Adaptive Partitioning for Chance-Constrained Problems with Finite Support

Marius Roland, Alexandre Forel, Thibaut Vidal

Abstract. This paper studies chance-constrained stochastic optimization problems with finite support. It presents an iterative method that solves reduced-size chance-constrained models obtained by partitioning the scenario set. Each reduced problem is constructed to yield a bound on the optimal value of the original problem. We show how to adapt the partitioning of the scenario set so that our adaptive method returns the optimal solution of the original chance-constrained problem in a finite number of iterations. At the heart of the method lie two fundamental operations: refinement and merging. A refinement operation divides a subset of the partition, whereas a merging operation combines a group of subsets into one. We describe how to use these operations to enhance the bound obtained in each step of the method while preserving the small size of the reduced model. Under mild conditions, we prove that, for specific refinement and merge operations, the bound obtained after solving each reduced model strictly improves throughout the iterative process. Our general method allows the seamless integration of various computational enhancements, significantly reducing the computational time required to solve the reduced chance-constrained problems. The method's efficiency is assessed through numerical experiments on chance-constrained multidimensional knapsack problems. We study the impact of our method's components and compare its performance against other methods from the recent literature.

1. Introduction

We consider Chance-Constrained Stochastic Programs (CCSPs). Solving a CCSP amounts to finding the optimal value of a decision variable vector $x \in \mathcal{X} \subseteq \mathbb{R}^n$ that minimizes an objective function $f : \mathcal{X} \to \mathbb{R}$. This decision variable has to satisfy a constraint that depends on an uncertain parameter $\xi \in \Xi$ with a probability of $(1-\tau)$. Here, the parameter $\tau \in [0, 1]$ represents the risk tolerance of the decision-maker. The CCSP reads

$$v^* = \min_{x \in \mathcal{X}} \quad f(x) \tag{1a}$$

$$\text{s.t.} \quad \mathbb{P}_\xi[x \in X(\xi)] \geq 1 - \tau. \tag{1b}$$

CCSPs were first introduced in [9]. Such models are used in a variety of fields such as power systems [10, 26], vehicle routing [13], finance [8], and contextual optimization [28].

More specifically, we focus on CCSPs whose uncertain parameters $\xi \in \Xi$ have finite support. The uncertain parameters belong to the set $\Xi := \{\xi^s : s \in S\}$ where each $\xi^s \in \mathbb{R}^d$ is a multi-dimensional vector representing a single realization of the uncertain parameters with probability q_s and S is a set of scenarios. When ξ has

Date: December 18, 2023.

2020 *Mathematics Subject Classification.* 90C15, 90C11.

Key words and phrases. Chance Constraints, Adaptive Partitioning, Stochastic Optimization.

finite support, Model (1) can be reformulated as

$$v^* = \min_{x \in \mathcal{X}} \quad f(x) \tag{2a}$$

$$\text{s.t.} \quad \sum_{s \in S} q_s \mathbb{1}\left(x \in X^s\right) \geq 1 - \tau, \tag{2b}$$

where $\mathbb{1}$ is the indicator function, and $X^s = X(\xi^s)$ is the set of feasible decisions for realization ξ^s. Models of the type (2) are for instance obtained when the generic CCSP (1) is approximatied using as Sample Average Approximation (SAA). The SAA method is a widely adopted approach, particularly when the distribution of ξ is unknown [3, 24]. This is primarily because of its simplicity and the theoretical guarantees it offers [21].

By introducing a binary variable $z_s \in \{0, 1\}$ for each $s \in S$, Model (2) can be reformulated as

$$v^* = \min_{x \in \mathcal{X}} \quad f(x) \tag{3a}$$

$$\text{s.t.} \quad z_s = \mathbb{1}(x \in X^s), \quad s \in S, \tag{3b}$$

$$\sum_{s \in S} q_s z_s \geq 1 - \tau, \tag{3c}$$

$$z_s \in \{0, 1\}, \quad s \in S \tag{3d}$$

Model (3) is regularly solved by introducing so-called "big-M" coefficients. Big-M coefficients allow to reformulate the indicator constraints (3b) without introducing additional variables or constraints. Let the feasible set of a scenario X^s be fully characterized by an inequality system $G^s(x) \leq 0$, where $G^s : \mathbb{R}^n \to \mathbb{R}^m$. Further, assume that an upper bound M_i^s exists on the maximum violation of the i-th constraint of scenario s for x in the set of feasible solutions of Model (3). Then, we may write

$$v^* = \min_{x \in \mathcal{X}} \quad f(x) \tag{4a}$$

$$\text{s.t.} \quad G^s(x) \leq M^s(1 - z^s), \quad s \in S, \tag{4b}$$

$$\sum_{s \in S} q_s z_s \geq 1 - \tau, \tag{4c}$$

$$z_s \in \{0, 1\}, \quad s \in S, \tag{4d}$$

where M^s is a vector of entries M_i^s. Large values for M_i^s result in a loose continuous relaxation of Model (3), see, e.g., [27]. This is one of the reasons why branch-and-bound solvers struggle to solve linear chance-constrained problems. In addition, finding tight values for M_i^s can be very time-consuming.

1.1. **Contributions.**

(1) We propose an adaptive partitioning method for solving CCSPs with finite support to optimality. The method is based on iteratively solving reduced-size CCSPs that yield lower bounds on the optimal objective of the original CCSP. The partitions are adapted so that solutions obtained in previous iterations of the method are excluded from the feasible set of the reduced problem. By construction, the adaptive partitioning method terminates after a finite number of iterations and returns the optimal solution to the original CCSP.

(2) We specify the main operations of the method that are based on mathematical arguments. Namely, we study how to refine and merge an existing partition of the scenario set so that the lower bound obtained by solving a partitioned problem is guaranteed to strictly increase. This property

has significant benefits when solving partitioned problems, it implies that big-M coefficients can be reused and tightened in successive iterations. This tightens the continuous relaxation of the reduced CCSPs that are solved and reduces the time needed to solve them. It further allows us to take increasing advantage of screening [26] as the number of iterations of the method increases.

(3) We discuss how to create partitions that, by construction, result in a lower bound on the original optimal objective that is tighter than the well-studied quantile bound [2]. In certain cases, this initial partition results in points that are feasible and hence optimal for the original CCSP.

(4) We compare the performance of the adaptive partitioning method for solving CCSPs with binary and continuous variables with state-of-the-art methods. The results demonstrate the computational advantage of the proposed method. We highlight why the proposed method performs well numerically by examining the effect of the main operations of the method.

1.2. **Related Literature.** This paper lies at the interface between two branches of literature. The first branch concerns solving CCSPs with finite support. Different families of valid inequalities have been proposed for solving CCSPs. The intersection between well-studied mixing inequalities [16] and CCSPs have received a lot of attention [20, 22, 1, 19]. Quantile cuts and their relationship with mixing inequalities [37] have also been studied [27, 18]. In [36] the authors propose to solve CCSPs using a branch-and-cut approach that exploits cuts based on irreducibly infeasible subsystems of scenarios. Other approaches that do not use valid inequalities have also been proposed. In [2] the authors study solution methods based on relaxations of the original CCSP. Moreover, they propose a scenario decomposition approach tailored to CCSPs with binary decision variables. The scenario decomposition approach is designed to iteratively solve the same relaxation of the original CCSP and add "no-good" cuts to exclude previously obtained solutions from the feasible set. In their conclusion, they highlight that a promising research direction is to investigate other techniques for excluding feasible solutions. This paper is in direct connection with our work since we take a similar direction for solving CCSPs. Moreover, our approach is independent of the types of variables that are considered. Further, some approaches are concerned with developing alternative problem formulations without binary variables [35, 34]. The tightening of big-M parameters has also received much attention [26, 27].

The second branch of literature related to this paper is scenario reduction for solving stochastic problems with large scenario sets. We differentiate between methods that carry out the scenario reduction separately from the optimization process and the approaches that consider both aspects simultaneously. A priori scenario reduction is performed by minimizing the Wasserstein distance [17, 12] between the original and a smaller size scenario set. This idea is extended by some authors by incorporating objective function information in the scenario reduction problem [23, 6]. Recently, in [31] the authors prove tight bounds on the approximation error of a reduced scenario set obtained via Wasserstein distance minimization. Combining scenario reduction and optimization is rather recent and has been applied to some stochastic optimization models. The generalized adaptive partition method (APM) for two-stage stochastic problems with fixed recourse in [33] has a strong link with our contribution. This method was initially proposed in [14] for CVaR minimization and is based on the ideas presented in [7]. Recently, in [30] the generalized APM ideas were extended to the case where the uncertain parameters follow a continuous distribution. Moreover, APMs have been combined with decomposition methods [25], stochastic dual dynamic programming [32], and Benders decomposition [29].

To the best of our knowledge, APMs have not been proposed for chance-constrained problems, despite being strongly connected to the scenario decomposition approach presented in [2]. We bridge this gap by proposing an adaptive partitioning approach for solving general CCSPs with finite support.

1.3. **Outline.** Section 2 presents the general adaptive partitioning method and discusses how to obtain upper and lower bounds on the original problem objective by solving partitioned problems. Section 3 studies how to efficiently refine partitions by excluding existing feasible solutions. Section 4 shows similar results for merge operations. Section 5 presents how to create partitions with a guaranteed tight lower bound on the objective of the original CCSP. Section 5 presents practical strategies that enhance the behavior of the algorithm. Section 6 evaluates the computational performance of our algorithm and compares it with state-of-the-art methods on classical instances of the CCSP literature. Finally, Section 7 summarizes our findings and suggests directions for future work.

2. Adaptive Partitioning Method

Our approach is based on creating partitions of the scenarios set, that is, grouping scenarios into subsets. We first formally define a partition. Then, we present two reduced-size models obtained via a partitioning of the scenario set. Both models yield bounds on the optimal objective of the original CCSP. After, we discuss how reduced-size models obtained via partitioning allow to compute tight big-M parameters for Model (4). Finally, we explain the APM and discuss its finite termination.

Definition 1. *A partition* $P = \{p_1, p_2, \dots, p_{|P|}\}$ *is a collection of non-empty subsets of the scenario set* S *such that* $\bigcup_{p\in P} p = S$ *and* $p_i \cup p_j = \emptyset$, *for all* $p_i, p_j \in P$.

2.1. **Partitioned CCSPs.** We now present how a reduced size CCSP is obtained by leveraging a partition P of a scenario set S. By aggregating the constraints of all the scenarios in a subset, we can construct two smaller-size chance-constrained problems that yield an upper and a lower bound on the optimal objective of the original CSSP. In the reduced CCSP each subset $p \in P$ represents a unique scenario and the feasible set for $p \in P$ is $X^p = \bigcap_{s\in p} X^s$.

In [2] the authors present the following reduced-size CCSP,

$$v^{\mathrm{L}}(P) = \min_{x\in\mathcal{X}} \quad f(x) \tag{5a}$$

$$\text{s.t.} \quad z_p = \mathbb{1}(x \in X^p), \quad p \in P, \tag{5b}$$

$$\sum_{p\in P} q_p z_p \geq \sum_{p\in P} q_p - \tau, \tag{5c}$$

$$z_p \in \{0,1\}, \quad p \in P, \tag{5d}$$

where for each subset $p \in P$ the probability $q_p = \min_{s\in p} q_s$.

Proposition 1 (from [2])**.** *The partitioned model* (5) *is a relaxation of the CCSP* (3)*, i.e.,* $v^L(P) \leq v^*$.

Conversely, if we set the probability of each subset $p \in P$ to $q_p = \sum_{s \in p} q_s$. Then, another reduced-size CCSP reads

$$v^{\mathrm{U}}(P) = \min_{x \in \mathcal{X}} \quad f(x) \tag{6a}$$
$$\text{s.t.} \quad z_p = \mathbb{1}(x \in X^p), \quad p \in P, \tag{6b}$$
$$\sum_{p \in P} q_p z_p \geq 1 - \tau, \tag{6c}$$
$$z_p \in \{0,1\}, \quad p \in P. \tag{6d}$$

Proposition 2. *The scenario grouping model (6) is a restriction of the CCSP (3), i.e., $v^* \leq v^U(P)$.*

Proof. Let $x' \in \mathcal{X}$ be any point inside the feasible set of Model (6) and let z'_p satisfy Constraint (6b). We set $z_s = z'_p$ for all $p \in P$ and $s \in p$. Then, Constraint (6c) yields $\sum_{s \in S} q_s z_s = \sum_{p \in P} q_p z_p \geq 1 - \tau$ and x' is feasible for Model (3). □

2.2. **Big-M Tightening.** As discussed in Section 1, tightening big-M coefficients is of major importance for problems with indicator variables [5]. The value of big-M coefficients directly influences the size of the continuous relaxation of Model (4). Thus, it also indirectly influences the time required to solve Model (4) to optimality. For any chance-constrained problem, the tightest value for the big-M parameter of the i-th constraint of a scenario s is obtained by solving the following optimization problem, see, e.g., [35],

$$\bar{M}_i^s = \max_{x \in \mathcal{X}} \quad G_i^s(x) \tag{7a}$$
$$\text{s.t.} \quad G^s(x) \leq M^s(1 - z^s), \quad s \in S, \tag{7b}$$
$$\sum_{s \in S} q_s z_s \geq 1 - \tau, \tag{7c}$$
$$z_s \in \{0,1\}, \quad s \in S, \tag{7d}$$

Model (7) is a chance-constrained problem and it can be as hard to solve as Model (4). Thus, it is largely unpractical to consider such an approach for every big-M that needs to be computed.

The partitioned CCSP (5) can be used to obtain tight big-M coefficients. Indeed, Proposition 4 states that Model (5) is a relaxation of Model (3) which means that Model (5) can be used to find a tight approximation of $\bar{M}_i^s$. Given a partition P, we state the partitioned big-M tightening model,

$$M_i^s = \max_{x \in \mathcal{X}} \quad G_i^s(x) \tag{8a}$$
$$\text{s.t.} \quad G^s(x) \leq M^s(1 - z^p), \quad p \in P, s \in p, \tag{8b}$$
$$\sum_{p \in P} q_p z_p \geq \sum_{p \in P} q_p - \tau, \tag{8c}$$
$$z_p \in \{0,1\}, \quad p \in P, \tag{8d}$$

where $\bar{M}_i^s \leq M_i^s$. We observe that any partition P of S allows to produce a valid big-M value.

2.3. **Adaptive Partitioning Method.** We now discuss how the APM works. The lower bound model (5) and the upper bound model (6) are core components of the APM. The partition P is modified so that tighter upper and lower bounds on the original CCSP are obtained. Algorithm 1 describes the general idea of the APM.

Algorithm 1 contains three main steps: computing a lower bound, computing an upper bound, and modifying the partition. The modification of the partition

Algorithm 1: Adaptive Partitioning Method.

Input : scenario set S, stopping criterion $\varepsilon \in (0, 1)$.
Output : optimal solution x^* of Model (3).
Initialize : $j \leftarrow 0$, $v^{\mathrm{U}} \leftarrow +\infty$, $v^{\mathrm{L}} \leftarrow -\infty$.
Design the first partition P^0.
while $(v^U - v^L)/v^U \geq \varepsilon$ **do**
  `/* Compute a tighter lower bound. */`
  Find $\underline{x}^j$, the solution of Model (5) for the partition P^j.
  if $v(\underline{x}^j) > v^L$ **then**
    Set $x^{\mathrm{L}} \leftarrow \underline{x}^j$ and $v^{\mathrm{L}} \leftarrow v(\underline{x}^j)$.
  end
  `/* Compute a tighter upper bound. */`
  Find $\bar{x}^j$, the solution of Model (6) for the partition P^j.
  if $v(\bar{x}^j) < v^U$ **then**
    Set $x^{\mathrm{U}} \leftarrow \bar{x}^j$ and $v^{\mathrm{U}} \leftarrow v(\bar{x}^j)$.
  end
  `/* Modify the partition. */`
  Modify P^j to obtain a new partition P^{j+1}.
  Increment iteration $j \leftarrow j + 1$.
end
return x^U

has the biggest influence on the computational performance of the APM. Typically, Algorithm 1 exhibits a trade-off between solving a large number of computationally tractable CCSPs or solving a small number of computationally intensive CCSPs. The performance of Algorithm 1 also depends on the design of the initial partition P^0 as it dictates the tightness of the initial bounds w.r.t. the optimal value v^*. The initial partition P^0 also influences the structure of any subsequent partition.

2.4. **Finite Termination.** As long as $|P^j|$ increases over the course of the iterations, Algorithm 1 terminates in a finite number of iterations and recovers the optimal solution of Model (3). Indeed, if $|P^{j+1}| > |P^j|$ holds for all j, the partition P^j will eventually contain exactly one scenario per subset $p \in P^j$. In that case, Model (5) reduces to Model (3), and Algorithm 1 returns the optimal solution of Model (3). The worst-case number of iterations needed by Algorithm 1 to recover the optimal solution of Model (3) is $|S| - |P^0|$.

Naturally, a method that modifies the partition at each iteration using refinement operations fits this property and terminates in a finite number of iterations. However, the aim when applying Algorithm 1 is to avoid reaching the point where P^j equals S because in that case there is no computational advantage of using the APM.

Our presentation of Algorithm 1 allows for flexibility in designing each of its main steps. For instance, computing the upper bound can be performed by a heuristic that does not necessarily solve a reduced size problem obtained via a partitioning of the scenarios. In the following, we present several methods to design and modify partitions that preserve finite termination of the method. In particular, we discuss how to create partitions of minimal size in Algorithm 1 that guarantee a strict increase of the resulting lower bound. Our results contribute to identifying what constitutes a good partition.

3. On Constructing Minimal Size Refinements

In this section, we present how we propose to refine a partition in Algorithm 1. We make the following assumption throughout the document.

Assumption 1. *All scenarios are equiprobable, i.e.* $q_s = 1/|S|$ *for all* $s \in S$.

Assumption 1 is introduced mainly to simplify the notation. For instance, it always holds when scenarios are obtained via a sample-average approximation. Further, any scenario set with rational probabilities can be modified to satisfy Assumption 1 by duplication of scenarios. Naturally, this increases the size of the scenario set but does not change the feasible set of Model (3).

Under Assumption 1, Constraint (5c) can be simplified as stated in the following proposition.

Proposition 3. *If Assumption 1 is satisfied, then Constraint* (5c) *is equivalent to*

$$\sum_{p \in P} z_p \geq |P| - \lfloor \tau |S| \rfloor. \tag{9}$$

Proof. Since, $q_p = \min_{s \in p} 1/|S| = 1/|S|$, Constraint (5c) can be reformulated as

$$\sum_{p \in P} z_p \geq |P| - \tau |S|. \tag{10}$$

Further, since the variables z_p are binary, Equation (10) is equivalent to

$$\sum_{p \in P} z_p \geq |P| - \lfloor \tau |S| \rfloor.$$

□

Remark 1. *Proposition 3 highlights that any partition* P *should satisfy* $|P| > \lfloor \tau |S| \rfloor$. *Indeed, if* $|P| \leq \lfloor \tau |S| \rfloor$, *then Constraint* (5c) *is always trivially satisfied and any* $x \in \mathcal{X}$ *is feasible for Model* (5) *with* P.

3.1. Solution Exclusion by Refinement.

Refinement operations constitute the main component of the APM. This is explained by the fact that refinement successive operations guarantee finite termination of the APM at the optimal solution of the original CCSP.

Definition 2. *A partition* R *is a refinement of a partition* P *if for any* $r \in R$ *there exists a subset* $p \in P$ *such that* $r \subseteq p$.

We now study how to perform efficient refinement operations. We refine a partition by excluding the solution obtained in the previous iteration of the APM. We prove that if this solution is unique, then we can create a minimal size refinement R of P such that $v^{\mathrm{L}}(R) > v^{\mathrm{L}}(P)$.

To carry out this proof, we proceed in the following way. First, in Proposition 4, we show that for any refinement R of a partition P, Model (5) with P is a relaxation of Model (5) with R. Second, in Proposition 5, we identify a class of refinement operations that guarantee the exclusion of any feasible point of Model (5) with P from the feasible set of Model (5) with R. Third, in Proposition 6, we show that this refinement is of minimal size. Finally, in Theorem 1, we use the previous results to show how a refinement R satisfying $v^{\mathrm{L}}(R) > v^{\mathrm{L}}(P)$ is obtained in Algorithm 1.

To simplify the presentation, we use the following notation throughout the remainder of this section. We let P denote a partition of the scenario set S and R be a partition resulting from a refinement of P. Given any point $x' \in \mathcal{X}$, we define $z_p(x') = \mathbb{1}(x' \in \bigcap_{s \in p} X^s)$ the value of the indicator variables associated with a subset $p \in P$, as well as $P_{\mathrm{F}}(x) := \{p \in P : \ x \in X^p\}$ the set of subsets that are

feasible for x and $P_{\mathrm{I}}(x) := \{p \in P : \ x \notin X^p\}$ the set of subsets of P that are infeasible for x. Naturally, $P = P_{\mathrm{F}}(x) \cup P_{\mathrm{I}}(x)$.

Proposition 4. *Model (5) with P is a relaxation of Model (5) with R, i.e.,*

$$v^* \geq v^L(R) \geq v^L(P).$$

Proof. In what follows, we make a distinction between $v^{\mathrm{L}}(P,R)$ and $v^{\mathrm{L}}(P,S)$. The value of $v^{\mathrm{L}}(P,R)$ is associated to solving the lower bound model (5) with P where P is considered a partition of R. So, R is considered as a standalone scenario set in this case. Further, the value of $v^{\mathrm{L}}(P,S)$ is associated to solving the lower bound model (5) with P where P is considered a partition of S. So, the lower bound $v^{\mathrm{L}}(P,S)$ is the same as $v^{\mathrm{L}}(P)$.

To prove the statement we show that the model solved for computing $v^{\mathrm{L}}(P,R)$ is equivalent to the model solved for computing $v^{\mathrm{L}}(P,S)$. If this holds, then $v^{\mathrm{L}}(R) \geq v^{\mathrm{L}}(P,R) = v^{\mathrm{L}}(P,S) = v^{\mathrm{L}}(P)$. First, we reformulate the model of $v^{\mathrm{L}}(R)$ as a CCSP. For every $r \in R$ we introduce

$$\kappa_r = \frac{q_r}{\sum_{r' \in R} q_{r'}}, \quad \varepsilon = \frac{\tau}{\sum_{r' \in R} q_{r'}}.$$

Then, we have that

$$\begin{aligned} v^{\mathrm{L}}(R) = \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{s.t.} \quad & z_r = \mathbb{1}(x \in X^r), \quad r \in R, \\ & \sum_{r \in R} \kappa_r z_r \geq 1 - \varepsilon, \\ & z_r \in \{0,1\}, \quad r \in R, \end{aligned}$$

Since the model of $v^{\mathrm{L}}(R)$ is a CCSP we may consider R as a standalone scenario set. We show that the model of $v^{\mathrm{L}}(P,R)$ is equivalent to the model of $v^{\mathrm{L}}(P,S)$. We write Constraint (5c) for the model of $v^{\mathrm{L}}(P,R)$, it reads,

$$\begin{aligned} \sum_{p \in P} q_p z_p &\geq \sum_{p \in P} q_p - \varepsilon, \\ \sum_{p \in P} \min_{r \in p}(\kappa_p) z_p &\geq \sum_{p \in P} \min_{r \in p}(\kappa_p) - \varepsilon, \\ \sum_{p \in P} \min_{r \in p}(\min_{s \in r}(q_p)) z_p &\geq \sum_{p \in P} \min_{r \in p}(\min_{s \in r}(q_p)) - \tau, \end{aligned}$$

which is equivalent to Constraint (5c) for the model of $v^{\mathrm{L}}(P,S)$. Similarly, we state Constraint (3b) for the model of $v^{\mathrm{L}}(P,R)$,

$$z_p = \mathbb{1}(x \in X^p) = \mathbb{1}(x \in \bigcap_{r \in p} X^r) = \mathbb{1}(x \in \bigcap_{r \in p} \bigcap_{s \in r} X^s),$$

which is equivalent to Constraint (3b) for the model of $v^{\mathrm{L}}(P,S)$. Hence, $v^{\mathrm{L}}(R,S) = v^{\mathrm{L}}(R,P)$ and Model (5) with P is a relaxation of Model (5) with R. □

Remark 2. *Proposition 4 holds whether Assumption 1 is satisfied or not.*

Remark 3. *By Proposition 4 we know that any big-M coefficient obtained for a partition P is valid for any refinement R of P. This property allows to keep the same big-M values through the iterations of Algorithm 1 if only refinements are carried out.*

Next, we show that there exists a refinement such that an optimal solution $\underline{x}$ of Model (5) with P is infeasible for Model (5) with R.

Proposition 5. *Let the point $\underline{x}$ be feasible for Model (5) with P but infeasible for Model (3). There exists a refinement R of P with size $|R| = |P| + \mu$, where*

$$\mu = \lfloor \tau |S| \rfloor + 1 - |P_I(\underline{x})|,$$

such that $\underline{x}$ is infeasible for Model (5) with R.

Proof. For what follows, we fix the value of parameter μ to $\lfloor \tau |S| \rfloor + 1 - |P_{\mathrm{I}}(\underline{x})|$. In addition, we introduce the sets

$$P_1(\underline{x}) = \{p \in P : \ |p \cap S_{\mathrm{I}}(\underline{x})| = 1\}, \quad S_1(\underline{x}) = S_{\mathrm{I}}(\underline{x}) \cap (\cup_{p \in P_1(\underline{x})} p),$$
$$P_2(\underline{x}) = \{p \in P : \ |p \cap S_{\mathrm{I}}(\underline{x})| \geq 2\}, \quad S_2(\underline{x}) = S_{\mathrm{I}}(\underline{x}) \cap (\cup_{p \in P_2(\underline{x})} p),$$

The definition of $P_1(\underline{x})$ ensures that$|P_1(\underline{x})| = |S_1(\underline{x})|$. Since $\underline{x}$ is infeasible for Model (3) we know that $|S_{\mathrm{I}}(\underline{x})| \geq \lfloor \tau |S| \rfloor + 1$. We can write

$$\begin{aligned} \mu &\leq |S_{\mathrm{I}}(\underline{x})| - |P_{\mathrm{I}}(\underline{x})|, \\ &\leq |S_1(\underline{x})| + |S_2(\underline{x})| - |P_1(\underline{x})| - |P_2(\underline{x})|, \\ &\leq |S_2(\underline{x})| - |P_2(\underline{x})|, \end{aligned} \tag{12}$$

By Equation (12) we know that there exist at least μ scenarios $s \in S_2(\underline{x})$ that can be removed from a set $p \in P_2(\underline{x})$ while insuring that $|P_{\mathrm{I}}(\underline{x})|$ stays unchanged. These scenarios can be used to create μ new infeasible subsets in R. In other words, each new subset $r \in R \setminus P$ satisfies the property $\underline{x} \notin X^r$ because r is contains at least one $s \in S_{\mathrm{I}}(\underline{x})$.

Next, we create this refinement R of partition P by creating exactly μ new subsets r for which $\underline{x} \notin X^r$. We have

$$\begin{aligned} |R| &= |P| + \mu, \\ &= |P_{\mathrm{F}}(\underline{x})| + \lfloor \tau |S| \rfloor + 1, \\ &= \sum_{r \in R} z_r(\underline{x}) + \lfloor \tau |S| \rfloor + 1, \end{aligned} \tag{13}$$

where we used that $|P(\underline{x})| = |P_{\mathrm{I}}(\underline{x})| + |P_{\mathrm{F}}(\underline{x})|$ and $\sum_{r \in R} z_r(\underline{x}) = |R_{\mathrm{F}}(\underline{x})| = |P_{\mathrm{F}}(\underline{x})| = \sum_{p \in P} z_p(\underline{x})$. By reorganizing terms in Equation (13), we get

$$\sum_{r \in R} z_r(\underline{x}) = |R| - \lfloor \tau |S| \rfloor - 1, \tag{14}$$

which, by Proposition 3, implies that $\underline{x}$ is not feasible for Model (5) with R. □

The proof of Proposition 5 describes a method that requires low computational effort for creating a refined partition R where $\underline{x}$ is excluded from the feasible set of Model (5) with R. Algorithm 2 describes this method. As explained in the proof of Proposition 5, R is created by splitting μ subsets that contain at least two infeasible scenarios. Each split creates two new subsets that contain at least one infeasible scenario. The next proposition shows that R obtained by applying Algorithm 2 is of minimal size.

Proposition 6. *Let the point $\underline{x}$ be feasible for Model (5) with P but infeasible for Model (3). If a refinement R of P is such that $\underline{x}$ is not feasible for Model (5) with R, then*

$$|R| - |P| \geq \mu = \lfloor \tau |S| \rfloor + 1 - |P_I(\underline{x})|.$$

Proof. It is not possible to refine subsets $p \in P_{\mathrm{F}}(\underline{x})$ into subsets $r \in R_{\mathrm{I}}(\underline{x})$ using only split operations. Hence, for any refinement R of partition P we have $|R_{\mathrm{F}}(\underline{x})| \geq |P_{\mathrm{F}}(\underline{x})|$. By assumption, $\underline{x}$ is infeasible for Model (5) with R, we have

$$|R| - \lfloor \tau |S| \rfloor - 1 \geq \sum_{p \in R} z_p(\underline{x}) = |P_{\mathrm{F}}(\underline{x})| = |P| - |P_{\mathrm{I}}(\underline{x})|. \tag{15}$$

Algorithm 2: Minimal Size Refinement.

Input : scenario set S, partition P of S, point $\underline{x} \in \mathcal{X}$.
Output : refinement R of P, where $\underline{x}$ is not feasible for Model (5) with R.
Initialize : $R \leftarrow P$ and $\mu \leftarrow \lfloor \tau |S| \rfloor + 1 - |P_{\mathrm{I}}(\underline{x})|$.

```
while |R| < |P| + μ do
    Select r_1 ∈ R_I(x) such that |r_1 ∩ S_I(x)| ≥ 2.
    Select two infeasible scenarios s_1, s_2 ∈ r_1 ∩ S_I(x).
    Set r_2 ← {s_1} and r_3 ← {s_2}.
    Allocate all remaining scenarios s ∈ r_1 \ {s_1, s_2} to r_2 and r_3.
    Set R ← R \ {r_1} ∪ {r_2, r_3}.
end
return R
```

Finally, by reorganizing terms in Equation (15) we get

$$|R| - |P| \geq \lfloor \tau |S| \rfloor + 1 - |P_{\mathrm{I}}(\underline{x})|.$$

□

Propositions 6 shows that by applying Algorithm 2 we create the smallest size refinement R of P such that $\underline{x}$ is infeasible for Model (5) on R. The following theorem is the main result used in Algorithm 1 for obtaining refinements R such that $v^{\mathrm{L}}(R) > v^{\mathrm{L}}(P)$.

Theorem 1. *Let the point $\underline{x}$ be an optimal solution of Model (5) with P. If $\underline{x}$ is unique and infeasible for Model (3), then the refinement R of P created by applying Algorithm 2 is the smallest size refinement of P such that*

$$v^{L}(R) > v^{L}(P).$$

Proof. Proposition 8 proves that any refinement R of P is such that Model (5) with P is a relaxation of Model (5) with R. Then, if R is created by applying Algorithm 2 we know by Proposition 5 that $\underline{x}$ is not feasible for Model (5) with R. Moreover, if $\underline{x}$ is the unique point such that solving Model (5) with P returns the value $v^{\mathrm{L}}(P)$, it follows that

$$v^{\mathrm{L}}(R) > v^{\mathrm{L}}(P).$$

In addition, Proposition 6 proves that R is of minimal size. □

4. On Constructing Minimal Size Mergers

Adaptive partitioning methods based uniquely on refinement operations strictly increase the size of the partition P with the iterations. As a consequence, the number of indicator variables z and the time required to solve Model (4) increase over time. We suggest adding merging operations to address this issue.

Definition 3. *A partition M of a set S is a merger of a partition P if for any $p \in P$ there exists $m \in M$ such that $p \subseteq m$.*

In general, it is not possible to perform a merge without increasing the lower bound obtained by solving Model (5). This is stated in the following corollary.

Corollary 1. *For any merger M of a partition P, we have $v^{L}(M) \leq v^{L}(P)$.*

Proof. The proof is a direct consequence of Definition 3. If M is a merger of P then P is a refinement of M and Proposition 4 holds. □

Corollary 1 suggests that a merging operation should not be performed in Algorithm 1 if we request a strict increasing lower-bound. However, in what follows, we show that an a-posteriori merge may be carried out.

4.1. **Solution Exclusion by Merging.** We now study how to perform efficient merging operations. We prove that, if certain conditions are satisfied, given a refinement R of a partition P, we can construct a merger M of R such that $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$. To carry out this proof, we proceed in the following way. First, in Propositions 7 and 8 we identify conditions that guarantee $v^{\mathrm{L}}(M) \geq v^{\mathrm{L}}(P)$. Second, in Corollary 2, we show that, if additional assumptions are made on M, then $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$. Third, in Proposition 9, we propose a procedure for creating a merger that satisfies the aforementioned conditions. Finally, in Theorem 2, we use the previous results to show how a merger M satisfying $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$ is obtained in Algorithm 1.

To simplify the presentation, we use the following notation throughout the remainder of this section. Let P be a partition of the scenario set S and let R be a refinement of P. Let M be a merger of R. Let N^A be the subsets of an arbitrary partition A that are not in P, i.e., $N^A = A \setminus P$. Further, let ρ_p denote the minimum cost of the solution that satisfies the constraint of a subset $p \in P$, i.e.,

$$\rho_p = \min_x \{f(x) : x \in X^p \cap \mathcal{X}\}. \tag{16}$$

Given any point $x' \in \mathcal{X}$, we define $z_p(x') = \mathbb{1}(x' \in X^p)$ the value of the indicator variables associated with a subset $p \in P$, as well as $P_{\mathrm{F}}(x) := \{p \in P : \ x \in X^p\}$ the set of subsets that are feasible for x and $P_{\mathrm{I}}(x) := \{p \in P : \ x \notin X^p\}$ the set of subsets of P that are infeasible for x. Naturally, $P = P_{\mathrm{F}}(x) \cup P_{\mathrm{I}}(x)$.

We now identify conditions that ensure $v^{\mathrm{L}}(M) \geq v^{\mathrm{L}}(P)$.

Proposition 7. *If M is such that $|M| \geq |P|$, then*

$$v^L(M) \geq \min\left\{v^L(P), \min_{m \in N^M} \rho_m\right\}.$$

Proof. Let $\underline{x}^M$ be the optimal solution of Model (5) with M. We distinguish two cases.
Case 1. If $z_{m'}(\underline{x}^M) = 0$ for all $m' \in N^M$ we know that $\underline{x}^M$ is also feasible for Model (5) with partition P and thus $v^{\mathrm{L}}(M) \geq v^{\mathrm{L}}(P)$.
Case 2. If there exists a $m' \in N^M$ such that $z_{m'}(\underline{x}^M) = 1$ it holds that $v^{\mathrm{L}}(M) \geq \rho_m$. □

Furthermore, we present a simpler method to evaluate situations where a merge operation guarantees that the lower bound is non-decreasing.

Proposition 8. *If M is such that $|M| \geq |P|$, then*

$$v^L(M) \geq \min\left\{v^L(P), \min_{r \in N^R} \rho_r\right\}.$$

Proof. Since the partition M is a merger of R,

$$\min_{m \in M} \rho_m \geq \min_{r \in R} \rho_r.$$

It follows from Proposition 7 that

$$v^{\mathrm{L}}(M) \geq \min(v^{\mathrm{L}}(P), \min_{m \in N^M}(\rho_m)) \geq \min(v^{\mathrm{L}}(P), \min_{r \in N^R}(\rho_r)).$$

□

Propositions 7 and 8 describe straightforward conditions on the value that ρ_p may take to guarantee $v^{\mathrm{L}}(M) \geq v^{\mathrm{L}}(P)$. If $\min_{p \in N^R} \rho_p \geq v^{\mathrm{L}}(P)$, Proposition 8 highlights that $v^{\mathrm{L}}(M) \geq v^{\mathrm{L}}(P)$ for any M obtained by merging the refinement R.

Moreover, if $\rho_r \leq v^{\mathrm{L}}(P)$ for at least one $r \in N^R$ it may be possible to construct a new partition M such that $\min_{m \in N^M} \rho_m \geq v^{\mathrm{L}}(P)$.

Corollary 2. *Let $\underline{x}$ be the optimal solution of Model (5) with P. Let M be such that $|M| \geq |P|$. Let $\min_{m \in N^M} \rho_m > v^L(P)$ or $\min_{r \in N^R} \rho_r > v^L(P)$. If $\underline{x}$ is unique and infeasible for Model (5) with M, then*

$$v^L(M) > v^L(P).$$

Proof. Let $\underline{x}^M$ be the optimal solution of Model (5) with M. To prove the statement we consider two cases.
Case 1. Let $z_{m'}(\underline{x}^M) = 0$ for all $m' \in N^M$. We know that $\underline{x}^M$ is also feasible for Model (5) with partition P. However, the point $\underline{x}$ is unique and not feasible for Model (5) with partition M, so $v^{\mathrm{L}}(M)$ cannot be equal to $v^{\mathrm{L}}(P)$, and $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$.
Case 2. Let a subset $m' \in N^M$ exist such that $z_{m'}(\underline{x}^M) = 1$. If $\min_{m \in N^M} \rho_m > v^{\mathrm{L}}(P)$ then $v^{\mathrm{L}}(M) \geq \rho_{m'} \geq \min_{m \in N^M} \rho_m > v^{\mathrm{L}}(P)$. Otherwise, if $\min_{r \in N^R} \rho_r > v^{\mathrm{L}}(P)$, then $v^{\mathrm{L}}(M) \geq \rho_{m'} \geq \min_{m \in N^M} \rho_m \geq \min_{r \in N^R} \rho_r > v^{\mathrm{L}}(P)$. □

Corollary 2 states a set of conditions that allows to identify when a merge operation leads to a strictly increasing lower bound. Notice that Model (6) does not need to be solved, instead it is sufficient to compute the value of $\min_{m \in N^M} \rho_m$ and $\min_{r \in N^R} \rho_r$.

We now show how to design a merger M such that $\underline{x}$ is not feasible for Model (5) with M. Moreover, we want M to be of minimal size, i.e., $|M| = |P|$. The following proposition describes a valid way to create such a partition.

Proposition 9. *Let the point $\underline{x}$ be feasible for Model (5) with P but infeasible for Model (3) and let R be a refinement of the partition P such that $|R| = |P| + \mu$, with $\mu = \lfloor \tau |S| \rfloor + 1 - |P_I(\underline{x})|$. Then, a merger M of the partition R satisfying $|M| = |P|$ exists such that the point $\underline{x}$ is not feasible for Model (5) with M.*

Proof. First, we show that if M satisfies the following two properties,

$$|M| = |P|, \tag{17}$$

$$|M_{\mathrm{F}}(\underline{x})| = |P_{\mathrm{F}}(\underline{x})| - \mu, \tag{18}$$

then $\underline{x}$ is not feasible for Model (5) with M. By combining Equations (17) and (18) we have

$$\begin{aligned} |M_{\mathrm{F}}(\underline{x})| &= |P_{\mathrm{F}}(\underline{x})| - \lfloor \tau |S| \rfloor - 1 + |P_{\mathrm{I}}(\underline{x})|, \\ &= |P| - \lfloor \tau |S| \rfloor - 1, \\ &= |M| - \lfloor \tau |S| \rfloor - 1, \end{aligned}$$

which, by Proposition 3, implies that $\underline{x}$ is not feasible for Model (5) with M.

Second, we describe how to construct a merger M of R that satisfies Equations (17) and (18). Observe that it is not possible to refine subsets $p \in P_{\mathrm{F}}(\underline{x})$ into subsets $r \in R_{\mathrm{I}}(\underline{x})$ using split operations. Hence, for any refinement R of partition P the inequality $|R_{\mathrm{F}}(\underline{x})| \geq |P_{\mathrm{F}}(\underline{x})|$ holds. We distinguish two cases.

Case 1. If $|P| = \lfloor \tau |S| \rfloor + 1$, then $\mu = |P| - |P_{\mathrm{I}}(\underline{x})| = |P_{\mathrm{F}}(\underline{x})| \leq |R_{\mathrm{F}}(\underline{x})|$. A valid merger M is obtained by merging μ subsets in $R_{\mathrm{F}}(\underline{x})$ together with exactly one subset in $R_{\mathrm{I}}(\underline{x})$. This is always possible because $R_{\mathrm{F}}(\underline{x})$ contains at least μ elements and $R_{\mathrm{I}}(\underline{x})$ is not empty since $\underline{x}$ would otherwise necessarily be feasible for Model (3). This yields a merger M of size $|P|$ satisfying Equation (17). Moreover, exactly μ sets $r \in R_{\mathrm{F}}(\underline{x})$ are merged with a set $r' \in R_{\mathrm{I}}(\underline{x})$ such that Equation (18) is satisfied.

Case 2. If $|P| \geq \lfloor \tau |S| \rfloor + 2$, then $\mu \leq |P| - |P_{\mathrm{I}}(\underline{x})| - 1 = |P_{\mathrm{F}}(\underline{x})| - 1 \leq |R_{\mathrm{F}}(\underline{x})| - 1$. We create the merger M by merging $\mu + 1$ sets inside $|R_{\mathrm{F}}(\underline{x})|$. This yields a merger M

of size $|P|$ satisfying Equation (17). Moreover, $\mu + 1$ sets are merged to form one subset $m \in M_{\mathrm{F}}(\underline{x})$ and Equation (17) is satisfied. □

Algorithm 3: Maximal Size Merger.

Input : scenario set S; partition P of S; refinement R of P; point $\underline{x} \in \mathcal{X}$.
Output : merger M of R, where $\underline{x}$ is not feasible for Model (5) with M.
Initialize : $M \leftarrow R$ and $\mu \leftarrow \lfloor \tau|S| \rfloor + 1 - |P_{\mathrm{I}}(\underline{x})|$.

```
if |P| = ⌊τ|S|⌋ + 1 then
    Select all m_1, ..., m_μ ∈ M_F(x) and m_{μ+1} ∈ M_I(x).
else if |P| ≥ ⌊τ|S|⌋ + 2 then
    Select m_1, ..., m_{μ+1} ∈ M_F(x).
end
Set m_{μ+2} ← ⋃_{i∈[μ+1]} m_i and M ← M \ {m_1, ..., m_{μ+1}} ∪ {m_{μ+2}}.
return M
```

Proposition 9 gives a simple procedure for creating a maximum-size merger that excludes the point $\underline{x}$. Algorithm 3 summarizes how to carry out this procedure. The following theorem is the main result used in Algorithm 1 for obtaining mergers M such that $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$.

Theorem 2. *Let $\underline{x}$ be the optimal solution of Model (5) with P. Let R be a refinement of P constructed by applying Algorithm 2 such that $\min_{R \in N^R} \rho_R > v^L(P)$. If $\underline{x}$ is unique and infeasible for Model (3), then the merger M of R obtained by applying Algorithm 3 is such that $|M| = |P|$ and*

$$v^L(M) > v^L(P).$$

Proof. The partition R is constructed by applying Algorithm 2. This means that $|R| = |P| + \mu$, with $\mu = \lfloor \tau|S| \rfloor + 1 - |P_{\mathrm{I}}(\underline{x})|$. By applying Algorithm 3, we construct a merger M as described in the proof of Proposition 9. Consequently, the point $\underline{x}$ is not feasible for Model (5) with M and $|M| = |P|$. Since $\underline{x}$ is not feasible for Model (5) with M and $\min_{r \in N^r} \rho_r > v^{\mathrm{L}}(P)$ or $\min_{m \in N^M} \rho_m > v^{\mathrm{L}}(P)$ we know by Corollary 2 that $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$. □

5. Strong Partitions and Practical Strategies

This section covers the remaining components that complement the presentation of the general APM summarized in Algorithm 1. First, we discuss how to build partitions from scratch such that solving Model (5) yields tight lower bounds on v^*. Second, we propose a method for refining selected subsets by maximizing the value of $\min_{r \in N^R} \rho_r$. Third, we explain how we select scenarios to be refined and merged to take advantage of the proposed refinement method. Finally, a heuristic for projecting an infeasible solution to the set of feasible solutions of the original model (3) is proposed.

5.1. Partitions for Tight Lower Bounds.

In what follows, we use the minimum subset cost coefficients ρ_p introduced in Equation (16). Moreover, we use the coefficients ρ_s, which are defined exactly as in Equation (16) but the set X^p is replaced by the set X^s.

We describe a procedure for building a partition P of size $\lfloor \tau|S| \rfloor + 1$ such that $v^{\mathrm{L}}(P) \geq v_{\mathrm{Q}}$, where v_{Q} is the well-known quantile bound [37]. Let ϕ be a permutation of S that satisfies $\rho_{\phi_1} \geq \cdots \geq \rho_{\phi_{|S|}}$. In [2] the quantile bound is defined as $v_{\mathrm{Q}} = \rho_{\phi_q}$ where $q = \min\{k \in \{1, \ldots, |S|\} : \sum_{l=1}^{k} q_{\phi_l} > \tau\}$.

Proposition 10 (from [2]). *The quantile bound v_Q is a lower bound on the optimal objective of the CCSP (3), i.e., $v_Q \leq v^*$.*

We now discuss how to obtain a partition P such that $v^{\mathrm{L}}(P) \geq v_{\mathrm{Q}}$. We use the permutation ϕ introduced earlier. First, we fix the size of P to $\lfloor \tau|S| \rfloor + 1$. Second, we assign the scenarios ϕ_i to the set $p := i \mod (|P|)$ for all $i \in \{1, \ldots, |S|\}$. In other words, scenario ϕ_1 is assigned to set p_1, scenario ϕ_2 is assigned to set p_2, and so forth, until the set $p_{|P|}$ is reached. This assignment is continued at p_1 until all scenarios are assigned to a set.

Proposition 11. *If P is created by a sequential assignment of the ordering given by the permutation ϕ into $\lfloor \tau|S| \rfloor + 1$ disjoint sets, then*

$$v^L(P) \geq v_Q.$$

Proof. From Proposition 3 it follows that

$$\sum_{p \in P} z_p \geq |P| - \lfloor \tau|S| \rfloor = 1.$$

Since we sequentially dispatch the scenarios based on ρ_s, each set $p \in P$ will contain at least one scenario ϕ_l with $l \in \{1, \ldots, \lfloor \tau|S| \rfloor + 1\}$. When Assumption 1 holds, we know that $q = \lfloor \tau|S| \rfloor + 1$ and it follows that

$$v^{\mathrm{L}}(P) \geq \min_{l \in \{1, \ldots, \lfloor \tau|S| \rfloor + 1\}} \rho_{\phi_l} = \rho_{\phi_q} = v_{\mathrm{Q}}.$$

□

Proposition 11 implies that if P is obtained by a sequential assignment of the permutation ϕ into $\lfloor \tau|S| \rfloor + 1$ disjoint sets, then $v^{\mathrm{L}}(P) \geq v_{\mathrm{Q}}$. This result, when combined with Theorems 1 and 2, ensures a strict increase of a tight lower bound produced by Algorithm 1, while keeping the size of the considered partitions as small as possible.

5.2. **Refinements that Promote Merging.** We now propose a model that, when solved to optimality, splits the subset r_1 into the subsets r_2 and r_3 as described in Algorithm 2. We propose to split r_1 in a way that maximizes the value of $\min_{r \in \{r_2, r_3\}} \rho_r$. Two observations support this approach. First, as described in Proposition 8, when the subsets r_2 and r_3 are such that $\min_{r \in \{r_2, r_3\}} \rho_r > v(P)$ we know that a merger M of size $|P|$ can be constructed so that $v^{\mathrm{L}}(M) > v^{\mathrm{L}}(P)$. Second, it is preferable that the value of $\min_{r \in \{r_2, r_3\}} \rho_r$ is as large as possible because the feasible regions X^{r_2} and X^{r_3} may be selected in the optimal solution of Model (5) with R. Indeed, if z_{r_2} or z_{r_3} are equal to one for the optimal solution of Model (5) with R, then $v(R) > \min(\rho_{r_2}, \rho_{r_3})$.

A naive way to maximize $\min_{r \in \{r_2, r_3\}} \rho_r$ is computing the value of $\min_{r \in \{r_2, r_3\}} \rho_r$ for every possible split of r_1. The number of models that need to be solved is given by the Stirling numbers of the second kind [15]. Hence, the complexity of this naive approach grows exponentially as the number of scenarios inside the subset r_1 increases.

We propose an optimization-based approach for chance-constrained linear problems that can be used when the decision variables are continuous or binary. Our approach is based on formulating the splitting problem as a bi-level optimization problem [11]. The upper-level yields an assignment of scenarios to the subsets r_2 and r_3 while maximizing the value of $\min_{r \in \{r_2, r_3\}} \rho_r$. The lower-level computes the subset costs ρ_{r_2} and ρ_{r_3} given a scenario assignment. For every $s \in r_1$ we introduce

two binary assignment variables π_{sr_2} and π_{sr_3} to track whether this scenario is assigned to subset r_2 or r_3. The upper-level problem is given by

$$\rho_{\text{div}} = \max_{\pi} \quad \min_{r\in\{r_2,r_3\}} \rho_r(\pi) \tag{19a}$$

$$\text{s.t.} \quad \sum_{r\in\{r_3,r_3\}} \pi_{sr} = 1, \quad s \in p \tag{19b}$$

$$\sum_{s\in r_1\cap S_{\text{I}}(\underline{x})} \pi_{sr} \geq 1, \quad r \in \{r_2, r_3\}, \tag{19c}$$

$$\pi \in \{0,1\}. \tag{19d}$$

Constraint (19b) states that every scenario is assigned to a subset. Constraint (19c) ensures that at least one $s \in S_{\text{I}}(\underline{x})$ is assigned to each subset.

As discussed earlier, the value of $\rho_r(\pi)$ for $r \in \{r_1, r_2\}$ is the solution of the lower level problem,

$$\rho_r(\pi) = \min_{x} \quad c^\top x \tag{20a}$$

$$\text{s.t.} \quad A^{\mathcal{X}} x \geq b^{\mathcal{X}}, \tag{20b}$$

$$\pi_{sr} A^s x \geq \pi_{sr} b^s, \quad s \in r_1 \tag{20c}$$

Model (20) computes the single subset cost ρ_r for a set of assignment variables π. Constraint (20b) is the equivalent of $x \in \mathcal{X}$ when we assume that Model (4) is linear. Constraint (20c) is the equivalent of $x \in X^s$ when we assume that Model (4) is linear.

Proposition 12. *Model (19) can be reformulated as a single-level problem of the form*

$$\rho_{div} = \max_{\pi,\eta} \quad \alpha \tag{21a}$$

$$s.t. \quad \alpha \leq (b^{\mathcal{X}})^\top \eta^{\mathcal{X}r} + \sum_{s\in r_1} (b^s)^\top \eta^{sr}, \quad r \in \{r_2, r_3\}, \tag{21b}$$

$$(A^{\mathcal{X}})^\top \eta^{\mathcal{X}r} + \sum_{s\in r_1} (A^s)^\top \eta^{sr} = c, \quad r \in \{r_2, r_3\}, \tag{21c}$$

$$1 - \pi_{sr} = \mathbb{1}(\eta_i^{sr} = 0), \quad s \in r_1, i \in I(s), r \in \{r_2, r_3\} \tag{21d}$$

$$\sum_{r\in\{r_2,r_3\}} \pi_{sr} = 1, \quad s \in r_1 \tag{21e}$$

$$\sum_{s\in r_1\cap S_I(\underline{x})} \pi_{sr} \geq 1, \quad r \in \{r_2, r_3\}. \tag{21f}$$

$$\pi \in \{0,1\}, \tag{21g}$$

$$\eta \geq 0. \tag{21h}$$

Proof. Since Model (20) is linear, strong duality holds. Let η be the dual variables of Model (20). The dual of Model (20) reads

$$\rho_r(\pi) = \max_{x} \quad (b^{\mathcal{X}})^\top \eta^{\mathcal{X}} + \sum_{s\in p} \pi_{sr} (b^s)^\top \eta^s \tag{22a}$$

$$\text{s.t.} \quad (A^{\mathcal{X}})^\top \eta^{\mathcal{X}} + \sum_{s\in p} \pi_{sr} (A^s)^\top \eta^s = c, \tag{22b}$$

$$\eta \geq 0. \tag{22c}$$

Model (22) is equivalent to

$$\rho_r(\pi) = \max_{x} \quad (b^{\mathcal{X}})^\top \eta^{\mathcal{X}} + \sum_{s \in p} (b^s)^\top \eta^s \tag{23a}$$

$$\text{s.t.} \quad (A^{\mathcal{X}})^\top \eta^{\mathcal{X}} + \sum_{s \in p} (A^s)^\top \eta^s = c, \tag{23b}$$

$$1 - \pi_{sp} = \mathbb{1}(\eta_i^s = 0), \quad s \in p, i \in I(s), \tag{23c}$$

$$\eta \geq 0, \tag{23d}$$

where $I(s)$ is the set of constraint indices for scenario $s \in S$.

Further, we introduce the variable $\alpha = \min_{r \in \{r_2, r_3\}} \rho_r(\pi)$. By construction, $\alpha \leq \rho_r(\pi)$ holds for all $r \in \{r_2, r_3\}$. Hence, Model (19) is equivalent to

$$\rho_{\text{best}} = \max_{\pi} \quad \alpha \tag{24a}$$

$$\text{s.t.} \quad \alpha \leq \rho_r(\pi), \quad r \in \{r_2, r_3\}, \tag{24b}$$

$$\sum_{r \in \{r_1, r_2\}} \pi_{sr} = 1, \quad s \in p \tag{24c}$$

$$\sum_{s \in r_1 \cap S_{\text{I}}(\underline{x})} \pi_{sr} \geq 1, \quad r \in \{r_2, r_3\}, \tag{24d}$$

$$\pi \in \{0, 1\} \tag{24e}$$

Finally, we replace $\rho_r(\pi)$ in Model (24) by Model (23), which yields Model (21). □

Remark 4. *The single-level reformulation of Model (21) can be generalized to the case where a subset is split into more than two subsets. We do not cover this case since splitting a subset into exactly two subsets always leads the highest possible value for ρ_{div}.*

Remark 5. *Model (21) can be used on the LP relaxation of a chance-constrained linear problem with binary variables. Indeed, if $\min_{r \in \{r_2, r_3\}} \rho_r > v(P)$ holds for the LP relaxation it necessarily also holds for the original problem with binary variables.*

5.3. **Subset Selection for Refinement and Merging.** We now explain how to select the subsets used in refinement and merging operations at each iteration of our adaptive method. Our methods aim to maximize the number of merging operations that are carried out.

We know from Proposition 8 that any refinement such that $\min_{r \in \{r_2, r_3\}} \rho_r > v(P)$ allows to perform a subsequent merge operation. Hence, when possible, we always select a subset $r_1 \in R_{\text{I}}(\underline{x})$ that satisfies $\rho_{\text{div}} > v^{\text{L}}(P)$ where ρ_{div} is obtained by solving Model (21). When several subsets satisfy the condition $\rho_{\text{div}} > v^{\text{L}}(P)$, we select the one with the smallest ρ_{div} value. When no subset satisfies the condition, i.e., all subsets are such that $\rho_{\text{div}} \leq v^{\text{L}}(P)$, we select $r_1 = \text{argmax}_{r \in R_{\text{I}}(\underline{x})} \rho_{\text{div}}$.

When merging, we select the feasible subsets $m \in M_{\text{F}}(\underline{x})$ with the largest single subset costs. When an infeasible subset is necessary to construct the merger (see the second case in Algorithm 3), we select the subset with the largest value for ρ_s. This strategy ensures that $v^{\text{L}}(M)$ is large when the newly created subset $m_{\mu+2}$ is selected in the next iteration.

5.4. **Recovering Feasible Solutions.** We propose a simple projection heuristic to recover valid primal solutions when the optimal solution $\underline{x}$ of Model (5) is not feasible for Model (3). We construct this feasible point $\bar{x}$ by selecting additional scenarios to be satisfied until Constraint (3c) is valid. We introduce the set E which contains all the scenarios that are satisfied for obtaining the point $\bar{x}$. Initially, this

set E is composed of every $s \in S_{\mathrm{F}}(\underline{x})$. Then, scenarios $s \in S_{\mathrm{I}}(\underline{x})$ are greedily added to E based on their feasibility w.r.t. $\underline{x}$. That is, we iteratively add the scenario with the smallest value of $\max_{i \in I(s)} G_i^s(\underline{x})$ to E until $|E| = |S| - \lfloor \tau |S| \rfloor$. Finally, we obtain $\bar{x}$ by solving the following model

$$\begin{aligned} v_{\mathrm{proj}} = \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{s.t.} \quad & G^s(x) \leq 0, \quad s \in E. \end{aligned}$$

Further, each time the value of ρ is computed for a scenario or a subset we check if the associated solution is feasible for Model (4) and improves the current best upper bound.

6. Numerical Study

This section presents the numerical results of the APM when compared to state-of-the-art methods. To assess the value of the APM and its components, we run repeated experiments on CCSP instances taken from the literature. Our numerical study investigates the performance of our method by measuring the time taken to solve an instance to optimality, or the optimality gap when the instance cannot be solved to optimality in the allocated time. To understand the strengths and weaknesses of our method, we also study a single run of the APM in detail. This allows us to analyze the occurrence and the effectiveness of the strategies proposed in Section 5. For instance, we show how the lower and upper bounds evolve and how the size of the partition evolves as the number of iterations increases.

All computations have been executed on a remote server. Each experiment is run on a single core of an Intel Gold 6148 Skylake with 2.4 GHz and is allocated 16 GB RAM. A time limit of 120 min is enforced. Our implementation is made with the programming language Python. All optimization models are solved using Gurobi 10.0.3. The code used to produce all numerical results is publicly available at the online repository: https://github.com/alexforel/AdaptiveCC.

6.1. Experimental Setting.

We follow the experimental setting of recent works on CCSPs in which multi-dimensional knapsack problems with either binary or continuous variables are used. All instances are generated according to the method described in [35] and [2]. Each chance-constrained instance is created by sampling a set of scenarios from a deterministic instance. A scenario is obtained by perturbing the left-hand side constraint matrix of the original deterministic instance. We consider three base instances: mk-10-10, mk-20-10, and mk-40-30, which have 10, 10, and 30 constraints per scenario, respectively, as well as 10, 20, and 40 decision variables, respectively. We generate five perturbed instances per deterministic instance.

6.1.1. *Implementation and Benchmarks.*

The final adaptive method, referred to as $\mathrm{P}_{\mathrm{final}}$, is implemented following the description of the APM displayed in Algorithm 1. Moreover, we implemented all the strategies discussed in Section 5. Obtaining big-M coefficients using Model (8) in each iteration of the APM is too time-consuming. Therefore, we use the less computationally heavy approach of [5] each time a new model is solved. In preliminary experiments, we observed that a large amount of big-M constraints (5c) are unnecessary for solving a reduced model (5). This is explained by the fact that each individual scenario is obtained by perturbing a deterministic set of constraints. We observe that, when scenarios are inside the same subset of a partition, a constraint of a scenario is often dominated by a constraint of another scenario. These dominated constraints are thus not necessary for representing the feasible set of the reduced model (5). As a consequence, the

computational performance of the APM improves when Gurobi's `Lazy` parameter is set to 1 for big-M constraints (5c).

We compare the APM with three benchmarks. The first two benchmarks use the big-M formulation of the original CCSP as given in Model (4). These two methods differ in the way the big-M parameters are computed. The first benchmark, referred to as "`Song Big-M`", follows the method proposed in [35] and also evaluated in [2] [2]. This method is tailored to chance-constrained packing problems and therefore to the multi-dimensional chance-constrained knapsack problems that we consider. It is based on solving a series of single-dimensional continuous knapsack problems from which extremely tight upper bounds on the big-M coefficients can be computed using a quantile argument. The second benchmark, referred to as "`Belotti Big-M`", is based on the problem-agnostic method of [5]. To obtain the big-M coefficients, a single-dimensional knapsack problem is solved for each scenario/constraint combination using a valid lower bound on the original CCSP. For both benchmarks, we do not see a distinct improvement when Gurobi's `Lazy` parameter is set to 1 for big-M constraints (4b), and therefore leave it to its default value.

The third benchmark is a naive partitioning method, referred to as $\mathrm{P}_{\mathrm{naive}}$. The method $\mathrm{P}_{\mathrm{naive}}$ is based on restricting Algorithm 1 to refinement operations. Each subset of the partition to be refined is chosen at random. The dispatch of scenarios to child subsets is also chosen at random. Nevertheless, the outline of Algorithm 2, i.e., how many scenarios are selected for refinement and which scenarios are considered for refinement, is respected. The initial partition of $\mathrm{P}_{\mathrm{naive}}$ is obtained by a random dispatch of scenarios to the subsets. To allow for a fair comparison between the results of different methods we include the time needed to compute the big-M coefficients in the total computation time.

6.1.2. *Big-M Computation Time.* Obtaining tight big-M coefficients can require a large amount of time. In particular, the method outlined in [35] requires solving $|S|^2|I|^2$ single-dimensional continuous knapsack problems and thus scales quadratically with the number of constraints and scenarios. Our implementation of [35] is coded in C++ and interfaced with our code using Cython [4]. Further, it exploits the symmetry of the problem to avoid performing unnecessary sorting operations. The time needed to obtain the big-M coefficients for our two benchmarks is given in Table 1.

As expected, the general method of [5] is very fast. In contrast, the method of [35], which is tailored for multi-dimensional knapsack problems, does not scale well with the problem size. It cannot terminate within the time limit for large problems. More precisely, big-M coefficients cannot be computed for the instance mk-40-30 when there are more than $|S| = 3000$ scenarios. This can be anticipated from the time needed to compute big-M coefficients when $|S| = 1000$ since increasing the scenario number threefold increases the computation times by a factor of nine. Still, we want to emphasize that our implementation of [35] is particularly efficient: the times needed to compute the big-M coefficients are approximately three times smaller than those presented in [2].

6.2. **Optimal Solutions and Bounds.** The main experimental results are presented in Table 2 and Table 3 for instances with continuous and binary variables, respectively. The tables are produced using the method described in [2]. Each row presents the performance metric of all methods averaged over the five perturbed instances. If all instances are solved, we show in the column called $\mathrm{T}_{\mathrm{avg}}$, the average time needed to solve the considered instance to optimality. If at least one instance cannot be solved, we show the average optimality gap over the non-solved instances

TABLE 1. Computation time needed to obtain all big-M parameters.

		Song Big-M	Belotti Big-M	
			Continuous	Binary
mk-10-10	500	3.96s	2.42s	7.45s
	1000	15.81s	5.37s	13.97s
	3000	140.77s	14.35s	41.59s
	5000	381.66s	24.29s	67.13s
mk-20-10	500	11.94s	4.50s	15.87s
	1000	46.86s	8.87s	52.52s
	3000	374.02s	25.79s	91.37s
	5000	1291.11s	43.58s	237.90s
mk-40-30	500	221.48s	21.85s	111.83s
	1000	861.83s	44.63s	205.68s
	3000	-	134.08s	626.63s
	5000	-	223.13s	1099.25s

and the number of solved instances in parentheses. In each row of the table, the best-performing method is shown in bold. We also show the average number of iterations, denoted as It_{avg}, for methods P_{final} and P_{naive}.

As can be observed in Table 2, the method P_{final} performs the best among the pool of compared methods when continuous variables are considered. More specifically, P_{final} is the best-performing method for 20 out of the 24 considered instances. For the remaining 4 instances the method `Song Big-M` performs best. Thanks to the extremely tight big-M coefficients that are produced with this method, the time needed to solve Model (4) and the final optimality gaps are always smaller compared to the `Belotti Big-M` method.

The results in Table 1 suggest that there is no dominating algorithm when binary variables are considered. In particular, the method `Song Big-M` makes it possible to solve most of the instances that have a large number of scenarios to optimality when binary variables are considered. Nevertheless, the method `Song Big-M` does not produce valid results when the time limit is reached for instance mk-40-30 with $|S| \geq 3000$. As explained earlier, due to the large amount of constraints and scenarios, too much time is consumed for computing the big-M coefficients. For these same instances the method P_{final} significantly reduces the optimality gap compared to the `Belotti Big-M` benchmark. We also observe that, for instance mk-10-10, only a few iterations are needed to solve the problem to optimality with method P_{final}. This demonstrates the strength of the partition presented in Section 5.1. In fact, the optimal solution of the original CCSP is, in some cases, found directly using the first partition.

In general, if we compare method P_{final} with method P_{naive}, we observe that the components presented in Section 5 improve the computational performance of the APM. Indeed, although P_{naive} is guaranteed to terminate finitely, this method is less competitive than the other methods. Still, it is interesting to note that this naive method yields lower optimality gaps than the `Belotti Big-M` benchmark when the number of scenarios is large.

Finally, we draw attention to the fact that when the number of scenarios is large, P_{final} consistently performs better than other methods for both continuous and binary variables. This is the strength of the APM. By design, it reduces the

TABLE 2. Computational comparison of a selection of methods for multi-dimensional knapsack problems with continuous variables.

			MILP_M		APM			
			`Song Big-M`	`Belotti Big-M`	P_naive		P_final	
Instance	τ	$\lvert S\rvert$	T_avg	T_avg	T_avg	It_avg	T_avg	It_avg
mk-10-10	0.1	500	**24.75s**	631.91s	1071.62s	43.0	113.94s	33.0
		1000	0.01%(4)	0.65%(0)	0.56%(0)	48.2	**1710.02s**	65.6
		3000	0.86%(0)	2.88%(0)	2.35%(0)	29.4	**0.59%(0)**	176.8
		5000	1.19%(0)	4.07%(0)	3.65%(0)	24.6	**0.94%(0)**	287.4
	0.2	500	1901.41s	0.15%(4)	0.19%(1)	63.4	**1497.49s**	58.8
		1000	0.52%(0)	1.55%(0)	2.13%(0)	39.0	**0.33%(0)**	89.0
		3000	1.34%(0)	4.77%(0)	4.19%(0)	33.0	**1.01%(0)**	202.8
		5000	2.08%(0)	6.73%(0)	8.11%(0)	32.2	**1.67%(0)**	313.0
mk-20-10	0.1	500	**0.02%(4)**	0.16%(4)	0.47%(0)	35.2	0.03%(4)	55.8
		1000	0.59%(0)	1.41%(0)	1.56%(0)	22.0	**0.34%(0)**	86.4
		3000	1.50%(0)	4.45%(0)	3.31%(0)	14.4	**1.09%(0)**	195.0
		5000	1.92%(0)	5.77%(0)	3.64%(0)	13.4	**1.39%(0)**	283.0
	0.2	500	**0.33%(0)**	0.89%(0)	1.78%(0)	29.2	0.40%(0)	52.6
		1000	1.20%(0)	2.49%(0)	4.60%(0)	19.2	**0.93%(0)**	83.0
		3000	2.57%(0)	7.75%(0)	6.23%(0)	14.8	**1.83%(0)**	197.2
		5000	3.18%(0)	8.78%(0)	6.41%(0)	17.2	**1.74%(0)**	318.2
mk-40-30	0.1	500	1.42%(0)	5.29%(0)	2.70%(0)	10.8	**1.37%(0)**	76.2
		1000	**2.88%(0)**	13.50%(0)	4.69%(0)	8.0	4.71%(0)	66.2
		3000	-	22.58%(0)	9.21%(0)	3.8	**6.67%(0)**	24.4
		5000	-	23.01%(0)	12.83%(0)	2.4	**8.36%(0)**	13.2
	0.2	500	2.87%(0)	8.90%(0)	6.27%(0)	10.2	**2.19%(0)**	77.4
		1000	5.22%(0)	24.60%(0)	11.09%(0)	6.8	**4.31%(0)**	129.6
		3000	-	41.08%(0)	24.01%(0)	3.4	**8.87%(0)**	143.0
		5000	-	42.53%(0)	31.85%(0)	2.0	**10.71%(0)**	83.4

amount of binary variables that are considered and, as a consequence, scales better with the size of the scenario set.

6.3. **Detailed Analysis.** To study the behavior of P_final and identify the strengths and weaknesses of this method, we provide a detailed analysis on a single instance: mk-20-10 with $|S| = 1000$ scenarios.

6.3.1. *Partition Size.* The number of subsets that compose the partition as a function of the number of iterations is shown in Figure 1. When we apply the method P_final, the partition size stays minimal for many iterations. More specifically, the behavior of P_final is divided into two phases. In the first phase, merge operations are performed at each iteration, which forces the partition to stay as small as possible. In this phase, the reduced-size model is solved very efficiently. Indeed, when the size of the partition is minimal, i.e., it is equal to $\lfloor \tau |S| \rfloor + 1$, the chance constraint (5c) reduces to $\sum_{p \in P} z_p \geq 1$. Hence, the optimal solution of the reduced model is given by the point that corresponds to $\min_{s \in S} \rho_s$.

In the second phase, fewer merging operations are carried out. As a result, the size of the partition increases steadily, and solving the reduced-size model requires more time in each iteration. Overall, more iterations are carried out when continuous variables are considered compared to when binary variables are considered in the

TABLE 3. Computational comparison of a selection of methods for multi-dimensional knapsack problems with binary variables.

			MILP_M		APM			
			`Song Big-M`	`Belotti Big-M`	P_naive		P_final	
Instance	τ	$\lvert S\rvert$	T_avg	T_avg	T_avg	It_avg	T_avg	It_avg
mk-10-10	0.1	500	**11.16s**	21.57s	32.69s	4.0	13.30s	2.4
		1000	**39.13s**	56.46s	101.67s	4.8	71.50s	5.4
		3000	163.17s	280.47s	382.67s	4.6	**120.38s**	3.2
		5000	800.21s	527.89s	1106.47s	8.2	**367.81s**	5.0
	0.2	500	**6.51s**	13.35s	36.56s	5.6	8.13s	1.0
		1000	20.58s	36.88s	83.75s	5.6	**16.36s**	1.2
		3000	154.22s	151.69s	947.74s	8.4	**36.81s**	1.0
		5000	406.18s	461.51s	1058.94s	14.0	**165.75s**	2.2
mk-20-10	0.1	500	**30.28s**	51.48s	442.98s	10.8	134.38s	9.2
		1000	**91.65s**	262.26s	1815.41s	13.4	333.44s	11.8
		3000	**635.70s**	2538.93s	1.70%(0)	11.4	2726.06s	19.0
		5000	**1999.27s**	2.73%(2)	2.99%(0)	8.0	0.27%(2)	16.6
	0.2	500	**34.98s**	81.76s	1339.09s	17.4	250.84s	14.4
		1000	**107.58s**	668.80s	0.37%(3)	21.2	940.53s	18.0
		3000	**1780.72s**	5.18%(0)	4.41%(0)	14.6	0.54%(1)	24.4
		5000	2.74%(0)	6.79%(0)	5.75%(0)	10.6	**1.34%(0)**	18.6
mk-40-30	0.1	500	**351.52s**	2503.72s	1.86%(1)	6.4	0.39%(4)	8.8
		1000	**1295.05s**	11.40%(0)	3.31%(0)	4.8	1.76%(0)	6.8
		3000	-	41.22%(0)	8.82%(0)	2.0	**3.88%(0)**	5.2
		5000	-	47.21%(0)	8.94%(0)	2.0	**5.76%(0)**	4.2
	0.2	500	**400.16s**	18.37%(1)	3.87%(0)	6.0	1.60%(0)	9.0
		1000	**2054.95s**	30.80%(0)	18.17%(0)	2.0	4.03%(0)	10.2
		3000	-	53.35%(0)	21.84%(0)	2.0	**6.14%(0)**	10.4
		5000	-	59.60%(0)	23.74%(0)	2.0	**5.72%(0)**	15.0

first phase of P_final, as could already be observed in Tables 2 and 3. Moreover, we observe that each time a refinement operation is carried out for an instance with continuous variables, exactly one additional subset is created. In contrast, in the binary case, the minimal-size refinement tends to create much more than one additional subset. This can be explained by the fact that, due to the integrality restrictions on the decision variables, the optimal solutions satisfy more sets X^p. It is also interesting to note that the final partition of P_naive is significantly larger than the one produced by P_final. The fact that partitions that necessitate 75 % less binary variables can yield optimal solutions to the original CCSP demonstrates the value of adaptive methods for CCSPs.

6.3.2. *Bound Evolution with Refinement and Merging.* In Figures 2 and 3, we examine how the lower and upper bounds, along with the iteration count, evolve over time. Figure 2 (a) shows how the method P_final achieves a computational advantage over the other methods when continuous variables are considered: it quickly finds a tight lower bound. This is explained by the large number of iterations performed in the first phase of the final APM when the partition size stays minimal. Figure 2 (a) shows that the first phase lasts around 450 s. Once the second phase is reached, the partition size increases, and more time is required to solve each

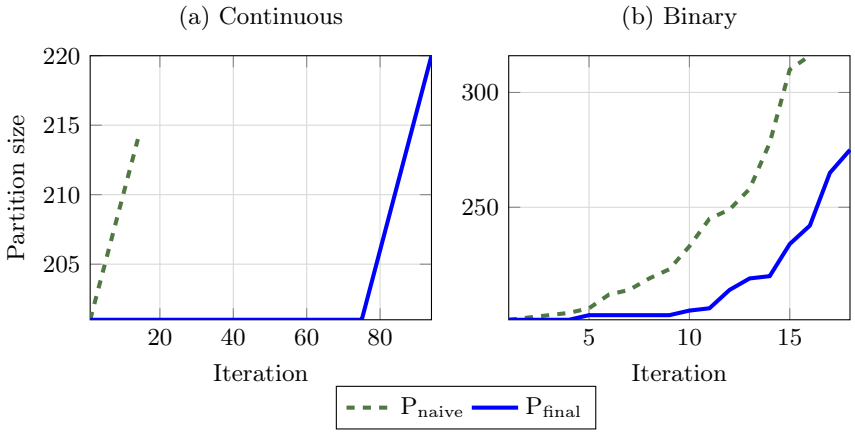

FIGURE 1. Partition size over iterations for mk-20-10 with 1000 scenarios with continuous and binary variables.

optimization problem. This can be observed in Figure 2 (b) as the number of iterations performed over time decreases after 450 s. Figure 3 (a) also highlights the computational advantage of the method `Song Big-M` when binary variables are considered. Thanks to its very tight big-M coefficients, this method quickly produces a feasible solution and can close the optimality gap in a short amount of time. On the other hand, P_{final} stays in the first phase for several iterations, resulting in a longer computation time.

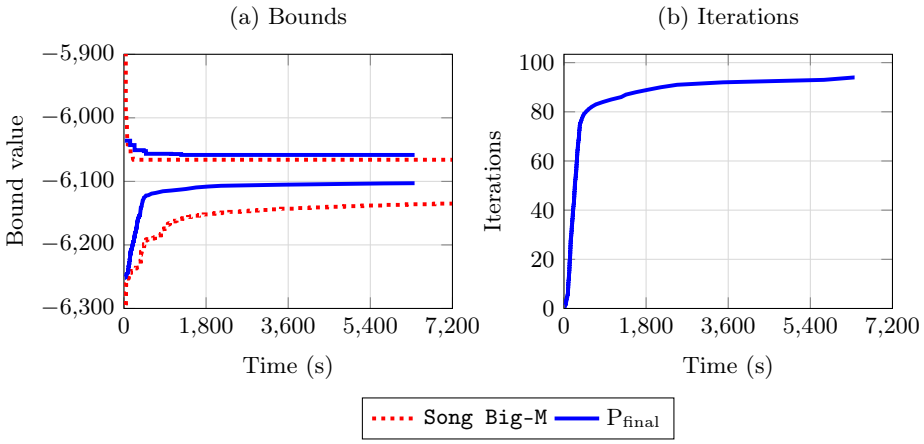

FIGURE 2. Convergence plot and number of iterations performed over time on mk-20-10 with 1000 scenarios and continuous variables.

7. CONCLUSION

This paper introduces a method for solving chance-constrained problems with finite support based on iteratively partitioning the scenario set and solving reduced models. The method provides valid upper bounds on the original stochastic problem and is guaranteed to recover its optimal solution in a finite number of iterations. The key idea of the proposed method is to modify the partition by excluding previously found solutions. We use mathematical arguments to find modifications

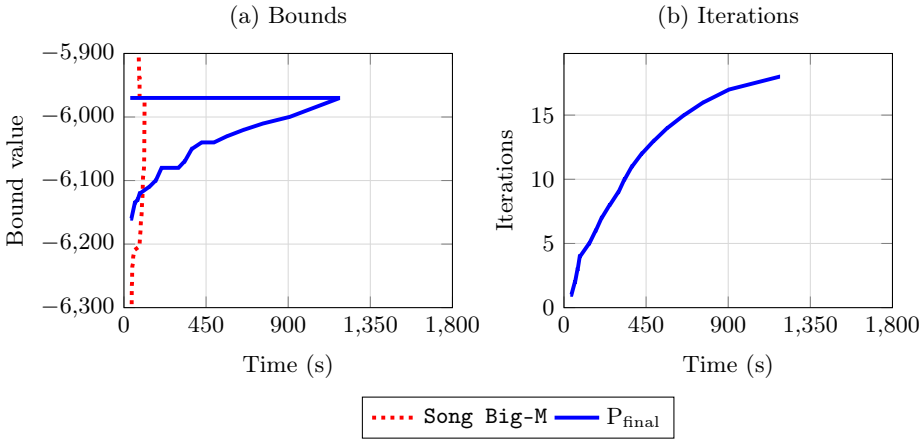

FIGURE 3. Convergence plot and number of iterations performed over time on mk-20-10 with 1000 scenarios and binary variables.

of the partition that ensure a strict increase of the lower bounds while keeping the size of the reduced model as small as possible.

The proposed method surpasses state-of-the-art benchmarks on standard instances with continuous variables and performs comparably on instances with binary variables. Its scalability improves notably when solving instances with a large number of scenarios and constraints.

Future research questions include exploring extensions for non-equiprobable scenarios, adapting the partitioning method for constraint aggregations within subsets, leveraging partitions to enhance techniques that exploit the quantile bound, and determining methods that allow obtaining the smallest partition yielding the optimal solution for the original problem. In a broader context, we consider exact optimization methods based on scenario reduction as a promising and evolving research direction.

ACKNOWLEDGMENTS

The authors want to thank Youssouf Emine for his help with the highly optimized implementation of a benchmark method, as well as Martine Labbé for her helpful comments on an earlier version of this manuscript. The support of IVADO, the Canada First Research Excellence Fund (Apogée/CFREF), as well as the computational infrastructure provided by Compute Canada are also gratefully acknowledged.

REFERENCES

[1] A. Abdi and R. Fukasawa. "On the mixing set with a knapsack constraint." In: *Mathematical Programming* 157 (2016), pp. 191–217. DOI: 10.1137/s10107-016-0979-5.

[2] S. Ahmed, J. Luedtke, Y. Song, and W. Xie. "Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs." In: *Mathematical Programming* 162 (2017), pp. 51–81. DOI: 10.1007/s10107-016-1029-z.

[3] S. Ahmed and A. Shapiro. "Solving Chance-Constrained Stochastic Programs via Sampling and Integer Programming." In: *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*. 2014. Chap. Chapter 12, pp. 261–269. DOI: 10.1287/educ.1080.0048.

[4] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. "Cython: The best of both worlds." In: *Computing in Science & Engineering* 13.2 (2011), pp. 31–39.

[5] P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, and D. Salvagnin. "On handling indicator constraints in mixed integer programming." In: *Computational Optimization and Applications* 65 (2016), pp. 545–566. DOI: 10.1007/s10589-016-9847-8.

[6] D. Bertsimas and N. Mundru. "Optimization-based scenario reduction for data-driven two-stage stochastic optimization." In: *Operations Research* (2022). DOI: 10.1287/opre.2022.2265.

[7] D. Bienstock and M. Zuckerberg. "Solving LP relaxations of large-scale precedence constrained problems." In: *International Conference on Integer Programming and Combinatorial Optimization*. Ed. by F. Eisenbrand and F. B. Shepherd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–14. DOI: 10.1007/978-3-642-13036-6_1.

[8] D. Cattaruzza, M. Labbé, M. Petris, M. Roland, and M. Schmidt. "Exact and Heuristic Solution Techniques for Mixed-Integer Quantile Minimization Problems." In: *INFORMS Journal on Computing* (2023). Forthcoming.

[9] A. Charnes, W. W. Cooper, and G. H. Symonds. "Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil." In: *Management science* 4.3 (1958), pp. 235–263. DOI: 10.1287/mnsc.4.3.235.

[10] J. Cho and A. Papavasiliou. "Exact Mixed-Integer Programming Approach for Chance-Constrained Multi-Area Reserve Sizing." In: *IEEE Transactions on Power Systems* (2023). DOI: 10.1109/TPWRS.2023.3279692.

[11] S. Dempe. *Foundations of bilevel programming*. Springer Science & Business Media, 2002. DOI: https://doi.org/10.1007/b101970.

[12] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. "Scenario reduction in stochastic programming." In: *Mathematical Programming* 95 (2003), pp. 493–511. DOI: 10.1007/s10107-002-0331-0.

[13] F. Errico, G. Desaulniers, M. Gendreau, W. Rei, and L.-M. Rousseau. "The vehicle routing problem with hard time windows and stochastic service times." In: *EURO Journal on Transportation and Logistics* 7 (2018), pp. 223–251. DOI: 10.1007/s13676-016-0101-4.

[14] D. Espinoza and E. Moreno. "A primal-dual aggregation algorithm for minimizing conditional value-at-risk in linear programs." In: *Computational Optimization and Applications* 59.3 (2014), pp. 617–638. DOI: 10.1007/s10589-014-9692-6.

[15] R. L. Graham, D. E. Knuth, O. Patashnik, and S. Liu. "Concrete mathematics: a foundation for computer science." In: *Computers in Physics* 3.5 (1989), pp. 106–107.

[16] O. Günlük and Y. Pochet. "Mixing mixed-integer inequalities." In: *Mathematical Programming* 90 (2001), pp. 429–457. DOI: 10.1007/PL00011430.

[17] H. Heitsch and W. Römisch. "Scenario reduction algorithms in stochastic programming." In: *Computational Optimization and Applications* 24 (2003), pp. 187–206. DOI: 10.1023/A:1021805924152.

[18] F. Klnc-Karzan, S. Kücükyavuz, and D. Lee. "Joint chance-constrained programs and the intersection of mixing sets through a submodularity lens." In: *Mathematical Programming* 195.1 (2021), pp. 283–326. DOI: 10.1007/s10107-021-01688-1.

[19] S. Kücükyavuz. "On mixing sets arising in chance-constrained programming." In: *Mathematical Programming* 132.1-2 (2012), pp. 31–56. DOI: 10.1007/s10107-010-0385-3.

[20] J. Luedtke. "A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support." In: *Mathematical Programming* 146.1-2 (2014), pp. 219–244. DOI: 10.1007/s10107-013-0684-6.

[21] J. Luedtke and S. Ahmed. "A sample approximation approach for optimization with probabilistic constraints." In: *SIAM Journal on Optimization* 19.2 (2008), pp. 674–699. DOI: 10.1137/070702928.

[22] J. Luedtke, S. Ahmed, and G. L. Nemhauser. "An integer programming approach for linear programs with probabilistic constraints." In: *Mathematical Programming* 122.2 (2010), pp. 247–272. DOI: 10.1007/s10107-008-0247-4.

[23] J. M. Morales, S. Pineda, A. J. Conejo, and M. Carrion. "Scenario reduction for futures market trading in electricity markets." In: *IEEE Transactions on Power Systems* 24.2 (2009), pp. 878–888. DOI: 10.1109/TPWRS.2009.2016072.

[24] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro. "Sample average approximation method for chance constrained programming: theory and applications." In: *Journal of Optimization Theory and Applications* 142.2 (2009), pp. 399–416. DOI: 10.1007/s10957-009-9523-6.

[25] B. S. Pay and Y. Song. "Partition-based decomposition algorithms for two-stage Stochastic integer programs with continuous recourse." In: *Annals of Operations Research* 284.2 (2020), pp. 583–604. DOI: 10.1007/s10479-017-2689-7.

[26] A. Porras, C. Dominguez, J. M. Morales, and S. Pineda. "Tight and Compact Sample Average Approximation for Joint Chance-Constrained Problems with Applications to Optimal Power Flow." In: *INFORMS Journal on Computing* (2023). DOI: 10.1287/ijoc.2022.0302.

[27] F. Qiu, S. Ahmed, S. S. Dey, and L. A. Wolsey. "Covering linear programming with violations." In: *INFORMS Journal on Computing* 26.3 (2014), pp. 531–546. DOI: 10.1287/ijoc.2013.0582.

[28] H. Rahimian and B. Pagnoncelli. "Data-driven approximation of contextual chance-constrained stochastic programs." In: *SIAM Journal on Optimization* 33.3 (2023), pp. 2248–2274. DOI: 10.1137/22M1528045.

[29] C. Ramirez-Pico, I. Ljubić, and E. Moreno. "Benders Adaptive-Cuts Method for Two-Stage Stochastic Programs." In: *Transportation Science* 57.5 (2023), pp. 1252–1275. DOI: 10.1287/trsc.2022.0073.

[30] C. Ramirez-Pico and E. Moreno. "Generalized adaptive partition-based method for two-stage stochastic linear programs with fixed recourse." In: *Mathematical Programming* 196.1-2 (2022), pp. 755–774. DOI: 10.1007/s10107-020-01609-8.

[31] N. Rujeerapaiboon, K. Schindler, D. Kuhn, and W. Wiesemann. "Scenario reduction revisited: Fundamental limits and guarantees." In: *Mathematical Programming* 191.1 (2022), pp. 207–242. DOI: 10.1007/s10107-018-1269-1.

[32] M. Siddig and Y. Song. "Adaptive partition-based SDDP algorithms for multistage stochastic linear programming with fixed recourse." In: *Computational Optimization and Applications* 81 (2022), pp. 201–250. DOI: 10.1007/s10589-021-00323-1.

[33] Y. Song and J. Luedtke. "An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse." In: *SIAM Journal on Optimization* 25.3 (2015), pp. 1344–1367. DOI: 10.1137/140967337.

[34] Y. Song and J. R. Luedtke. "Branch-and-cut approaches for chance-constrained formulations of reliable network design problems." In: *Mathematical Programming Computation* 5.4 (2013), pp. 397–432. DOI: 10.1007/s12532-013-0058-3.

[35] Y. Song, J. R. Luedtke, and S. Kücükyavuz. "Chance-constrained binary packing problems." In: *INFORMS Journal on Computing* 26.4 (2014), pp. 735–747. DOI: 10.1287/ijoc.2014.0595.

[36] M. W. Tanner and L. Ntaimo. "IIS branch-and-cut for joint chance-constrained stochastic programs and application to optimal vaccine allocation." In: *European Journal of Operational Research* 207.1 (2010), pp. 290–296. DOI: 10.1016/j.ejor.2010.04.019.

[37] W. Xie and S. Ahmed. "On quantile cuts and their closure for chance constrained optimization problems." In: *Mathematical Programming* 172 (2018), pp. 621–646. DOI: 10.1007/s10107-017-1190-z.

(M. Roland, A. Forel, T. Vidal) André-Aisenstadt Pavillon, 2920 Tour Road, Montreal, Quebec H3T 1N8, Canada

Email address: mmmroland@gmail.com, alexandre.forel@polymtl.ca, thibaut.vidal@cirrelt.ca