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

Inexact stabilized Benders' decomposition approaches

with application to chance-constrained problems with finite support

W. van Ackooij · A. Frangioni · W. de Oliveira

Received: date / Accepted: date

Abstract We explore modifications of the standard cutting-plane approach for minimizing a convex nondifferentiable function, given by an oracle, over a combinatorial set, which is the basis of the celebrated (generalized) Benders' decomposition approach. Specifically, we combine stabilization—in two ways: via a trust region in the L_1 norm, or via a level constraint—and inexact function computation (solution of the subproblems). Managing both features simultaneously requires a nontrivial convergence analysis; we provide it under very weak assumptions on the handling of the two parameters (target and accuracy) controlling the *informative on-demand inexact oracle* corresponding to the subproblem, strengthening earlier know results. This yields new versions of Benders' decomposition, whose numerical performance are assessed on a class of hybrid robust and chance-constrained problems that involve a random variable with an underlying discrete distribution, are convex in the decision variable, but have neither separable nor linear probabilistic constraints. The numerical results show that the approach has potential, especially for instances that are difficult to solve with standard techniques.

Keywords :

Benders' decomposition, chance-constrained problems, mixed-integer optimization, nonsmooth optimization, stabilization, inexact function computation

Mathematics Subject Classification (2010) MSC 90C15 · MSC 90C25 · MSC 49M27 · MSC 90C11

1 Introduction

Motivated by application of Generalized Benders decomposition (GBD) algorithms to Chance-Constrained Optimization (CCO) problems, we investigate in this work modifications to the standard Cutting-Plane Method (CPM) for minimizing an oracle-provided convex nondifferentiable function over a combinatorial

Wim van Ackooij
EDF R&D, OSIRIS
7, Boulevard Gaspard Monge, 91120 Palaiseau Cedex, France
E-mail: wim.van-ackooij@edf.fr

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italia
E-mail: frangio@di.unipi.it

Welington de Oliveira
Universidade do Estado do Rio de Janeiro - UERJ
Rua São Francisco Xavier 524, 20550-900, Rio de Janeiro, Brazil
E-mail: welington@ime.uerj.br

set. The main aim of our investigation is to improve the practical performance of GBD by applying it (actually, to the CPM that sits at its core) two techniques that have been successful in related but different situations: *stabilization* and *inexact subproblem solution*. While these have sometimes been separately employed in the past, their simultaneous use involves a nontrivial interaction that had not been fully theoretically investigated before. Since GBD is the most prominent application of our results we mainly discuss them in that context. However, it has to be remarked that our theory is completely general, and it applies to any situation where the minimization of a convex (possibly nondifferentiable) function described by a standard (computationally costly) first-order oracle is required, provided that the oracle can be modified as to become an *informative on-demand inexact* one, formally defined below.

The observation that certain problems can be made much easier by temporarily fixing a subset of variables, and that solving the thusly simplified problem may provide information for improving the allocations of these fixed variables, was originally made for linear problems [11]. It was also soon realized that it was particularly promising for stochastic ones [62]. Since it basically relies on duality, it was readily generalized to problems with some underlying convexity [38]; we refer to [16, 33] for a general overview. In a nutshell, GBD is a variant of Kelley's Cutting-Plane Method [41] applied to the value function of the simplified problem. As it is well-known that CPM suffers from serious computational issues, it is not surprising that methods aimed at improving the performances of GBD have been proposed. Several of these have centered on the appropriate choice for the linearizations (Bender's cuts) used to construct the model, as it was quickly recognized that this can have a significant impact on (practical) convergence [42]. Different strategies have been proposed for generating good linearizations [26, 57, 61, 65, 68], occasionally with a problem-dependent flavor, e.g., based on duality theory [32] or on combinatorial arguments [15].

We tackle the issue of efficiency of GBD from a different angle, with a two-pronged approach. On the one hand we aim at decreasing the number of iterations by *stabilizing* the CPM. On the other hand we aim at decreasing the cost of each iteration by allowing the corresponding subproblem to be solved only inexactly. That the latter can be beneficial as well as it is intuitive was already recognized as early as [69]. However, in the corresponding convergence analysis the sequence of accuracy parameters is chosen a priori to converge to zero, so it is not adaptive to the actual needs of the algorithm. In [67] the idea is applied to the stochastic context, developing a specific rule for that application that allows to entirely skip evaluation of some subproblems (scenarios) at properly chosen iterations; this is shown to combine the advantages of disaggregate and aggregate variants. However, the approach is proposed for the level method only, and is tailored on the quite specific sum-function structure of two-stage stochastic programs. Besides, we consider the interaction between inexact subproblem solution and both known forms of stabilization of the CPM, namely, level and trust region/proximal. The combination of the two devices requires a technically rather involved theoretical analysis in the convex case; see [51] for the proximal stabilization, and [4, 28, 50] for the level one. It may be worth mentioning that, in the convex case, inexact computations can also be used in the context of fast gradient-like approaches [18, 64], but these methods are not easily adapted to the combinatorial setting. In our case, as it happened e.g., in [35], finiteness helps to simplify the arguments somewhat. We exploit this to thoroughly analyze the rules for selecting the two parameters controlling the oracle—the *target* and the *accuracy*—in order to devise methods with the weakest possible requirements. This allows us to cover a large set of different rules for choosing them. Stabilization of combinatorial Benders' approaches has already been proposed (e.g., [6, 58]), but usually with exact oracles. The only previous work we are aware of where both approaches are used simultaneously in the combinatorial case is [70], but only for the level stabilization. Moreover, our convergence arguments are significantly more refined.

It is worth remarking that, while we primarily discuss our results in the context where the subproblems are convex ones, they also apply—and, indeed, are even more likely to be significant—to the extensions of GBD where convexity is not present. These require different techniques according to the nature of the non-convexity in the subproblem: for instance, polyhedral techniques to approximate the convex hull [14, 59] or generalized "logic" duality (inference duality) [40] for integer linear subproblems, linearizations techniques to iteratively refine the description of the convex envelope for nonlinear nonconvex subproblems [44], or generalized concavity [71]. In all these cases the subproblems are significantly harder to solve, which makes their approximate solution all too relevant.

This work is organized as follows. Section 2 reviews (generalized) Benders' decomposition for binary nonlinear optimization problems, in order to set the stage for our results. In Section 3 we revisit the CPM to solve the resulting binary problem, extending it to handle oracles with on-demand accuracy. Two different stabilized variants of the CPM are then discussed in Section 4. Section 5 discusses the application of the developed techniques to a large set of randomly generated hybrid robust and nonlinear CCO problems, which is in itself new. Those COO problems can be recast as "monolithic" Mixed-Integer Second-Order Cone Problems (MISOCP), and therefore solved with standard software; our tests show that under many (although not all) conditions the new approaches significantly outperform the direct use of general-purpose MINLP tools. Finally, Section 6 closes with some concluding remarks.

2 Generalized Benders decomposition

Consider the following Mixed-Integer Non-Linear Problem (MINLP)

$$\min\left\{\, f(x) \;:\; G(x) \leq Tz \;\;,\;\; z \in Z \;\;,\;\; x \in X \,\right\} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $G : \mathbb{R}^n \to \mathbb{R}^p$ are convex mappings, $X \subseteq \mathbb{R}^n$ is a convex compact set, $T \in \mathbb{R}^{p\times m}$, and Z is some subset of $\{0,1\}^m$ which is "simple to describe". In (1), the binary z act as *complicating variables*, in that for fixed z the problem reduces to the convex Non-Linear Problem (NLP)

$$v(z) := \min\left\{\, f(x) \;:\; G(x) \leq Tz \;\;,\;\; x \in X \,\right\} \;\;, \tag{2}$$

which is much easier to solve. This is the core idea behind Generalized Benders' Decomposition (GBD): the problem is restated in terms of the complicating variables only as

$$v^* := \min\left\{\, v(z) \;:\; z \in Z \,\right\} \;\;, \tag{3}$$

using the *value function* $v : \{0,1\}^m \to \mathbb{R} \cup \{\infty\}$ of the NLP (2). The optimal values of (1) and (3) coincide: if z^* solves the latter and x^* solves (2) with $z = z^*$, then (x^*, z^*) solves (1). We recall in the following lemma some known properties of v.

Lemma 1 *The mapping v is proper, convex, and bounded from below. If for a given $z \in \mathcal{D}\mathrm{om}(v)$ (2) satisfies some appropriate constraint qualification (e.g., Slater's condition) so that the set $\Lambda(z) \subset \mathbb{R}^p_+$ of optimal Lagrange multipliers of the constraints $G(x) \leq Tz$ in (2) is nonempty, then $\partial v(z) = -T^{\mathsf{T}}\Lambda(z)$.*

Note that other constraint qualification can be invoked in the Lemma apart from Slater's condition, such the QNCQ one. What is required is just that solving the convex NLP (2) for fixed z be equivalent to solving its dual, i.e., that after having computed $v(z)$ an optimal dual solution $\lambda^*(z) \in \Lambda(z)$ is available that can be used to construct a *subgradient* for v in z. This brings us in the classical setting for NonSmooth Optimization (NSO): an *oracle* is available that provides function values and subgradients for the function.

The simplest approach for NSO problems, irrespectively of z being continuous or discrete, is perhaps the *Cutting-Plane Method* [41]. It relies on using the first-order information provided by the oracle to construct a *model* of v. At a given iteration k a set of iterates $\{z^1, \ldots, z^{k-1}\}$ has been generated and an index set $\mathcal{O}_k \subset \{1, \ldots, k-1\}$ gathers the points z^j whose oracle information $(v^j = v(z^j), w^j \in \partial v(z^j))$ is (still) available. The standard *cutting-plane model* for v is

$$\check{v}_k(z) := \max\{\, v^j + \langle w^j, z - z^j \rangle \;:\; j \in \mathcal{O}_k \,\} \;\leq\; v(z) \;\;, \tag{4}$$

where the inequality follows from convexity of v. Each of the inequalities in (4) is, in the parlance of Benders' decomposition, an *optimality cut*. Minimizing $\check{v}_k$ over the feasible set Z provides a lower bound v_k^{low} for v^* and a new candidate solution z^k to problem (3), at which the oracle is to be called. If $\check{v}_k(z^k) = v(z^k)$ then clearly z^k is optimal for (3), as $v_k^{\mathrm{low}} = \check{v}_k(z^k) \leq v^* \leq v(z^k)$; otherwise, the newly obtained function value (and subgradient) changes the model, and one can iterate. This brief account is, in a nutshell, the CPM applied to the solution of (3), i.e., the celebrated Benders' decomposition method applied to (1). Remarkably, this process does not depend on the fact that Z is a continuous or discrete set, provided that one can efficiently minimize $\check{v}_k$ over Z. Which is not to say that the process is identical for discrete and continuous feasible sets, as a number of relevant details differ. For instance, if $z^{k+1} \notin \mathcal{D}\mathrm{om}(v)$, i.e., if subproblem (2) is infeasible, then z^{k+1} must be eliminated from the feasible set of (3) by adding a *feasibility cut* to the set of constraints, which is usually obtained by means of the *unbounded ray* of the dual problem. In the binary setting, one can alternatively use a simpler *no-good-type* cut [17], along the lines of the *combinatorial* Benders' cuts proposed in [15]. In particular, let $S(z) = \{\, s \;:\; z_s = 0 \,\}$ and $S^k = S(z^k)$; then

$$\textstyle\sum_{s \in S^k} z_s \geq 1 \tag{5}$$

is a feasibility cut that excludes the point z^k from the feasible set of (3). This corresponds to restricting Z to a subset $Z^{k+1} \supseteq \mathcal{D}\mathrm{om}(v)$, or equivalently to force the model $\check{v}_{k+1}$ to agree with v in z^k ($\check{v}_k(z^k) < \check{v}_{k+1}(z^k) = v^k = +\infty$). Hence, together with the index set $\mathcal{O}_k$ of optimality cuts, one only needs to keep the index set $\mathcal{F}_k \subseteq \{1, \ldots, k-1\}$ of feasibility cuts, and define the *master problem*

$$z^{k+1} \in \arg\min \{\, \check{v}_k(z) \;:\; z \in Z^k \,\} \quad \equiv \quad \begin{cases} \min r \\ \text{s.t. } \; r \geq v^j + \langle w^j, z - z^j \rangle & j \in \mathcal{O}_k \\ \qquad \textstyle\sum_{s \in S^j} z_s \geq 1 & j \in \mathcal{F}_k \\ \qquad z \in Z \;,\; r \in \mathbb{R} \end{cases} \tag{6}$$

It is easy to show that no optimal solution of (3) is excluded by feasibility cuts, and that the CPM solves problem (3) (and, consequently, problem (1)). Note that (6) is unbounded below if $\mathcal{O}_k = \emptyset$, i.e., no optimality cut has been generated yet. This will surely happen for $k = 0$, and may happen even for a large k (for instance, for all k if Z^* is empty). A simple way to avoid this issue is to add to (6) the single constraint $r \geq 0$ in case $\mathcal{O}_k = \emptyset$, making (6) a feasibility problem seeking just any point in Z^k. That constraint is removed as soon as an optimality cut is generated.

Algorithm 1 Combinatorial Cutting-Plane Method (CCPM)

```
Step 0. (Initialization) Let $\delta_{\mathrm{Tol}} \geq 0$ be the stopping tolerance. $v_0^{\mathrm{up}} \leftarrow \infty$, $\mathcal{O}_1 = \mathcal{F}_1 \leftarrow \emptyset$, and $k \leftarrow 1$.
Step 1. (Master) Find $z^k$ by solving problem (6), and let $v_k^{\mathrm{low}}$ be its optimal value.
Step 2. (Stopping test) $\Delta_k \leftarrow v_{k-1}^{\mathrm{up}} - v_k^{\mathrm{low}}$. If $\Delta_k \leq \delta_{\mathrm{Tol}}$, stop: $z^{\mathrm{up}}$ is a $\delta_{\mathrm{Tol}}$-optimal solution.
Step 3. (Oracle call) Solve (2) with $z$ replaced by $z^k$.
    – If (2) is infeasible then $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{k\}$, $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k$, $v_k^{\mathrm{up}} \leftarrow v_{k-1}^{\mathrm{up}}$ and go to Step 4.
    – Else, obtain $v^k = v(z^k)$ and a subgradient $w^k$ as in Lemma 1, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$ and $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k \cup \{k\}$.
      $v_k^{\mathrm{up}} \leftarrow \min\{\, v^k \,,\, v_{k-1}^{\mathrm{up}} \,\}$. If $v^k = v_k^{\mathrm{up}}$ then $z^{\mathrm{up}} \leftarrow z^k$.
Step 4. (Loop) $k \leftarrow k + 1$ and go to Step 1.
```

Theorem 1 *Algorithm 1 terminates after finitely many steps either returning a δ_{Tol}-optimal solution to problem (3) or proving that it is infeasible.*

Proof First, note that the algorithm cannot stop for $k = 1$: $v_0^{\mathrm{up}} = \infty$ implies $\Delta_k = \infty > \delta_{\mathrm{Tol}}$ ($v_1^{\mathrm{low}} = 0$ because $\mathcal{O}_1 = \emptyset$, cf. the added constraint $r \geq 0$ in (6)). Once z^k is determined in Step 1 and v^k is computed in Step 3, we have two cases. If $v^k = \infty$, a feasibility cut is added which excludes (at least) z^k from the feasible region for good. As long as no feasible solution is found (i.e., $v^j = \infty$ for all $j \leq k$) one keeps having $v_k^{\mathrm{low}} = 0 \Longrightarrow \Delta_k = \infty$ unless (6) is infeasible, in which case $v_k^{\mathrm{low}} = v_k^{\mathrm{up}} = \infty$ and hence $\Delta_k = 0$: the algorithm stops, with $Z \subseteq Z^k = \emptyset$ proving that (3) is empty. Because at each iteration Z^k strictly shrinks and Z has a finite cardinality, after finitely many steps either the algorithm establishes that (3) is infeasible or it generates an optimality cut.

In this last case, from (6) we have that $v_k^{\mathrm{low}} = \check{v}_k(z^k) \geq v^j + \langle w^j, z^k - z^j \rangle$ for all $j \in \mathcal{O}_k$. Therefore, $\|w^j\|\|z^j - z^k\| \geq \langle w^j, z^j - z^k \rangle \geq v^j - v_k^{\mathrm{low}} \geq v_{k-1}^{\mathrm{up}} - v_k^{\mathrm{low}} = \Delta_k$, where the last inequality comes from the fact that $v_{k-1}^{\mathrm{up}} \leq v^j$ for all $j \in \mathcal{O}_k$. Hence, z^k differs from all the previously generated iterates as long as the $\Delta_k > 0$. Since Z contains only finitely many points, the algorithm finitely stops. □

The CPM can therefore be used to solve (3) (and thus (1)); it can also be generalized somewhat, see e.g., [66]. However, CPM is well-known to suffer from slow convergence in practice. It is therefore attractive to mimic "more advanced" NSO approaches, such as one of the several available variants of *bundle methods*, developed for the continuous case (e.g., [34,39,43] among many others). Adapting these methods to the case where Z is *combinatorial* requires specific work; this is done in Section 4, whereas in the next one we introduce another mechanism that can improve the performances of the approach.

3 Extending the cutting-plane method: inexact oracles

Algorithm 1 has two potentially costly steps: Step 1, that requires solving a MI(N)LP, and Step 3 that requires solving a (convex) NLP. A way to reduce the computational burden in Step 3 is to only *approximately solve* the NLP; in the parlance of [18, 50, 51] this is employing an *inexact oracle* for the value function v. While we discuss this with reference to GBD, the idea of inexact oracles clearly has wider application, as (10) will make apparent.

To illustrate what an inexact oracle is in the GBD case, we cast the oracle problem (2) in a Lagrangian setting. That is, for the vector λ of Lagrangian multipliers, the dual function of (2)

$$\theta(\lambda) := \min\left\{ f(x) + \langle \lambda, G(x) \rangle \ : \ x \in X \right\} - \langle \lambda, Tz \rangle \tag{7}$$

has the property that $\theta(\lambda) \leq v(z)$ for each $\lambda \in \mathbb{R}^p_+$. Hence, the Lagrangian dual

$$v(z) = \max\left\{ \theta(\lambda) \ : \ \lambda \in \mathbb{R}^p_+ \right\} \tag{8}$$

is (under appropriate constraint qualification, cf. Lemma 1) equivalent to (2). In fact, in correspondence to the optimal solution λ^* to (8) one can find an optimal solution to $x^* \in X$ to (7) (with $\lambda = \lambda^*$) such that $G(x^*) \leq Tz$ and $\langle \lambda^*, G(x^*) - Tz \rangle = 0$ (complementary slackness); one then has $f(x^*) \geq v(z) \geq \theta(\lambda^*) = f(x^*)$. *Approximately* solving (2) to any given accuracy ε reduces to finding a feasible primal-dual pair $(\bar{x}, \bar{\lambda})$ (i.e., $\bar{x} \in X$, $G(\bar{x}) \leq Tz$, $\bar{\lambda} \geq 0$) such that $(0 \leq) f(\bar{x}) - \theta(\bar{\lambda}) \leq \varepsilon$. Every conceivable algorithm for the problem has to provide both information in order to be able to stop at a certified (approximately) optimal solution: the Lagrangian setting is not the only possible one. When the form of $G(\cdot)$ allows for an appropriate algebraic dual, such as in the case of our experiments (cf. §5), more complex dual feasibility conditions can take the place of the minimization in (7) to ensure weak duality. Yet, such a dual solution would comprise λ as well as other dual variables for the other constraints representing $x \in X$. We will therefore not explicitly distinguish the two cases, and we will stick with the Lagrangian notation for ease of discussion.

Algorithmically speaking, there are two different approaches that can yield useful approximated oracles:

- *Dual approach*: directly tackle problem (8) via some appropriate optimization algorithm; as θ is most often nonsmooth, usually a NSO approach is required. This typically constructs a sequence $\{\lambda^h\}$ of iterates with nondecreasing objective value $\theta(\lambda^h)$ that approximate $v(z)$ from below. Any such algorithm eventually constructs a primal feasible (at least, up to some specified numerical tolerance) solution $\bar{x}$ such that $f(\bar{x}) - \theta(\lambda^h)$ is appropriately small [30], thereby providing the stopping criterion. Although it did not turn out to be computationally effective in our specific application, the dual approach has indeed shown to be efficient in several cases to solve large-scale continuous [36, 37] and combinatorial [3] programs.
- *Primal-dual approach*: under appropriate conditions, subproblem (2) can be solved by primal-dual interior-point methods (e.g., [10] among the many others). These typically construct a sequence of primal-dual pairs (x^h, λ^h) which track the *central path* towards the optimal primal-dual solutions (x^*, λ^*). Possibly after an initial infeasible phase, both solutions are *feasible*. In particular, it is well known (see for instance [12, §11.2]) that every central point $x^h \in X$ yields a dual feasible point λ^h. Hence, once again λ^h yields a lower bound on the optimal value, and x^h an upper bound; the algorithm stops when the two are suitably close.

In both cases, a sequence $\{\lambda^h\}$ is produced, each one of which can be used to construct *approximate linearizations*. In fact, $\underline{v} = \theta(\lambda) \leq v(z)$ and $w = -T^\mathsf{T}\lambda$ are such that

$$v(\cdot) \geq \underline{v} + \langle w, \cdot - z \rangle \ . \tag{9}$$

Note that (9) only hinges on weak duality, and therefore does not require constraint qualification. Thus, often a sequence of candidates for the next linearization in Algorithm 1 is available, provided that one allows them not to be tight, i.e., $\underline{v} < v(z)$. Indeed, *proving* that a linearization *is* tight, or at least that the error $\varepsilon = v(z) - \underline{v}$ is "small", requires the entirely different information x^h that provides an *upper bound* $\bar{v} \geq v(z)$. In this context, taking a leaf from [50] and (separately) [51], we define an *informative on-demand inexact oracle* as any procedure that, given $z \in Z$, a *descent target* $\texttt{tar} \in \mathbb{R} \cup \{-\infty, +\infty\}$ and a *desired accuracy* $\varepsilon \geq 0$, returns:

$$\left[\begin{array}{l} \text{i) as function information, } \textit{two} \text{ values } \underline{v} \text{ and } \bar{v} \text{ such that } \underline{v} \leq v(z) \leq \bar{v} \\ \text{ii) as first-order information, a vector } w \in \mathbb{R}^p \text{ such that (9) holds} \\ \quad \text{under the condition that, if } \underline{v} \leq \texttt{tar} \text{ then } \bar{v} - \underline{v} \leq \varepsilon \end{array}\right. \tag{10}$$

Setting $\texttt{tar} = \infty$ and $\varepsilon = 0$ gives the standard exact oracle for v. We will aim at defining the weakest possible conditions on the way to set the two parameters that still guarantee convergence of the algorithm. Intuitively, this means that we want to keep $\texttt{tar}$ "small", and ε "large". Indeed, when $\underline{v} > \texttt{tar}$ (which, for instance, surely holds if $\texttt{tar} = -\infty$), not much at all is required from $\bar{v}$: $\bar{v} = \infty$ is a perfectly acceptable answer, corresponding to "no feasible solution to (3) has been found yet" (which may be just because no feasible solution exists). Note that the oracle must be able to detect when (2) is infeasible, so that a feasibility cut is added to (6). This is signaled by returning $\underline{v} = \infty$ (which obviously implies $\bar{v} = \infty$), incidentally making the value of $\texttt{tar}$ irrelevant. Usually, in this case w should be required to describe a valid inequality for $\mathcal{D}\text{om}(v)$ that cuts away the current z, which requires the oracle to find an *unbounded feasible descent direction* for (8). Although doing so may be beneficial in practice, in our setting we can use (5) instead when $\underline{v} = \infty$, disregarding w; we therefore assume this, just in order to simplify the notation. Moreover, setting a finite target $\texttt{tar} < \infty$ actually allows us to dispense with feasibility cuts altogether. In fact, suppose that (2) is infeasible for the fixed z: since X in (1) is compact, we can argue by using [39, Proposition 2.4.1, Chapter XII] that (7) is unbounded. Thus, any dual or primal-dual algorithm applied to its solution will typically construct a sequence $\{\lambda^h\}$ such that $\theta(\lambda^h) \to \infty$, while (clearly) never constructing any feasible x^h. It is therefore easy to add the simple check $\theta(\lambda^h) > \texttt{tar}$ to the oracle, and stop it immediately when this happens (or at any moment thereafter). If $\texttt{tar} < \infty$ this will typically ensure finite termination of the oracle even in the unbounded case (which is not trivial for pure dual approaches), while still ensuring by weak duality that $\underline{v} = \theta(\lambda^h) > \texttt{tar}$ and $w = -T^\mathsf{T}\lambda^h$ provide the required information (whatever the value of $\bar{v}$, e.g., $\bar{v} = \infty$). This yields the following simple but useful remark:

Remark 1 (Valid cuts for function v) Given $z \in Z$ and a finite target $\texttt{tar} < \infty$ as input, an oracle (10) solving (7) can provide a *valid cut* w for v independently of whether or not (2) is feasible for z. Hence, $\mathcal{F}_k$ in (6) can remain empty for all k even if $Z \setminus \mathcal{D}\text{om}(v) \neq \emptyset$. □

As we will see, this information is enough for the CPM, making it irrelevant to "formally prove" if (2) is or not feasible. This shows that when inexact computations are allowed for, and a finite target is specified, the separation between feasibility and optimality cuts blurs somewhat.

3.1 A cutting-plane method for inexact oracles

We now adapt Algorithm 1 for dealing with the (informative, on-demand) inexact oracle (10). We start by providing the necessary change for defining the next iterate: having the (inexact) oracle information at hands, the inexact cutting-plane approximation for v is just

$$\check{v}_k(z) := \max\left\{\ \underline{v}^j + \langle w^j, z - z_j \rangle \ : \ j \in \mathcal{O}_k\ \right\} \ , \tag{11}$$

where $\mathcal{O}_k$ is as in Algorithm 1. Since (9) yields $\check{v}_k(z) \leq v(z)$ for all $z \in Z$, minimizing $\check{v}_k$ still provides a valid global lower bound over the optimal value of (3). The algorithm is then given below.

Algorithm 2 Inexact Combinatorial Cutting-Plane Algorithm (ICCPM)

```
Step 0. (Initialization) As in Step 0 of Algorithm 1. In addition, choose $\varepsilon_1 \geq 0$ and $\gamma > 0$.
Step 1. (Master) As in Step 1 of Algorithm 1, but with model given in (11).
Step 2. (Stopping test) As in Step 2 of Algorithm 1.
Step 3. (Oracle call) Choose $\mathtt{tar}_k$, send the triple $(z^k, \varepsilon_k, \mathtt{tar}_k)$ to oracle (10), receive $\underline{v}^k$, $\bar{v}^k$, and $w^k$.
    – If the subproblem is infeasible ($\underline{v}^k = \infty$), proceed as in Algorithm 1.
    – Otherwise, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k \cup \{k\}$, $v^{\rm up}_k \leftarrow \min\{\, \bar{v}^k \,,\, v^{\rm up}_{k-1} \,\}$. If $\bar{v}^k = v^{\rm up}_k$ then $z^{\rm up} \leftarrow z^k$.
        Step 3.1 (Accuracy control) If $\underline{v}^k \leq v^{\rm low}_k + \gamma$ then choose $\varepsilon_{k+1} \in [0, \varepsilon_k)$, otherwise choose $\varepsilon_{k+1} \in [0, \infty)$ arbitrarily.
Step 4. (Loop) $k \leftarrow k + 1$ and go to Step 1.
```

The algorithm is essentially the same as the original one with the obvious modifications regarding upper and lower estimates. The only real novel mechanisms are the ones regarding the handling of the target $\mathtt{tar}_k$ and of the accuracy ε_k. These are stated in a general form that allows many different implementations. The crucial requirement is that, eventually, iterations where $\underline{v}^k \leq \mathtt{tar}_k$ *and* $\varepsilon_k \leq \delta_{\rm Tol}$ are performed. With this simple device the algorithm can be proven to converge.

Theorem 2 *Assume that the choice of $\mathtt{tar}_k$ in Step 3 and that of ε_k in Step 3.1 are implemented in such a way that the following properties holds: i) if $v^{\rm up}_k = \infty$, then $\mathtt{tar}_k = \infty$, and ii) if ε_k is reduced in a sequence of* consecutive *iterations, then $\mathtt{tar}_k \geq v^{\rm low}_k + \gamma$ and $\varepsilon_k \leq \delta_{\rm Tol}$ holds for k large enough. Under these conditions, Algorithm 2 finitely terminates with either a $\delta_{\rm Tol}$-optimal solution to problem* (3) *or a proof that it is infeasible.*

Proof We will begin by showing that the algorithm establishes infeasibility of problem (3) after finitely many iterations. Since $v(z) = \infty$ for all $z \in Z$, the algorithm can never produce $\bar{v}^j < \infty$, and as a consequence $v^{\rm up}_k = \infty$ for all k. Then, by assumption i) $\mathtt{tar}_k = \infty$, which means that oracle (10) must necessarily return $\underline{v}^k = \infty$ for all k. The algorithm behaves exactly as the original one and the proof is completed.

Assume now that $\underline{v}^k < \infty$ at least once: the problem admits a feasible solution. In fact, at the first such k one has $\mathtt{tar}_k = \infty$, and therefore $\underline{v}^k < \mathtt{tar}_k$: by (10), $\infty > \underline{v}^k + \varepsilon_k \geq \bar{v}^k \geq v(z)$, hence at least one feasible point exists and $v^{\rm up}_k < \infty$ as well. Since the total number of possible "infeasible" iterations (with $\underline{v}^k = \infty$) is finite, for the sake of finite convergence arguments we can assume that none happens after iteration k. Note that, due to Remark 1, $\mathcal{F}_k$ may be empty even if some z^k really was infeasible. Hence, we can assume that a feasible z^k is determined in Step 1: from (6) with $\check{v}_k$ given in (11) we have that $v^{\rm low}_k = \check{v}_k(z^k) \geq \underline{v}^j + \langle w^j, z^k - z^j \rangle$ for all $j \in \mathcal{O}_k$, whence by the Cauchy-Schwarz inequality

$$\|w^j\| \|z^j - z^k\| \geq \langle w^j, z^j - z^k \rangle \geq \underline{v}^j - v^{\rm low}_k \quad . \tag{12}$$

For all the iterations $j \in \mathcal{O}_k$ where $\underline{v}^j > v^{\rm low}_k$, (12) gives $\|z^j - z^k\| > 0$, i.e., $z^k \neq z^j$ as in the proof of Theorem 1. If one could prove that $\underline{v}^j > v^{\rm low}_k$ holds always, the finiteness of Z would complete the proof.

However, since we have two different values $\bar{v}^j \geq v(z^j) \geq \underline{v}^j$, it may well happen that $\bar{v}^j - v^{\rm low}_k \geq \Delta_k > \delta_{\rm Tol}$ while one is performing a *tight* iteration: $\mathcal{T}_k = \{\, j \in \mathcal{O}_k \,:\, \underline{v}^j \leq v^{\rm low}_k \,\}$. For $j \in \mathcal{T}_k$ we could have $z^k = z^j$, which exposes the algorithm to the danger of cycling. Note that (12) cannot be used with $j = k$ because $k \notin \mathcal{O}_k$: the linearization corresponding to z^k was not in the definition of $\check{v}_k$ when the master problem was solved. That is, one can repeat the same iterate (say, $z^{k+1} = z^k$), and in this case $\underline{v}^k > v^{\rm low}_k$ only implies that (possibly, but not even necessarily) $v^{\rm low}_{k+1} > v^{\rm low}_k$. Thus, the lower bound on v^* may increase arbitrarily slowly; and similarly for the upper bound. In fact, when condition i) is no longer in effect, and until condition ii) is triggered, $\mathtt{tar}_k$ can be chosen arbitrarily (possibly, $\mathtt{tar}_k = -\infty$), which means that one can have $\bar{v}^k = \infty$ (although, at this point of the argument, $v^{\rm up}_k < \infty$). Thus, we need to distinguish among the iterations that are *in-target*, i.e., belong to $\mathcal{I}_k = \{\, j \leq k \,:\, \underline{v}^j \leq \mathtt{tar}_j \,\}$, and those that are not. The rationale is that only for $j \in \mathcal{I}_k$ the value of $\bar{v}_j$ actually is "meaningful", while for $j \notin \mathcal{I}_k$ nothing much can be said, $\bar{v}^j = \infty$ being possible. The relevant set of iterations is then $\mathcal{T}'_k = \mathcal{T}_k \cap \mathcal{I}_k$, and for these iterations one has

$$\begin{aligned} v^{\mathrm{up}}_{k-1} &= \min\{\, \bar{v}^j \ : \ j \in \mathcal{O}_k \,\} \leq \min\{\, \bar{v}^j \ : \ j \in \mathcal{T}'_k \,\} \leq \\ &\leq \min\{\, \underline{v}^j + \varepsilon_j \ : \ j \in \mathcal{T}'_k \,\} \leq v^{\mathrm{low}}_k + \min\{\, \varepsilon_j \ : \ j \in \mathcal{T}'_k \,\} \ , \end{aligned} \tag{13}$$

where the first inequality comes from $\mathcal{T}'_k \subseteq \mathcal{O}_k$, the second inequality comes from the assumption on oracle (10) when $j \in \mathcal{T}'_k \subseteq \mathcal{I}_k$, and the third one comes from the definition of $\mathcal{T}_k$. Note that $\mathcal{T}'_k = \emptyset$ may happen, making (13) useless, because either no tight iteration has been performed, or none of them is an in-target one, the latter clearly depending on how $\mathtt{tar}_k$ is chosen.

Let us now assume by contradiction that the algorithm does not finitely terminate. For iteration k, either $\underline{v}^k \leq v^{\mathrm{low}}_k + \gamma$, or $\underline{v}^k > v^{\mathrm{low}}_k + \gamma$. Let us first consider the case when $\underline{v}^k > v^{\mathrm{low}}_k + \gamma$ occurs infinitely many times (not necessarily consecutively). By taking subsequences if necessary we can assume that the condition actually holds at every k. Since the set Z is finite and the algorithm is assumed to loop forever, eventually it necessarily re-generates iterates that have already been found earlier. Hence, consider two iterates $h > k$ such that $z^h = z^k$: one has

$$v^{\mathrm{low}}_h = \check{v}_h(z^h) = \max\{\, \underline{v}^j + \langle w^j, z^h - z^j \rangle \ : \ j \in \mathcal{O}_h \,\} \geq \underline{v}^k + \langle w^j, z^h - z^k \rangle = \underline{v}^k > v^{\mathrm{low}}_k + \gamma \tag{14}$$

where the first equalities are just the fact that z^h is optimal for $\check{v}_h$, the leftmost inequality comes from $k \in \mathcal{O}_h$, and the following equality comes from $z^h = z^k$. Actually, if the algorithm runs forever then at least one of the iterates z^k has to be generated *infinitely many* times. Since $\gamma > 0$, repeating (14) on the corresponding sub-sequence proves that $v^{\mathrm{low}}_k \to \infty$ as $k \to \infty$, which contradicts $v^{\mathrm{low}}_k \leq v^* \leq v^{\mathrm{up}}_k < \infty$.

Hence, $\underline{v}^k > v^{\mathrm{low}}_k + \gamma$ can only occur finitely many times: if the algorithm runs forever, then eventually a long enough sequence of *consecutive* iterations where ε_k is reduced has to be performed. The assumption ii) on the choice of the oracle parameters now applies: eventually, $\mathtt{tar}_k \geq v^{\mathrm{low}}_k + \gamma \; (\geq \underline{v}^k \implies k \in \mathcal{I}_k)$ and $\varepsilon_k \leq \delta_{\mathrm{Tol}}$. This ensures that $\bar{v}^k - \underline{v}^k \leq \varepsilon_k \leq \delta_{\mathrm{Tol}}$: eventually, iterates become "accurate enough". Now, consider two (large enough, hence accurate enough) iterations $k > j$: if $j \in \mathcal{T}_k$, then by (13) one would have $\Delta_k = v^{\mathrm{up}}_{k-1} - v^{\mathrm{low}}_k \leq \bar{v}^j - \underline{v}^j \leq \varepsilon_j \leq \delta_{\mathrm{Tol}}$ (obviously, $v^{\mathrm{up}}_{k-1} \leq \bar{v}^j$), contradicting the fact that the algorithm does not stop. One should therefore have that $\underline{v}^j > v^{\mathrm{low}}_k$ always holds; however, in this case (12) shows that $z^k \neq z^j$ for all the infinitely many $k > j$, which contradicts finiteness of Z. Altogether, we have shown that the algorithm must therefore stop after finitely many iterations. □

We now discuss why, save for a few minor twists, the conditions on ε_k and $\mathtt{tar}_k$ in Theorem 2 appear to be basically the weakest possible ones.

- Having $\mathtt{tar}_k = \infty$ as long as $v^{\mathrm{up}}_k = \infty$, as required by condition i), may seem a harsh request, but it appears to be unavoidable in order to account for the case where (1) is infeasible. In fact, if the algorithm were allowed to set a finite target $\mathtt{tar}_1$, then the oracle may return any finite $\underline{v}_1 > \mathtt{tar}_1$ together with $w_1 = 0$ (and, obviously, $\bar{v}_1 = \infty$). Then, at the next iteration the algorithm would have $v^{\mathrm{low}}_2 = \underline{v}_1 > \mathtt{tar}_1$, and may well produce $z^2 = z^1$. Hence, an infinite sequence of iterations may start where $z^k = z^1$ and $v^{\mathrm{low}}_k \to \infty$, but the algorithm never stops because $v^{\mathrm{up}}_k = \infty$ as well (and ε_k is finite). That is, the algorithm would spend all the time trying to compute $v(z^1) = \infty$ by finitely approximating it from below. Thus, setting $\mathtt{tar}_k = \infty$ is required until the problem is proven feasible.
- A similar observation justifies the need for the constant $\gamma > 0$, both in the definition of the threshold for reducing ε_k in Step 3.1, and in the condition on $\mathtt{tar}_k$. In fact, if one would require decrease only if $\underline{v}^k \leq v^{\mathrm{low}}_k$, then a sequence of iterations all producing the same z may ensue where $\underline{v}^k > v^{\mathrm{low}}_k$, but "only very slightly so". Hence, while one may have $v^{\mathrm{low}}_{k+1} > v^{\mathrm{low}}_k$, the increment may be vanishingly small, and since ε_k would not be decreased, ultimately v^{low}_k may never become close enough to v^{up}_k (a similar argument, in the continuous case, is made in [18, Observation 2.7]). Analogously, if one would set $\mathtt{tar}_k = v^{\mathrm{low}}_k$, then again a sequence of iterations producing the same z^k could be performed where $\underline{v}^k$ may be fractionally larger than v^{low}_k. While this would force the decrease of ε_k, none of these iterations would be in-target, which would not allow us to use (13). That is, while v^{low}_k may indeed be converging to v^*, oracle (10) may never report a close enough upper bound; in fact, one may well have $\bar{v}^k = \infty$ for all k.

A few minor improvements could be added:

- It is possible to weaken somewhat condition i) by requiring that $\mathtt{tar}_k = \infty$ holds *after finitely many iterations where* $v_k^{\mathrm{up}} = \infty$, whatever mechanism be used to ensure this.
- Running the algorithm with fixed $\varepsilon_k = \delta_{\mathrm{Tol}}$ and $\mathtt{tar}_k = \infty$ does not require any γ (which is somehow obvious since that parameter only influences the choice of ε_k and $\mathtt{tar}_k$). An alternative way to get the same result would be to ask that $\varepsilon_k = \delta_{\mathrm{Tol}}$ and $\mathtt{tar}_k$ is "large enough" (e.g., $\mathtt{tar}_k = v_{k-1}^{\mathrm{up}}$) eventually, with some mechanism, say a fixed number of iterations, to ensure it.
- A fixed $\gamma > 0$ is only the simplest possible option. All one needs is that each time when the same iterate is repeated, v_k^{low} has "significantly increased". As customary in other settings (e.g., [18]), one may obtain this by ensuring that for every subsequence of the sequence $\{\gamma_k\}$ the series diverges (even if, say, $\gamma_k \to 0$). Note that this allows $\gamma_k = 0$ to happen, but only finitely many times: eventually $\gamma_k > 0$ has to happen, albeit it does not need to be bounded away from zero.
- Alternatively, for $\delta_{\mathrm{Tol}} > 0$ the simple choice $\gamma_k = \alpha \Delta_k$ for some fixed $\alpha > 0$ obviously works.
- A slightly weaker version of Step 3.1 is possible where ε_k is only reduced if $\underline{v}^k \leq v_k^{\mathrm{low}} + \gamma_k$ *and* z^k *coincides with a previously generated tight iterate*, for this cannot happen infinitely many times. However, the check that $z^k \neq z^j$ for all $j \in \mathcal{T}_k$ could ultimately be costly in both time and memory.

3.2 Accuracy handling in the ICCPM

Theorem 2 provides ample scope for constructing different strategies to handle the oracle parameters ε_k and $\mathtt{tar}_k$, besides the obvious one where $\varepsilon_k = \delta_{\mathrm{Tol}}$ and $\mathtt{tar}_k = \infty$ throughout. In fact, intuitively having high-accuracy computations at the initial phases of the algorithm is unnecessary, and that starting with a "large" ε_k and a "low" $\mathtt{tar}_k$ would be better. This has in fact been proven true computationally in the continuous case [5, 50, 67]. There are several possible ways of doing it:

- One may choose (even a-priori) a sequence $\{\varepsilon_k\} \to \delta_{\mathrm{Tol}}$ (finitely), while still keeping $\mathtt{tar}_k = \infty$. This again works, but it is clearly non adaptive.
- Keeping ε_k "large" and $\mathtt{tar}_k = -\infty$ is possible for most of the time. That is, if the condition at Step 3.1 is not satisfied, then one can immediately reset ε_{k+1} to some "large" value (say, ε_1), and leave it there until forced to reduce it. This generalizes the rule presented in [51, Algorithm 5.4] for the continuous case.

All this shows that there is no need to solve the oracle with even a modicum of accuracy, both from the primal and the dual viewpoint, unless the algorithm has reached the global optimum of the current model $\check{v}$; only then the accuracy has to be increased. Said otherwise, the function ideally only have to be computed with *provable* accuracy δ_{Tol} only at the optimal solution z^*. Insisting that ε_k and $\mathtt{tar}_k$ are kept "coarse" for most of the iterations is therefore possible, and likely justified in cases where the oracle cost is a preponderant fraction of the overall computational burden.

However, there can be cases where the master problem cost may be non-negligible, in particular because it is a combinatorial program. Furthermore, working with a "coarse" model of v—even coarser then the ordinary cutting-plane model arguably already is—may increase the number of iterations, and therefore finally prove not computationally convenient. Hence, more "eager" mechanisms for handling the accuracy parameters may be preferable, which can be obtained in several ways.

- For instance, $\mathtt{tar}_k \leftarrow \max\{\, v_k^{\mathrm{up}} \,,\, v_k^{\mathrm{low}} + \gamma_k \,\}$ may make sense. Basically, one is trying to improve on the upper bound, although doing so early on has only a limited effect on the algorithm behavior (the only role of v_k^{up} being in the stopping condition).
- Condition ii) can be ensured by choosing $\alpha \in (0, 1)$ and setting $\varepsilon_{k+1} = \max\{\, \delta_{\mathrm{Tol}} \,,\, \min\{\, \alpha\Delta_k \,,\, \varepsilon_k \,\}\,\}$. This is for instance the strategy that has been employed in [67, 69], and it is intuitively attractive because ε_k is nonincreasing with the iterations and converges to δ_{Tol}. A (minor) issue arises when $\delta_{\mathrm{Tol}} = 0$, since the algorithm may not finitely terminate: basically, once identified the optimal solution z^* one would keep on calling the oracle on z^* infinitely many times, with a sequence of ε_k (quickly)

converging to zero, but never really reaching it. This issue is easily resolved by setting $\varepsilon_k \leftarrow \delta_{\text{Tol}}$ when ε_k has reached a sufficiently small value.
- The analysis in Theorem 2 directly suggests to choose $\varepsilon_{k+1} < \min\{\, \varepsilon_j \;:\; j \in \mathcal{T}'_k \,\}$, with some mechanism ensuring $\varepsilon_k \leq \delta_{\text{Tol}}$ eventually. This has the advantage that the right-hand side of the equation is ∞ while $\mathcal{T}'_k = \emptyset$, so one starts to decrease ε_k only when tight iterates are indeed performed.
- The above mechanisms still rigidly alternates master problem and subproblem computations. If the master problem cost is significant, when the oracle error is found to be excessive one may rather prefer to avoid the computation of z^{k+1} (which may well produce z^k once again) and directly re-compute the function with increased accuracy. A potential advantage of this eager mechanism is that, while the oracle is still called several times on the same iterate z^k, (at least, some of) the calls happen *consecutively*. This means that *warm starts* can be used in the oracle making a sequence of calls with ε_k decreasing up to some $\bar{\varepsilon}$ not significantly more costly than a single call with accuracy $\bar{\varepsilon}$.

Thus, several strategies exist for handling the accuracy in the original CPM. In the next paragraph we will discuss how these can be adapted when stabilized versions of the approach are employed.

4 Extending the cutting-plane method: stabilization

In this section we explore a different strategy to improve the performances of the CPM: rather than decreasing the iteration cost, we aim at requiring fewer iterations. For this we propose two different *stabilization techniques* which, in analogy with what has been done for the continuous case, have the potential of improving the quality of the first-order information by decreasing the *instability* of the CPM, i.e., the fact that two subsequent iterates can be "very far apart". This has been shown to be very beneficial (e.g., [8, 35, 58] among the many others) for two reasons: fewer subproblems are solved, and therefore master problems of smaller sizes need be solved. Also, stabilization involves modifications of the master problem, which may (or may not) lead to an even further reduction of its computational cost. It has also been reported (cf. [70] in the discrete setting and [5] in the continuous one) that stabilization in GBD improves feasibility of master iterates, thus reducing the number of feasibility cuts.

4.1 Trust-region stabilization

The most widespread stabilization technique in the continuous setting is the *proximal* one, whereby a term is added to the objective function of the master problem to discourage the next iterate to be "far" from one properly chosen point (say, the best iterate found so far). In our setting, we find it more appropriate to use trust-regions [48]; note that in the continuous case the two approaches are in fact "basically the same" [8, 34]. For this we modify the master problem (6) by selecting one *stability center* $\hat{z}^k$, which can initially be thought of as being the current best iterate, and the radius $\mathcal{B}_k \geq 1$ of the current trust region. Then, we restrict the feasible region of (6) as

$$z^k \in \arg\min \left\{\, \check{v}_k(z) \;:\; z \in Z^k \;,\; \|z - \hat{z}^k\|_1 \leq \mathcal{B}_k \,\right\} \;, \tag{15}$$

which just amounts at adding the single linear *local branching constraint* [31]

$$\textstyle\sum_{s \,:\, \hat{z}^k_s = 1}(1 - z_s) + \sum_{s \,:\, \hat{z}^k_s = 0} z_i \leq \mathcal{B}_k \quad .$$

A similar approach was used in [58] and [49], both with exact oracles. In addition to consider inexact oracles, our analysis is significantly more refined. The effect of the trust region is to force the next iterate to lie in a *neighbourhood* of the stability center where only at most $\mathcal{B}_k$ variables can change their state w.r.t. the one they have in $\hat{z}^k$. A benefit of this choice is that the complement of that neighbourhood, $\|z - \hat{z}^k\|_1 \geq \mathcal{B}_k + 1$, can be represented by an analogous linear constraint. Note that in a convex setting any such set would be non-convex, thus making the master problem significantly harder to solve if one were to add it; yet, in our case this is just another linear constraint in a(n already nonconvex) MILP. It is therefore reasonable to add these constraints, and we will denote by $\mathcal{R}_k$ the set of iterations at which the *reverse region* constraints are added. We remark in passing that, since for $z_s \in \{0,1\}$ one has $z_s^2 = z_s$, also $\|z - \hat{z}^k\|_2^2$ has a similar linear expression (this is in fact used in §4.2). The ICCPM is modified as follows:

Algorithm 3 Trust Region Inexact Combinatorial Cutting-Plane Algorithm (TRICCPM)

```
Step 0. (Initialization) As in Step 0 of Algorithm 2, plus $\mathcal{R}_1 \leftarrow \emptyset$, $\hat{v}^1 = \infty$, choose $\mathcal{B}_1 \geq 1$, $\beta > 0$ and $\hat{z}^1 \in Z$ arbitrarily.
Step 1. (Master) As in Step 1 of Algorithm 2 except using (15).
Step 2. (Stopping test) $\Delta_k \leftarrow v_{k-1}^{\text{up}} - v_k^{\text{low}}$. If $\Delta_k > \delta_{\text{Tol}}$ then $\mathcal{R}_{k+1} \leftarrow \mathcal{R}_k$, $\mathcal{B}_{k+1} \leftarrow \mathcal{B}_k$ and go to Step 3.
    If $\mathcal{B}_k = m$ then stop: $z^{\text{up}}$ is a $\delta_{\text{Tol}}$-optimal solution. Otherwise, $\mathcal{R}_{k+1} \leftarrow \mathcal{R}_k \cup \{k\}$ and choose $\mathcal{B}_{k+1} \in (\mathcal{B}_k\,,\, m]$.
Step 3. (Oracle call) As in Step 3 of Algorithm 2, except add the following before Step 3.1:
    If $\bar{v}_k \leq \hat{v}_k - \beta$ then $\hat{z}^{k+1} \leftarrow z^k$, $\hat{v}_{k+1} \leftarrow \bar{v}_k$, choose $\varepsilon_{k+1} \in [0,\infty)$ arbitrarily and go to Step 4.
    Otherwise $\hat{z}^{k+1} \leftarrow \hat{z}^k$, $\hat{v}_{k+1} \leftarrow \hat{v}_k$ and proceed to Step 3.1.
Step 4. (Loop) $k \leftarrow k+1$ and go to Step 1.
```

A few remarks on the algorithm are useful. The initial stability center $\hat{z}^1$ may or may not be feasible; to be on the safe side, by initializing $\hat{v}_1 = \infty$ we assume it is not. If a feasible iterate z^k is produced, the new mechanism at Step 3 ensures that the stability center is moved to the feasible point. This is called a "serious step" (SS) in the parlance of bundle methods. Note that the initial master problem, not really having an objective function, may well return $z^1 = \hat{z}^1$. In this case one may be doing a "fake" SS where the stability center does not really change, while its objective function value does. Adding the reverse region constraint when $\mathcal{B}_k$ is increased in Step 2 is not strictly necessary, but it is cheap and ensures that iterates in the previous neighbourhood are no longer produced. Reducing the feasible region of (15), this hopefully makes it easier to solve.

Theorem 3 *Under the assumptions of Theorem 2, Algorithm 3 terminates after finitely many steps with either a δ_{Tol}-optimal solution to problem* (3) *or a proof that this problem is infeasible.*

Proof The analysis of Theorem 2 applied to a fixed stability center, and therefore to the fixed (finite) feasible region $Z \cap (\,\|\cdot - \hat{z}\|_1 \leq \mathcal{B}\,)$, shows that if no SS is done, eventually the stopping condition at Step 2 is triggered. If this happens but $\mathcal{B} < m$, then *local* δ_{Tol}-optimality of z^{up} (which, as we shall see later on, is *not* necessarily the same as $\hat{z}$) in the current neighbourhood has been proven. If the master problem were convex this would be the same as global optimality, but since this is not the case we need to increase $\mathcal{B}$ up until eventually the whole Z is covered; obviously, this can happen only a finite number of times. When $\mathcal{B}_k = m$, global optimality is ensured; in particular, if no feasible solution has been found throughout, then (3) clearly is infeasible.

Hence, we must prove that changing $\hat{z}^k$—performing a SS—can only happen finitely many times. Note that a SS do not trigger a forced decrease of ε_k (although it is possible to reduce it), so we cannot rely on that mechanism to prove finite convergence. However, since $\beta > 0$, a SS can only be declared when a "significant" decrease is obtained, which can only happen finitely many times. □

Again, the assumptions on the SS mechanism are (almost) minimal. Indeed, with $\beta = 0$ it would be possible that the same iterate z is produced infinitely many times, each time the oracle providing a vanishingly smaller value for $\bar{v}$ and therefore triggering a "fake" SS (where $\hat{z}$ is not actually changed). This in itself would not impair finite convergence, but an even more devious case exists: the one where $\hat{z}$ "cycles through" a set of iterates (possibly with the very same, and optimal, value of $v(z)$), each time fractionally reducing v^{up} but never really producing the true value: $\beta > 0$ rules this out. Similarly to the remark concerning γ in ICCPM: a fixed value $\beta > 0$ is the easiest choice. A non-summable sequence $\{\beta_k\}$ or, for $\delta_{\text{Tol}} > 0$, $\beta_k = \alpha\Delta_k$ for $\alpha \in (0,1]$ are other options that work as well. We remark in passing that the rules deciding when a SS has to be declared are crucial in the continuous case when dealing with inexact oracles [51]. It is also apparent that the analysis extends to the "eager" accuracy control of §3.1.

A relevant practical detail is how $\mathcal{B}$ is increased in Step 2 when local optimality is detected. The simple rule $\mathcal{B}_{k+1} \leftarrow \mathcal{B}_k + 1$ would work, but previous experience (in the exact setting) [6] suggests that this is not computationally effective: once a local optima for a "small" region has been found, it is usually best to try to prove its global optimality. Therefore, in our computational experiences we have used (without any significant amount of tuning) the simple approach where $\mathcal{B}$ is updated by moving through a restricted set of sizes, as described in §5.3. Similarly to what happens for ε_k and $\mathtt{tar}_k$, it is possible to reset $\mathcal{B}$ to a "small" value each time a SS is computed, since this only happens finitely many times. This differs from [58] that employs the trust-region constraint only in the initial iterations.

4.2 Level stabilization

In this section we explore *level stabilization*, which is, even in the continuous case, significantly different to analyze [43]. Continuous level bundle methods have been analyzed in the inexact case in [50], and discrete level bundle methods have been analyzed in the exact case in [20]; to the best of our knowledge, apart from the recent [70] this is the first work investigating a discrete level bundle method with inexact oracles. Our results are more general, as we deal with oracles with on-demand (as opposed to fixed) accuracy, and the hypotheses we require to establish convergence are significantly weaker.

Level stabilization centers on defining a *level parameter* $v_k^{\text{lev}} \in (v_k^{\text{low}}, v_{k-1}^{\text{up}})$ and the *level set*

$$\mathbb{Z}^k := \left\{ z \in Z^k \ : \ \check{v}_k(z) \le v_k^{\text{lev}} \right\} \ . \tag{16}$$

The next iterate z^k is then chosen in $\mathbb{Z}^k$, whenever it is nonempty. If $\mathbb{Z}^k = \emptyset$, then $v_k^{\text{low}} < v_k^{\text{lev}} \le v^*$, i.e., a lower bound on the optimal value that is *strictly better* then v_k^{low} has been generated. As a result, the current lower bound v_k^{low} can be updated by the simple rule $v_k^{\text{low}} \leftarrow v_k^{\text{lev}}$.

Choosing a new iterate in $\mathbb{Z}^k$ can be done in several ways. A good strategy is to choose z^k by satisfying some criterion of proximity with respect to a given stability center $\hat{z}^k \in \{0,1\}^m$, which in this case may not even belong to Z. That is, we find z^k by minimizing a convex *stability function* $\varphi(\cdot; \hat{z}^k)$ over $\mathbb{Z}^k$; in the continuous case, the most common choice for φ is $\varphi = \| \cdot \|_2^2$. In our setting it may be natural to take the ℓ_1 or ℓ_∞ norms instead, as done in [20], since this leads to a MILP master problem. Yet, as already mentioned, for binary variables the Euclidean norm also is linear: $\|z - \hat{z}^k\|_2^2 = \langle \frac{1}{2}\mathbf{1} - \hat{z}^k, z \rangle + \|\hat{z}^k\|_2^2$, $\mathbf{1}$ being the all-one vector of appropriate dimension. Taking the ℓ_2 norm as a stability function leads to

$$z^k \in \arg\min \left\{ \varphi(z; \hat{z}^k) \ : \ z \in \mathbb{Z}^k \right\} \quad \equiv \quad \begin{cases} \min \ \langle \frac{1}{2}\mathbf{1} - \hat{z}^k, z \rangle \\ \text{s.t.} \ \ \underline{v}^j + \langle w^j, z - z^j \rangle \le v_k^{\text{lev}} \ j \in \mathcal{O}_k \\ \qquad z \in Z^k \, . \end{cases} \tag{17}$$

The level version of the ICCPM then reads as follows:

Algorithm 4 Level Inexact Combinatorial Cutting-Plane Algorithm (LICCPM)

```
Step 0. (Initialization) Run Algorithm 2 with the added condition in Step 2: stop (also) if $\bar{v}^k = v_k^{\text{up}} < \infty$.
    If $\Delta_k \le \delta_{\text{Tol}}$ then terminate.
Step 1. (Stopping test) As in Step 2 of Algorithm 2.
Step 2. (Master) Choose arbitrarily $\hat{z}^k \in \{0,1\}^m$. Choose $v_k^{\text{lev}} \in [v_k^{\text{low}}, v_{k-1}^{\text{up}} - \delta_{\text{Tol}})$. Solve (17): if it is infeasible
    then $v_k^{\text{up}} \leftarrow v_{k-1}^{\text{up}}$, choose $v_k^{\text{low}} \in [v_k^{\text{lev}}, v^*]$ and go to Step 4, else $z^k$ is available.
Step 3. (Oracle call) Choose $\mathtt{tar}_k$. Send the triple $(z^k, \varepsilon_k, \mathtt{tar}_k)$ to oracle (10), receive $\underline{v}^k$, $\bar{v}^k$, and $w^k$.
    – If $\underline{v}^k = \infty$ then proceed as in Algorithm 1.
    – Otherwise, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k \cup \{k\}$, $v_k^{\text{up}} \leftarrow \min\{ \bar{v}^k \, , \, v_{k-1}^{\text{up}} \}$. If $\bar{v}^k = v_k^{\text{up}}$ then $z^{\text{up}} \leftarrow z^k$.
      Step 3.1 (Accuracy control) If $\underline{v}^k \le v_k^{\text{lev}}$ then choose $\varepsilon_{k+1} \in [0, \varepsilon_k)$, otherwise choose $\varepsilon_{k+1} \in [0, \infty)$ arbitrarily.
Step 4. $v_{k+1}^{\text{low}} \leftarrow v_k^{\text{low}}$, $k \leftarrow k+1$ and go to Step 1.
```

As usual, a few remarks on the algorithm can be useful:

– The level master problem (17) does not need any trick, as (6) and (15) do, when $\mathcal{O}_k = \emptyset$. In fact, as long as this happens, the value of v_k^{lev} is completely irrelevant: $\mathbb{Z}^k = Z^k$, and one is seeking the nearest feasible point to $\hat{z}^k$ (i.e., $\hat{z}^k$ itself if $\hat{z}^k \in Z^k$).

- The algorithm does not actually need an *optimal* solution to (17): any *feasible* point is enough. This opens the way for applying heuristics to (17), for instance by solving the continuous relaxation and then applying randomized rounding. One can also append the formulation (17) with additional constraints, such as those induced by precedence relations [55], if these can be exhibited.
- Conversely, (17) does not automatically produce a valid (local) lower bound v_k^{low} as (6) (and (15)) do, while—at least if $\mathcal{O}_k \neq \emptyset$—requiring one for defining v_k^{lev}. Thus, the algorithm requires an initialization phase which essentially uses the standard CPM. A different initialization step will be discussed later on.

Theorem 4 *Assume that Step 2 is implemented in such a way that v_k^{lev} changes or problem* (3) *is found to be infeasible* only finitely many times. *Furthermore, assume that the choice of $\mathtt{tar}_k$ in Step 3 and that of ε_k in Step 3.1 satisfy the assumptions of Theorem 2, only with $\mathtt{tar}_k \geq v_k^{\text{lev}}$ replacing $\mathtt{tar}_k \geq v_k^{\text{low}} + \gamma$. Then, Algorithm 4 finitely terminates with either a δ_{Tol}-optimal solution to problem* (3) *or a proof that it is infeasible.*

Proof Step 0 of the algorithm—possibly a complete run of Algorithm 2—finitely terminates due to Theorem 2: either $v_{k-1}^{\text{up}} = v_k^{\text{low}} = \infty \implies \Delta_k = 0$, proving that (3) is infeasible, or a feasible z^k is eventually found, at which point Step 0 terminates. Since this happens at Step 2 of Algorithm 2, v_k^{low} is available, which is what the initialization aimed at. Also, note that one could have $v_{k-1}^{\text{up}} \leq v_k^{\text{low}} + \delta_{\text{Tol}} < \infty$, i.e., the algorithm stops at Step 0 because the initialization has already found a δ_{Tol}-optimal solution.

The crucial assumption on the level management is that v_k^{lev} can change and (3) be infeasible only finitely many times. Consequently, if the algorithm does not terminate, there exists an iteration $\bar{k}$ such that $v_k^{\text{lev}} = v_{\bar{k}}^{\text{lev}}$ and $\mathbb{Z}^k \neq \emptyset$ for all $k \geq \bar{k}$. Let us therefore assume $k \geq \bar{k}$: we have $\underline{v}^j + \langle w^j, z^k - z^j \rangle \leq v^{\text{lev}}$ for all $j \in \mathcal{O}_k$, where $v^{\text{lev}} = v_k^{\text{lev}} = v_{\bar{k}}^{\text{lev}}$, from which we get the analogous of (12)

$$\|w^j\| \|z^j - z^k\| \geq \langle w^j, z^j - z^k \rangle \geq \underline{v}^j - v^{\text{lev}} \quad . \tag{18}$$

Defining $\mathcal{T}_k = \{ j \in \mathcal{O}_k : \underline{v}^j \leq v^{\text{lev}} \}$, (18) immediately shows that $z^k \neq z^j$ for $j \notin \mathcal{T}_k$. Again, only tight iterations can repeat previous iterates. Therefore, $\underline{v}^k > v^{\text{lev}}$ can only happen finitely many times. In this case one has $k \notin \mathcal{T}_{k+1}$, and more in general $k \notin \mathcal{T}_h$ for all $h > k$. So, each time a non-tight iteration is performed, its iterate must be different from all these previous non-tight iterations. Thus, finiteness of Z ensures that either the algorithm stops, or eventually only tight iterates can be performed. Let us therefore assume that $\bar{k}$ is large enough so that for all $k \geq \bar{k}$ the iterate is tight.

From the hypotheses on $\mathtt{tar}_k$ and ε_k, eventually $\mathtt{tar}_k \geq v^{\text{lev}}$ and $\varepsilon_k \leq \delta_{\text{Tol}}$, i.e, $\bar{v}^k - \underline{v}^k \leq \varepsilon_k \leq \delta_{\text{Tol}}$. Copying (13) (keeping $\mathcal{I}_h$ and $\mathcal{T}'_h$ unchanged), we similarly conclude

$$v_{k-1}^{\text{up}} \leq \min\{ \bar{v}^j \ : \ j \in \mathcal{T}_k \} \leq \min\{ \underline{v}^j + \varepsilon_j \ : \ j \in \mathcal{T}'_k \} \leq v_k^{\text{lev}} + \min\{ \varepsilon_j \ : \ j \in \mathcal{T}'_k \} \leq v^{\text{lev}} + \delta_{\text{Tol}} \quad .$$

But the choice of v^{lev} at Step 2 now requires that $v^{\text{lev}} < v_{k-1}^{\text{up}} - \delta_{\text{Tol}} \leq v^{\text{lev}}$, a contradiction: hence, the algorithm must terminate finitely. □

The assumptions on Step 2 are not trivial to satisfy. This is because the general rule in Step 2, $v_k^{\text{lev}} \in [v_k^{\text{low}}, v_{k-1}^{\text{up}} - \delta_{\text{Tol}})$, *requires* changing the value of v^{lev} from that of the previous iteration when $v_{k-1}^{\text{lev}} > v_{k-1}^{\text{up}} - \delta_{\text{Tol}}$, i.e., one has found a better upper bound at the previous iteration that forces v^{lev} to be decreased. Furthermore, when $\mathbb{Z}^k = \emptyset$ it is easy to choose v_{k+1}^{lev} in a way that causes this to happen again at the next iteration: just increase v^{lev} of a vanishingly small fraction. Hence, ensuring that none of this happens infinitely many often requires careful choices in the updating mechanism. This is especially true if $\delta_{\text{Tol}} = 0$, because it means that eventually one must have $v_k^{\text{low}} = v_k^{\text{lev}} = v_{k-1}^{\text{up}} = v^*$, quite a high call. Indeed, we will show that this is not, in fact, possible unless one provides the algorithm with a significant helping hand under the form of a way to compute "tight" lower bounds. Different working examples of v^{lev}-selection mechanisms can be developed, though, as we discuss below.

The analysis should also plainly extend to the case when the feasible set Z is not finite, but still bounded, $\mathcal{D}\text{om}(v) \subset Z$ (ensuring thus that ∂v is locally bounded) and $\delta_{\text{Tol}} > 0$. This case is analyzed in [70], under stricter assumptions on the oracle. We have not pursued this extension because our analysis is rather focussed on the handling of the accuracy parameters in the oracle: very similar results could be expected in the non-finite compact case, but this would make the arguments harder to follow.

4.3 Accuracy handling in the LICCPM

The short proof of Theorem 4 somehow hides the fact that the level parameter v^{lev} has to be properly managed for the assumptions to be satisfied. We now discuss possible mechanisms which obtain that.

If $\delta_{\mathrm{Tol}} > 0$, then the assumption of the Theorem can be satisfied by the following simple mechanism: in Step 3, whenever $\mathbb{Z}^k = \emptyset$ we set $v_k^{\mathrm{low}} \leftarrow v_k^{\mathrm{lev}}$. Furthermore, denoting by $h(k) < k$ the iteration where v_k^{lev} has last changed ($h(1) = 1$), for some fixed $\alpha \in (0,1)$ we set

$$v_k^{\mathrm{lev}} \leftarrow \begin{cases} v_{k-1}^{\mathrm{up}} - \max\{\, \delta_{\mathrm{Tol}} \,,\, \alpha \varDelta_k \,\} & \text{if } v_{k-1}^{\mathrm{up}} < v_{h(k)}^{\mathrm{up}} - \delta_{\mathrm{Tol}} \text{ or } \mathbb{Z}_{k-1} = \emptyset \\ v_{h(k)}^{\mathrm{lev}} & \text{otherwise} \end{cases} \quad . \tag{19}$$

In plain words, v_k^{lev} is updated whenever v^{low} needs be revised upwards, or v^{up} is "significantly" revised downwards. This mechanism ensures that v_k^{lev} cannot change infinitely many times. In fact, even if the same iterate z^k (possibly, the optimal solution) is generated more than once, updating the upper bound $\bar{v}^k$ by vanishingly small amounts, v_k^{lev} only changes if the upper bound decreases "significantly", i.e., by at least $\delta_{\mathrm{Tol}} > 0$. Similarly, $\mathbb{Z}^k = \emptyset$ cannot happen infinitely many times: in fact, whenever this happens

$$\varDelta_{k+1} = v_k^{\mathrm{up}} - v_k^{\mathrm{low}} \le v_{k-1}^{\mathrm{up}} - v_{k-1}^{\mathrm{lev}} = \max\{\, \delta_{\mathrm{Tol}} \,,\, \alpha \varDelta_k \,\} \quad .$$

In other words, the gap shrinks exponentially fast until eventually $\varDelta_k \le \delta_{\mathrm{Tol}}$, triggering the stopping condition. Note, however, that for $\delta_{\mathrm{Tol}} = 0$ (19) only gives $\varDelta_k \to 0$, but not finite convergence.

Although not particularly significant in practice, it may be worth remarking that the fact that (19) only works with $\delta_{\mathrm{Tol}} > 0$, unlike those of Algorithms 2 and 3, is not not due to a weakness of the analysis, but rather to an inherent property of level-based methods. Indeed, using a level stabilization one does not have a way to prove that a given value v_k^{lev} is a *sharp* lower bound on v^*: when $\mathbb{Z}^k = \emptyset$ we can conclude $v_k^{\mathrm{lev}} < v^*$. The fact that the inequality is strict shows that a level-based approach will never be able to prove that $v_k^{\mathrm{lev}} = v^*$. This is why one is not allowed to pick $v_k^{\mathrm{lev}} = v_{k-1}^{\mathrm{up}}$: if $v_{k-1}^{\mathrm{up}} = v^*$, one would never be able to prove this because $\mathbb{Z}^k \neq \emptyset$. Indeed, consider a stylized problem with $Z = \{\bar{z}\}$. At the first iteration (in Step 0), the oracle may provide $\bar{v}_1 = v(\bar{z}) = v^*$ and some $\underline{v}_1 = v^* - \varepsilon_1$ (with $\varepsilon_1 > 0$), so that Step 0 ends with $v_1^{\mathrm{low}} = v^* - \varepsilon_1$. Even if one sets $\varepsilon_k = 0$, an infinite sequence of iterations then follows whereby $\mathbb{Z}^k = \emptyset$ always happens and $v_k^{\mathrm{low}} \to v^*$—say, using (19)—but never quite reaching it. This is a known (minor) drawback of this form of stabilization.

Actually, LICCPM can finitely converge even if $\delta_{\mathrm{Tol}} = 0$, but only if the initial v_k^{low} provided by Step 1 happens to be precisely v^*. This is because that v_k^{low} is produced by different means, which do allow to prove that $v_k^{\mathrm{lev}} \le v^*$. This observation suggests a variant of the algorithm which can work with $\delta_{\mathrm{Tol}} = 0$: just prior to setting the level parameter one solves (6) and updates v_k^{low} with its minimum value. This was proposed when level methods were introduced [43], in the continuous case. The condition ensures that $\mathbb{Z}^k \neq \emptyset$ will happen at *every* iteration, thus making the relevant part of the assumption in Theorem 4 moot. However, this re-introduces the risk that v_k^{low} increases infinitely many times. Furthermore, (19) now no longer rule out the risk that decreases of v_k^{up} are vanishing. To avoid these problems, one may for instance introduce some mechanism whereby eventually $v_k^{\mathrm{lev}} = v_k^{\mathrm{low}}$: basically, at some point the algorithm reverts to the standard CPM. Hybrid versions where (6) is solved "from time to time" are also possible. While in principle applicable, we do not see this approach as promising in our specific setting because our master problem is combinatorial, and hence possibly computationally costly. For this reason we do not pursue its analysis further.

A somewhat opposite approach would be to dispense the need of finding a v_k^{low} that is a guaranteed lower bound on v^* from the start, and hence the need of solving (6) at least once. To do that, one can avoid the call to (the modified) Algorithm 2 in Step 0, and instead initialize $v_1^{\text{low}} < \infty$ arbitrarily. Then, the following has to be added right before Step 3.1:

Step 3.0 (v_k^{low} update) If $\mathbb{Z}^k = \emptyset$ has not happened yet and $v_k^{\text{up}} - \delta_k < v_k^{\text{low}}$, then
$v_k^{\text{low}} \leftarrow \min\{\, v_k^{\text{up}}\,,\, v_k^{\text{low}}\,\} - \delta_k$, choose $\varepsilon_{k+1} \in [0, \infty)$ arbitrarily and go to Step 4

where $\delta_k \in (\delta_{\text{Tol}}, \bar{\delta}]$ for some $\bar{\delta} < \infty$. With this modification, similar to the one present in [50], the algorithm replaces the dependable *lower* bound v_k^{low} on v^* with a guess, produced using the best available *upper* bound and a displacement (this is called a "target value" approach [18]). Step 3.0 cannot be executed infinitely many times: each time it does v_k^{low} decreases by an amount bounded away from zero, and $v^* > -\infty$. Hence, eventually v_k^{low} will be a valid lower bound on $v^* - \bar{\delta}$, and $v_k^{\text{up}} - \delta_k < v_k^{\text{low}}$ can no longer happen unless v_k^{low} increases. But the latter only happens when $\mathbb{Z}^k = \emptyset$, at which point Step 3.0 is disabled: a dependable lower bound has been found, and the normal course of the algorithm, as analyzed in Theorem 4, starts. Basically, all the iterations up to that point take the place of the Step 0 where (6) is solved. Note that the algorithm cannot stop before that $\mathbb{Z}^k = \emptyset$ at least once, since the target will always "overrun" v_{k-1}^{up} by at least $\delta_k > \delta_{\text{Tol}}$. Hence, the analysis of Theorem 4 still applies. In fact, similarly to §4.1, we can disable the accuracy control when the target decreases.

As a final remark, the analysis clearly extends to the more "eager" accuracy control versions of §3.1, with the corresponding computational trade-offs.

4.4 Bundle resets: making master problems easier to solve

All the master problems considered in this work are combinatorial problems, and hence in principle difficult to solve. It can be expected that the size of the two bundles $\mathcal{F}_k$ and $\mathcal{O}_k$, respectively of feasibility and optimality cuts, may have an impact on the solution time (this may be true also for $\mathcal{R}_k$ of the reverse region constraints in TRICCPM). It is therefore possible—although by no means certain—that reducing the bundle sizes helps in reducing the master problem time.

In the convex case it is sometimes possible to reduce the size to $\mathcal{O}_k$ all the way down to $|\mathcal{O}_k| = 2$ by the so-called *aggregation technique*. However, this cannot be done for $\mathcal{F}_k$, and even for $\mathcal{O}_k$ this only works for certain stabilizations: the standard CPM and trust region approaches, for instance, do not have any particularly elegant way of resetting the bundle [34, §5.3], while proximal (under specific assumptions) [34, §5.2] and level do. However, the aggregation technique heavily relies on convexity of the master problem (in particular by using the dual optimal solution), and therefore does not extend to our discrete case, even for the stabilizations that would allow it in the continuous one.

There is a practical way in which one can reset $\mathcal{O}_k$ (and, by the same token, $\mathcal{F}_k$): it can be done arbitrarily, only provided that this happens *finitely many times*. The standard convergence proofs then apply after that the last reset has occurred. This is not a very satisfactory mechanism, and in particular it does not allow to set any a-priori bound on $|\mathcal{O}_k|$; however, it is, basically, the only mechanism that works in general even in the convex case, unless strong properties allow otherwise [34, §5.2]. If $\delta_{\text{Tol}} > 0$, for instance, a simple way to reset $\mathcal{O}_k$ (and, similarly, $\mathcal{F}_k$) is to initialize $\bar{k} \leftarrow 1$, pick $\alpha \in (0, 1)$, and employ the following rule (e.g., at the beginning of Step 4)

$$\text{If } \Delta_k \leq \alpha \Delta_{\bar{k}} \text{ then choose } \mathcal{O}_{k+1} \supseteq \{k\}, \bar{k} \leftarrow k, \text{ else } \mathcal{O}_{k+1} = \mathcal{O}_k \cup \{k\} \ .$$

That is, the bundle can be reset each time the optimality gap "decreases enough"; this can happen only a finite number of times. Similar rules could check for "substantial changes" in v_k^{low} or v_k^{up} separately.

There is a non-trivial trade-off regarding bundle management. On one hand, keeping the bundle as small as possible may save on master problem time. On the other hand, accruing information is what drives the algorithm, and therefore discarding information too quickly may be very detrimental to convergence rates. A fortiori in the discrete case, only computational experiments can provide guidance on the best way to perform bundle resets.

5 Application to probabilistically constrained optimization

To test our approaches we will consider chance-constrained optimization (CCO) problems of the form

$$f_* := \min \left\{ \, f(x) \; : \; \mathbb{P}[\, g(x,\xi) \leq 0 \,] \geq p \; , \; x \in X \, \right\} \tag{20}$$

where $\xi \in \mathbb{R}^r$ is a random variable, $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, $g = [g_i]_{i \in I}$ is a mapping over a finite index set I such that each $g_i : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}$ is convex in the first argument, and $X \neq \emptyset$ is a bounded convex set. The *joint probabilistic constraints* require that all the inequalities $g_i(x,\xi) \leq 0$ for $i \in I$ hold simultaneously with high enough probability $p \in (0,1]$, and arise in many applications such as water management, finance and power generation (e.g., [2, 56, 63] and references therein). For introductory texts on joint probabilistic programming we refer to [22, 53].

This large class of problems contains cases of different difficulty, even for the same n, r and $|I|$, depending on the underlying assumptions on the probabilistic constraint. For instance, setting $p = 1$ essentially eliminates the difficulty, reducing (20) to a standard convex nonlinear optimization problem, albeit potentially with many constraints [13]. This does not mean that such a problem is trivial, since the functions f or g can be nonsmooth or/and difficult to evaluate. For such cases, specialized approaches can be required: the methods of choice currently being constrained bundle ones [4, 5, 29].

When $p \in (0,1)$ instead, one of the fundamental differentiating factors is whether the distribution of ξ is continuous or discrete. In the former case, one can face hard nonconvex nonlinear optimization problems, and a careful theoretical study of the properties of the probability function (e.g., differentiability [1]) is needed. We will rather consider the case where ξ takes values in a finite set $\Xi = \{ \, \xi^s \; : \; s \in S \, \} \subseteq \mathbb{R}^r$ of possible realizations (or *scenarios*), with associated weights π_s (summing to one). The continuous case can clearly be approximately reduced to the discrete one by drawing an appropriate finite sample; the key question then concerns the minimal sample size which allows to assert feasibility for the original problem with a given confidence level (e.g., [47] and the references therein).

Numerical methods for problems with discrete distributions are, as mentioned in [21, § 2.5], necessarily based on combinatorial techniques. Indeed, there are now $|S|$ blocks of constraints $g(x,\xi^s) \leq 0$, one for each $s \in S$, and (20) requires to minimize f over the intersection of X and all possible ways to select a set of scenarios $P \subseteq S$ such that $\sum_{s \in P} \pi_s \geq p$. In other words, while one is allowed not to satisfy all blocks of constraints $g(x,\xi^s) \leq 0$, the set of these that are not satisfied by the chosen solution x must be a low-probability one [60, Chapter 4]. To develop solution methods, simplifying assumptions are frequently made. A common one is that all the constraint functions g_i, $i \in I$ are separable, i.e., $g_i(x,\xi) = \xi - \tilde{g}_i(x)$ for given concave functions $\tilde{g}_i$. This assumption is crucial for optimization algorithms based on *p-efficient points*, a concept introduced in [52] and used to obtain equivalent problem formulations, as well as necessary and sufficient optimality conditions [22, 53]. Methods based on p-efficient points are diverse: see [23,54] for primal and dual CPM, [25] for cone generation methods, and [24] for augmented Lagrangian and bundle methods. All in all, numerical techniques for this class of problems are well-developed.

When the constraints are not separable, p-efficient approaches are no longer suitable. In this case, one frequently encounters the assumption that the constraints are linear with respect to x, e.g., $g(x,\xi) = A(\xi)x - b(\xi)$. This allows to reformulate (20) as a MINLP, which is actually a MILP if f and X are also linear. This is by far the most widely studied class of CCO problems with finite support and non-separable constraints. For instance, in the recent [46] an approach similar in spirit to combinatorial Benders' cuts is proposed whereby valid inequalities are derived to strengthen the MILP formulation of the problem, making it easier to solve large-scale instances with standard MILP tools. In this analysis, linearity plays a crucial role. The approach is extended to a wider class of problems in [45], but again linearity is essential.

In this work we will neither assume that the constraints g are linear with respect to x, nor separable: our only assumptions are that f and g are convex in x, X is compact and convex, and ξ has finite support Ξ. Then, the probability constraint in (20) can be modeled by a standard *disjunctive reformulation*. That is, a binary variable $z_s \in \{0, 1\}$ for each $s \in S$ is introduced which dictates whether or not the block of constraints $g(x,\xi^s) \leq 0$ is going to be satisfied by the optimal solution x. This requires to estimate, for each $i \in I$ a large enough constant M_i^s that makes the constraint redundant over X:

$$M_i^s \geq \max\{\ g_i(x, \xi^s)\ :\ x \in X\ \}\ . \tag{21}$$

We remark that while (21) is an easy problem in the linear case, as it amounts to maximizing a linear function over a well-behaved convex set, this is in principle no longer true in the nonlinear case, as maximizing a convex function (even a quadratic one) over a convex set (even a hypercube) is in general $\mathcal{NP}$-hard [19]. However, this issue can be tackled in different ways, such as defining an appropriate concave upper approximation of g_i (say the *concave envelope* of g_i over X, e.g., [7]) or approximating X as an ellipsoid which, if g_i is quadratic, makes the problem tractable [19]. We will therefore assume that constants M_i^s are available: then, (20) can be reformulated as a MINLP using

$$\mathbb{P}[\, g(x, \xi) \leq 0\,] \geq p\ \equiv \left\{ \begin{array}{ll} g_i(x, \xi^s) \leq M_i^s z_s & i \in I\ ,\ s \in S \\ \sum_{i \in S} \pi_s\, z_s \leq 1 - p & \\ z_s \in \{0, 1\} & s \in S \end{array} \right\} \equiv\ \Big\{\ G(x) \leq Tz\ ,\ z \in Z\ \Big\} \tag{22}$$

for the obviously defined $G(x) = [g(x, \xi^s)]_{s \in S}$, T and Z. Therefore, (20) fits the general scheme (1), and hence we can solve it via GBD. To the best of our knowledge, GBD has never been used as a main tool for solving CCO programs of this form, although some mechanics of the approach are used in [46] in a less general setting.

5.1 Application: a hybrid robust/chance-constrained model

We consider the minimization of some objective function $f : \mathbb{R}^n \to \mathbb{R}$ subject to linear constraints

$$Ax \leq \xi\ \ , \tag{23}$$

where both ξ and A are subject to uncertainty. It occurs in many cases that there are different sources of uncertainty, not all equally well understood. This setting is of interest for instance in energy management, where x represents an energy production schedule (e.g., [63] for the unit-commitment problem), and (23) means that we wish to produce sufficient energy in all but the most extreme and implausible scenarios. Knowledge of the distribution of ξ (energy demand) is available, since its characterization has received considerable attention, while A is related to the underlying physics of generation plants and/or to the behavior of other generation companies, and much less information is available.

We can therefore employ a *hybrid robust/chance-constrained* approach. Let $A = [a_i]_{i \in I}$; we will assume that the uncertainty about the coefficients matrix can be expressed in the form $a_i(u) = \bar{a}_i + P_i u$, where $\bar{a}_i \in \mathbb{R}^n$, P_i is an $n \times n_i$ matrix, and the *uncertainty set* $u \in \mathcal{U}_i = \{u \in \mathbb{R}^{n_i}\ :\ \|u\| \leq \kappa_i\}$ is the ball of radius κ_i in the ℓ_2 norm. For the sake of notation we define $\mathcal{U} = [\mathcal{U}_i]_{i \in I}$, and we write $A(u)$ for $u \in \mathcal{U}$ to mean $[a_i(u_i)]_{i \in I}$, where $u_i \in \mathcal{U}_i$. On the other hand, $\xi \in \mathbb{R}^m$ is a random variable with known distribution, in our setting represented by a finite set $\varXi$ of realizations (possibly obtained by appropriate sampling). We can then express our requirement under the form of the *robust chance-constraint*

$$\mathbb{P}\big[\, A(u)x \leq \xi \quad \forall u \in \mathcal{U}\,\big] \geq p\ \ . \tag{24}$$

For a fixed ξ, the well-established theory of robust optimization (e.g., [9]) applies: $a_i(u)^{\mathsf{T}} x \leq \xi_i$ for all $u \in \mathcal{U}_i$ if and only if $\max\{\, a_i(u)^{\mathsf{T}} x\ :\ u \in \mathcal{U}_i\,\} \leq \xi_i$, which due to our choice of $a_i(u)$ reduces to

$$\max\{\, a_i(u)^{\mathsf{T}} x\ :\ u \in \mathcal{U}_i\,\} = \bar{a}_i^{\mathsf{T}} x + \max\{\, (P_i^{\mathsf{T}} x)^{\mathsf{T}} u\ :\ u \in \mathcal{U}_i\,\} = \bar{a}_i^{\mathsf{T}} x + \kappa_i \|P_i^{\mathsf{T}} x\|\ \ .$$

Consequently, (24) reduces to $\mathbb{P}\big[\, \bar{a}_i^{\mathsf{T}} x + \kappa_i \|P_i^{\mathsf{T}} x\| \leq \xi_i \quad i \in I\,\big] \geq p$, which readily falls in the setting of §2, as $g_i(x, \xi) = \bar{a}_i^{\mathsf{T}} x + \kappa_i \|P_i^{\mathsf{T}} x\| - \xi_i$ is convex in x (since, obviously, $\kappa_i > 0$). The approach easily extends to $\mathcal{U}$ being defined by any convex ℓ_p norm, resulting in the conjugate norm in the constraint above, but this is well-known and we stick to the ℓ_2 case for simplicity; here, g_i is a Second-Order Cone representable function.

It is interesting to remark that (24) implies the weaker condition

$$\mathbb{P}\big[\,A(u)x \leq \xi\,\big] \geq p \quad \forall u \in \mathcal{U} \ . \tag{25}$$

Indeed, because in (24) the probability constraint has to hold for the *maximum over all* $u \in \mathcal{U}$ of $A(u)$, a fortiori it has to hold for any specific choice. The inverse is not true in general, as shown by the following counterexample.

Example 1 Take $\mathcal{U} = \{u_1, u_2\}$ and $\Xi = \{\xi_1, \xi_2\}$ with probability $\pi_1 = \pi_2 = 0.5$. Moreover, suppose that there exists $\bar{x}$ such that

$$A(u_1)\bar{x} \leq \xi_1 \text{ but } A(u_1)\bar{x} \not\leq \xi_2 \quad , \quad A(u_2)\bar{x} \not\leq \xi_1 \text{ but } A(u_2)\bar{x} \leq \xi_2 \ .$$

Therefore, $\bar{x}$ is feasible for (25) for the choice $p = 0.5$: $\mathbb{P}[\,A(u)\bar{x} \leq \xi\,] \geq 0.5$ however chosen $u \in \mathcal{U}$. However, $\mathbb{P}[\,A(u)\bar{x} \leq \xi \quad \forall u \in \mathcal{U}\,] = \mathbb{P}[\emptyset] = 0$. Numerical data may be picked as $\bar{x} = (1,1)$, $\xi_1 = (2,0)$, $\xi_2 = (0,2)$, $A(u_1) = \begin{pmatrix} 1\,1 \\ 0\,0 \end{pmatrix}$, $A(u_2) = \begin{pmatrix} 0\,0 \\ 1\,1 \end{pmatrix}$. □

The example shows that the weaker model (25) may not behave satisfactorily: since we do not know the underlying distribution of u, when decision x has been taken and $u \in \mathcal{U}$ turns up, the actual probability $\mathbb{P}[\,A(u)x \leq \xi\,]$ may turn out to be arbitrarily low. This is in fact analogous to the difference between joint and individual probabilistic constraints (e.g., [2]).

5.2 Generation of problem instances

For our experiments we focussed on problems of the form

$$\min\Big\{\, c^{\mathsf{T}}x \;:\; \mathbb{P}\big[\,A(u)x \leq \xi \quad \forall u \in \mathcal{U}\,\big] \geq p \ , \ \ 0 \leq x \leq \bar{x} \,\Big\} \ , \tag{26}$$

where $\bar{x} \in \mathbb{R}^n$ is a bound, the objective function is linear, $\xi \in \mathbb{R}^r$ has finite support Ξ and $A(u)$, $\mathcal{U}$ are as described in the previous paragraph.

Instances of problem (26) were generated by using the following procedure. We begin by setting the problem dimensions n, $|I|$ and $|S|$. We also set $p_i = n$ the common dimension of the matrices P_i involved in the constraints. Finally, we took $\kappa_i = 1/2$, $\pi_s = 1/|S|$ uniformly, and $p = 0.8$. The next step consists of randomly generating the matrix $\bar{A} = [\bar{a}_i]_{i\in I}$, with entries in $[-10, 10]$. The matrices P_i and vector c were generated likewise with entries in $[-1, 1]$. Because coefficient matrices are usually sparse in real-world problems, we generated both $\bar{A}$ and P_i sparse. Finally, we generated a random candidate solution x^c with entries in $[0, 10]$ and we computed $\bar{\xi}$ such that $\bar{\xi}_i = \bar{a}_i^{\mathsf{T}} x^c + \kappa_i \|P_i^{\mathsf{T}} x^c\|$ for $i \in I$. Scenarios for ξ were generated as $\xi^s = \bar{\xi} + r^s$, where r^s was chosen in two different ways. For (at least) a fraction p of the scenarios, r^s was chosen with entries in $[0, 20]$, so that $\xi^s \geq \bar{\xi}$. For the remaining (at most) $1-p$ fraction of scenarios (properly rounding taking place), r^s is allowed to have entries spanning $[-20, 20]$. Thus, it is immediate to realize that x^c is feasible for problem (26) by construction. The constant M, identical for all scenarios, has been set by carefully analyzing the data of the instances with an ad-hoc approach.

The choice of the ℓ_2 norm for $\mathcal{U}$ means that problem (3) has a Mixed-Integer Second-Order Cone formulation, that can be directly solved by off-the-shelf tools like `Cplex` provided that it is reformulated as a Quadratically-Constrained Quadratic Problem. By introducing auxiliary variables y_i for all $i \in I$, (26) can be rewritten by means of the following constraints

$$\begin{aligned} &\bar{a}_i^{\mathsf{T}} x + \kappa_i y_i - \xi_i^s \leq M_i^s z_s && i \in I \ , \ \ s \in S \\ &x^{\mathsf{T}}(P_i P_i^{\mathsf{T}})x \leq y_i^2 && i \in I \end{aligned} \tag{27}$$

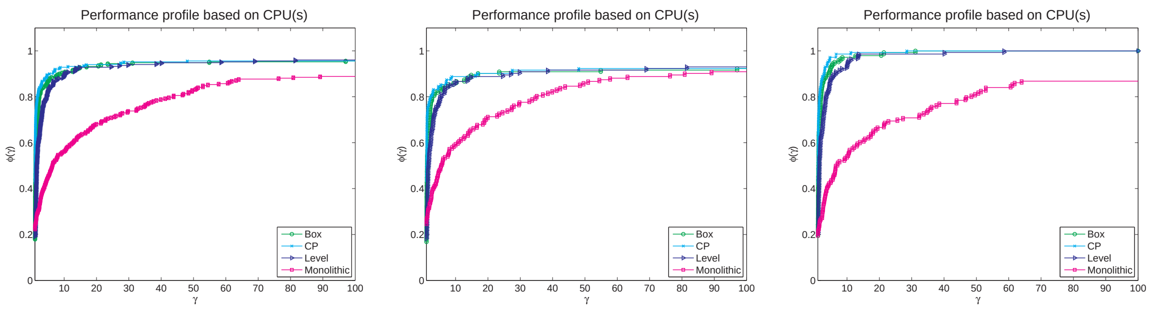
which are appropriately dealt with by `Cplex` even if the Hessian matrix of (27) is not, strictly speaking, positive semi-definite. Subproblem (2) can be written and solved with the same tools.

5.3 Setup and results

We have generated several problem instances as follows. First we have chosen n and $|I|$ ranging over $\{50, 100\}$ and $|S| \in \{50, 100, 500\}$, for a total of 12 combinations. We also varied the sparsity of $\bar{A}$ and P_i in the set $\{1\%, 0.1\%, 0.01\%\}$, but only considering the combinations of adjacent sparsity levels, i.e., avoiding the combinations $(1\%, 0.01\%)$ and $(0.01\%, 1\%)$, for a total of 7 cases. For each of the above we generated 3 instances changing the seed of the random number generator, for a total of $12 \cdot 7 \cdot 3 = 252$ instances. We have experimented with two different values for δ_{Tol}: 10^{-4}, the default optimality tolerance for MINLPs, and 10^{-3}, considered "coarse" but still acceptable in some cases. Both are to be intended as *relative*, which, via appropriate scalings, does not impact our analysis in §3 and §4, where *absolute* tolerances were used for simplicity. We also set a time limit for each method of 100000 seconds, on a cluster with Intel Xeon X5670 CPUs, each with 8 Gb of reserved memory. Both the "monolithic" approach and all the optimization problems in the GBD one (the MILP master problem and the SOCP subproblems) have been solved with `Cplex` 12.4, single-threaded. Other than that, the time limit and optimality tolerance, no other parameters of `Cplex` were tuned.

We have compared the "`Monolithic`" approach and three variants of GBD: `CP`, the CPM of Algorithm 1, `Box`, the TRICCPM of Algorithm 3, and `Level`, the LICCPM of Algorithm 4. In all cases a primal-dual oracle was employed, where `Cplex` was used to solve the SOCP formulation of the subproblem (cf. (27)). Several experiments were carried out with a dual oracle, but it was found not competitive in this setting, and we do not report the corresponding results. The TRICCPM changes the stability center whenever a better solution is found, i.e., $\beta = 0$, and moves through box sizes $\{0.005, 0.5, 1\}\,|S|$. The LICCPM uses mechanism (19) described after Theorem 4 with $\alpha = 0.9$, updates the lower bound v_k^{low} by also solving the standard master problem (6) at every iteration as discussed after Theorem 4, systematically chose the last iterate as the next stability center $\hat{z}^k$, and uses a stabilization parameter close to the value of 0.18 which is optimal in the continuous setting [43]. Some tuning was performed, for instance about other choices for the center update rule, but the results were quite stable.

We compare the solvers by means of performance profiles [27], which read as follows: the parameter γ represents a scalar value, and $\phi(\gamma)$ the percentage of problems on which a given solver was no slower than γ times the fastest solver on that particular problem. The value $\phi(0)$ shows which solver is the fastest one and the value $\phi(\infty)$ is the percentage of instances that the solver managed to solve. Looking at $\phi(0)$ in Figure 1(a) shows that `CP` is faster in roughly 40% of the instances, whereas the other methods are roughly each at 20%. The GBD approaches are noticeably more robust than the monolithic one.

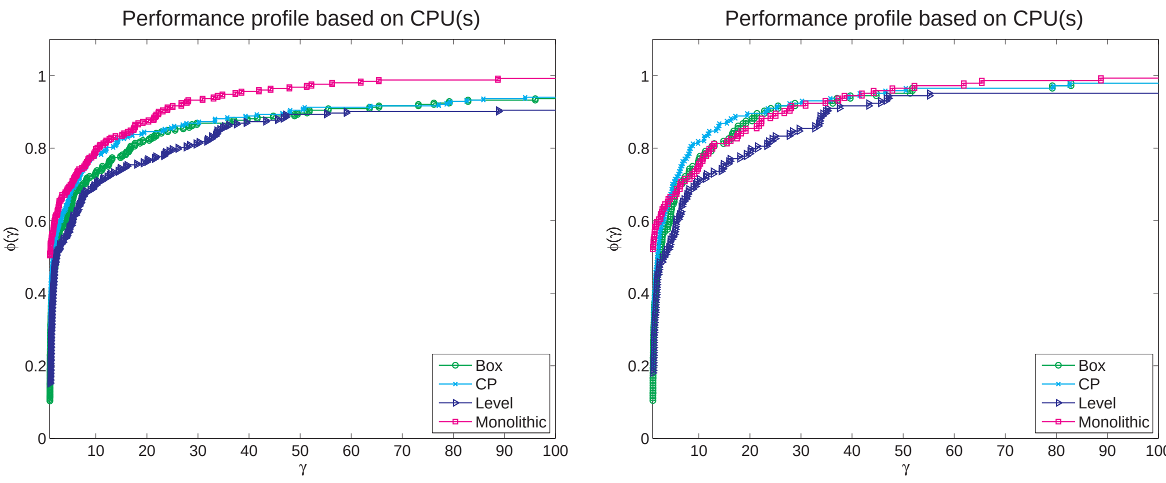


(a) All instances (b) Instances with low sparsity (c) Instances with high sparsity

Fig. 1 Performance profiles for the different methods based on cpu time

The impact of sparsity is illustrated on Figures 1(b) and 1(c), which report performance profiles restricted respectively to "low sparsity" instances ($\{1\%, 0.1\%\}$) and "high sparsity" ones ($\{0.1\%, 0.01\%\}$): `Monolithic` gets more and more outperformed by the GBD as sparsity increases. This shows the potential of decomposition methods, since real-world problems are often highly sparse (even below 0.001%).

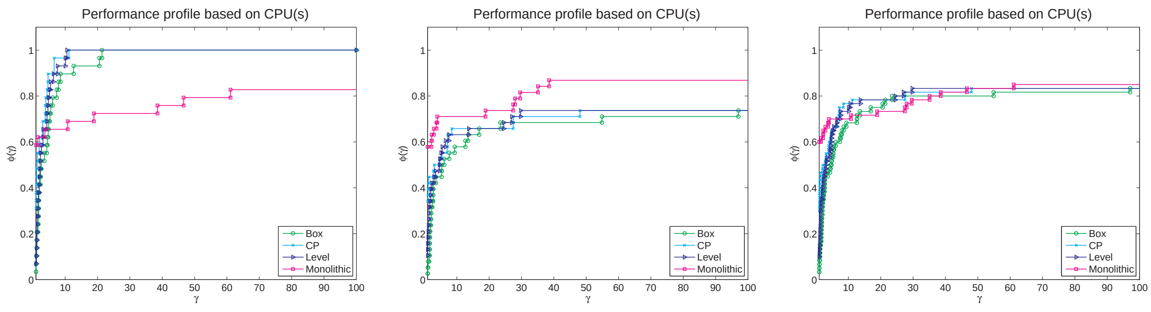
The impact of the stopping tolerance can be gauged by comparing Figure 1(a) with Figure 2(a), where the performances are reported with $\delta_{\mathrm{Tol}} = 10^{-3}$ is required. For this coarser tolerance, the monolithic approach outperforms the GBD ones. However, this is mostly true for dense instances only: as shown in Figure 2(b), for sparser instances the GBD remain competitive even at a lower precision.



(a) All instances (b) Instances with high sparsity

Fig. 2 Performance profiles for the different methods based on cpu time with $\delta_{\mathrm{Tol}} = 10^{-3}$

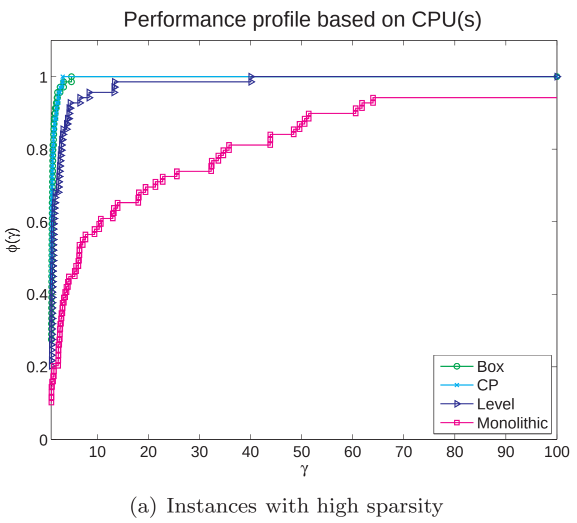
One issue with performance profiles is that they do not discriminate among instances of widely varying "difficulty". That is, two solvers tested on two instances such that the first one has a running time of 2 and 1000 while and the second one has a running time of 1 and 2000 (in whichever units) would show to have exactly the same performance profiles, while one may be interested in knowing that the first solver is "better on harder instances although worse on easier ones". To investigate this issue we subdivided our instances in three classes: easy if the fastest solver takes less then one minute, intermediate if it takes between 1 and 10 minutes, and hard otherwise. Figures 3(a), 3(b) and 3(c) show the performance of the solvers on the easy instances (with $\delta_{\mathrm{Tol}} = 10^{-4}$.



(a) Instances with high sparsity (b) Instances with low sparsity (c) All instances

Fig. 3 Performance profiles for the different methods based on cpu time on the easy instances

The figures show that while `Monolithic` is competitive for easy instances, this is only so for dense problems: for sparse ones, although `Monolithic` is fastest in around 60% of the cases, the solvers based on GBD are far more robust. The picture is even clearer for hard instances, as shown in Figures 4(a) and 4(b): there, `Monolithic` is significantly outperformed also on low sparsity instances. Similar results were obtained for the intermediate instances, and therefore we avoided to report the corresponding profiles.

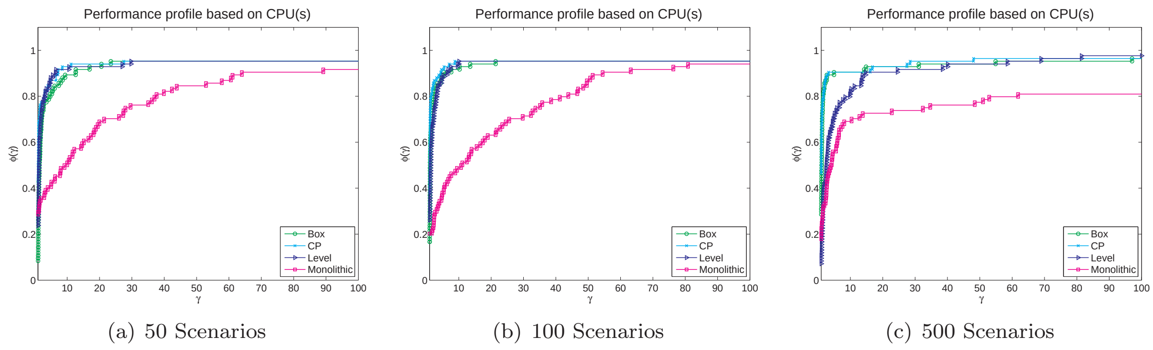


(a) Instances with high sparsity

(b) Instances with low sparsity

Fig. 4 Performance profiles for the different methods based on cpu time on the hard instances

The impact of the number of scenarios is illustrated in Figures 5(a), 5(b) and 5(c). The results show that GBD is quite stable as $|S|$ varies, whereas `Monolithic` is less and less robust as $|S|$ increases.



(a) 50 Scenarios (b) 100 Scenarios (c) 500 Scenarios

Fig. 5 Performance profiles for the different methods based on cpu time, discriminated according to the number of scenarios $|S|$

Finally, we have performed some experiments related to the effect of setting a target `tar` $< \infty$. This has proved somewhat more complex than anticipated for a number of reasons. On one hand, the dual oracle, which is the one to which the approach is most suited, was not particularly efficient in our case. On the other hand, the *unfeasible start* primal-dual approach implemented in `Cplex` turned out to produce feasible solutions (in particular, dual ones) rather late in the optimization. Hence, even intercepting them as early as possible using the appropriate callbacks did not significantly improve the running times. We therefore resorted to a somewhat abstract setting, by executing the dual oracle *twice* at each iteration: once with $\texttt{tar}_k = \infty$, and once with a finite target (the best current value). This has been done in the ICCPM, and only the dual solution of the exact oracle has been used to compute the feasibility cuts. We have then compared the oracle time when the target is specified with that when it is not. On average over all the 252 data sets, using the target resulted in a reduction of oracle time of about 77%, with a standard deviation of around 20% and cases where the reduction was above 99%. Only in two instances the running time increased, by a relatively minor fraction. Although these experiments disregard the impact on the convergence speed of the different feasibility cuts produced, they do indicate that using an inexact oracle may be beneficial.

All in all, our results prove that, on this class of problems, and for our specific test set, the GBD approaches are in general competitive with the monolithic one. In particular, `Monolithic` is competitive (but not necessarily the best choice on sparse instances) when $\delta_{\text{Tol}} = 10^{-3}$, and should be preferred only for "easy" and "dense" instances with $\delta_{\text{Tol}} = 10^{-4}$. In all the other cases, the GBD approaches are significantly better.

6 Conclusions and perspectives

In this paper we have studied the combination of two approaches for improving the performances of algorithms for the minimization of a convex function $v(z)$, given by a first-order oracle, over a finite domain Z. One idea is to relax the conditions on the information produced by the oracle, requiring it to only be an *informative on-demand inexact* one (10), so as to make each iteration cheaper. The second idea is to employ different forms of stabilization (trust region and level) to reduce the number of oracle calls. Employing both techniques simultaneously requires some care in the handling of the accuracy parameters; our convergence results seem to require very weak conditions, which basically show that the objective function may need to be computed accurately only in a small fraction of the iterates. Our analysis should also plainly extend to the case setting of [70], i.e., the feasible set Z is not finite but is bounded, and ∂v is locally bounded. Our results would then extend these obtained in [70], under stricter assumptions on the oracle and $\delta_{\text{Tol}} > 0$.

Our analysis is primarily interesting for improving the performances of Generalized Benders' Decomposition approaches. An interesting property of oracles with on-demand accuracy is that they are able to provide linearizations even in absence of constraint qualification for the underlining convex problem (defining the value function). Moreover, linearizations can even be computed at points not belonging to the domain of the value function, which may allow to implement the algorithm without using feasibility cuts (unless the problem is infeasible). We remark that while we discuss the inexact oracle in the case where the subproblem is convex, our analysis also applies (and it is possibly even more relevant) to the case where it is a hard problem (e.g., [14, 40, 44, 59]), which makes obtaining good upper and lower estimates even more time consuming.

In order to test the computational significance of the developed techniques, we have applied them on a class of hybrid Robust and Chance-Constrained Optimization problems, arising when a linear program is subject to two different sources of uncertainty, that need to be dealt with with different techniques (an uncertainty set plus a finite set of scenarios). These problems allow a monolithic formulation that a commercial solver such as `Cplex` can solve; however, our results show that GBD approaches are often much more efficient, in particular for high sparsity, a large number of scenarios, and a higher final accuracy. Also, the experiments indicate that the ideas developed in this work are promising to improve the performances of decomposition techniques.

To conclude, this work is significant in three areas: general algorithms for the minimization of oracle-provided convex functions over discrete sets, Generalized Benders' Decomposition approaches, and Chance-Constrained Optimization. Of course, further improvements are possible in all the three areas. However, we believe that our results already show that the combination of different concepts such as *probability valid inequalities*, *strengthening formulations* for combinatorial problems and oracles with *on-demand accuracy* from nonsmooth optimization will prove fruitful for solving problems as (20).

Acknowledgements The authors gratefully acknowledge financial support from the Gaspard-Monge program for Optimization and Operations Research (PGMO) project "Consistent Dual Signals and Optimal Primal Solutions". The first and second authors would also like to acknowledge networking support by the COST Action TD1207.

References

1. van Ackooij, W., Henrion, R.: Gradient formulae for nonlinear probabilistic constraints with Gaussian and Gaussian-like distributions. SIAM Journal on Optimization **24**(4), 1864–1889 (2014)

2. van Ackooij, W., Henrion, R., Möller, A., Zorgati, R.: Joint chance constrained programming for hydro reservoir management. Optimization and Engineering **15**, 509–531 (2014)
3. van Ackooij, W., Malick, J.: Decomposition algorithm for large-scale two-stage unit-commitment. Annals of Operations Research **238**(1), 587–613 (2016). DOI 10.1007/s10479-015-2029-8
4. van Ackooij, W., de Oliveira, W.: Level bundle methods for constrained convex optimization with various oracles. Computation Optimization and Applications **57**(3), 555–597 (2014)
5. van Ackooij, W., Sagastizábal, C.: Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. SIAM Journal on Optimization **24**(2), 733–765 (2014)
6. Baena, D., Castro, J., Frangioni, A.: Stabilized Benders methods for large-scale combinatorial optimization: applications to data privacy (2015)
7. Bao, X., Sahinidis, N., Tawarmalani, M.: Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. Optimization Methods and Software **24**, 485–504 (2009)
8. Ben Amor, H., Desrosiers, J., Frangioni, A.: On the Choice of Explicit Stabilizing Terms in Column Generation. Discrete Applied Mathematics **157**(6), 1167–1184 (2009)
9. Ben-Tal, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization. Princeton University Press (2009)
10. Ben-Tal, A., Nemirovski, A.: Lectures on Modern Convex Optimization: Analysis, Algorithms, Engineering Applications. MPS-SIAM Series on Optimization. SIAM, Philadelphia (2001)
11. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**(1), 238–252 (1962)
12. Boyd, S., Vandenberghe, L.: Convex optimization. Available at http://www.stanford.edu/ boyd/cvxbook **ISBN 0 521 83378 7** (2006)
13. Calafiore, G.C., Campi, M.C.: Uncertain convex programs: Randomized solutions and confidence levels. Mathematical Programming **102**(1), 25–46 (2005)
14. Caroe, C.C., Tind, J.: L-shaped decomposition of two-stage stochastic programs with integer recourse. Math. Programming **83**, 451–464 (1998)
15. Codato, G., Fischetti, M.: Combinatorial benders' cuts for mixed-integer linear programming. Operations Research **54**(4), 756–766 (2006)
16. Costa, A.M.: A survey on benders decomposition applied to fixed-charge network design problems. Computers & Operations Research **32**(6), 14291450 (2005)
17. d'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: On Interval-Subgradient Cuts and No-Good Cuts. Operations Research Letters **38**, 341–345 (2010)
18. d'Antonio, G., Frangioni, A.: Convergence Analysis of Deflected Conditional Approximate Subgradient Methods. SIAM Journal on Optimization **20**(1), 357–386 (2009)
19. de Klerk, E.: The complexity of optimizing over a simplex, hypercube or sphere: a short survey. Central European Journal of Operations Research **16**(2), 111–125 (2008)
20. de Oliveira, W.: Regularized nonsmooth optimization methods for convex minlp problems. TOP pp. 1–28 (2016). DOI 10.1007/s11750-016-0413-4
21. Dentcheva, D.: Optimization models with probabilistic constraints. In: G. Calafiore, F. Dabbene (eds.) Probabilistic and Randomized Methods for Design under Uncertainty, 1st edn., pp. 49–97. Springer (2006)
22. Dentcheva, D.: Optimisation models with probabilistic constraints. In: A. Shapiro, D. Dentcheva, A. Ruszczyński (eds.) Lectures on Stochastic Programming. Modeling and Theory, *MPS-SIAM series on optimization*, vol. 9, pp. 87–154. SIAM and MPS, Philadelphia (2009)
23. Dentcheva, D., Lai, B., Ruszczyński, A.: Dual methods for probabilistic optimization problems. Mathematical Methods of Operations Research **60**(2), 331–346 (2004)
24. Dentcheva, D., Martinez, G.: Regularization methods for optimization problems with probabilistic constraints. Math. Programming (series A) **138**(1-2), 223–251 (2013)
25. Dentcheva, D., Prékopa, A., Ruszczyński, A.: Concavity and efficient points for discrete distributions in stochastic programming. Mathematical Programming **89**, 55–77 (2000)
26. Dinter, J.V., Rebenack, S., Kallrath, J., Denholm, P., Newman, A.: The unit commitment model with concave emissions costs: a hybrid benders decomposition with nonconvex master problems. Annals of Operations Research **210**(1), 361–386 (2013)
27. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002). URL `http://dx.doi.org/10.1007/s101070100263`
28. Fábián, C.: 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.), pp. 35–55 (2000)
29. Fábián, C., Wolf, C., Koberstein, A., Suhl, L.: Risk-averse optimization in two-stage stochastic models: computational aspects and a study. SIAM Journal on Optimization **25**(1), 28–52 (2015)
30. Feltenmark, S., Kiwiel, K.: Dual applications of proximal bundle methods, including lagrangian relaxation of nonconvex problems. SIAM Journal on Optimization **10**(3), 697–721 (2000)
31. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming **98**(1-3), 23–47 (2003)
32. Fischetti, M., Salvagnin, D., Zanette, A.: A note on the selection of Benders cuts. Mathematical Programming **124**(1), 175–182 (2010)
33. Floudas, C.A.: Generalized benders decomposition. In: C.A. Floudas, P.M. Pardalos (eds.) Encyclopedia of Optimization, 2nd edn., pp. 1163–1174. Springer - Verlag (2009)
34. Frangioni, A.: Generalized bundle methods. SIAM Journal on Optimization **13**(1), 117–156 (2002)
35. Frangioni, A., Gendron, B.: A stabilized structured dantzig-wolfe decomposition method. Mathematical Programming B **104**(1), 45–76 (2013)
36. Frangioni, A., Gorgone, E.: Generalized bundle methods for sum-functions with "easy" components: Applications to multicommodity network design. Mathematical Programming **145**(1), 133–161 (2014)

37. Frangioni, A., Lodi, A., Rinaldi, G.: New approaches for optimizing over the semimetric polytope. Mathematical Programming **104**(2-3), 375–388 (2005)
38. Geoffrion, A.M.: Generalized benders decomposition. Journal of Optimization Theory and Applications **10**(4), 237–260 (1972)
39. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II, 2nd edn. No. 306 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag Berlin Heidelberg (1996)
40. Hooker, J.N., Ottosson, G.: Logic-based benders decomposition. Math. Programming **96**, 33–60 (2003)
41. Kelley, J.: The cutting-plane method for solving convex programs. Journal of the Society for Industrial and Applied Mathematics **8**(4), 703–712 (1960)
42. Kolokolov, A., Kosarev, N.: Analysis of decomposition algorithms with benders cuts for p-median problem. Journal of Mathematical Modelling and Algorithms **5**(2), 189–199 (2006)
43. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Programming **69**(1), 111–147 (1995)
44. Li, X., Chen, Y., Barton, P.I.: Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems. Ind. Eng. Chem. Res. **51**(21), 7287–7299 (2012)
45. Liu, X., Küçükyavuz, S., Luedtke, J.: Decomposition algorithm for two-stage chance constrained programs. Mathematical Programming Series B pp. 1–25 (2014). DOI 10.1007/s10107-014-0832-7
46. Luedtke, J.: A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. Mathematical Programming **146**(1-2), 219–244 (2014)
47. Luedtke, J., Ahmed, S.: A sample approximation approach for optimization with probabilistic constraints. SIAM Journal on Optimization **19**, 674–699 (2008)
48. Marsten, R., Hogan, W., Blankenship, J.: The BOXSTEP method for large-scale optimization. Operations Research **23**(3), 389–405 (1975)
49. Oliveira, F., Grossmann, I., Hamacher, S.: Accelerating Benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. Computers & Operations Research **49**(1), 47–58 (2014)
50. de Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on demand accuracy. Optimization Methods and Software **29**(6), 1180–1209 (2014)
51. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Prog. Series B **148**, 241–277 (2014)
52. Prékopa, A.: Dual method for a one-stage stochastic programming problem with random rhs obeying a discrete probabiltiy distribution. Z. Operations Research **34**, 441–461 (1990)
53. Prékopa, A.: Probabilistic programming. In: A. Ruszczyński, A. Shapiro (eds.) Stochastic Programming, *Handbooks in Operations Research and Management Science*, vol. 10, pp. 267–351. Elsevier, Amsterdam (2003)
54. Prékopa, A., Vízvári, B., Badics, T.: Programming under probabilistic constraints with discrete random variable. In: F. Giannessi, S. Komlósi, T. Rapcsák (eds.) New Trends in Mathematical Programming : Hommage to Steven Vajda, *Applied Optimization*, vol. 13, pp. 235–255. Springer (1998)
55. Ruszczyński, A.: Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. Mathematical Programming **93**, 195–215 (2002)
56. Ruszczyński, A.: Decomposition methods. In: A. Ruszczyński, A. Shapiro (eds.) Stochastic Programming, *Handbooks in Operations Research and Management Science*, vol. 10, pp. 141–211. Elsevier, Amsterdam (2003)
57. Sahiridis, G.K.D., Minoux, M., Ierapetritou, M.G.: Accelerating benders method using covering cut bundle generation. International Transactions In Operational Research **17**, 221–237 (2010)
58. Santoso, T., Ahmed, S., Goetschalcks, M., Shapiro, A.: A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research **167**(1), 96–115 (2005)
59. Sen, S., Sherali, H.: Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. Mathematical Programming **106**, 203–223 (2006)
60. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on Stochastic Programming. Modeling and Theory, *MPS-SIAM series on optimization*, vol. 9. SIAM and MPS, Philadelphia (2009)
61. Sherali, H., Lunday, B.J.: On generating maximal nondominated benders cuts. Annals of Operations Research **210**(1), 57–72 (2013)
62. van Slyke, R., Wets, R.B.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM Journal of Applied Mathematics **17**, 638–663 (1969)
63. Tahanan, M., van Ackooij, W., Frangioni, A., Lacalandra, F.: Large-scale unit commitment under uncertainty: a literature survey. 4OR **13**(2), 115–171 (2015). DOI 10.1007/s10288-014-0279-y
64. Tran-Dinh, Q., Necoara, I., Diehl, M.: Fast inexact decomposition algorithms for large-scale separable convex optimization. Optimization (to appear) pp. 1–33 (2015). DOI 10.1080/02331934.2015.1044898
65. Wentges, P.: Accelerating benders' decomposition for the capacitated facility location problem. Mathematical Methods of Operations Research **44**(2), 267–290 (1996)
66. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optimization and Engineering **3**, 253–280 (2002)
67. Wolf, C., Fábián, C.I., Koberstein, A., Stuhl, L.: Applying oracles of on-demand accuracy in two-stage stochastic programming a computational study. European Journal of Operational Research **239**(2), 437–448 (2014)
68. Yang, Y., Lee, J.M.: A tighter cut generation strategy for acceleration of benders decomposition. Computers and Chemical Engineering **44**, 84–93 (2012)
69. Zakeri, G., Philpott, A., Ryan, D.M.: Inexact cuts in benders decomposition. SIAM Journal on Optimization **10**(3), 643–657 (2000)
70. Zaourar, S., Malick, J.: Quadratic stabilization of benders decomposition pp. 1–22 (2014). Draft submitted; Privately communicated
71. Zappe, C.J., Cabot, A.V.: The application of generalized benders decomposition to certain nonconcave programs. Computers Math. Applic. **21**(6/7), 181–190 (1991)