Università di Pisa

Dipartimento di Informatica

Technical Report: TR-15-01

Inexact stabilized Benders' decomposition approaches to chance-constrained problems with finite support

Wim van Ackooij
EDF R&D OSIRIS
1, avenue du Général de Gaulle, 92141 Clamart Cedex, France
wim.van-ackooij@edf.fr

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

Welington de Oliveira
Instituto Nacional de Matemática Pura e Aplicada – IMPA
Rua Dona Castorina 110, 22460-320, Rio de Janeiro, Brazil
wlo@impa.br

February 23, 2015

ADDRESS: Largo B. Pontecorvo 3, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

Inexact stabilized Benders' decomposition approaches to chance-constrained problems with finite support

Wim van Ackooij
EDF R&D OSIRIS
1, avenue du Général de Gaulle, 92141 Clamart Cedex, France
wim.van-ackooij@edf.fr

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

Welington de Oliveira
Instituto Nacional de Matemática Pura e Aplicada – IMPA
Rua Dona Castorina 110, 22460-320, Rio de Janeiro, Brazil
wlo@impa.br

February 23, 2015

Abstract

Motivated by a class of chance-constrained optimization problems, we explore modifications of the (generalized) Benders' decomposition approach. The chance-constrained problems we consider involve a random variable with an underlying discrete distribution, are convex in the decision variable, but their probabilistic constraint is neither separable nor linear. The variants of Benders' approach we propose exploit advances in cutting-plane procedures developed for the convex case. Specifically, the approach is stabilized in the two ways; via a proximal term/trust region in the L_1 norm, or via a level constraint. Furthermore, the approaches can use inexact oracles, in particular *informative on-demand inexact* ones. The simultaneous use of the two features requires a nontrivial convergence analysis; we provide it under what would seem to be the weakest possible assumptions on the handling of the two parameters controlling the oracle (target and accuracy), strengthening earlier know results. Numerical performance of the approaches are assessed on a class of hybrid robust and chance-constrained conic problems. 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*

1 Introduction

This work is motivated by chance-constrained optimization (CCO) problems of the form

$$f_* := \min \left\{ \; f(x) \; : \; \mathbb{P}[\, g(x,\xi) \leq 0 \,] \geq p \; , \; x \in X \; \right\} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, $g = [g_i]_{i \in I}$ is a mapping over a finite index set I such that each $g_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is convex in the first argument, and $X \neq \emptyset$ is a bounded convex set that can be "conveniently represented" in a sense specified below. More generally, f and g can be taken to be finite-valued only in an open neighbourhood of X. The probabilistic constraint $\mathbb{P}[\, g(x,\xi) \leq 0 \,] \geq p$ means that we wish that all the random inequalities $g_i(x,\xi) \leq 0$ for $i \in I$ hold simultaneously with high enough probability $p \in (0,1)$. These are referred to as *joint probabilistic constraints*, and arise in many applications such as water management, finance and power generation (e.g., [58,64,69] and references therein). For introductory texts on joint probabilistic programming we refer to [23,41,53,54]. See also [8,26,40,48,52,55,72] and [42] for classic works on this topic.

This class of problems is very large, and contains cases of wildly different difficulty, even for the same n, m and $|I|$, depending on the assumptions made on the probabilistic constraint. For instance, the choice $p = 1$ essentially eliminates the probabilistic constraint and its underlying difficulties, reducing (1) to a standard convex nonlinear optimization problem, albeit potentially with many constraints. This choice was investigated e.g., in [13]. This is not to say that such a problem is necessarily trivial to solve, since the functions f or g can be nonsmooth or/and difficult to evaluate. Consequently, specialized approaches can be required to deal with such cases, the methods of choice currently being constrained bundle ones [67,71].

When $p \in (0,1)$ instead, one of the fundamental differentiating factors is whether the distribution of ξ is assumed to be 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 is needed (e.g., differentiability, see for instance: [66,68]). We will rather follow the alternative path and assume that ξ takes values in a finite set $\Xi = \{ \, \xi^s \; : \; s \in S \, \} \subseteq \mathbb{R}^m$ of possible realizations (or *scenarios*), with associated weights π_s (summing to one). Popular sample based approaches reduce to the discrete case, even if ξ is a continuously distributed random variable, by drawing an appropriate finite sample. Then the key question concerns the sample size which allows to assert feasibility for the original unsampled problem with a given confidence level. The analysis in [48] provides such a link and considers bounds on optimality as well. In [13] the idea is to enforce the constraints on all drawn samples, and a link between the sample size and feasibility for the original problem is exhibited. That approach was extended for large safety levels p in [50,51], while heuristic ways of discarding scenarios has been investigated in [14].

Numerical methods for problems of the form (1) with underlying discrete distributions are therefore, as mentioned in [22, § 2.5], necessarily based on combinatorial techniques; see for instance [8,26,52, 55,72] and [47]. Indeed, there are now $|S|$ blocks of constraints $g(x,\xi^s) \leq 0$, one for each $s \in S$, and (1) 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 [62, Chapter 4]. However, simplifying assumptions are frequently made on the structure of the problem. A common one is that all the constraint functions g_i, $i \in I$ are separable, i.e., $g_i(x,\xi)$ is of the form $\xi - \tilde{g}_i(x)$ 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 [22,23,54].

We mention that p-efficient point based algorithms are diverse: see [55] and [24] for primal and dual cutting-plane approaches, [26, 72] for cone generation methods, and [25] for augmented Lagrangian and bundle methods. All in all, numerical techniques for this class of problems are well-developed.

However, when the constraints are not separable, p-efficient approaches are no longer suitable. In this case, one frequently encounters the assumption that the constraints $g(x, \xi)$ are linear with respect to variable x, e.g., $g(x, \xi) = A(\xi)x - b(\xi)$, where both the coefficient matrix $A(\xi)$ and the right-hand side $b(\xi)$ depend on the random parameter ξ. This allows to reformulate the problem as a Mixed-Integer Linear Program (MILP) if f and X are also linear, and as a linearly-constrained Mixed-Integer NonLinear Program (MINLP) otherwise. From a practical viewpoint, this is perhaps the most general setting algorithmically addressed in the literature of chance-constrained problems with finite support and non-separable constraints. This configuration was for instance adopted in the recent [47], where an approach similar in spirit to combinatorial Benders' cuts is proposed whereby powerful valid inequalities are derived which significantly strengthen the MILP formulation of the problem, making it easier to solve large-scale instances with standard MILP tools. However, linearity plays a crucial role there.

In this work we will neither assume that the constraints g are linear with respect to x, nor separable: our only (mild) assumptions are that f and g are convex in x, and X is compact and convex. Under these assumptions, one can solve (1) via the *(generalized) Benders' decomposition* approach [38], which is basically Kelley's cutting-plane method [45] with a discrete master problem. To the best of our knowledge, (generalized) Benders' decomposition has never been used as a main tool for solving CCO programs of this form, although some mechanics of the approach are used e.g., in [47] in a less general setting.

Motivated by this application, we discuss two classes of improvements over the standard approach. On one hand we allow the cut generation subproblem to be solved only inexactly, hopefully reducing the corresponding running time. This has been done in [76] but only for the continuous case. Furthermore, the sequence $\{\varepsilon_i\}$ of errors is chosen a priori to converge to zero. Oracles with on-demand accuracy (see the formal definition in (11) below) are employed in the recent [75] in the context of stochastic programming, but again in the continuous case. On the other hand we propose two different stabilization techniques that can decrease the number of iterations required to attain convergence, and/or the average computational cost of the master problem. Both improvements are inspired by similar devices developed for convex cutting-plane algorithms, but are adapted to the current set of *combinatorial* cutting-plane approaches. Stabilization of combinatorial Benders' approaches has already been proposed in [2], but with exact oracles. The only previous work we are aware of where both approaches are used simultaneously is [77], but only for the level stabilization. Moreover, our convergence arguments are significantly more refined. In the continuous case, the simultaneous use of stabilization and inexact oracles requires a technically rather involved theoretical analysis; see [21] for the proximal case, and [20, 29, 67] for the level one. 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. Hopefully, the most efficient rule for any given class of applications can be selected computationally.

We test the proposed algorithm over a large data set of randomly generated hybrid robust and nonlinear CCO problems that 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.

This work is organized as follows. Section 2 recalls how one can use (generalized) Benders' decomposition to reformulate (1) as a binary nonlinear optimization problem. In Section 3 we revisit a cutting-plane algorithm to solve the resulting binary problem, extending it to handle oracles with on-demand accuracy. Two different stabilized variants of the cutting-plane method are discussed in Section 4. Section 5 assesses the numerical performance of the presented algorithms in some academic CCO problems. Finally, Section 6 closes with some concluding remarks.

2 Generalized Benders decomposition

When ξ has finite support Ξ, the probability constraint in (1) can be modelled by a standard *disjunctive reformulation*. That is, one defines a binary variable $z_s \in \{0, 1\}$ for each $s \in S$ 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. However, in order to do that, we need to be able to estimate for each $i \in I$ a large enough constant M_i^s that makes the constraint redundant over X; that is, we need

$$M_i^s \geq \max\{\, g_i(x, \xi^s) \;:\; x \in X \,\} \;. \tag{2}$$

We remark that while (2) can be considered 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 generic convex function (even a very simple quadratic one) over a generic convex set (even a very simple hypercube) is in general $\mathcal{NP}$-hard [18]. This issue can conceivably be tackled in different ways. One could, for instance, define an appropriate concave upper approximation of g_i, the best possible being the *concave envelope* of g_i over X. While this is in general a complex task, it has been extensively studied and it can be at least efficiently approximated in many cases (see e.g., [3, 56, 65] among the many others). Alternatively, if g_i is quadratic then the problem can be approached by approximating X. In fact, we recall that for the generic ellipsoid $E(Q, c) = \{\, x \in \mathbb{R}^n \;:\; (x - c)^\mathsf{T} Q (x - c) \leq 1 \,\}$, where $\mathbb{R}^{n \times n} \ni Q \succeq 0$ and $c \in \mathbb{R}^n$, its volume is proportional to $det(Q)^{-1/2}$. Then, the well-known *Minimum-Volume Covering Ellipsoid* (see e.g., [63] and the references therein) is a convex program, since negative powers of the determinant are convex, and in particular SDP-representable [7]. Hence, in several cases, e.g., if X is SDP-representable (say, a polytope), we can efficiently find an ellipsoidal outer approximation of X. Then, one can (approximately) solve (2) for a quadratic g_i since the problem can be reduced to a minimum eigenvalue computation [18]. We remark that in (2) one may in principle replace X with the optimal solution set X^* of (1), but if one knew it, or any better approximation $X^* \subset X' \subset X$, there would no need to solve the problem, or at least one would rather use X' instead of X throughout.

If constants M_i^s are available, then (1) can be reformulated as a (possibly, very-large-scale) "monolithic" mixed-integer program. For notational purposes we now introduce the (nonlinear) mapping $G : \mathbb{R}^n \to \mathbb{R}^{|I| \times |S|}$ as $G(x) = [g(x, \xi^s)]_{s \in S}$, and the matrix (linear mapping) $T = [T^s]_{s \in S}$ where each $T^s \in \mathbb{R}^{|I| \times |S|}$ has rows $T_i^s = M_i^s u_s$, $u_s \in \mathbb{R}^{|S|}$ being the vector having 1 in the column selecting z_s and 0 otherwise. We also introduce the combinatorial set $Z = \{\, z \in \{0,1\}^{|S|} \;:\; \sum_{s \in S} \pi_s \, z_s \leq 1 - p \,\}$ of all "low probability subsets" of scenarios. We can then rewrite

$$\mathbb{P}[\, g(x, \xi) \leq 0 \,] \geq p \;\equiv\; \left\{ \begin{array}{ll} g_i(x, \xi^s) \leq M_i^s z_s & i \in I \;,\; s \in S \\ \sum_{i \in S} \pi_s \, z_s \leq 1 - p & \\ z_s \in \{0, 1\} & s \in S \end{array} \right\} \;\equiv\; \Big\{\, G(x) \leq Tz \;,\; z \in Z \,\Big\}. \tag{3}$$

Therefore, problem (1) becomes the mixed-binary optimization problem

$$\min \{\ f(x)\ :\ G(x) \leq Tz\ \ ,\ \ x \in X\ \ ,\ \ z \in Z\ \}. \tag{4}$$

In this formulation, the binary variables z then act as *complicating variables*. If z is fixed, problem (4) becomes a convex nonlinear programming problem, which is much easier to solve. This feature suggests to solve problem (1) via generalized Benders' decomposition [38], where the complicating variables are separated from the core problem.

This can be done by defining the value function $v : \mathbb{R}^{|S|} \to \mathbb{R} \cup \{\infty\}$

$$v(z) := \min \{\ f(x)\ :\ G(x) \leq Tz\ \ ,\ \ x \in X\ \} \tag{5}$$

where z is now understood as being a parameter. Then, the optimal value f_* of problem (1) coincides with

$$v^* := \min \{\ v(z)\ :\ z \in Z\ \}\ . \tag{6}$$

Moreover, let z^* be a solution to problem (6) and x^* be a solution to the problem defined in (5) by replacing z by z^*: then x^* also solves (1). We recall in the following lemma some important properties of the value function v.

Lemma 1 *The mapping v defined in* (5) *is a proper convex mapping, bounded from below. Moreover, assume that for a given $z \in \mathcal{D}\mathrm{om}(v)$ problem (5) satisfies some constraint qualification so that the set $\Lambda(z) \subset \mathbb{R}_+^{|I|\times|S|}$ of optimal Lagrange multiplies of the constraints $G(x) \leq Tz$ in problem* (5) *is nonempty, then, the subdifferential of v at z is given by $\partial v(z) = -T^{\mathsf{T}}\Lambda(z)$.*

Proof. Convexity of v is well-known, e.g., [59, Lemma 4.24]. For any given $z \in \mathcal{D}\mathrm{om}(v)$, since X is bounded and f is continuous, it follows that $v(z) > -\infty$. Now, the standard value function $\tilde{v}(y) := \min\{\ f(x)\ :\ G(x) \leq y\ ,\ x \in X\ \}$ is such that $v(z) = \tilde{v}(Tz)$; if $z \in \mathcal{D}\mathrm{om}(v)$, then $y = Tz \in \mathcal{D}\mathrm{om}(\tilde{v})$ and $\Lambda(z)$ is also the set of Lagrange multipliers of the above problem, with $y = Tz$. Therefore, [59, Theorem 4.26] yields $\partial\tilde{v}(y) = -\Lambda(z)$; the rest thus follows from the chain rule for (sub)differentiation $\partial v(z) = T^{\mathsf{T}}\partial\tilde{v}(Tz) = T^{\mathsf{T}}\partial\tilde{v}(y) = -T^{\mathsf{T}}\Lambda(z)$. ■

Computing v for fixed z amounts to solving a convex continuous problem, or equivalently (under appropriate constraint qualifications) its dual. Any optimal dual solution $\lambda^*(z) \in \Lambda(z)$ provides a *subgradient* for v in z. Since every known approach for solving the problem also provides an (at least approximate) dual optimal solution, we can assume that (at least) one $\lambda^*(z)$ is in fact known after having computed $v(z)$. We are therefore, at least as the objective function is concerned, in the classical setting for nonsmooth optimization: an *oracle* is available that provides function values and subgradients for the function.

Perhaps the simpler algorithm that can be used for solving nonsmooth optimization problems of this kind (irrespectively of the fact that the feasible region is continuous or discrete), is the *cutting-plane method* [45]. It relies on the oracle to extract first-order information on the objective function, out of which a *model* of $v(\cdot)$ is constructed. More specifically, at a given iteration k a set of iterates $\{z^1, \ldots, z^{k-1}\}$ has been generated. In general one wants to keep an index set $\mathcal{O}_k \subset \{1, \ldots, k-1\}$ gathering the points z^j whose oracle information $(v(z^j), w^j \in \partial v(z^j))$ is (still) available. The standard *cutting-plane model* for v is

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

where the inequality follows from convexity of v. Each of the inequalities in the definition is, in the parlance of Benders' decomposition, known as an *optimality cut*. By minimizing this model over the feasible set Z we obtain a lower bound v_k^{low} for v^* and a new candidate solution z^k to problem (6), at which the oracle is to be called: if $\check{v}_k(z^k) = v(z^k)$, then clearly z^k is optimal for (6) (as $v_k^{\text{low}} = \check{v}_k(z^k) \leq v^* \leq v(z^k)$), otherwise the newly obtained function value (and subgradient) changes the model, and one can iterate. Remarkably, this process does not depend on the fact that Z is a continuous or discrete set, although minimizing $\check{v}_k(z)$ over Z is much harder in the second case. This brief account is, in a nutshell, the celebrated Benders' decomposition method applied to our problem (1).

However, the cutting-plane method is well-known to suffer from a number of issues that often make it inefficient in practice. Thus, one could think to use "more advanced" nonsmooth optimization approaches, such as one of the several available variants of *bundle methods*; see [34, 44, 46] among many others. Yet, bundle methods have traditionally been developed for the case of a continuous feasible set Z, whereas in our case Z is *combinatorial* (albeit deceptively simple). This makes any *master problem* managing the z variables (to find the next iterate z^k) a combinatorial program, and thus requires specific adaptations.

Before going into further details, it is appropriate to remark about the role of the M_i^s tolerances in the two approaches. A potential advantage of Benders' decomposition in our setting is that the the value function v *is independent from the values of* M_i^s. This is easily seen when considering the solution (5) for a fixed feasible z. In fact, if $z_s = 0$ then the (block of) constraint(s) $g(x, \xi^s) \leq 0$ is present in the problem; the value of M_i^s is irrelevant. When $z_s = 1$ instead, then the (block of) constraint(s) is *simply not there*. In other words, the M_i^s have to be chosen in such a way that all the constraints are inactive at the optimal solution x^* of (5): complementary slackness ensure that the corresponding block of multipliers $\lambda^*(z)^s$ must be all-zero. Again, the value of M_i^s is irrelevant. This is not to say that the decomposition approach is not sensitive to the choice of the big-Ms: as Lemma 1 shows, these do appear in the computation of the subgradients of v, and therefore of the model $\check{v}_k$, ultimately influencing the algorithm. Indeed, consider a single variable index s: if $z_s = 1$, then all the constraints corresponding to that scenario will be inactive, hence all the corresponding optimal dual variables λ will be null, and therefore the corresponding entry w_s of the subgradient will be null as well. Conversely, if $z_s = 0$ one expects at least some of the corresponding λ to be strictly positive, and therefore $w_s > 0$; it is immediate to realize that the larger are the M_i^s, the larger will be w_s. However, *the actual shape of* v *does not depend on* M_i^s *in the least*, as already discussed.

So, consider for the sake of illustration two feasible points z' and z" which only differ for one scenario s, i.e., $z'_s = 0$ and $z''_s = 1$. The "discrete derivative" $v(z'') - v(z')$ along component s (easily seen to be non-negative) does not depend on the M_i^s, whereas its estimate w_s obtained in z' (non-negative as well) does, and grows as they do. This confirms that one would like to have the smallest possible value of the M_i^s, that compatible with the "true shape" of v. Any larger value will likely lead to a $\check{v}_k$ "underestimating more" v in the critical region where many components of z are 1 (an all-one z would clearly be the optimal solution if it were feasible), therefore to worse lower bounds v_k^{low}, and ultimately to worse performances. An analogous situation arises when using the monolithic formulation: larger values for the big-Ms are well-known to translate into a weaker continuous relaxation, hence looser lower bounds, hence ultimately less performant algorithms. Indeed, the rationale of the approach in [47] is precisely to avoid the definition of big-Ms in the problem, albeit at the cost of a larger number of inequalities. Thus, for the both approaches a trade-off may exist between the cost of computing tight estimates for the M_i^s constants and the

efficiency of the solution phase. This is clearly worse in our nonlinear case than in the linear one, since in the latter one only needs to solve (possibly, a large number of) linear programs.

It is also fair to add that the monolithic formulation (4) can in fact be solved without any direct reference to the M_i^s. For instance, for the class of problems in §5 we use the general-purpose solver `Cplex` to solve (4), which allows to directly include "logical" *indicator constraints*

$$z^s = 0 \quad \longrightarrow \quad g(x, \xi^s) \leq 0$$

in the formulation instead of their standard MI(N)LP formulation having the big-Ms in the right-hand side. This potentially comes at a cost, as the use of the algebraic form of the constraints paired with "tight" big-Ms, if they are available, is in general computationally more efficient. However, it does save the need to solve potentially difficult estimation problems. Adapting the Benders' approach to the case where no estimates of the M_i^s are available looks more difficult, and we do not pursue this line of ideas further in this paper.

3 Extending the cutting-plane method: inexact oracles

We aim at deriving more efficient approaches to the solution of (6) than the standard cutting-plane algorithm, borrowing from ideas already developed for the continuous case. This is not immediate, as a number of details of the process actually are different for discrete feasible sets. For instance, $z^{k+1} \notin \mathcal{D}\mathrm{om}(v)$, i.e., if subproblem (5) is infeasible, then z^{k+1} must be eliminated from the feasible set of (6). The usual way to do that is by adding a *feasibility cut* to the set of constraints; this can usually be obtained by means of the *unbounded ray* of the dual problem. In the binary setting, one can rather use a simpler *no-good-type cut* [16], along the lines of the *combinatorial* Benders' cuts proposed in [15]. In particular, let $S(z) = \{\, s \in S \;:\; z_s = 0 \,\}$, and $S^k = S(z^k)$; then

$$\textstyle\sum_{s \in S^k} z_s \geq 1 \tag{7}$$

is a feasibility cut that excludes the point z^k from the feasible set of (6). 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 ($\check{v}_{k+1}(z^k) = v(z^k) = +\infty$). 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 mixed-binary *master problem*

$$z^{k+1} \in \arg\min \left\{\, \check{v}_k(z) \;:\; z \in Z^k \,\right\} \quad \equiv \quad \left\{ \begin{array}{lll} \min & r & \\ \text{s.t.} & r \geq v(z^j) + \langle w^j, z - z^j \rangle & j \in \mathcal{O}_k \\ & \sum_{s \in S^j} z_s \geq 1 & j \in \mathcal{F}_k \\ & z \in Z \;,\; r \in \mathbb{R} & \end{array} \right. \tag{8}$$

It is easy to show that no optimal solution of (6) is excluded by feasibility cuts. Indeed, if the optimal set Z^* of (6) is empty (that is, $Z \cap \mathcal{D}\mathrm{om}(v) = \emptyset \Longrightarrow v^* = +\infty$), then clearly no cut can make it less so. If, otherwise, an optimal solution z^* exists, then $v^* = v(z^*) < \infty$. Consequently, z^* can never be detected to be infeasible to problem (5). Now, let z^k be infeasible for (5): if $\sum_{s \in S^k} z_s^* < 1$, then $\sum_{s \in S^k} z_s^* = 0$, i.e., $z_s^* = 0$ for all $s \in S^k$. Consequently, problem (5) corresponding to z^* is at least as constrained as problem (5) related to z^k which was infeasible. This means that $\infty > v(z^*) \geq v(z^k) = \infty$, a contradiction, so that $\sum_{s \in S^k} z_s^* \geq 1$ must hold.

The cutting-plane algorithm for solving problem (6) (and thus problem (1)) is formally stated below; see [73] and [74] for more general settings. Note that (8) can be 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 (1) the single constraint $r \geq 0$ in case $\mathcal{O}_k = \emptyset$, making it a feasibility problem seeking just any point in Z^k. The last constraint is then removed as soon as an optimality cut is generated.

Algorithm 1 Combinatorial cutting-plane algorithm

Step 0. (Initialization) Let $\delta_{\text{Tol}} \geq 0$ be the stopping tolerance. $v_0^{\text{up}} \leftarrow \infty$, $\mathcal{O}_1 = \mathcal{F}_1 \leftarrow \emptyset$, $k \leftarrow 1$.

Step 1. (Master) Find z^k by solving problem (8), and let v_k^{low} be its optimal value.

Step 2. (Stopping test) $\Delta_k \leftarrow v_{k-1}^{\text{up}} - v_k^{\text{low}}$. If $\Delta_k \leq \delta_{\text{Tol}}$, stop: z^{up} is a δ_{Tol}-optimal solution.

Step 3. (Oracle call) Solve (5) with z replaced by z^k.

- If the problem is infeasible then $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{k\}$, $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k$, $v_k^{\text{up}} \leftarrow v_{k-1}^{\text{up}}$, go to Step 4.
- Else, obtain $v(z^k)$ and a subgradient w^k as in Lemma 1, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$ and $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k \cup \{k\}$. $v_k^{\text{up}} \leftarrow \min\{\, v(z^k)\,,\, v_{k-1}^{\text{up}}\,\}$. If $v(z^k) = v_k^{\text{up}}$ then $z^{\text{up}} \leftarrow z^k$.

Step 4. (Loop) $k \leftarrow k + 1$ and go to Step 1.

Theorem 2 *Algorithm 1 terminates after finitely many steps either returning a δ_{Tol}-optimal solution to problem (6) or proving that it is infeasible.*

Proof. First, note that the algorithm cannot stop for $k = 1$: $v_0^{\text{up}} = \infty$ implies $\Delta_k = \infty > 0$ (remember that $v_1^{\text{low}} = 0$ because $\mathcal{O}_1 = \emptyset$, cf. the added constraint $r \geq 0$ in (8)). Once z^k is determined in Step 1 and $v(z^k)$ is computed in Step 3, we have two cases: either $v(z^k) = \infty$, or not. In the first case a no-good 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., $v(z^j) = \infty$ for all $j \leq k$) one keeps having $v_k^{\text{low}} = 0 \Longrightarrow \Delta_k = \infty$; this is unless (8) is empty, in which case $v_k^{\text{low}} = v_k^{\text{up}} = \infty$ and hence $\Delta_k = 0$. In the latter case the algorithm stops: $Z^k = \emptyset$, which proves that (6) is empty. Because at each iteration Z^k strictly shrinks and Z has a finite cardinality, this has to happen eventually if no feasible solution is ever found.

Otherwise, from problem (8) we have that $v_k^{\text{low}} = \check{v}_k(z^k) \geq v(z^j) + \langle w^j, z^k - z^j \rangle$ for all $j \in \mathcal{O}_k$. Therefore, $\|w^j\| \|z^j - z^k\| \geq \langle w^j, z^j - z^k \rangle \geq v(z^j) - v_k^{\text{low}} \geq v_{k-1}^{\text{up}} - v_k^{\text{low}} = \Delta_k$, where the last inequality comes from the fact that $v_{k-1}^{\text{up}} \leq v(z^j)$ for all $j \in \mathcal{O}_k$. Hence, in either case the next iterate z^k differs from all the previously generated points as long as the optimality gap is strictly positive. Since Z contains only finitely many points, the algorithm finitely stops. ■

Algorithm 1 has two costly steps: Step 1, that requires solving a MI(N)LP, and Step 2 that requires solving a large-scale a convex nonlinear optimization problem. In what follows we discuss ways to reduce the computational burden in Step 2 by employing an inexact oracle.

3.1 Inexact oracles

Obtaining the pair $(v(z), w)$ for each given point $z \in Z$ amounts at solving the (very possibly, large-scale) convex optimization problem (5) to optimality. Intuitively, when z is far from the optimal

solution of (6) such an exact computation might be wasteful. This brings us to explore the use in our setting of inexact oracles along the lines of [20, 21].

To see what an inexact oracle for our case looks like, we cast our oracle problem (5) in a Lagrangian setting. Because the problem is solved for a fixed z, as previously noted we do not need to consider all the constraints $G(\cdot)$, but only the part that is "active" at the current z, i.e., $\bar{G}(x) = [g(x, \xi^s)]_{s \in S(z)}$. Hence, introducing an appropriately sized vector λ of Lagrangian multipliers, we can consider the dual function of (5)

$$\theta(\lambda) := \Big\{ f(x) + \langle \lambda, \bar{G}(x) \rangle \ : \ x \in X \Big\}, \tag{9}$$

which provides lower estimates of the optimal value, i.e., $\theta(\lambda) \leq v(z)$. By assuming some constraint qualification, problem (5) is equivalent to its Lagrangian dual, i.e.,

$$v(z) = \max \Big\{ \theta(\lambda) \ : \ \lambda \in \mathbb{R}_+^{|I| \times |S(z)|} \Big\}. \tag{10}$$

In fact, in correspondence to the optimal solution λ^* to (10) one can find an optimal solution to $x^* \in X$ to (9) (with $\lambda = \lambda^*$) such that $\bar{G}(x^*) \leq 0$ and $\langle \lambda^*, \bar{G}(x^*) \rangle = 0$ (complementary slackness). One then has $f(x^*) \geq v(z) \geq \theta(\lambda^*) = f(x^*)$. *Approximately* solving (5) to any given accuracy ε reduces to finding a feasible primal-dual pair $(\bar{x}, \bar{\lambda})$ (i.e., $\bar{x} \in X$, $\bar{G}(\bar{x}) \leq 0$, $\bar{\lambda} \geq 0$) 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, strictly speaking, 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 (9) to ensure the same property, i.e., that a dual feasible solution provides a lower bound on the optimal value. 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 make an explicit distinction between these 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 (10) via some appropriate nonsmooth optimization algorithm, e.g., a bundle method. 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 $f(\bar{x}) - \theta(\bar{\lambda^h})$ is arbitrarily 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 programs (e.g., [31, 36, 37]) or even in the combinatorial case [70].

  A number of techniques can be used to speed-up the solution process: for instance, since the constraints $\bar{G}(x) \leq 0$—even though already restricted to the "active" scenarios—can be many, resulting in a (9) with a λ in a highly dimensional space, one could think of using dynamic bundle methods [4, 28, 36, 37], wherein constraints are dualized in an incremental manner. Also, the approach allows for *warm start* procedures when z changes. Indeed, during the solution for one specific z the dual function is computed for all the iterates λ^h. Thus, a sequence of primal solutions $x^h \in X$ (typically infeasible, i.e., $\bar{G}(x^h) \not\leq 0$) is constructed out of which lower linearizations for θ, and hence cutting-plane models $\check{\theta}_h$, are built. After having

solved (10) for one z, a "rich" model $\check{\theta}$ is typically available. It is immediate to realize that, when a new iterate z' arises, all the primal solutions x^h remain feasible for (9). Thus, we can readily construct a cutting-plane model $\check{\theta}'$ for the new problem by applying appropriate shifts to the subgradients and function values defining the old $\check{\theta}$. This has a potential to significantly speed-up the solution process (see [70] for more on the efficiency of bundle hot-starting).

- *Primal-dual approach*: under appropriate conditions, subproblem (5) has a form which is amenable to solution by primal-dual interior-point methods (e.g., [7] 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 [9, §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. These algorithms have the best known worst-case complexity for solving convex programs and they are also, in general, very efficient in practice, although hot-starting can be more of an issue.

We remark that "purely primal" approaches may also exist. However, since they are less useful in our context and hence we choose not to discuss them. Anyway, in most cases the most computationally effective approaches to convex optimization are the two above ones.

In this context, taking a leaf (separately) from [20] and [21] we can define an *informative on-demand inexact oracle* as any procedure that, given $z \in Z$, a *descent target* $\mathtt{tar} \in \mathbb{R} \cup \{-\infty, +\infty\}$ and a *desired accuracy* $\varepsilon \geq 0$, returns:

$$\left[\begin{array}{l} \text{as function information, } \textit{two} \text{ values } \underline{v} \text{ and } \bar{v} \text{ such that } \underline{v} \leq v(z) \leq \bar{v} \\ \text{as first-order information, a vector } w \in \mathbb{R}^{|S|} \text{ such that } v(\cdot) \geq \underline{v} + \langle w, \cdot - z \rangle \\ \text{under the condition that, if } \underline{v} \leq \mathtt{tar} \text{ then } \bar{v} - \underline{v} \leq \varepsilon \end{array}\right. \tag{11}$$

Clearly, the choice $\mathtt{tar} = \infty$ and $\varepsilon = 0$ results in an exact oracle for v. We will aim at defining the weakest possible conditions on the way to set the two input parameters $\mathtt{tar}$ and ε that still guarantee convergence of the algorithm. Intuitively, this means that we want to keep $\mathtt{tar}$ "small", and ε "large". Indeed, when $\underline{v} > \mathtt{tar}$ (which, for instance, surely holds if $\mathtt{tar} = -\infty$), not much at all is required from $\bar{v}$: $\bar{v} = \infty$ is a perfectly acceptable answer, corresponding to "no feasible solution to (6) has been found yet". This may be so just because no feasible solution exists. In fact, one in principle has to require that (11) can detect when problem (5) is infeasible, which would prompt us to add a feasibility cut to the problem. This case could be easily signaled by the oracle returning $\underline{v} = \infty$ (which obviously implies $\bar{v} = \infty$), incidentally making the value of $\mathtt{tar}$ irrelevant. Also, in this case w might return a valid inequality for $\mathcal{D}\mathrm{om}(v)$ that "cuts away" the current z, which is usually provided for free by the oracle detecting an *unbounded feasible descent direction* for the problem. However, as previously discussed in our setting we do not need this information, and w can be simply disregarded when $\underline{v} = \infty$.

Moreover, setting a finite target $\mathtt{tar} < \infty$ actually allows us to dispense with feasibility cuts altogether. This has the added bonus to allow us to work even if problem (5) does not satisfy a constraint qualification. In fact, weak duality still ensures that the optimal value of (10) still provides a (potentially, strict) lower bound on $v(z)$. Thus, with arbitrary λ, $\underline{v} = \theta(\lambda)$ and $w = -T^{\mathsf{T}}\lambda$ (where λ is here considered automatically extended with zeros for the scenarios $s \notin S(z)$) still gives the subgradient inequality in (11), i.e., we can still derive a valid cutting-plane for v even when (5) does not satisfy constraint qualification. Suppose now that (5) is infeasible for the fixed z: since

X in (1) is compact, we can argue by using [44, Proposition 2.4.1, Chapter XII] that the dual (9) can be made to go to infinity. Thus, any algorithm applied to its solution will typically construct a sequence λ^h such that $\theta(\lambda^h) \to \infty$, while (clearly) never constructing any feasible x^h. This yields the following simple but useful remark:

Remark 3 (Valid cuts for function v) *Given $z \in Z$ and a finite target $\texttt{tar} < \infty$ as input, an oracle (11) solving problem (9) can provide a* valid cut *w for v independently of whether or not problem (5) is feasible for z. Hence, $\mathcal{F}_k$ in* (8) *can remain empty for all k even if $Z \setminus \mathcal{D}\mathrm{om}(v) \neq \emptyset$.*

That is, for a given $z \in Z$ any dual or primal-dual oracle will produce a sequence λ^h, whose objective function value is (hopefully) increasing and which may, or may not, diverge to ∞. 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., $\bar{v} = \infty$). In other words, this information is entirely sufficient to prove that the current point z does not provide any improvement with respect to the target $\texttt{tar}$. This, as we will see, is enough for the algorithms, making it irrelevant to actually "formally prove" if (5) is or not feasible (the dual problem (10) is or not unbounded).

3.2 A cutting-plane method for inexact oracles

We now adapt Algorithm 1 for dealing with the (informative, on-demand) inexact oracle (11). 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

$$\check{v}_k(z) := \max \left\{ \underline{v}^j + \langle w^j, z - z_j \rangle \ : \ j \in \mathcal{O}_k \right\} \ , \tag{12}$$

where $\mathcal{O}_k$ is as in Algorithm 1. Of course, (11) yields $\check{v}_k(z) \leq v(z)$ for all $z \in Z$, and therefore minimizing $\check{v}_k$ still provides a valid global lower bound over the optimal value of (6). The algorithm is then given below.

Algorithm 2 Inexact combinatorial cutting-plane algorithm

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 (12).

Step 2. (Stopping test) As in Step 2 of Algorithm 1.

Step 3. (Oracle call) Choose $\texttt{tar}_k$, send $(z^k, \varepsilon_k, \texttt{tar}_k)$ to oracle (11), 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, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k \cup \{k\}$, $v_k^{\mathrm{up}} \leftarrow \min\{ \bar{v}^k , v_{k-1}^{\mathrm{up}} \}$. If $\bar{v}^k = v_k^{\mathrm{up}}$ then $z^{\mathrm{up}} \leftarrow z^k$.

  Step 3.1 (Accuracy control) If $\underline{v}^k \leq v_k^{\mathrm{low}} + \gamma$ 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. The mechanisms 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* $\varepsilon_k \leq \delta_{\mathrm{Tol}}$ are performed. With this simple device, the algorithm will find a δ_{Tol}-optimal solution to the problem, as shown in the following theorem.

Theorem 4 *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* $v_k^{\mathrm{up}} = \infty$*, then* $\mathtt{tar}_k = \infty$*, and ii) if* ε_k *is reduced in a sequence of* consecutive *iterations, then* $\mathtt{tar}_k \geq v_k^{\mathrm{low}} + \gamma$ *and* $\varepsilon_k \leq \delta_{\mathrm{Tol}}$ *holds for* k *large enough. Under these conditions, Algorithm 2 finitely terminates with either a* δ_{Tol}*-optimal solution to problem* (6) *or a proof that it is infeasible.*

Proof. We will begin by showing that the algorithm establishes infeasibility of problem (6) after finitely many iterations. In fact, since $v(z) = \infty$ for all $z \in Z$, the algorithm can never produce $\bar{v}^j < \infty$, and as a consequence $v_k^{\mathrm{up}} = \infty$ for all k. Then, by assumption i) $\mathtt{tar}_k = \infty$, which means that oracle (11) must necessarily return $\underline{v}^k = \infty$ all along. Then the algorithm behaves exactly as the original one, and needs no further discussion. Note that in this case Remark 3 does not apply, since optimality cuts can be used in place of feasibility ones only when a finite target $\mathtt{tar} < \infty$ can be used. Clearly, feasibility cuts are necessary if the problem is indeed infeasible. We can therefore restrict our discussion to the case where $\underline{v}^k < \infty$ at least once: then, the problem solution set is nonempty. In fact, at the first such k one has $\mathtt{tar}_k = \infty$, and therefore $\underline{v}^k < \mathtt{tar}_k$: by (11), $\infty > \underline{v}^k + \varepsilon_k \geq \bar{v}^k \geq v(z)$, hence at least one feasible point exists and $v_k^{\mathrm{up}} < \infty$ 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 (just wait long enough for the last one). Note that, due to Remark 3, there may in fact be none of these even if some z^k really happens to be infeasible.

Hence, let a feasible z^k be determined in Step 1: from (8) with $\check{v}_k$ given in (12) we have that $v_k^{\mathrm{low}} = \check{v}_k(z^k) \geq \underline{v}^j + \langle w^j, z^k - z^j \rangle$ for all $j \in \mathcal{O}_k$, whence by the Cauchy-Schwarz inequality

$$\|w^j\| \|z^j - z^k\| \geq \langle w^j, z^j - z^k \rangle \geq \underline{v}^j - v_k^{\mathrm{low}} \quad . \tag{13}$$

For all the iterations $j \in \mathcal{O}_k$ where $\underline{v}^j > v_k^{\mathrm{low}}$, (13) gives $\|z^j - z^k\| > 0$, i.e., $z^k \neq z^j$ as in the proof of Theorem 2. If one could prove that this always happens, the usual finiteness argument would apply.

However, in the inexact case things are rather more involved. The point is that here one has the two different values $\bar{v}^j \geq v(z^j) \geq \underline{v}^j$, hence it may well happen that $\bar{v}^j - v_k^{\mathrm{low}} \geq \Delta_k > \delta_{\mathrm{Tol}}$ while one is performing a *tight* iteration: $\mathcal{T}_k = \{\ j \in \mathcal{O}_k\ :\ \underline{v}^j \leq v_k^{\mathrm{low}}\ \}$. For $j \in \mathcal{T}_k$ it may thus happen that $z^k = z^j$, which exposes the algorithm to the danger of cycling. Note that (13) 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 $\underline{v}^k > v_k^{\mathrm{low}}$ only implies that (possibly, but not even necessarily) $v_{k+1}^{\mathrm{low}} > v_k^{\mathrm{low}}$. 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 well have $\bar{v}^k = \infty$ (although, at this point of the argument, $v_k^{\mathrm{up}} < \infty$). That is, also the decrease of the upper bound towards v^* has to be kept under control. For this we also have 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 simple: 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 a definite possibility. Thus, the relevant set of iterations is $\mathcal{T}'_k = \mathcal{T}_k \cap \mathcal{I}_k$; only for these one has

$$\begin{aligned} v^{\rm up}_{k-1} &= \min\{\, \bar{v}^j \,:\, j \in \mathcal{O}_k \,\} \leq \min\{\, \bar{v}^j \,:\, j \in \mathcal{T}'_k \,\} \leq \\ &\leq \min\{\, \underline{v}^j + \varepsilon_j \,:\, j \in \mathcal{T}'_k \,\} \leq v^{\rm low}_k + \min\{\, \varepsilon_j \,:\, j \in \mathcal{T}'_k \,\}, \end{aligned} \tag{14}$$

where the first inequality comes from $\mathcal{T}'_k \subseteq \mathcal{O}_k$, the second inequality comes from the assumption on oracle (11) 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 (14) 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.

We want to prove that the algorithm finitely terminates, so let's assume by contradiction that it does not. For the new iteratation k, either $\underline{v}^k \leq v^{\rm low}_k + \gamma$, or not. Let us first consider the case where $\underline{v}^k > v^{\rm low}_k + \gamma$ occurs, on the whole, 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

$$v^{\rm low}_h = \check{v}_h(z^h) = \max\{\, \underline{v}^j + \left\langle w^j, z^h - z^j \right\rangle \,:\, j \in \mathcal{O}_h \,\} \geq \underline{v}^k + \left\langle w^j, z^h - z^k \right\rangle = \underline{v}^k > v^{\rm low}_k + \gamma \tag{15}$$

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 (15) on the corresponding sub-sequence proves that $v^{\rm low}_k \to \infty$ as $k \to \infty$, which contradicts $v^{\rm low}_k \leq v^* \leq v^{\rm up}_k < \infty$.

Hence, $\underline{v}^k > v^{\rm low}_k + \gamma$ 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, $\mathtt{tar}_k \geq v^{\rm low}_k + \gamma$ ($\geq \underline{v}^k \Longrightarrow k \in \mathcal{I}_k$) and $\varepsilon_k \leq \delta_{\rm Tol}$. This ensures that $\bar{v}^k - \underline{v}^k \leq \varepsilon_k \leq \delta_{\rm Tol}$: eventually, iterates become "accurate enough". Now, consider two (large enough, hence accurate enough) iterations $k > j$: if $j \in \mathcal{T}_k$, then by (14) one would have $\Delta_k = v^{\rm up}_{k-1} - v^{\rm low}_k \leq \bar{v}^j - \underline{v}^j \leq \varepsilon_j \leq \delta_{\rm Tol}$ (obviously, $v^{\rm up}_{k-1} \leq \bar{v}^j$), contradicting the fact that the algorithm does not stop. One should therefore have that $\underline{v}^j > v^{\rm low}_k$ always holds; however, in this case (13) 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. ■

Theorem 4 provides ample scope for constructing different strategies to handle the accuracy parameter ε_k and the target parameter $\mathtt{tar}_k$. The most obvious one is to keep $\varepsilon_k = \delta_{\rm Tol}$ and $\mathtt{tar}_k = \infty$ throughout, thereby insisting to always having both a primal and a dual solution to the oracle subproblem with the same accuracy as the final one required to the algorithm. Then, (14) immediately proves the theorem without any need for Step 3.1 to ever be executed (all iterations are in-target ones). However, intuitively one would expect that having high-accuracy computations at the initial phases of the algorithm is unnecessary, and that starting with a "large" ε_k and a "low" $\mathtt{tar}_k$, revising them as the algorithm proceeds, may be more effective. This has in fact been proven true computationally in the continuous case [20, 28, 71, 75].

The conditions in Theorem 4 on the way in which the oracle parameters are managed are very weak. This corresponds to a "lazy" approach, whereby accuracy is increased only if the algorithm is detected to be at risk of repeatedly generating the same iterate. Having $\texttt{tar}_k = \infty$ as long as $v_k^{\rm up} = \infty$, as required by condition i) in the Theorem, may seem to contradict this, as one actually starts with a "very high" target (albeit, possibly, with a large ε as well). However, it is easy to realize that this is basically unavoidable if the problem is infeasible. In fact, if the algorithm was allowed to set a finite target $\texttt{tar}_1$, then the oracle may return any finite $\underline{v}_1 > \texttt{tar}_1$ together with $w_1 = 0$ (and, obviously, $\bar{v}_1 = \infty$). Then, at the next iteration the algorithm would have $v_2^{\rm low} = \underline{v}_1 > \texttt{tar}_1$, and may well produce $z^2 = z^1$. Hence, an infinite sequence of iterations may start where $z^k = z^1$ and $v_k^{\rm low} \to \infty$, but the algorithm never stops because $v_k^{\rm up} = \infty$ 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 $\texttt{tar}_k = \infty$ is required until the problem is proven feasible. It is however possible to weaken somewhat condition i) by requiring that $\texttt{tar}_k = \infty$ holds *after finitely many iterations where* $v_k^{\rm up} = \infty$, whatever mechanism be used to ensure this.

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 $\texttt{tar}_k$. In fact, if one would require decrease only if $\underline{v}^k \leq v_k^{\rm low}$, then a sequence of iterations all producing the same z may ensue where $\underline{v}^k > v_k^{\rm low}$, but "only very slightly so". Hence, while one may have $v_{k+1}^{\rm low} > v_k^{\rm low}$, the increment may be vanishingly small, and since ε_k would not be decreased, ultimately $v_k^{\rm low}$ may never become close enough to $v_k^{\rm up}$ (a similar argument, in the continuous case, is made in [17, Observation 2.7]). Analogously, if one would set $\texttt{tar}_k = v_k^{\rm low}$, then again a sequence of iterations producing the same iterate could be performed where $\underline{v}^k$ may be fractionally larger than $v_k^{\rm low}$. While this would force the decrease of ε_k, none of these iterations would be in-target, which would not allow us to use (14). That is, while $v_k^{\rm low}$ may indeed be converging to v^*, oracle (11) may never report a close enough upper bound. In fact, one may well have $\bar{v}^k = \infty$ for all k. Note that running the algorithm with fixed $\varepsilon_k = \delta_{\rm Tol}$ and $\texttt{tar}_k = \infty$, does not require any γ (which is somehow obvious since the parameter only influences the choice of ε_k and $\texttt{tar}_k$). An alternative way to get the same result would be to ask that $\varepsilon_k = \delta_{\rm Tol}$ and $\texttt{tar}_k$ is "large enough" (e.g., $\texttt{tar}_k = v_{k-1}^{\rm up}$) eventually, with some mechanism, say a fixed number of iterations, to ensure it.

Hence, our conditions on ε_k and $\texttt{tar}_k$ appear to be basically the weakest possible ones, save for a few twists. In fact, a fixed $\gamma > 0$ is only the simplest possible option. All one needs is that each time when the same iterate is repeated, $v_k^{\rm low}$ has "significantly increased". As customary in other settings (e.g., [17]), one may obtain the same results by choosing a-priori a sequence $\{\gamma_k\}$ such for every subsequence 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 $\delta_{\rm Tol} > 0$ the choice $\gamma_k = \alpha\Delta_k$ for some fixed $\alpha > 0$ would work. All this means that γ_k is ideally "small", i.e., that $\texttt{tar}_k$ is "close" to $v_k^{\rm low}$ even when anything at all is required about it. For most of the algorithm, $\texttt{tar}_k = -\infty$ is possible. Similarly, the analysis allows to always keep ε_k as large as possible unless there is compelling evidence that reducing it is needed. That is, if the condition at Step 3.1 is not satisfied, than 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 [21, Algorithm 5.4] for the continuous case. One could also implement an even *weaker* version of Step 3.1 where ε_k is only reduced if $\underline{v}^k \leq v_k^{\rm low} + \gamma_k$ *and* z^k coincides with a previously generated tight iterate, for this cannot happen infinitely many times (albeit the check that $z^k \neq z^j$ for all $j \in \mathcal{T}_k$ could ultimately be costly in both time and memory).

Basically, 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 there are clear indications that the algorithm has reached the global optimum of the approximation to $v(\cdot)$ corresponding to the currently available solution. Only then the accuracy has to be increased. Said otherwise, the function ideally only have to be computed with *provable* accuracy δ_{Tol} (and thus a primal solution need be produced at all) only at the optimal solution z^*. Insisting that ε_k and $\texttt{tar}_k$ 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, especially since it is a combinatorial program. Furthermore, working with a coarse model of v—even coarser then the ordinary cutting-plane model—may result in a larger number of iterations for reaching convergence, and therefore finally prove not computationally convenient. For all these reasons, more "eager" mechanisms for handling the accuracy parameters may be preferable, which can be obtained in several ways. For instance, $\texttt{tar}_k \leftarrow \max\{\, v_k^{\text{up}} \,,\, v_k^{\text{low}} + \gamma_k \,\}$ may make sense. Basically, one is trying to improve on the upper bound whenever it seems there may be a chance to, although improving the upper bound early on has only a limited effect on the algorithm behavior (the only role of v_k^{up} being in the stopping condition). For ε_k, several forms of more "eager" handling of ε_k and $\texttt{tar}_k$ can be imagined. We will list a few possibilities among many:

- Condition ii) in the hypothesis of the Theorem can be obtained, when $\delta_{\text{Tol}} > 0$, by simply choosing $\alpha \in (0, 1)$ and $\varepsilon_{k+1} = \max\{\, \delta_{\text{Tol}} \,,\, \min\{\, \alpha\Delta_k \,,\, \varepsilon_k \,\} \,\}$. This ensures that ε_k is nonincreasing with the iterations, which makes sense intuitively since it has eventually to reach δ_{Tol} anyway. This is for instance the strategy that has been employed in [75, 76]. A (minor) issue arises when $\delta_{\text{Tol}} = 0$, 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. Apart from the fact that this is actually a moot point in numerical optimization, where some nonzero accuracy is usually required anyway, the issue is easy to solve: some mechanism has just to ensure that $\varepsilon_k \leq \delta_{\text{Tol}}$ eventually, which may simply amount at choosing a fixed $\delta > 0$, and if $\varepsilon_k \leq \delta_{\text{Tol}} + \delta$ then $\varepsilon_k \leftarrow \delta_{\text{Tol}}$.

- The analysis in Theorem 4 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. Yet, some form of control is required on $\texttt{tar}_k$ to ensure that eventually these will be in-target.

- 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. This can be simply obtained by modifying Step 3.1 so that whenever ε_k is reduced the algorithm jumps directly to Step 3 rather then passing through Step 1. 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 whereby the previously interrupted computation is basically just resumed from the point it reached when it was stopped due to having reached the previous (larger) accuracy. This potentially makes a sequence of calls with ε_k decreasing up to some $\bar{\varepsilon}$ not significantly more costly than a single call with accuracy $\bar{\varepsilon}$, even if, say, a primal-dual approach is used.

Thus, several strategies exist for handling the accuracy in the original cutting-plane algorithm. 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 cutting-plane algorithms: rather than decreasing the iteration cost we aim at requiring fewer iterations. To this aim 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 accrued first-order information by decreasing the *instability* of pure cutting-plane algorithms, i.e., the fact that two subsequent iterates can be "very far apart". This has been shown to be very beneficial (e.g., [5, 11, 35] 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. [77] in the discrete setting and [71] in the continuous one) that stabilization 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, for reasons to become clear shortly we find it more appropriate to rather go the trust-region way [49]. Anyhow, even in the continuous case the two approaches are in fact basically the same [5, 34].

Consider the (inexact) cutting-plane model defined in (12). We modify the master problem (8) by selecting one *stability center* $\hat{z}^k$, which can initially be thought of as being (although it is not exactly) the current best iterate, and the radius $\mathcal{B}_k \geq 1$ of the current trust region. Then, we restrict the feasible region of (8) by adding the box constraint $\|z - \hat{z}^k\|_1 \leq \mathcal{B}_k$, leading to

$$z^k \in \arg\min \left\{ \check{v}_k(z) \ : \ z \in Z^k \ , \ \|z - \hat{z}^k\|_1 \leq \mathcal{B}_k \right\} \ . \tag{16}$$

This is particularly attractive in our binary setting because the new condition can be expressed by the single linear constraint

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

which is well-known, e.g., under the moniker of *local branching constraint* [32]. The effect is to force the next iterate to lie in a *neighbourhood* of the stability center where only at most $\mathcal{B}_k$ scenarios 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 obviously be represented by an analogous linear constraint. Note that in a convex setting any such set would be *concave*, 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 may therefore be 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, due to the fact that for a binary variable z_s 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). Thus, we might as well have mimicked the standard proximal bundle approach by adding that term, weighted by an appropriate constant, to the objective function of (8). However, the rationale of using a quadratic proximal term in the continuous case is that it somehow "gives second-order information" (albeit "poorman's one") to the model. We do not think that this is significant in our setting, where the master problem is not only linear, but even worse combinatorial. The (inexact, combinatorial) cutting-plane algorithm is then modified as follows:

Algorithm 3 Trust region inexact combinatorial cutting-plane algorithm

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 (16).

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 $\mathcal{B}_k = |S|$ then stop: z^{up} is a δ_{Tol}-optimal solution. Otherwise, $\mathcal{R}_{k+1} \leftarrow \mathcal{R}_k \cup \{k\}$ and choose $\mathcal{B}_{k+1} \in (\mathcal{B}_k\,,\,|S|]$.

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 ever produced, the new mechanism at Step 3 ensures that the stability center is moved to the feasible point. This is called a "serious step" 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" serious step where the stability center does not really change, while its objective function value does. If (6) has no solution this will never happen, and the stability center will remain fixed in $\hat{z}^1$.

The analysis of Theorem 4 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 serious step is done, eventually the stopping condition at Step 2 is triggered. If this happens but $\mathcal{B} < |S|$, 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 the size of the neighbourhood up until eventually the whole Z is covered. Fortunately, this can happen only a finite (and "small") number of times. Adding the reverse region constraint in this occurrence is not strictly necessary, but it costs little while ensuring that the master problem will no longer produce any iterate in the previous neighbourhood, which has "already been thoroughly explored". Only the "doughnut" $\mathcal{B}_k < \|\cdot - \hat{z}^k\|_1 \leq \mathcal{B}_{k+1}$ need now be explored. This reduces the size of the feasible region of the master problem, hopefully making it easier to solve. Obviously, if $\mathcal{B}_k = |S|$ then global optimality is ensured, considering that local optimality in all the "inner rings" has already been proven. In particular, if no feasible solution

has been found throughout, then (6) clearly is infeasible.

The only mechanism that still requires discussion is then the "serious step" one, where $\hat{z}^k$ is modified: we need to ensure that this cannot happen infinitely many times. Note that our choice has been that a serious step never triggers a forced decrease of ε_k (while it is entirely possible to still reduce it if one wants), so we cannot rely on that mechanism to prove finite convergence. The issue could be quickly dismissed invoking the finiteness of Z, but this would be a superficial treatment that does not take into account the specificity of our inexact setting. Indeed, in our case it would, in principle, be entirely possible that the same iterate z is produced infinitely many times, with each oracle call providing a smaller value for $\bar{v}$. Whenever the same iterate is produced but with a smaller $\bar{v}$, a(n again) "fake" serious step could be triggered in Step 3 whereby $\hat{z}$ is not actually changed. With $\beta = 0$, decreases may be vanishingly small: the sequence of values may not even converge towards $v(z)$, but even if it does it may take infinitely many iterations to do so. This in itself would not be a problem: the feasible region of the master problem would not change, and the analysis would still apply proving that there cannot be infinitely many iterations. Yet, a more devious case need to be taken into account: that where $\hat{z}$ "cycles through" a set of iterates (possibly with the very same, and optimal, value of $v(z)$), each time reducing v^{up} a tiny little bit but never really producing the true value. This is precisely the reason for $\beta > 0$: a serious step can only be declared when a "significant" decrease is obtained, which avoids this scenario. Again, a fixed value is only the most obvious choice: a non-summable sequence $\{\beta_k\}$ or, for $\delta_{\text{Tol}} > 0$, $\beta_k = \alpha\Delta_k$ for $\alpha \in (0, 1]$ would work as well. We remark in passing that rules of this kind are common in the continuous case, where deciding when a serious step has to be declared is one of the delicate choices, in particular in the inexact case [21]. Our discrete setting makes the analysis easier.

This discussion proves the following result, which to the best of our knowledge is new:

Theorem 5 *Under the assumptions of Theorem 4, Algorithm 3 terminates after finitely many steps with either a δ_{Tol}-optimal solution to problem* (6) *or a proof that this problem is infeasible.*

It is apparent that the analysis also extends to the "eager" accuracy controls of §3.2.

At least a nontrivial practical detail still need to be discussed: how the size of the trust region is increased in Step 2 when local optimality is detected. In principle the simple rule $\mathcal{B}_{k+1} \leftarrow \mathcal{B}_k + 1$ could be used, but previous experience (in the exact setting) [2] suggests that this is not computationally effective. Conversely, once one has found a good local optima for a "small" region, it usually makes sense to quickly proceed at trying 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, i.e., $\mathcal{B}_k \in \{0.5\%|S|, 50\%|S|, 100\%|S|\}$. Other rules are clearly possible.

4.2 Level stabilization

In this section we rather explore *level stabilization*, which is, even in the continuous case, significantly different to analyze [46]. Continuous level bundle methods have been analyzed in the inexact case in [20], and discrete level bundle methods have been analyzed in the exact case in [19]. To the best of our knowledge, apart from the recent [77] this is the first work investigating a discrete level bundle method with inexact oracles. In contrast to [77] that handles oracles with fixed accuracy only, our approach deals with more general oracles with on-demand accuracy. Accordingly, the

hypotheses required to establish convergence of our approach are significantly weaker than the ones assumed in [77].

At iteration k, let v_k^{low} and v_{k-1}^{up} be the available (inexact) lower and upper bounds for the optimal value v^* of problem (6). Thus, $\Delta_k = v_{k-1}^{\text{up}} - v_k^{\text{low}}$ is the (inexact) optimality gap. The level stabilization approach centers on defining a *level parameter* $v_k^{\text{lev}} \in (v_k^{\text{low}}, v_{k-1}^{\text{up}})$ and the *level set*

$$\mathbb{Z}^k := \left\{ z \in Z^k \ : \ \check{v}_k(z) \leq v_k^{\text{lev}} \right\} \ . \tag{17}$$

Note that it may happen that $v_{k-1}^{\text{up}} = \infty$, in which case v_k^{lev} is just taken arbitrarily (with the safeguards discussed below) satisfying $v_k^{\text{lev}} > v_k^{\text{low}}$. The next iterate z^k is chosen in the level set $\mathbb{Z}^k$, whenever it is nonempty. It is immediate to realize that if $\mathbb{Z}^k = \emptyset$, then $v_k^{\text{low}} < v_k^{\text{lev}} \leq v^*$, 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 (again) a given stability center $\hat{z}^k \in \{0,1\}^{|S|}$, 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 continuous optimization, the most common choice for the function φ is the Euclidean norm. In the mixed-integer setting it may be natural to take the ℓ_1 or ℓ_∞ norms instead, as done in [19], since this leads to a MILP master problem. Yet, as already mentioned, in the binary setting 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. Consequently, taking the ℓ_2 norm as a stability function leads to the binary linear programming problem

$$z^k \in \arg\min \left\{ \varphi(z; \hat{z}^k) \ : \ z \in \mathbb{Z}^k \right\} \quad \equiv \quad \left\{ \begin{array}{ll} \min & \langle \frac{1}{2}\mathbf{1} - \hat{z}^k, z \rangle \\ \text{s.t.} & \underline{v}^j + \langle w^j, z - z^j \rangle \leq v_k^{\text{lev}} \quad j \in \mathcal{O}_k \\ & z \in Z^k \, . \end{array} \right. \tag{18}$$

We remark that (18) does not need any trick, as (8) and (16) did, 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 $\hat{z}^k \in Z^k$). Also, Algorithm 4 below does not actually need an *optimal* solution to (18): a *feasible* point would be enough. This opens the way for applying efficient heuristics for inexactly solving (18), for instance by solving the continuous relaxation and then applying randomized rounding. We can also append the formulation (18) with additional constraints, such as those induced by precedence relations [57], if these can be exhibited. Conversely, note that (18) does not automatically produce a valid lower bound v_k^{low}, 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 cutting-plane algorithm. Another initialization step will be discussed later on.

Algorithm 4 Level inexact combinatorial cutting-plane algorithm

Step 0. (Initialization) Run Algorithm 2 with the added condition in Step 2: stop (also) if $\bar{v}^k = v_k^{\rm up} < \infty$. If $\Delta_k \leq \delta_{\rm Tol}$ then terminate.

Step 1. (Stopping test) As in Step 2 of Algorithm 2.

Step 2. (Master) Choose arbitrarily $\hat{z}^k \in \{0,1\}^{|S|}$. Choose $v_k^{\rm lev} \in [v_k^{\rm low}, v_{k-1}^{\rm up} - \delta_{\rm Tol})$. Solve (18): if it is infeasible then $v_k^{\rm up} \leftarrow v_{k-1}^{\rm up}$, choose $v_k^{\rm low} \in [v_k^{\rm lev}, v^*]$ and go to Step 4, else z^k is available.

Step 3. (Oracle call) Choose $\texttt{tar}_k$. Send $(z^k, \varepsilon_k, \texttt{tar}_k)$ to oracle (11), receive $\underline{v}^k$, $\bar{v}^k$, and w^k.

- If $\underline{v}^k = \infty$ then proceed as in Algorithm 1.
- Otherwise, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, $\mathcal{O}_{k+1} \leftarrow \mathcal{O}_k \cup \{k\}$, $v_k^{\rm up} \leftarrow \min\{\, \bar{v}^k \,,\, v_{k-1}^{\rm up} \,\}$. If $\bar{v}^k = v_k^{\rm up}$ then $z^{\rm up} \leftarrow z^k$.

  Step 3.1 (Accuracy control) If $\underline{v}^k \leq v_k^{\rm lev}$ then choose $\varepsilon_{k+1} \in [0, \varepsilon_k)$, otherwise choose $\varepsilon_{k+1} \in [0, \infty)$ arbitrarily.

Step 4. $v_{k+1}^{\rm low} \leftarrow v_k^{\rm low}$, $k \leftarrow k+1$ and go to Step 1.

It is easy to see that this algorithm is close to Algorithm 2. The main differences are the use of master problem (18) in place of (8), the corresponding selection of the level parameter at Step 1, and the use of $v_k^{\rm lev}$ instead of $v_k^{\rm low} + \gamma$ in Step 3.1. It is not surprising, then, that a similar analysis can be devised.

Theorem 6 *Assume that Step 2 is implemented in such a way that $v_k^{\rm lev}$ changes or problem* (6) *is found to be infeasible* only finitely many times. *Furthermore, assume that the choice of $\texttt{tar}_k$ in Step 3 and that of ε_k in Step 3.1 satisfy the assumptions of Theorem 4, only with $\texttt{tar}_k \geq v_k^{\rm lev}$ replacing $\texttt{tar}_k \geq v_k^{\rm low} + \gamma$. Then, the same conclusions as in Theorem 4 hold.*

Proof. We start by remarking that the assumptions on Step 2 are not trivial to satisfy. This is because the general rule in Step 2, $v_k^{\rm lev} \in [v_k^{\rm low}, v_{k-1}^{\rm up} - \delta_{\rm Tol})$, *requires* changing the value of $v^{\rm lev}$ from that of the previous iteration when $v_{k-1}^{\rm lev} > v_{k-1}^{\rm up} - \delta_{\rm Tol}$, i.e., one has found a better upper bound at the previous iteration that forces a "significant" downward revision of $v^{\rm lev}$. Furthermore, when $\mathbb{Z}^k = \emptyset$ it is "very easy" to choose $v_{k+1}^{\rm lev}$ in a way that causes this to happen again at the next iteration: just increase $v^{\rm lev}$ 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 $\delta_{\rm Tol} = 0$, because it means that eventually one must have $v_k^{\rm low} = v_k^{\rm lev} = v_{k-1}^{\rm up} = v^*$, 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 $v^{\rm lev}$-selection mechanisms can be developed, though, as we discuss below.

Step 0 of the algorithm—possibly a complete run of Algorithm 2—finitely terminates due to Theorem 4. That is, either the algorithm terminates with $v_{k-1}^{\rm up} = v_k^{\rm low} = \infty \Longrightarrow \Delta_k = 0$ proving that (6) is infeasible, or a feasible z^k if eventually found, at which point Step 0 terminates. Note that, crucially, this happens at Step 2 of Algorithm 2, meaning that $v_k^{\rm low}$ is also available, which is the reason for the initialization. Also, note that there is the odd chance that $v_{k-1}^{\rm up} \leq v_k^{\rm low} + \delta_{\rm Tol} < \infty$, i.e., the algorithm stops at Step 0 because the initialization has already found a $\delta_{\rm Tol}$-optimal solution (as opposed to having proven the problem empty).

The crucial assumption on the level management is that $v_k^{\rm lev}$ can change and (6) be infeasible only finitely many times. Thus, either the algorithm finitely terminates, or there exists an iteration $\bar{k}$ such that $v_k^{\rm lev} = v_{\bar{k}}^{\rm lev}$ and $\mathbb{Z}^k \neq \emptyset$ for all $k \geq \bar{k}$. Hence, we can assume all our iterates to be large enough for this to happen. We will thus simply denote $v_k^{\rm lev} = v_{\bar{k}}^{\rm lev}$ as $v^{\rm lev}$. This gives

$$\underline{v}^j + \langle w^j, z^k - z^j \rangle \leq v^{\rm lev} \quad \text{for all} \quad j \in \mathcal{O}_k$$

from which we get the analogous of (13)

$$\|w^j\| \|z^j - z^k\| \geq \langle w^j, z^j - z^k \rangle \geq \underline{v}^j - v^{\rm lev} \ . \tag{19}$$

Defining $\mathcal{T}_k = \{ \ j \in \mathcal{O}_k \ : \ \underline{v}^j \leq v^{\rm lev} \ \}$, (19) immediately shows that $z^k \neq z^j$ for $j \notin \mathcal{T}_k$. Again, only tight iterations can repeat previous iterates. Therefore, $\underline{v}^k > v^{\rm lev}$ can only happen finitely many times. In fact, 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 this is happening (all indices are large enough).

The consequence is that the hypotheses on $\texttt{tar}_k$ and ε_k now apply: eventually, $\texttt{tar}_k \geq v^{\rm lev}$ and $\varepsilon_k \leq \delta_{\rm Tol}$, i.e, $\bar{v}^k - \underline{v}^k \leq \varepsilon_k \leq \delta_{\rm Tol}$. Copying (14) (keeping $\mathcal{I}_h$ and $\mathcal{T}'_h$ unchanged), we similarly conclude

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

But the choice of $v^{\rm lev}$ at Step 2 now requires that

$$v^{\rm lev} < v_{k-1}^{\rm up} - \delta_{\rm Tol} \leq v^{\rm lev} \ ,$$

a contradiction: hence, the algorithm must terminate finitely. ■

The analysis should also plainly extend to the case when the feasible set Z is not finite, but still bounded, $\mathcal{D}{\rm om}(v) \subset Z$ (ensuring thus that ∂v is locally bounded) and $\delta_{\rm Tol} > 0$. This case is analyzed in [77], 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.

The short proof of Theorem 6 somehow hides the fact that the level parameter $v^{\rm lev}$ has to be properly managed for the assumptions to be satisfied. We now discuss possible mechanisms which obtain that.

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

$$v_k^{\rm lev} \leftarrow \begin{cases} v_{k-1}^{\rm up} - \max\{ \ \delta_{\rm Tol} \ , \ \alpha \Delta_k \ \} & \text{if } v_{k-1}^{\rm up} < v_{h(k)}^{\rm up} - \delta_{\rm Tol} \text{ or } \mathbb{Z}_{k-1} = \emptyset \\ v_{h(k)}^{\rm lev} & \text{otherwise} \end{cases} \ . \tag{20}$$

In plain words, $v_k^{\rm lev}$ is updated whenever $v^{\rm low}$ needs be revised upwards, or $v^{\rm up}$ is "significantly" revised downwards. This mechanism ensures that $v_k^{\rm lev}$ 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 $\delta_{\text{Tol}} > 0$. Similarly, $\mathbb{Z}^k = \emptyset$ cannot happen infinitely many times: in fact, whenever this happens

$$\Delta_{k+1} = v_k^{\text{up}} - v_k^{\text{low}} \leq v_{k-1}^{\text{up}} - v_{k-1}^{\text{lev}} = \max\{\, \delta_{\text{Tol}}\,,\, \alpha\Delta_k \,\} \quad .$$

In other words, the gap shrinks exponentially fast until eventually $\Delta_k \leq \delta_{\text{Tol}}$, triggering the stopping condition. Note, however, that for $\delta_{\text{Tol}} = 0$ (20) only gives $\Delta_k \to 0$ (fast), but cannot guarantee finite convergence.

It is useful to remark that the need of having $\delta_{\text{Tol}} > 0$ in the above derivation—which means the convergence properties of the algorithm are ever so slightly weaker than these of the previous two ones. Note that this is not very relevant in practice, and not due to a weakness of the analysis, but rather to an inherent property of level-based methods. The point is that 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^*. Indeed, only when $\mathbb{Z}^k = \emptyset$ can we conclude $v_k^{\text{lev}} < v^*$. The fact that the inequality in the previous relationship is strict shows that a level-based approach will never be able to prove that $v_k^{\text{lev}} = v^*$. This is why one is not allowed to pick $v_k^{\text{lev}} = v_{k-1}^{\text{up}}$: if $v_{k-1}^{\text{up}} = v^*$, one would not 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 $v_1^{\text{low}} = v^* - \varepsilon_1$. Even if one sets $\varepsilon_k = 0$, an infinite sequence of iterations then ensues whereby $\mathbb{Z}^k = \emptyset$ always happens and $v_k^{\text{low}} \to v^*$—say, using (20)—but never quite reaching it. This is a known (minor) drawback of this form of stabilization.

It is worth mentioning that the algorithm actually has a chance to work even if $\delta_{\text{Tol}} = 0$, but only if the initial v_k^{low} provided by Step 1 happens to be precisely v^*. This is because that lower bound is produced by different means, which do allow to prove that a specific value is a sharp lower bound. This observation reveals that a variant of the algorithm is possible, which can work with $\delta_{\text{Tol}} = 0$: just prior to setting the level parameter one solves (8) and updates v_k^{low} as its minimum value. This was in fact proposed when level methods were introduced [46] in the continuous setting. The condition ensures that $\mathbb{Z}^k \neq \emptyset$ will happen at *every* iteration, thus making the relevant part of the assumption in Theorem 6 moot. However, this re-introduces the risk that v_k^{low} increases infinitely many times. Furthermore, (20) 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 $v_k^{\text{lev}} = v_k^{\text{low}}$: basically, at some point the algorithm reverts to the standard cutting-plane approach. Hybrid versions where (8) 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 could allow to dispense with 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 (8) 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 and add the following 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 $\bar{\delta} < \infty$. The rationale for the modification, similar to the one present in [10], is clear: not having a dependable *lower* bound v_k^{low} on v^*, one proceeds by guessing a value

using the best available *upper* bound and a displacement (this is called a "target value" approach in other settings [17]). 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, sooner or later 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 for good anyway: a dependable lower bound has been found, and the normal course of the algorithm, as analyzed in Theorem 6, starts. Basically, all the iterations up to that point take the place of the Step 0 where (8) 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 (6) still applies. In fact, similarly to §4.1, we can disable the accuracy control when the target decreases.

As a very final remark, the analysis clearly extends to the more "eager" accuracy control versions of §3.2, with the corresponding computational trade-offs.

4.3 Bundle resets: making subproblems easier to solve

All the considered master problems considered in this work employ two bundles: one of feasibility cuts, represented by $\mathcal{F}_k$, and another of linearizations of v, represented by $\mathcal{O}_k$. In addition, the trust-region variant has a "bundle of reverse region constraints", but that is not essential. In order to prevent previous iterates $z^k \notin \mathcal{D}\text{om}(v)$ to be visited again by the algorithms, the bundle of feasibility cuts must satisfy $\mathcal{F}_k \subseteq \mathcal{F}_{k+1}$ for all k. On the other hand, the bundle of linearizations $\mathcal{O}_k$ can be reduced in some particular configurations, making the master problem smaller and thus possibly saving CPU time for computing the next trial point.

While there are sensible ways for reducing the bundle in the convex case when some specific forms of stabilization are used, the non-stabilized cutting plane algorithm and other forms of stabilization, like trust region, do not have any particularly sensible way of doing reset even in the convex case [34, §4.3]. This is, clearly, even worse if the master problem is discrete. Basically, the bundle $\mathcal{O}_k$ can be reset arbitrarily provided that this happens *finitely many times*: the standard convergence proofs then apply after that the last reset has occurred. A simple rule to manage the bundle 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)

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

In other words, the bundle can be reset each time the optimality gap "decreases enough"; this can happen only a finite number of times if $\delta_{\text{Tol}} > 0$. Although the above rule can be employed by all the presented algorithms, it appears to be more practical when applied by stabilized algorithms. Similar rules could be employed checking for "substantial changes" in v_k^{low} or v_k^{up} separately, or any other means.

There is, clearly, a non-trivial trade-off regarding bundle management. Indeed, on one hand keepinf $\mathcal{O}_k$ as small as possible may save on master problem time. On the other hand, accruing information is what ultimately drives the convergence, and therefore discarding information too quickly may be very detrimental to convergence rates. The trade-off is dependent on the ration between master problem time and subproblem time, and in general on the details of the problem at hand, even in the continuous case (e.g. [36, Table 7], [11]). We can expect the trade-off to be even more specific when the master problem is discrete, and therefore in general more difficult to solve.

5 Application: a hybrid robust/chance-constrained model

We consider the following situation, that has many potential applications. Some objective function $f : \mathbb{R}^n \to \mathbb{R}$, possibly linear, has to be optimized subject to (linear) constraints of the form

$$Ax \leq \xi \ , \tag{21}$$

where both ξ and A are subject to uncertainty. In this setting we are concerned with obtaining solutions x satisfying (21) in most, but not necessarily all, scenarios. It occurs in many cases that there are different sources of uncertainty, not all equally well understood. For instance, while ξ (the right-hand side of the constraint, say demands or costs of some commodity) may have a well characterized distribution, for A (the coefficient matrix, say depending on the choices of other actors in the same market or on the functioning of some complex equipment) one does not dispose of such a fine characterization. This setting is of interest for instance in energy management, where x could represent an energy production schedule (e.g., [64] for the uncertain unit-commitment problem). The condition $Ax \leq \xi$ could mean that we wish to produce sufficient energy in all but the most extreme and implausible scenarios. Knowledge of ξ (energy demand) would be available, since its characterization has received much attention from statisticians/econometrists, while A could be related to the underlying physics of generation plants and/or to the behavior of other generation companies, and much less information could be available.

For this reason we will explore a *hybrid robust/chance-constrained* approach. Let $A = [a_i]_{i\in I}$; we will assume that the uncertainty about the matrix can be expressed in the form $a_i(u) = \bar{a}_i + P_i u$, where $\bar{a}_i \in \mathbb{R}^n$, the *uncertainty set* $u \in \mathcal{U}_i = \{u \in \mathbb{R}^{n_i} \ : \|u\|_{r_i} \leq \kappa_i\}$ is the ball of radius κ_i in some convex norm (i.e., $r_i \geq 1$), and P_i is an $n \times n_i$ matrix (e.g., [6]). For the sake of simplifying the notation we define by $\mathcal{U} = [\mathcal{U}_i]_{i\in I}$, 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 well known distribution. In our setting it will be represented by a finite set Ξ of realizations, possibly obtained by appropriate sampling, although this is not crucial for most of the analysis here.

We will express our requirement under the form of the *robust chance-constraint*

$$\mathbb{P}\big[\, A(u)x \leq \xi \quad \forall u \in \mathcal{U} \,\big] \geq p \ . \tag{22}$$

This setting lends itself to a well-known treatment: for a fixed ξ, $a_i(u)^\mathsf{T} x \leq \xi_i$ for all $u \in \mathcal{U}_i$ if and only if $\max\{\, a_i(u)^\mathsf{T} x \ : \ u \in \mathcal{U}_i \,\} \leq \xi_i$. Following our choice of $a_i(u)$, we have that

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

where $\|.\|_{s_i}$ is the (convex) conjugate norm of $\|.\|_{r_i}$, i.e., $1/r_i + 1/s_i = 1$, interpreted with usual conventions when either r_i or s_i equals 1. This follows from well known results of optimizing linear forms over norm constrained sets (e.g., [6, Chapter 1] [1, Lemma 3.1]), and more in general from convex duality results. Consequently, (22) is equivalent to

$$\mathbb{P}\big[\, \bar{a}_i^\mathsf{T} x + \kappa_i \|P_i^\mathsf{T} x\|_{s_i} \leq \xi_i \quad i \in I \,\big] \geq p \ ,$$

which readily falls in the setting of §2, as $g_i(x, \xi) = \bar{a}_i^\mathsf{T} x + \kappa_i \|P_i^\mathsf{T} x\|_{s_i} - \xi_i$ is clearly convex in x (since, obviously, $\kappa_i > 0$). The special cases $r_i = 1$ or $r_i = \infty$ result in g being polyhedral, whereas $r_i = 2$ makes g_i a Second-Order Cone representable function.

It is interesting to remark that (22) implies the weaker condition

$$\mathbb{P}\big[\, A(u)x \leq \xi \,\big] \geq p \quad \forall u \in \mathcal{U} \ . \tag{23}$$

Indeed, because in (22) 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 7 *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* (23) *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 also shows that the weaker model (23) does not behave satisfactorily. Indeed we do not know, or wish to make explicit, the underlying distribution of u. Therefore, 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 a joint probabilistic constraint and an individual probabilistic constraint. While the former (like (22)) provides a sound probabilistic guarantee, the latter (like (23)) may yield insufficient robustness, as shown e.g., in [69].

5.1 Generation of problem instances

For our experiments we focussed on problems of the form

$$\min \left\{ c^{\mathsf{T}} x \ : \ \mathbb{P}\big[\, A(u)x \leq \xi \quad \forall u \in \mathcal{U} \,\big] \geq p \ , \ 0 \leq x \leq \bar{x} \right\} \ , \tag{24}$$

where $\bar{x} \in \mathbb{R}^n$ is a bound, $\xi \in \mathbb{R}^m$ has finite support Ξ and $A(u), \mathcal{U}$ are as described in the previous paragraph (with $r_i = s_i = 2$ for all i). We have taken a linear objective function $c \in \mathbb{R}^n$ for simplicity.

Instances of problem (24) were generated by using the following procedure. We begin by setting the problem dimensions n, $m = |I|$ 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 with sparsity (percentage of non-zero entries) in the set $\{1\%, 0.1\%, 0.01\%\}$. Finally, we generate a random candidate solution x^c with entries in $[0, 10]$ and we compute $\bar{\xi}$ such that $\bar{\xi}_i = \bar{a}_i^{\mathsf{T}} x^c + \kappa_i \|P_i^{\mathsf{T}} x^c\|_2$ 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 (24) 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 Euclidean norm in $\mathcal{U}$ means that problem (6) has a Mixed-Integer Second-Order Cone formulation, that can be directly solved by off-the-shelf tools like `Cplex`. In fact, current solvers require a slight transformation of the model into a Quadratically-Constrained (convex)

Quadratic Problem. By introducing auxiliary variables y_i for all $i \in I$, (24) can be reformulated by using (apart from the other, obvious ones) the following linear and quadratic constraints

$$\begin{aligned} &\bar{a}_i^{\mathsf{T}} x + \kappa_i y_i - \xi_i^s \leq M_i^s z_s && i \in I \ , \ s \in S \\ &x^{\mathsf{T}}(P_i P_i^{\mathsf{T}})x \leq y_i^2 && i \in I \end{aligned} \tag{25}$$

which are appropriately dealt with by `Cplex` even if the Hessian matrix of (25) is not, strictly speaking, positive semi-definite. Subproblem (5) can be written and solved with the same tools.

5.2 Setup and results

We have generated several problem instances as follows. First we have taken the dimensions of the problem with all choices of n, m 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 $\{1\%, 0.1\%, 0.01\%\}$. Here we have only considered the combinations of adjacent sparsity levels, i.e., avoided the combinations $(1\%, 0.01\%)$ and $(0.01\%, 1\%)$, for a total of 7 cases (instead of 9). Finally, for each of the above we generated 3 instances just changing the seed of the random number generator, for a grand total of $12 \cdot 7 \cdot 3 = 252$ randomly generated instances. For our experiments we chose two different values for δ_{Tol}, i.e., `1e-4` (considered the default tolerance for hard Mixed-Integer problems) and `1e-3` (considered a "coarse", but still acceptable in some cases, tolerance). Both are to be intended as *relative*, which, via appropriate scalings, does not substantially change our analysis in the previous paragraphs where *absolute* tolerances were used for simplicity. We also set a time limit for each method of 100000 seconds, which is slightly more than a day. Both the "monolithic" approach and all the optimization problems in the Benders' one (the MILP master problem and the SOCP subproblems) have been solved with the state-of-the-art, general-purpose solver `Cplex` 12.4. Other than setting the time limit and relative gap accuracy, no other parameters of `Cplex` were set. The runs have been performed on Intel Xeon X5670 westmere computer cluster nodes with 8 Gb of reserved memory. All `Cplex` runs were done monothreaded.

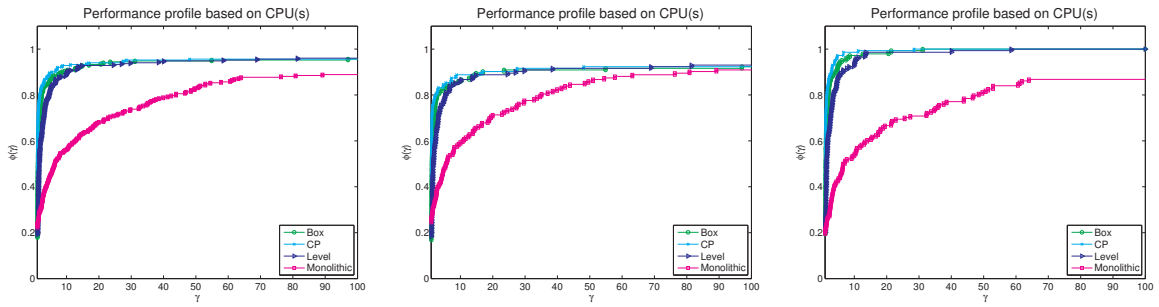
In our experiments, we have compared the "`Monolithic`" approach and three variants of the generalized Benders' one:

1. `CP`, the cutting-plane method of Algorithm 1;
2. `Box`, the trust region cutting-plane method of Algorithm 3;
3. `Level` the level cutting-plane method of Algorithm 4.

Each of these methods used the primal-dual oracle where `Cplex` was used to solve the SOCP formulation of the subproblem (cf. (25)). Several experiments were carried out with a dual oracle, but it was not found to be competitive in this setting and we do not report the corresponding results. The standard cutting planes method (Algorithm 2) adds a feasibility cut whenever problem (5) is found to be infeasible and an optimality cut otherwise. The box stabilization mechanism added on top of this method changes the stability center whenever a better solution is found, i.e., $\beta = 0$ (up to machine precision) and moves through box sizes $\{0.005, 0.5, 1\}\,|S|$. The level stabilized method uses mechanism (20) described after Theorem 6 with $\alpha = 0.9$. The lower bound v_k^{low} is updated by also solving the standard master problem (8) at every iteration as discussed after Theorem 6. We also systematically chose the last iterate as the next stability center $\hat{z}^k$; we have experimented

with several other choices for the center update rule and found little changes. The stabilization parameter is not very large, but close to the optimal continuous choice 0.18 exhibited in [46].

In order to compare the solvers we will make use of performance profiles [27], which read as follows: the parameter λ represents a scalar value, and $R(\lambda)$ the percentage of problems on which a given solver was no slower than λ times the fastest solver on that particular problem. The value $R(0)$ shows which solver is the fastest one and the value $R(\infty)$ is the percentage of instances that the solver managed to solve. In figure 1(a) we can observe that the generalized Benders methods manage to outperform the monolithic solver on different fractions of problems. The cutting planes variants outperforms the monolithic solver in roughly 40% of the instances, whereas the other methods are roughly each at 20%.



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

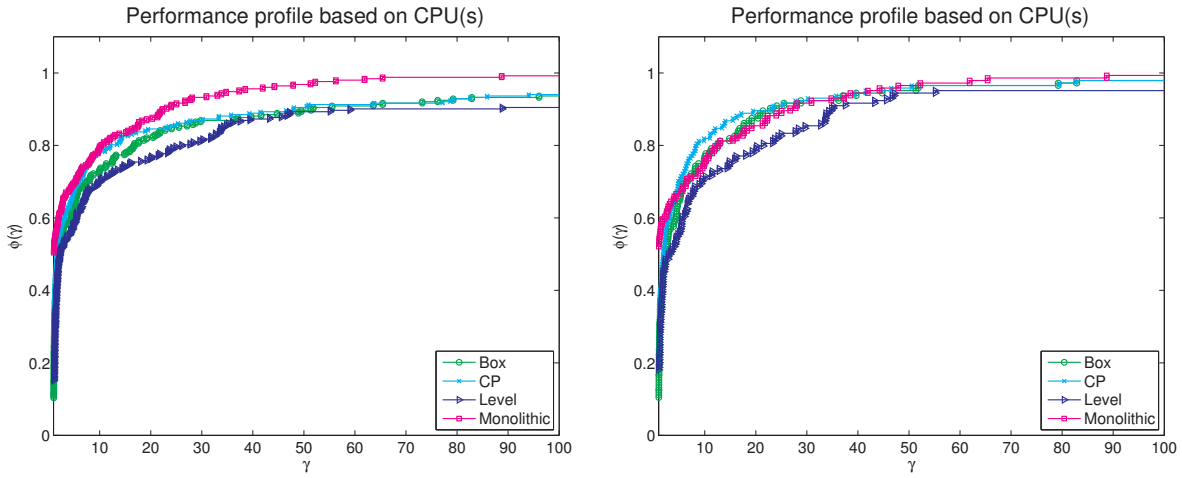
Figure 1: Performance profiles for the different methods based on cpu time

The importance 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 ($\{0.1\%, 0.01\%\}$). The monolithic solver gets more and more outperformed by the generalized Benders methods as sparsity increases. Since in practice, problems are often highly sparse (even below 0.001%), this shows the potential for generalized Benders based methods.

The impact of the stopping tolerance can be gauged by comparing figure 1(a) with figure 2(a), where the performance profiles are reported when a stopping tolerance $\delta_{\text{Tol}} = \texttt{1e-3}$ is required. For this coarser tolerance, the monolithic approach outperforms the generalized Benders based methods. However, this is mostly true for dense instances only: as shown in figure 2(b), for sparser instances the cutting plane based methods still remain competitive even at a lower precision.

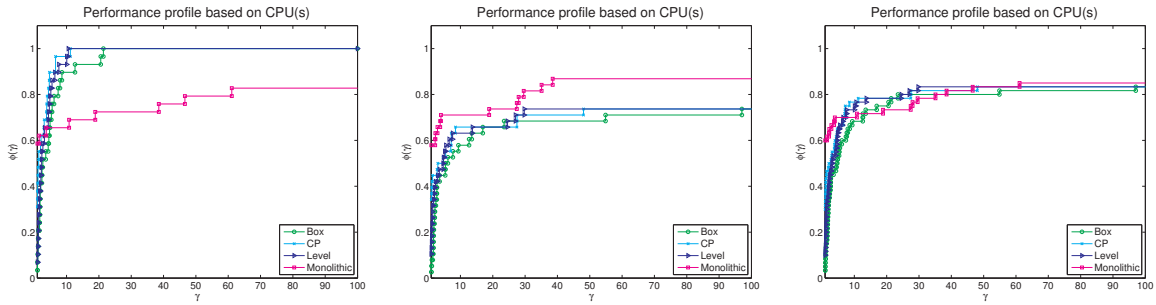
One issue with performance profiles is that they do not discriminate among instances of widely varying "difficulty". That is, two solvers tested on two instances such that the first one has a running time of 2 and 1000 while and the second one has a running time of 1 and 2000 (in whichever units) would 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 take 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 $\delta_{\text{Tol}} = \texttt{1e-4}$).

The performance profiles show that while the monolithic approach can be considered competitive for easy instances, this is only so for dense problems: for sparse ones, although the monolithic is the fastest approach in around 60% of the cases, the other solvers based on the generalized Benders decomposition are far more robust. The picture is even clearer for hard instances, as shown in



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

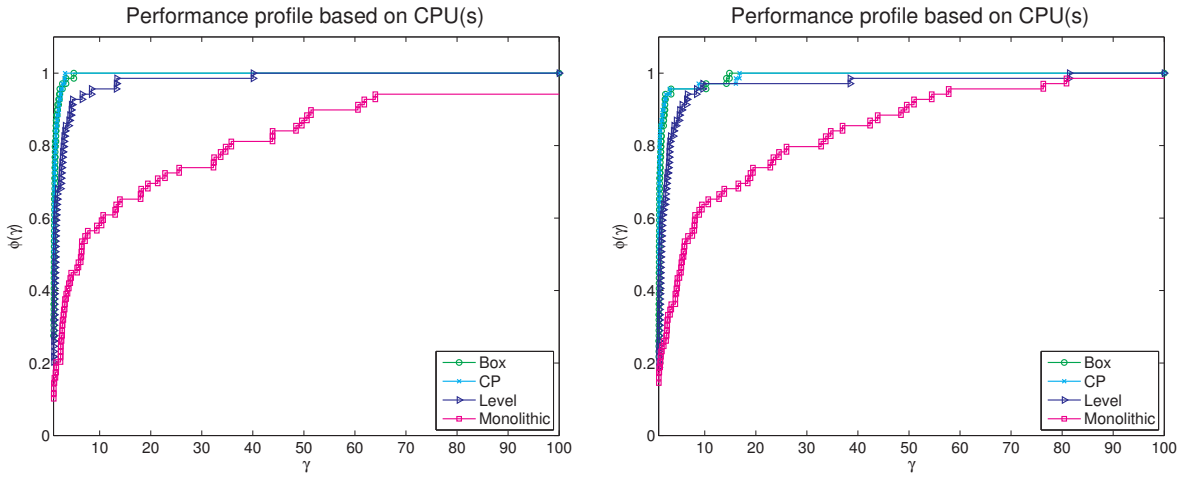
Figure 2: Performance profiles for the different methods based on cpu time with $\delta_{\text{Tol}} = \texttt{1e-3}$



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

Figure 3: Performance profiles for the different methods based on cpu time on the easy instances

figures 4(a) and 4(b): in these cases, the monolithic approach is significantly outperformed also on low sparsity instances, although slightly less so than on high sparsity ones. Very similar results were obtained for the intermediate instances, and therefore we choose not to report the corresponding profiles.



(a) Instances with high sparsity

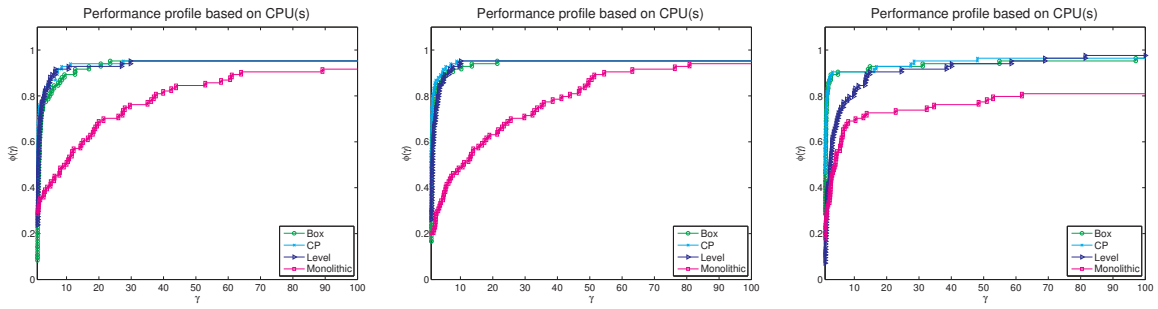
(b) Instances with low sparsity

Figure 4: Performance profiles for the different methods based on cpu time on the hard instances

In order to illustrate the effect of using a target $\texttt{tar} < \infty$, we have resorted to the use of the dual oracle. As the primal oracle is concerned, we have experimented with several `Cplex` features, such as callbacks and it turns out that a feasible primal and dual solution to problem (5) are only returned at the last of some 20 iterations. In a very similar way, changing ε_k in the primal oracle did not allow us to obtain a correct lower estimate $\underline{v}(z)$ of $v(z)$. In order to measure the gain in time we have executed the dual oracle twice at each iteration : once with $\texttt{tar}_k = \infty$ and once with a finite target (best current value). We have performed this comparison with the cutting planes method. We have measured the ratio oracle time(target) / oracle time (∞) and call efficiency 1 minus this ratio. This provides us over the 252 data sets with an average efficiency of 77.23 %, with a 20.73 % standard deviation and maximum efficiency of 99.4 %. In only two data sets a negative efficiency was observed (-36.7 % and -25.2 %).

We also separately examined the impact of the number of scenarios. We observed that the generalized Benders solvers have a rather stable behaviour as $|S|$ is concerned, whereas the robustness of the monolithic solver worsens as $|S|$ increases. This is illustrated in figures 5(a), 5(b) and 5(c).

Our results prove that, on this class of problems, and for our specific test set, the generalized Benders' approaches are in general competitive with the monolithic one. Actually, our results seem also to provide some guidelines for choosing the best method. In particular, the monolithic should be preferred only for "easy" and "dense" instances with $\delta_{\text{Tol}} = \texttt{1e-4}$, whereas it is always competitive (but not necessarily the best choice on sparse instances) when $\delta_{\text{Tol}} = \texttt{1e-3}$. In all the other cases, the cutting plane approaches are significantly better.



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

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

6 Conclusions and perspectives

In this paper we have studied the combination of two approaches for improving the performances of generalized Benders' decomposition approaches: different forms of stabilization (proximal/trust region and level), and the use of inexact oracles. We have also applied our results on a class of chance-constrained optimization problems with underlying discrete distribution arising by mixing two different sources of uncertainty in an uncertain linear program. The resulting randomly generated instances span different classes of difficulty. We have voluntarily remained within a class of problems that a monolithic commercial solver such as `Cplex` can solve.

Our approach consists of splitting the underlying combinatorial part from the underlying convexity structure through the use of a convex value function, defined by a convex programming problem. Computing a linearization for the value function through an oracle is performed in an efficient way by employing the concept of oracles with on-demand accuracy. Both dual and primal-dual oracles can be used for computing linearizations of the value function. Although the dual oracle was not competitive for the considered test problems, it may prove so for other ones (for instance in large-scale 2-stage unit-commitment problems wherein the primal oracle is simply not computable, e.g., [70]). An interesting property of oracle with on-demand accuracy is that they are able to provide linearizations regardless of the existence 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. This feature makes the algorithm converge regardless of the use of feasibility cuts (unless the problem is infeasible).

The theoretical framework laid out here allows for many variants of the rules governing the two main parameters dictating the behaviour of the inexact oracle, namely, the target and the accuracy. Although motivated by the specific CCO application, the framework is completely generic: possibly barring a few minor technical details (the form of the trust-region constraints and the reverse region ones in §4.1, the exact form of the objective function in §4.2), it directly applies to any minimization problem with finite domain Z such that the function $v(z)$ can be computed by an *informative on-demand inexact oracle* (11). This comprises many variants of (generalized) Benders' approaches where the master problem is a pure integer (most often, pure binary) problem, as well as other applications. 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., [61]), which makes obtaining good upper and lower estimates even more time

consuming. Our 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 [77], 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, but very similar results could be expected in the non-finite compact case.

The choices are so versatile that we believe that this might provide important tools for tackling the problems amenable by the Benders' decomposition approach in practice. In particular, our numerical results show that, at least in our application, even a simple set-up of the generalized Benders' methods can significantly outperform a monolithic approach. This is particularly true under some conditions about the sparsity level of the entry data, the number of underlying scenarios and the required final accuracy, which we computationally analyze.

We conclude by mentioning 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 provide, in our opinion, important tools for solving problems as (1). We therefore expect that this study will be useful for future research on combinatorial algorithms for chance-constrained problems with finite support, as well as providing inspiration for improving the performances of Benders' decomposition approaches to many other applications.

Acknowledgements

The authors gratefully acknowledges financial support from the Gaspard-Monge program for Optimization and Operations Research (PGMO) project "Consistent Dual Signals and Optimal Primal Solutions".

References

[1] R. Apparigliato. *Règles de décision pour la gestion du risque: Application á la gestion hebdomadaire de la production électrique*. PhD thesis, École Polytechnique, Juin 2008.

[2] D. Baena, J. Castro, and A. Frangioni. Stabilized Benders methods for large-scale combinatorial optimization: applications to data privacy, 2015.

[3] X. Bao, N.V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. *Optimization Methods and Software*, 24:485–504, 2009.

[4] A. Belloni and C. Sagastizábal. Dynamic bundle methods. *Math. Programming Series A*, 120(2):289–311, 2009.

[5] H. Ben Amor, J. Desrosiers, and A. Frangioni. On the Choice of Explicit Stabilizing Terms in Column Generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.

[6] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[7] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, Engineering Applications*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2001.

[8] P. Beraldi and A. Ruszczyński. A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software*, 17:359–382, 2002.

[9] S. Boyd and L. Vandenberghe. Convex optimization. *Available at http://www.stanford.edu/ boyd/cvxbook*, ISBN 0 521 83378 7, 2006.

[10] U. Brannlund, K. C. Kiwiel, and P. O. Lindberg. A descent proximal level bundle method for convex nondifferentiable optimization. *Operations Research Letters*, 17(3):121 – 126, 1995.

[11] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2):299–344, 2008.

[12] G. Calafiore and F. Dabbene (Eds). *Probabilistic and Randomized Methods for Design under Uncertainty*. Springer, 1st edition, 2006.

[13] G. C. Calafiore and M. C. Campi. Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

[14] M. C. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, 2011.

[15] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[16] C. d'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. On Interval-Subgradient Cuts and No-Good Cuts. *Operations Research Letters*, 38:341–345, 2010.

[17] G. d'Antonio and A. Frangioni. Convergence Analysis of Deflected Conditional Approximate Subgradient Methods. *SIAM Journal on Optimization*, 20(1):357–386, 2009.

[18] E. de Klerk. The complexity of optimizing over a simplex, hypercube or sphere: a short survey. *Central European Journal of Operations Research*, 16(2):111–125, 2008.

[19] W. de Oliveira. Regularized optimization methods for convex minlp problems. Technical report, 2014. Available at `http://www.bcamath.org/documentos_public/archivos/publicaciones/elbm.pdf`.

[20] W. de Oliveira and C. Sagastizábal. Level bundle methods for oracles with on demand accuracy. *Optimization Methods and Software*, 29(6):1180–1209, 2014.

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

[22] D. Dentcheva. Optimization models with probabilistic constraints. In G. Calafiore and F. Dabbene, editors, *Probabilistic and Randomized Methods for Design under Uncertainty*, pages 49–97. Springer, 1st edition, 2006.

[23] D. Dentcheva. *Optimisation Models with Probabilistic Constraints. Chapter 4 in [62].* MPS-SIAM series on optimization. SIAM and MPS, Philadelphia, 2009.

[24] D. Dentcheva, B. Lai, and A. Ruszczyński. Dual methods for probabilistic optimization problems. *Mathematical Methods of Operations Research*, 60(2):331–346, 2004.

[25] D. Dentcheva and G. Martinez. Regularization methods for optimization problems with probabilistic constraints. *Math. Programming (series A)*, 138(1-2):223–251, 2013.

[26] D. Dentcheva, A. Prékopa, and A. Ruszczyński. Concavity and efficient points for discrete distributions in stochastic programming. *Mathematical Programming*, 89:55–77, 2000.

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

[28] G. Emiel and C. Sagastizábal. Incremental like bundle methods with applications to energy planning. *Computational Optimization and Applications*, 46(2):305–332, 2009.

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

[30] S. Feltenmark and K.C. Kiwiel. Dual applications of proximal bundle methods, including lagrangian relaxation of nonconvex problems. *SIAM Journal on Optimization*, 10(3):697–721, 2000.

[31] E.C. Finardi and E.L. Da Silva. Solving the hydro unit commitment problem via dual decomposition and sequential quadratic programming. *IEEE Transactions on Power Systems*, 21(2):835–844, 2006.

[32] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003.

[33] C.A. Floudas and P.M. Pardalos (Eds). *Encyclopedia of Optimization.* Springer - Verlag, 2nd edition, 2009.

[34] A. Frangioni. Generalized bundle methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

[35] A. Frangioni and B. Gendron. A Stabilized Structured Dantzig-Wolfe Decomposition Method. *Mathematical Programming*, 140:45–76, 2013.

[36] A. Frangioni and E. Gorgone. Generalized Bundle Methods for Sum-Functions with "Easy" Components: Applications to Multicommodity Network Design. *Mathematical Programming*, 145(1):133–161, 2014.

[37] A. Frangioni, A. Lodi, and G. Rinaldi. New approaches for optimizing over the semimetric polytope. *Mathematical Programming*, 104(2-3):375–388, 2005.

[38] A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[39] F. Giannessi, S. Komlósi, and T. Rapcsák (Eds). *New Trends in Mathematical Programming : Hommage to Steven Vajda*, volume 13 of *Applied Optimization.* Springer, 1st edition, 1998.

[40] R. Henrion. *Qualitative stability of convex programs with probabilistic constraints. Chapter 12 in [43].* Springer Berlin Heidelberg, 2000.

[41] R. Henrion. Introduction to chance constraint programming. *Tutorial paper for the Stochastic Programming Community HomePage, http://www.wias-berlin.de/people/henrion/publikat.html*, 2004.

[42] R. Henrion and A. Möller. A gradient formula for linear chance constraints under Gaussian distribution. *Mathematics of Operations Research*, 37:475–488, 2012.

[43] N. Van Hien, J-J. Strodiot, and P. Tossings (Eds). *Optimization : Proceedings of the 9th Belgian-French-German Conference on Optimization Namur*, volume 481 of *Lecture Notes in Economics and Mathematical Systems*. Springer Berlin Heidelberg, 2000.

[44] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*. Number 306 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 2nd edition, 1996.

[45] J.E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[46] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Math. Programming*, 69(1):111–147, 1995.

[47] J. Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, 146(1-2):219–244, 2014.

[48] J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19:674–699, 2008.

[49] R.E. Marsten, W.W. Hogan, and J.W. Blankenship. The BOXSTEP method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

[50] A. Nemirovski and A. Shapiro. Scenario approximations of chance constraints. *Preprint : http://www.optimization-online.org/DB_HTML/2004/11/1000.html*, pages 1–45, 2004.

[51] A. Nemirovski and A. Shapiro. *Scenario Approximations of Chance Constraints (Chapter 1 in [12]).* Springer, 2006.

[52] A. Prékopa. 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] A. Prékopa. *Stochastic Programming.* Kluwer, Dordrecht, 1995.

[54] A. Prékopa. *Probabilistic programming. In [60] (Chapter 5).* Elsevier, Amsterdam, 2003.

[55] A. Prékopa, B. Vízvári, and T. Badics. *Programming under probabilistic constraints with discrete random variable. In [39].* Springer, 1998.

[56] A. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimimization*, 10:425–437, 1997.

[57] A. Ruszczyński. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, 93:195–215, 2002.

[58] A. Ruszczyński. *Decomposition Methods (Chapter 3 in [60])*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, Amsterdam, 2003.

[59] A. Ruszczyński. *Nonlinear Optimization*. Princeton. Princeton University Press, 2006.

[60] A. Ruszczyński and A. Shapiro. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, Amsterdam, 2003.

[61] S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106:203–223, 2006.

[62] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming. Modeling and Theory*, volume 9 of *MPS-SIAM series on optimization*. SIAM and MPS, Philadelphia, 2009.

[63] P. Sun and R.M. Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004.

[64] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra. Large-scale unit commitment under uncertainty: a literature survey. *4OR*, available online, 2015.

[65] M. Tawarmalani, J.-P.P. Richard, and C. Xiong. Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming*, 138(1-2):531–577, 2013.

[66] S. Uryas'ev. *Derivatives of probability and Integral functions: General Theory and Examples. Appearing in [33]*. Springer - Verlag, 2nd edition, 2009.

[67] W. van Ackooij and W. de Oliveira. Level bundle methods for constrained convex optimization with various oracles. *Computation Optimization and Applications*, 57(3):555–597, 2014.

[68] W. van Ackooij and R. Henrion. Gradient formulae for nonlinear probabilistic constraints with gaussian and gaussian-like distributions. *Siam Journal on Optimization*, 24(4):1864–1889, 2014.

[69] W. van Ackooij, R. Henrion, A. Möller, and R. Zorgati. Joint chance constrained programming for hydro reservoir management. *Optimization and Engineering*, 15:509–531, 2014.

[70] W. van Ackooij and J. Malick. Decomposition algorithm for large-scale two-stage unit-commitment. *draft submitted*, pages 1–26, 2014.

[71] W. van Ackooij and C. Sagastizábal. Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. *Siam Journal on Optimization*, 24(2):733–765, 2014.

[72] B. Vízvári. The integer programming background of a stochastic integer programming algorithm of Dentcheva-Prékopa-Ruszczyński. *Optimization Methods and Software*, 17:543–559, 2002.

[73] T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex minlp problems. *Computers & Chemical Engineering*, 19, Supplement 1(0):131 – 136, 1995.

[74] T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3:253–280, 2002.

[75] C. Wolf, C. I. Fábián, A. Koberstein, and L. Stuhl. Applying oracles of on-demand accuracy in two-stage stochastic programming a computational study. *European Journal of Operational Research*, 239(2):437–448, 2014.

[76] G. Zakeri, A.B. Philpott, and D.M. Ryan. Inexact cuts in benders decomposition. *SIAM Journal on Optimization*, 10(3):643–657, 2000.

[77] S. Zaourar and J. Malick. Quadratic stabilization of benders decomposition. pages 1–22, 2014. Draft submitted; Privately communicated.