm{m}}(x_{k+1}) + \langle \hat{g}_k, x - x_{k+1}\rangle \\ &= f_{\hat{x}_k} - \big(f_{\hat{x}_k} - f_k^{\mathrm{m}}(x_{k+1})\big) + \langle \hat{g}_k, x - \hat{x}_k\rangle + \langle \hat{g}_k, \hat{x}_k - x_{k+1}\rangle \\ &= f_{\hat{x}_k} - \delta_k^{\mathrm{m}} + \langle \hat{g}_k, x - \hat{x}_k\rangle + \langle \hat{g}_k, t_k\hat{g}_k\rangle \\ &= f_{\hat{x}_k} - \hat{e}_k - \langle \hat{g}_k, \hat{x}_k\rangle + \langle \hat{g}_k, x\rangle = f_{\hat{x}_k} - \phi_k + \langle \hat{g}_k, x\rangle\,, \end{aligned}$$

where we have used (16) in the third equality, and (24) in the two last equalities. We have thus proved (25). Finally, suppose there exists some infinite set of indices $\mathcal{I}$ as hypothesized. Combining (25) with $\hat{\varepsilon}_k$ being a bound on the oracle error at $\hat{x}_k$, we obtain that $f(\hat{x}_k) - \hat{\varepsilon}_k \leq f_{\hat{x}_k} \leq f(x) + \phi_k - \langle \hat{g}_k, x\rangle$ for all $x \in K^*$, which we may rearrange into $f(\hat{x}_k) \leq f(x) + \phi_k - \langle \hat{g}_k, x\rangle + \hat{\varepsilon}_k$ for all $x \in K^*$. Taking the $\liminf$ over $k \in \mathcal{I}$ in this inequality, we conclude that $\liminf_{k\in\mathcal{I}} f(\hat{x}_k) \leq f(x) + \hat{\varepsilon}_{\mathcal{I}}$ for all $x \in K^*$. Now, since $\{f(\hat{x}_k)\}$ is a nonincreasing sequence, it must converge to its $\liminf$, so we can strengthen the previous statement to $\lim_{k\to\infty} f(\hat{x}_k) \leq f(x) + \hat{\varepsilon}_{\mathcal{I}}$ for all $x \in K^*$. Finally, if $\hat{x}$ is any limit point of $\{\hat{x}_k\}$, the continuity of f and the convergence of $\{f(\hat{x}_k)\}$ guarantee that $f(\hat{x}) = \lim_{k\to\infty} f(\hat{x}_k)$ and thus that $f(\hat{x}) \leq f(x) + \hat{\varepsilon}_{\mathcal{I}}$ for all $x \in K^*$. □

As we have just shown, the quality of the solution estimates $\hat{x}_k$ generated by our algorithm depends only on the oracle errors at serious iterates. Hence, if the oracle is eventually exact at serious iterates, an exact solution to problem (5) can be identified if we can guarantee the validity of condition (26).

Corollary 3.3. *Suppose that (26) holds for some infinite $\mathcal{I} \subseteq \mathbb{N}$. In addition,*

(a) Suppose that only finitely many serious iterates are generated. let $\hat{x} := x_j$ be the last serious iterate, calculated at iteration $j-1$. Moreover, let $\hat{\varepsilon}_{\mathcal{I}} = \sum_{i=1}^m \varepsilon_j^i$, where $\{\varepsilon_j^i\}$ are the oracle error bounds passed as input to Algorithm 1 at iteration j. Then $\hat{x}$ is $\hat{\varepsilon}_{\mathcal{I}}$-solution to problem (5).

(b) If infinitely many serious descent iterates are generated and $\lim_k \hat{\varepsilon}_k = 0$, then every cluster point of $\{\hat{x}_k\}_{k\in\mathcal{I}}$ is an exact solution to problem (5).

We are now in position to present our disaggregate proximal bundle method in full detail, as shown in Algorithm 2.

In the algorithm, note that the equivalence of the master problem formulations (15) and (19) ensures that $f_k^{\mathrm{m}}(x_{k+1}) = \sum_{i=1}^m \bar{r}^i$ in Step 1. Therefore, the definition of the predicted decrease δ_k^{m} in Step 1 is equivalent to $\delta_k^{\mathrm{m}} = f_{\hat{x}_k} - f_k^{\mathrm{m}}(x_{k+1})$.

One important feature of the algorithm we have not yet discussed is noise attentuation. With an exact oracle, the aggregate error $\hat{e}_k$ defined in (24) satisfies $\hat{e}_k \geq 0$ for all k, and therefore $\delta_k^{\mathrm{m}} = \hat{e}_k + t_k\|\hat{g}_k\|^2 \geq 0$. However, when the oracle is inexact, the aggregate error and predicted decrease may in some cases be negative. Without some kind of safeguard, such a negative predicted decrease would make it possible to pass the "descent" test in line 21 despite a worstened objective estimate $f_{x_{k+1}} > f_{x_k}$. In order to prevent this situation, we use the technique proposed in [19], which increases the proximal step t_k whenever one has, as in line 14 of the algorithm,

$$\hat{e}_k < -\beta t_k\|\hat{g}_k\|^2 \quad \text{ for some fixed } \beta \in (0,1)\,. \tag{27}$$

Algorithm 2 Disaggregate proximal bundle algorithm for lower inexact oracles

Step 0: Initialization
Choose $x_1 \in K^*$, $\kappa, \beta \in (0,1)$, $t_{\min} > 0$, $c > 1$, and stopping tolerances $\texttt{Tol}_{\hat{g}}, \texttt{Tol}_{\phi} > 0$
Set $\hat{x}_1 \leftarrow x_1$ and call Algorithm 1 with $\gamma_1 = \infty$ to compute $f_{\hat{x}_1}$ and $(f^i_{x_1}, g^i_{x_1})$ for $i = 1, \ldots, m$.
Set $\mathcal{B}^i_1 \leftarrow \{1\}$ for all $i = 1, \ldots, m$, $k \leftarrow 1$ and $\texttt{na} = 0$ ($\texttt{na}$ is the noise attenuation flag).
for $k = 1, 2, \ldots$ **do**
    Step 1: Solve master problem to obtain new iterate
    Solve (19) to obtain the optimal solution x_{k+1} and $\bar{r}^i$, $i = 1, \ldots, m$
    $\delta^{\mathrm{m}}_k \leftarrow f_{\hat{x}_k} - \sum_{i=1}^m \bar{r}^i$, $\hat{g}_k \leftarrow (\hat{x}_k - x_{k+1})/t_k$, $\hat{e}_k \leftarrow \delta^{\mathrm{m}}_k - t_k \|\hat{g}_k\|^2$, $\phi_k \leftarrow \hat{e}_k + \langle \hat{x}_k, \hat{g}_k \rangle$
    Step 2: Stopping test
    if $\|\hat{g}_k\| \le \texttt{Tol}_{\hat{g}}$ and $\phi_k \le \texttt{Tol}_{\phi}$ **then**
        Stop: return $\hat{x}_k$ and $f_{\hat{x}_k}$
    end if
    Step 3: Noise attenuation test
    if $\hat{e}_k < -\beta t_k \|\hat{g}_k\|^2$ **then**
        Set $\texttt{na} \leftarrow 1$, $t_{k+1} \leftarrow c\, t_k$, $\hat{x}_{k+1} \leftarrow \hat{x}_k$ ▷ Noise buildup, repeat master with larger t_k
    else ▷ Otherwise, continue to evaluate new iterate
        Step 4: Oracle calls
        Define $\gamma_k \leftarrow f_{\hat{x}_k} - \kappa \delta^{\mathrm{m}}_k$ and choose the bounds $\varepsilon^i_k \ge 0$, $i = 1, \ldots, m$
        Call Algorithm 1 to compute $f_{x_{k+1}}$, L_{k+1}, and $(f^i_{x_{k+1}}, g^i_{x_{k+1}})$ for each $i \in L_{k+1}$
        Step 5: Descent test and next serious iterate
        if $f_{x_{k+1}} \le \gamma_k$ **then** ▷ Serious step
            Set $\hat{x}_{k+1} \leftarrow x_{k+1}$, $\texttt{na} \leftarrow 0$ and choose $t_{k+1} \ge t_k$
        else ▷ Null step
            Set $\hat{x}_{k+1} \leftarrow \hat{x}_k$ and choose $t_{k+1} \in [(1 - \texttt{na})t_{\min} + (\texttt{na})t_k,\, t_k]$
        end if
        Step 6: Bundle management
        Update the bundles $\mathcal{B}^i_{k+1}$, $i = 1, \ldots, m$, in any manner that satisfies (22).
    end if
end for

Lemma 3.4. [5, Corollary 5.3] *Suppose that, starting at the value $k = \hat{k}$ of the iteration counter, Algorithm 2 loops infinitely with the condition $\hat{e}_k < -\beta t_k \|\hat{g}_k\|^2$ at line 14 holding true, with steps 4-6 never again being executed. Then condition (26) is satisfied for $\mathcal{I} = \mathbb{N}$ and the last serious iterate $\hat{x} := \hat{x}_{\hat{k}}$ is an $\hat{\varepsilon}$-solution to the problem of minimizing f.*

We omit the proof of this result, since it is given in [5]; however, we remark that it depends crucially on (27). Whenever step 4 is reached at iteration k, then the test in step 3 ensures that the predicted decrease is nonnegative, because it implies

$$\delta_k^{\mathrm{m}} = \hat{e}_k + t_k \|\hat{g}_k\|^2 \geq (1-\beta) t_k \|\hat{g}_k\|^2 \geq 0\,, \tag{28}$$

making line 21 of the algorithm a true descent test. As a result, the sequence $\{f_{\hat{x}_k}\}$ generated by Algorithm 2 is nonincreasing.

Since Algorithm 1 initializes $f_{L_{k+1}}$ by using the model value $f_k^{\mathrm{m}}(x_{k+1})$, the test $f_{L_{k+1}} \leq \gamma_k$ in the first iteration of Algorithm 1 becomes $f_k^{\mathrm{m}}(x_{k+1}) \leq f_{\hat{x}_k} - \kappa \delta_k^{\mathrm{m}}$, which is equivalent to

$$0 \leq f_{\hat{x}_k} - f_k^{\mathrm{m}}(x_{k+1}) - \kappa \delta_k^{\mathrm{m}} = (1-\kappa)\delta_k^{\mathrm{m}}\,. \tag{29}$$

The inequality (29) must hold because $\kappa \in (0,1)$ and (28) is satisfied. Therefore, Algorithm 1 must enter its inner loop and solve the subproblem (5a) for at least one index i. As a result, the set of subproblem indices L_{k+1} in Algorithm 1 must be nonempty and at least one new linearization will be introduced into the subproblem (19) at each iteration k.

As already mentioned, Step 4 may not solve all m subproblems (5a) for a some candidate solutions x_{k+1}. If x_{k+1} is identified to be a null step, Step 4 of the algorithm is interrupted and the approximate value $f_{x_{k+1}}$ may be only a rough estimate of the exact value $f(x_{k+1})$. However, the error $f(x_k) - f_{x_k}$ is locally bounded in the following sense:

Remark 1 (Locally boundedness)**.** *Algorithms 1 and 2 ensure that*

i. the nonnegative sequence of errors $\{f(\hat{x}_k) - f_{\hat{x}_k}\}$ is bounded by $\{\hat{\varepsilon}_k\}$, with $\hat{\varepsilon}_k = \sum_{i=1}^m \varepsilon_{j(k)}^i$, where $j(k)$ is the index of the iteration first resulting in the serious iterate $\hat{x}_k = x_{j(k)}$;

ii. the nonnegative sequence of errors $\{f(x_k) - f_{x_k}\}$ is bounded if $\{x_k\}$ is a bounded sequence, as shown in Lemma 2.5.

The efficiency of Algorithm 2 depends on how one updates the proximal stepsize t_k. One efficient rule for managing t_k is proposed in [21]. Typically, the value of the parameter c is quite large, most commonly $c = 10$. When dealing with inexact oracles, the noise attenuation step thus sharply increases t_k whenever the oracle error is detected to be excessive, *i.e.*, (27) holds. In this case, the "flag" variable `na` denoting *noise attenuation* is set to 1. Referring to line 24 of the algorithm, this setting prevents any decrease in t_k in Step 5 until a new serious step occurs. This property is crucial to the convergence analysis [19, 5].

3.4 Convergence result

Algorithm 2 resembles the inexact proximal bundle algorithm introduced in [19], but differs in its: (i) cutting-plane model definition f_k^{m}; (ii) management of oracles calls in Step 4 (see Algorithm 1); (iii) assumptions regarding the sequence of errors $\{f(x_k) - f_{x_k}\}$. The algorithms presented in [19, 8] assume that the error $f(x) - f_x$ is bounded for all $x \in K^*$. The analysis in [5] extends the results of [19, 8] and considers the following weaker assumption on the errors (see Equation (6.8) in [5]):

$$\forall\, R \geq 0,\, \exists\, \eta(R) \geq 0 \;\text{ such that }\; \|x\| \leq R \;\Rightarrow\; f(x) - f_x \leq \eta(R)\,. \tag{30}$$

As noted in [5, Remark 6.5], this assumption can be replaced by the following:

$$\text{If there is a last serious step } \hat{k},\ \text{then } \exists\, E \geq 0 : f(x_k) - f_{x_k} \leq E \;\; \forall\, k > \hat{k}. \tag{31}$$

In other words, if the algorithm executes an infinite string of null steps, the oracle error remains bounded over those steps. This turns out to be a weaker assumption than (30). In what follows, we will show that Algorithm 2 (combined with Algorithm 1) satisfies (31) by recalling Remark 1 and by showing that any infinite string of null steps must be bounded; this latter result can also be used to show that (30) implies (31) and thus that (31) is indeed the weaker of the two conditions.

Lemma 3.5. *Consider Algorithm 2 applied to problem* (5) *and assume that all bounds* $\varepsilon_k^i \geq 0$ *for the oracle errors provided to Algorithm 1 are bounded: there exists* $E \geq 0$ *such that* $\varepsilon_k^i \leq E$ *for all* $i = 1, \ldots, m$ *and* $k \geq 0$. *Suppose (as already assumed above) that the sets* P^i *in* (5) *are compact and maps* g^i *are continuous. Moreover, assume that there exists a last serious iterate* $\hat{x}$ *identified at iteration* $\hat{k}$, *and that the sequence of prox-parameters* $\{t_k\}_{k>\hat{k}}$ *is nonincreasing. Then the sequence of null iterates* $\{x_k\}_{k>\hat{k}}$ *is bounded and property* (31) *holds true.*

Proof. Under these hypotheses, $\hat{x}_k = \hat{x}$ for all $k \geq \hat{k}$. Since the sequence $\{t_k\}$ is assumed to be nonincreasing, we may use the results of [5, Lemma 6.3], where the objective function of (15) is denoted by

$$f_k^{\mathrm{S}}(x) := f_k^{\mathrm{m}}(x) + \tfrac{1}{2t_k}\|x - \hat{x}\|^2 .$$

As Step 6 of the algorithm manages the bundle in order to satisfy (22), Lemma 3.3 of [19] (see also Lemma 6.3(i) of [5]) shows that the sequence $\{f_k^{\mathrm{S}}(x_{k+1})\}$ is increasing. Therefore, there exists some constant $C < \infty$ such that $-C \leq f_k^{\mathrm{S}}(x_{k+1})$ for all $k \geq \hat{k}$. We note that since the model f_k^{m} given in (11) approximates f from below, [5, Equation (4.11)] holds true for $\eta^{\mathrm{M}} = 0$, Lemma 6.3(ii) in [5] then demonstrates that (because $\eta^{\mathrm{M}} = 0$ therein)

$$\tfrac{1}{2t_k}\|x_{k+1} - \hat{x}\|^2 \leq f(\hat{x}) - f_k^{\mathrm{S}}(x_{k+1}) \leq f(\hat{x}) + C < \infty \quad \text{for all } k \geq \hat{k},$$

showing that $\{x_k\}_{k\geq\hat{k}}$ is a bounded sequence (because $\{t_k\}_{k>\hat{k}}$ is a nonincreasing by hypothesis). Since ε_k^i, $i = 1, \ldots, m$, in Algorithm 1 are bounded, Lemma 2.5 ensures that (31) is satisfied. □

Since (31) holds true, we are in the setting of [5]. In the proof below, we will rely on analysis from [5] to prove the convergence of Algorithm 2. Although the analysis in [5] focuses on the unconstrained case $K^* = \mathbb{R}^n$ in (5), its results also apply when the constraint $x \in K^* \neq \mathbb{R}^n$ is included the master problem, as in (19); see [5, § 7.3] for more information.

Theorem 3.6. *Consider problem* (5) *and assume that* $f^{\inf} > -\infty$ *and all bounds* $\varepsilon_k^i \geq 0$ *for the oracle errors provided to Algorithm 1 are (globally) bounded. Suppose that the tolerances* $\texttt{Tol}_{\hat{g}}$, $\texttt{Tol}_{\phi}$ *in Algorithm 2 are set to zero. Then, if the algorithm halts at iteration* k, *we have that* $f_{\hat{x}_k} \in [f^{\inf} - \hat{\varepsilon}_k, f^{\inf}]$. *Otherwise, the algorithm generates a sequence of iterates satisfying* (26) *for some infinite index set* $\mathcal{I} \subseteq \mathbb{N}$. *In this case, the following hold, where* $\hat{\varepsilon} = \liminf_{k\to\infty} \hat{\varepsilon}_k$ *as defined above:*
– *We have* $f_\infty := \lim_{k\to\infty} f_{\hat{x}_k}$ *satisfies* $f_\infty \in [f^{\inf} - \hat{\varepsilon}, f^{\inf}]$.
– *Every cluster point (if any) of the subsequence* $\{\hat{x}_k\}$ *is a* $\hat{\varepsilon}$*-solution to problem* (5).

Proof. First, we consider the case in which the algorithm halts. Since $\texttt{Tol}_{\hat{g}} = \texttt{Tol}_{\phi} = 0$ by assumption, the stopping test in Step 2 of the algorithm ensures that $\phi_k \leq 0$ and $\|\hat{g}_k\| = 0$. From (25), we immediately conclude that $f_{\hat{x}_k} \leq f(x)$ for all $x \in K^*$, that is, $f_{\hat{x}_k} \leq f^{\inf}$. On the other hand, we also know that $f_{\hat{x}_k} \geq f(\hat{x}_k) - \hat{\varepsilon}_k \geq f^{\inf} - \hat{\varepsilon}_k$. This completes the proof for the finite case.

From now on, we thus assume that the algorithm generates infinitely many iterates. We consider the following possible cases:

(a) the algorithm loops forever in noise attentuation mode, with only a finite number of invocations of the oracle;

(b) every string of noise-attenuation iterations is finite, meaning that Algorithm 1 is called an infinite number of times

  (b.1) infinitely many serious iterates are generated;

  (b.2) only finitely many serious iterates are generated and $\{t_k\}$ is unbounded;

  (b.3) the same as (b.2), but $\{t_k\}$ is bounded.

In each case, we will establish that (26) holds for some sequence of indices $\mathcal{I}$. In case (a), Lemma 3.4 immediately shows that (26) holds for $\mathcal{I} = \mathbb{N}$, so we proceed to case (b).

Since the oracle errors are bounded at serious iterates as noted Remark 1(i), case (b.1) follows from standard analysis of inexact proximal bundle methods under the assumption that $f^{\inf} > -\infty$; see for instance [19, Lemma 3.8] or Propositions 6.1 and 6.9 in [5].

Let us now consider cases (b.2) and (b.3), in both of which there exists a last serious iterate $\hat{x} = \hat{x}_{\hat{k}}$ first identified at iteration $\hat{k}$. Note that $\hat{x}_k = \hat{x}$ for all $k \geq \hat{k}$. Since Step 5 of Algorithm 2 does not increase t_k at null steps, the only manner in which one can have an unbounded sequence of proximal stepsizes $\{t_k\}$ as stipulated in case (b.2) is if there are infinitely many noise attenuation steps in Step 3 of the algorithm. Since the noise attenuation flag is only cleared at serious steps, there therefore exists a index $k_1 \geq \hat{k}$ such the noise attenuation flag `na` will remain fixed at 1 for all $k \geq k_1$. In this case, the sequence $\{t_k\}_{k \geq k_1}$ is monotone nondecreasing by Step 5 and having an infinite number of noise attenuation steps means that $t_k \to \infty$. Let $\mathcal{K} \subset \mathbb{N}$ be the set of indices of iterations at which noise attenuation occurs. Now, $\lim_{k \in \mathcal{K}} t_k = \infty$ and (27) imply by the boundedness of $\hat{\varepsilon}_k$ that $\limsup_{k \in \mathcal{K}} \hat{e}_k \leq 0$ and $\lim_{k \in \mathcal{K}} \hat{g}_k = 0$. In this case, (24) becomes $\phi_k = \hat{e}_k + \langle \hat{x}, \hat{g}_k \rangle$ for $k \geq \hat{k}$, and (26) holds true for the choice $\mathcal{I} = \mathcal{K}$.

Finally, let us consider case (b.3). In this case the sequence $\{t_k\}_{k > \hat{k}}$ is nonincreasing and thus Lemma 3.5 applies: the sequence of function estimation errors is bounded. As an result, Theorem 6.4 and Remark 6.5 in [5] provide the following useful inequality

$$\limsup_{k > \hat{k}} \left\{ f_{x_k} - f_{k-1}^{\mathrm{m}}(x_k) \right\} \leq 0 \,.$$

Since $f_{x_k} > \gamma_{k-1}$ for $k > \hat{k}$ (all subsequent steps being null steps), the above inequality yields

$$0 \geq \limsup_{k > \hat{k}} \left[\gamma_{k-1} - f_{k-1}^{\mathrm{m}}(x_k) \right] = \limsup_{k > \hat{k}} \left[f_{\hat{x}} - \kappa \delta_{k-1}^{\mathrm{m}} - f_{k-1}^{\mathrm{m}}(x_k) \right] = (1 - \kappa) \limsup_{k > \hat{k}} \, \delta_{k-1}^{\mathrm{m}} \,.$$

By (28) and $t_k \geq t_{\min} > 0$ we conclude that $\limsup_{k > \hat{k}} \hat{e}_k \leq 0$ and $\limsup_{k > \hat{k}} \|\hat{g}_k\| = 0$. Given the definition $\phi_k = \hat{e}_k + \langle \hat{x}, \hat{g}_k \rangle$ for $k \geq \hat{k}$, (26) holds for $\mathcal{I} = \mathbb{N}$.

We have now shown that a suitable index set $\mathcal{I}$ exists in all cases in which the algorithm runs for an infinite number of iterations. The remainder of the proof follows from the oracle relation $f_{\hat{x}_k} \in [f(\hat{x}_k) - \hat{\varepsilon}_k, f(\hat{x}_k)]$, monotonicity of the sequence $\{f_{\hat{x}_k}\}$, and Proposition 3.2 (see also Theorem 4.5 in [5]). □

We now briefly consider the case that the oracle error ε_k^i is set to zero for all subproblems i and iterations k. In this case Algorithm 2 is a variant of the partially inexact bundle method proposed in [15], except that our method uses a disaggregate cutting-plane model, while [15] uses an aggregate model. Despite having all the tolerances ε_k^i set to zero, our algorithm is partially inexact because its oracle management subroutine (Algorithm 1) may terminate without evaluating every possible subproblem in the $f_{x_{k+1}} > \gamma_k$ case (which necessarily leads to a null step). Corollary 3.3(ii) ensures that, under the assumptions of Theorem 3.6, Algorithm 2 finds an exact solution to problem 5 even though the function and subgradient evaluations may be inexact for null iterates. Moreover, since the aggregate linearization error $\hat{e}_k$ is nonnegative ($\hat{e}_k \geq -\hat{\varepsilon}_k = 0$), inequality (27) never holds and consequently, noise attenuation never occurs, that is, $\mathtt{na} = 0$ throughout the Algorithm 2. Further information on partially inexact proximal bundle method can be found in [15] and in [4, Section 7.1.1].

Remark 2. *If Algorithm 2, as it is currently stated, enters an infinite string of null or noise attentuation steps starting at iteration k, there is no opportunity to improve its accuracy beyond $\hat{\varepsilon}_k$. Although such situations do not appear to be of practical concern, one could easily remedy them by inserting an optional, additional oracle call after Step 6, in which one invokes the oracle again at $\hat{x}_k$ with $\gamma_k = f_{\hat{x}_k}$ and smaller values of the ε_k^i.*

3.5 Bundle compression

As already shown in Lemma 2.3, the aggregate cutting-plane model f_k^{m} given in (11) can be expected to provide a better model of the true objective function f than the aggregate model (10). However, there is price to pay for this improvement: the model f_k^{m} may require more storage than the aggregate one $\check{f}_k$. The bundle management strategy (21) is an attempt to keep the master problem (15) easy to solve. Although very useful in practice, this strategy does not guarantee that all the bundles $\mathcal{B}_k^i$, $i = 1, \ldots, m$, remain of bounded size. In other words, Algorithm 2, which as stated always makes $\mathcal{B}_k^i$ at least as big as specified in rule (21), might not be a limited-memory algorithm. In order to make Algorithm 2 a limited-memory algorithm, the bundles $\mathcal{B}_k^i$ may need to be compressed by dropping active constraints at some iterations, as

described below. We now show how convergence can still be guaranteed even when active constraints are dropped.

The convergence analysis given in [5], which applies to Algorithm 2 as showed in Theorem 3.6, considers a very general model f_k^{M} approximating the objective function. In [5], the model only needs to satisfy the following inequalities

$$\max\{\bar{f}_{k+1}(x),\, \bar{f}_k^{\mathrm{a}}(x)\} \leq f_{k+1}^{\mathrm{M}}(x) \leq f(x) + \eta^M\,, \tag{32}$$

where $\bar{f}_{k+1}(x) = f_{x_{k+1}} + \langle g_{x_{k+1}}, x - x_{k+1}\rangle$ is the natural linearization of f at iteration x_{k+1}, $\bar{f}_k^{\mathrm{a}}$ is the aggregate linearization given in (17), and η^M is a constant. In our setting, $\eta^M = 0$ because the oracle is of the lower type. If f_{k+1}^{M} is the disaggregate model f_{k+1}^{m} with bundle management strategy (22), then the inequalities in (32) hold true (with $\eta^M = 0$). This claim follows from [3, Remark 4.2] by recalling (13)-(14) and that Step 6 of Algorithm 2 yields $k+1 \in \mathcal{B}_{k+1}^i$ for all $i \in L_{k+1}$ and $\mathcal{B}_k^{i,\mathrm{act}} \subseteq \mathcal{B}_{k+1}^i$ for all $i = 1, \ldots, m$.

However, we can still satisfy (32) using much smaller bundles than $\mathcal{B}_k^{i,\mathrm{act}}$. One possible approach to satisfying (32) is the following:

- Choose the bundles $\mathcal{B}_k^i$ in such a manner that $\sum_{i=1}^m f_{k+1}^{\mathrm{m}}(x) \geq \bar{f}_{k+1}(x)$ for all $x \in K^*$ and $k \geq 0$.
- Take $f_{k+1}^{\mathrm{M}}(x) := \max\{f_{k+1}^{\mathrm{m}}(x), \bar{f}_k^{\mathrm{a}}(x)\}$.

The first condition above may be satisfied by requiring that $\mathcal{B}_{k+1}^i \ni k+1$ if $i \in L_{k+1}$, and $\mathcal{B}_{k+1}^i \ni j(i,k+1)$ otherwise, where we recall that $j(i,k+1)$ is any index maximizing $f_{x_j}^i + \langle g_{x_j}^i, x_{k+1} - x_j\rangle$ over $j \in \mathcal{B}_k^i$. Thus, we could have bundles as small as a single element. Using the model function $f_{k+1}^{\mathrm{M}}(x) = \max\{f_{k+1}^{\mathrm{m}}(x), \bar{f}_k^{\mathrm{a}}(x)\}$, the master problem becomes

$$\left\{\begin{array}{lrcll} \min_{x,r} & \sum_{i=1}^m r^i + \frac{1}{2t_k}\|x - \hat{x}_k\|^2 & & & \\ \text{s.t.} & \bar{f}_j^i(x) & \leq & r^i & \forall\, j \in \mathcal{B}_k^i,\ i = 1, \ldots, m \\ & \bar{f}_{k-1}^{\mathrm{a}}(x) & \leq & \sum_{i=1}^m r^i & \\ & x & \in & K^*,\ r \in \mathbb{R}^m. & \end{array}\right. \tag{33}$$

Defining $j(i,k) = k$ in the case that $i \in L_k$, then according to the convergence analysis in [5], Theorem 3.6 continues to hold so long as $\mathcal{B}_k^i \ni j(i,k)$. This means that there could be as few as $m+1$ constraints in the model (33), not counting any required to express $x \in K^*$. Since the number of these latter constraints is presumably fixed, we thus have the possibility of placing a fixed upper bound on the number of constraints in the master problem, making our method a limited-memory algorithm. Furthermore, if we do have $\mathcal{B}_k^i \supseteq \mathcal{B}_k^{i,\mathrm{act}}$ for all $i = 1, \ldots, m$ at some iteration k, then the constraint $\bar{f}_{k-1}^{\mathrm{a}}(x) \leq \sum_{i=1}^m r^i$ in (33) is redundant, and may be dropped for that iteration.

While theoretical convergence is not impaired by employing such bundle compression strategies, it has been observed that frequent bundle compression reduces the empirical speed of convergence.

4 Conclusions

In this work we have proposed a bundle method for solving difficult optimization problems with additive structure. The method is of the *on-demand accuracy type* and allows to abandon null iterates without fully evaluating their objective values. Our methods function estimates at such abandoned points may be very imprecise, but have shown that such inaccuracies do not impair the quality of the algorithm's solution estimates. In particular, if function evaluations are asymptotically exact at serious steps, then our proposed algorithm is able to obtain a asymptotically exact solution estimates for the underlying optimization problem.

The proposed algorithm is in the bundle method family, with a number of features that have essentially all appeared in prior proposed algorithms, but not in the particular combination required by our applications. The principal applications we have in mind are stochastic integer programs, as applied to challenging problems such as multistage electrical generation system unit commitment planning in the presence of relatively unpredictable renewable power sources.

Numerical experiences with bundle methods employing the on-demand accuracy ideas have been encouraging [4, 32], although these methods require that oracle errors be uniformly bounded throughout the iterative process. Since our approach does not impose this assumption on null steps, computational effort on unpromising trial points can be reduced and therefore a even better performance may be expected. Research of computational applications is ongoing.

References

[1] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical Optimization: Theoretical and Practical Aspects*, Springer-Verlag, 2nd ed., 2006.

[2] A. Borghetti, A. Frangioni, F. Lacalandra, and C. Nucci, *Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment*, IEEE Transactions on Power Systems, 18 (2003), pp. 313–323.

[3] R. Correa and C. Lemaréchal, *Convergence of some algorithms for convex minimization*, Mathematical Programming, 62 (1993), pp. 261–275.

[4] W. de Oliveira and C. Sagastizábal, *Level bundle methods for oracles with on-demand accuracy*, Optimization Methods and Software, 29 (2014), pp. 1180–1209.

[5] W. de Oliveira, C. Sagastizábal, and C. Lemaréchal, *Convex proximal bundle methods in depth: a unified analysis for inexact oracles*, Mathematical Programming, 148 (2014), pp. 241–277.

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

[7] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming, 55 (1992), pp. 293–318.

[8] G. Emiel and C. Sagastizábal, *Incremental like bundle methods with applications to energy planning*, Computational Optimization and Applications, 46 (2009), pp. 305–332.

[9] C. Fábián, *Bundle-type methods for inexact data*, in Proceedings of the XXIV Hungarian Operations Researc Conference (Veszprém, 1999), vol. 8 (special issue, T. Csendes and T. Rapcsk, eds.), 2000, pp. 35–55.

[10] C. Fábián, *Computational aspects of risk-averse optimisation in two-stage stochastic models*, tech. report, Institute of Informatics, Kecskemét College, Hungary, 2013. Optimization Online report.

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

[12] M. Fortin and R. Glowinski, *On decomposition-coordination methods using an augmented Lagrangian*, in Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, eds., North-Holland: Amsterdam, 1983, pp. 97–146.

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

[14] D. Gabay, *Applications of the method of multipliers to variational inequalities*, in Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, eds., North-Holland: Amsterdam, 1983, pp. 299–331.

[15] M. Gaudioso, G. Giallombardo, and G. Miglionico, *An incremental method for solving convex finite min-max problems*, Math. of Oper. Res., 31 (2006).

[16] W. E. Hart, C. Laird, J.-P. Watson, and D. L. Woodruff, *Pyomo — Optimization Modeling in Python*, Springer, 2012.

[17] W. E. Hart, J.-P. Watson, and D. L. Woodruff, *Pyomo: modeling and solving mathematical programs in Python*, Math. Program. Comput., 3 (2011), pp. 219–260.

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

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

[20] C. Lemaréchal, *Constructing bundle methods for convex optimization*, in Fermat Days 85: Mathematics for Optimization, vol. 129 of North-Holland Mathematics Studies, North-Holland, 1986, pp. 201–240.

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

[22] B. Martinet, *Détermination approchée d'un point fixe d'une application pseudo-contractante. Cas de l'application prox*, C. R. Acad. Sci. Paris Sér. A-B, 274 (1972), pp. A163–A165.

[23] W. Oliveira and C. Sagastizábal, *Bundle methods in the* xxi*st century: A birds'-eye view*, Pesquisa Operacional, 34 (2014), pp. 647 – 670.

[24] R. Rockafellar, *Convex Analysis*, Princeton University Press, 1st ed., 1970.

[25] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Mathematics of Operations Research, 1 (1976), pp. 97–116.

[26] R. T. Rockafellar and R. J.-B. Wets, *Scenarios and policy aggregation in optimization under uncertainty*, Mathematics of Operations Research, 16 (1991), pp. 119–147.

[27] A. Ruszczyński, *A regularized decomposition method for minimizing a sum of polyhedral functions*, Mathematical Programming, 35 (1986), pp. 309–333.

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

[29] W. van Ackooij and W. de Oliveira, *Level bundle methods for constrained convex optimization with various oracles*, Computation Optimization and Applications, 57 (2014), pp. 555–597.

[30] J.-P. Watson and D. L. Woodruff, *Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems*, Comput. Manag. Sci., 8 (2011), pp. 355–370.

[31] J.-P. Watson, D. L. Woodruff, and W. E. Hart, *PySP: modeling and solving stochastic programs in Python*, Math. Program. Comput., 4 (2012), pp. 109–149.

[32] C. Wolf, C. I. Fábián, A. Koberstein, and L. Suhl, *Applying oracles of on-demand accuracy in two-stage stochastic programming a computational study*, European Journal of Operational Research, 239 (2014), pp. 437 – 448.

[33] G. Zakeri, A. B. Philpott, and D. M. Ryan, *Inexact cuts in Benders decomposition*, SIAM J. Optim., 10 (2000), pp. 643–657.