We study assignment problem with chance constraints (CAP) and its distributionally robust counterpart (DR-CAP). We present a technique for estimating big-M in such a formulation that takes advantage of the ambiguity set. We consider a 0-1 bilinear knapsack set to develop valid inequalities for CAP and DR-CAP. This is used within a probability cut framework to solve DR-CAP. A computational study on problem instances obtained from using real hospital surgery data shows that the developed techniques allow us to solve certain model instances, and reduce the computational time for others. The use of Wasserstein ambiguity set in the DR-CAP model improves the out-of-sample performance of satisfying the chance constraints more significantly than the one possible by increasing the sample size in the sample average approximation technique. The solution time for DR-CAP model instances is of the same order as that for solving the CAP instances. This finding is important because chance constrained optimization models are very difficult to solve when the coefficients in the constraints are random.

*Key words*: chance-constrained assignment problem, distributionally robust optimization, bilinear program, branch-and-cut, valid inequalities, operating room planning
1. Introduction

In the chance-constrained assignment problem, we assign the items with random weights to available bins and minimize the assignment cost while satisfying the bin capacity constraints with probability at least \(1 - \varepsilon\). In a motivating example, surgeries with random durations are assigned to available operating rooms, and we want to ensure that the assigned surgeries complete within a specified duration with a high probability. More specifically, we study the chance-constrained assignment problem:

\[
\begin{align*}
\text{(CAP)} & \quad \min_{\mathbf{y} \in \{0,1\}^{|\mathcal{I}| \times |\mathcal{J}|}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} y_{ij} \\
\text{subject to} & \quad \sum_{j \in \mathcal{J}} y_{ij} = 1, \quad \forall i \in \mathcal{I}, \quad (1a) \\
& \quad \sum_{i \in \mathcal{I}} y_{ij} \leq \rho_j, \quad \forall j \in \mathcal{J}, \quad (1b) \\
& \quad \mathbb{P} \left\{ \sum_{i \in \mathcal{I}} \xi_i y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, \quad \forall j \in \mathcal{J}, \quad (1d)
\end{align*}
\]

where \(\mathcal{I} := \{1, \ldots, |\mathcal{I}|\}\) is the set of items, \(\mathcal{J} := \{1, \ldots, |\mathcal{J}|\}\) is the set of bins, \(|\cdot|\) is the cardinality of a set, \(c_{ij}\) is the nonnegative cost for assigning item \(i\) to bin \(j\), \(\rho_j\) is the quantitative restriction of bin \(j\), and \(t_j\) is the capacity of bin \(j\). \(\xi_i\) is the random weight of item \(i\). The binary decision variable \(y_{ij}\) indicates if item \(i\) is assigned to bin \(j\). Let \(\mathbf{y}_j := (y_{1j}, \ldots, y_{|\mathcal{I}|j})^\top\) for \(j \in \mathcal{J}\), and \(\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_{|\mathcal{J}|})^\top\). The objective \((1a)\) minimizes the total cost of assigning the items to the bins. Constraints \((1b)\) ensure that item \(i\) is assigned to only one bin. Constraints \((1c)\) ensure that at most \(\rho_j\) items are assigned to bin \(j\). Constraints \((1d)\) ensure that the capacity for bin \(j\) is satisfied with probability \(1 - \varepsilon\), where \(\varepsilon \in [0,1]\). The chance-constrained assignment problem has a wide range of applications such as in healthcare (Zhang et al. 2015), facility location (Peng et al. 2019), and cloud computing (Cohen et al. 2019), among others.

There are several challenges in solving the chance-constrained assignment problem. First, \((\text{CAP})\) is not a convex optimization problem, given that the variables in \((\text{CAP})\) are binary and chance constraints \((1d)\) might not induce a convex set. Moreover, the chance constraint is generally difficult to evaluate. In the chance-constrained programming (CCP) literature, it is commonly assumed that the probability distributions of the random weights \(\xi_i\) are known and finitely supported. Incomplete knowledge of the probability distribution of \(\xi_i\) can be addressed by using an ambiguity set \(\mathcal{P}\) that allows a family of distributions. The chance constraints \((1d)\) are satisfied over all probability distributions within the ambiguity set \(\mathcal{P}\), resulting in the formulation:

\[
\begin{align*}
\text{(DR-CAP)} & \quad \min_{\mathbf{y} \in \{0,1\}^{|\mathcal{I}| \times |\mathcal{J}|}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} y_{ij} \\
& \quad \mathbb{P} \left\{ \sum_{i \in \mathcal{I}} \xi_i y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, \quad \forall j \in \mathcal{J}, \quad (2a)
\end{align*}
\]
subject to (1b), (1c),

\[
\inf_{\mathbb{P} \in \mathcal{P}} \left\{ \sum_{i \in I} \xi_i y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, \quad \forall j \in \mathcal{J}.
\]  

In this paper we assume that the probability distribution \( \mathbb{P} \) has finite support \( \xi := (\xi_1, \ldots, \xi_N)^T \), where \( \xi_\omega := (\xi_1^\omega, \ldots, \xi_N^\omega)^T \) for \( \omega \in \Omega := \{1, \ldots, N\} \). \( \xi_i^\omega \) denotes the weight of item \( i \) for scenario \( \omega \in \Omega \), and \( p_\omega \) is the probability of scenario \( \omega \in \Omega \) such that \( p_\omega \geq 0 \) and \( \sum_{\omega \in \Omega} p_\omega = 1 \). We further assume that \( \xi_i^\omega \) and \( t_j \) are non-negative integers, and without loss of generality, \( p_\omega \leq \varepsilon \) and \( \xi_i^\omega \leq t_j \), for \( i \in I, j \in \mathcal{J}, \omega \in \Omega \).

The model framework corresponds to an approach where a sample average approximation replaces the original distribution of a random vector with a finite number of samples (Luedtke and Ahmed 2008, Pagnoncelli et al. 2009). The SAA approach may provide a good candidate solution for the ‘true’ chance-constrained program (Shapiro et al. 2009, Calafiore and Campi 2006). This has motivated a number of studies for solving CCPs by formulating it as a mixed-integer program (see, e.g., Luedtke et al. (2010), Küçükyavuz (2012), Abdi and Fukasawa (2016), Liu et al. (2019), Zhao et al. (2017), Peng et al. (2019)).

### 1.1. Chance-Constrained Programs with Random Technology Matrices

The model in (1) has randomness in the coefficients of the constraints, i.e., it has a random technology matrix. CCPs with random technology matrices are significantly more difficult to solve than the case where only the right-hand-side vector is random (Tanner and Ntaimo (2010)). Tanner and Ntaimo (2010) used irreducible infeasible subsystems to derive a class of valid inequalities for such problems. Luedtke (2014) used a technique similar to the one for generating valid inequalities for CCPs with random right-hand side to develop strong valid inequalities, and proposed a branch-and-cut decomposition algorithm for CCPs. Qiu et al. (2014) proposed an iterative scheme to improve the coefficient estimation in a big-M formulation, and observed that the coefficient strengthening technique can significantly decrease the solution time. van Ackooij et al. (2016) investigated a generalized Benders decomposition approach with stabilization and inexact function computation to solve CCP. Liu et al. (2016) studied two-stage CCPs and developed a Benders decomposition approach with strengthened optimality cuts to solve the problem. More recently, Xie and Ahmed (2018) projected the mixing inequalities onto the original space to derive a family of quantile cuts for such problems.

### 1.2. Integer Chance-Constrained Programs

For the integer programming problem with chance constraints, Beraldi and Bruni (2010) formulated the problem as an integer program with knapsack constraints, and used the feasible solutions of the knapsack constraints to divide the feasible region of the problem within a branch and bound scheme. Song and Luedtke (2013) studied a chance-constrained reliable network design problem.
They derived valid inequalities for this problem. Song et al. (2014) considered a chance-constrained packing problem. This problem is to select a subset of items that maximizes the total profit while satisfying a single chance constraint. The problem is viewed as a probabilistic cover problem, and the probabilistic cover inequalities are developed by using a lifting technique from Zemel (1989). Deng and Shen (2016) investigated a chance-constrained appointment scheduling problem and used a decomposition algorithm with formulation strengthening strategies to solve this problem. Wu and Küçükyavuz (2017) studied a chance-constrained combinatorial optimization problem and presented an exact method for solving the problem under the assumption that the chance probability can be calculated. Wang et al. (2019) studied the chance-constrained multiple bin-packing problem and used the lifting technique to develop a family of valid inequalities.

1.3. Distributionally Robust Optimization

In the distributionally robust optimization (DRO) framework, the probability distribution of the random variables lies in an ambiguity set. Two widely used ambiguity sets are the moment-based ambiguity sets (see, e.g., Delage and Ye (2010), Wiesemann et al. (2014), Mehrotra and Papp (2014), and Bansal et al. (2018)) and the statistical distance-based sets (see, e.g., Ben-Tal et al. (2013), Jiang and Guan (2018), Esfahani and Kuhn (2018), and Luo and Mehrotra (2019)). For the distributionally robust chance-constrained programs, Chen et al. (2010) and Zymler et al. (2013) developed tractable approximations of ambiguous chance constraints under the moment-based ambiguity sets. Hanasusanto et al. (2017) studied the ambiguous joint chance constraints where the ambiguity set is characterized by the mean and an upper bound on the dispersion, and presented a convex reformulation under some conditions. Jiang and Guan (2016) studied a data-driven distributionally robust chance-constrained model using a $\phi$-divergence measure-based set. They showed that this problem is equivalent to a classical chance-constrained problem with a perturbed risk level. As an important type of statistical distance, the Wasserstein metric can be used to define an ambiguity set, which has a polyhedral structure. Thus, several studies have investigated the use of distributionally robust chance-constrained problems with the Wasserstein ambiguity set (see, e.g., Xie (2018), Chen et al. (2018)). For the distributionally robust chance-constrained binary programs, Cheng et al. (2014) considered the distributionally robust chance-constrained quadratic knapsack problem and assumed that the first and second moments, and the joint support of random variables are known. They provided a semidefinite programming (SDP) relaxation for the binary constraints. Zhang et al. (2015) assumed that only the mean and the variance are known, and investigated the two-stage distributionally robust chance-constrained bin-packing problem with continuous bin extension decisions. They developed a branch-and-price approach based on a column generation reformulation to solve the mixed-integer reformulation. Deng et al. (2016) studied chance-constrained surgery planning by
using a $\phi$-divergence measure-based ambiguity set, and used a branch-and-cut algorithm to solve the mixed-integer linear reformulation of this problem. Zhang et al. (2018) considered the distributionally robust chance-constrained bin-packing problem in which only the mean and the covariance matrix are known. They reformulated the problem as a binary second-order cone (SOC) program, and developed valid inequalities for the SOC program by using the submodularity and the bin-packing structure of the model. Finally, we refer interested readers to a recent survey by Rahimian and Mehrotra (2019) for more details about DRO.

1.4. Contributions of This Paper

This paper makes the following specific contributions:

- We use a big-M approach to formulate (CAP) and (DR-CAP) as binary and semi-infinite integer programs, respectively. We first present a coefficient strengthening approach for the semi-infinite integer reformulation and provide alternative bilinear reformulations for (CAP) and (DR-CAP). The big-M strengthening approach presented here takes advantage of the ambiguity set in its computations, and it is applicable for more general problems.

- We develop a new family of valid inequalities for the binary bilinear knapsack set from a single row and scenario in the bilinear constraints and the constraints (1c). More specifically, we use the lifting technique for the binary bilinear knapsack set to derive lifted cover inequalities and show that these inequalities are facet-defining under certain conditions. Furthermore, we present stronger valid inequalities for (CAP) and (DR-CAP) by further restricting the feasible region of $y$ in a lifting problem.

- We consider the intersection of multiple binary bilinear knapsacks with a general 0-1 knapsack constraint, and a cardinality constraint. By using the lifting technique and a heuristic procedure, we obtain another new family of valid inequalities for this intersection set. These valid inequalities are a generalization of the cover inequalities.

- We develop separation heuristics that efficiently obtain the violated inequalities and incorporate the inequalities in a branch-and-cut framework to solve the strengthened big-M binary reformulation of (CAP). We then propose a branch-and-cut algorithm with probability cuts, which uses a distribution separation procedure, the valid inequalities developed in this paper, and the feasibility/probability cuts, to solve the strengthened big-M semi-infinite reformulation of (DR-CAP). A convergence proof of this algorithm is provided.

- We perform a computational study for an assignment problem based on real data from a hospital to show the benefits of the techniques developed in this paper. Using the techniques developed in this paper, we solve (CAP) instances with up to 1,500 scenarios within ten hours when $\varepsilon = 0.08, 0.1, 0.12$, and obtain a smaller optimality gap for instances with $\varepsilon = 0.06$. For (DR-CAP) using the Wasserstein
metric, we solved all instances with $N = 1,500$ within two hours for $\varepsilon = 0.1$. We performed an out-of-sample estimation of the chance constraint satisfaction for the solutions obtained from (CAP) and (DR-CAP). The (DR-CAP) solutions achieve the desirable probability target more reliably, though we find that both (CAP) and (DR-CAP) models may violate the chance constraint out-of-sample when the sample size and the radius of the Wasserstein set are small. We find that robustness of the solution is improved when using a moderate size sample and a Wasserstein ambiguity set. As expected, the (DR-CAP) solutions are more ‘costly’. (DR-CAP) instances are solved in about four times the time required to solve (CAP).

1.5. Organization

The remainder of this paper is organized as follows. Section 2 formulates (CAP) as a binary integer program using the big-M technique. Subsequently, in this section, we formulate (DR-CAP) as a semi-infinite program and present a big-M coefficient strengthening procedure for this formulation. We then present alternative bilinear formulations for (CAP) and (DR-CAP), respectively. We exploit the structure of the bilinear formulations to develop two classes of valid inequalities in Section 3. Specifically, in Section 3.1 we utilize the sequential lifting technique to develop the lifted cover inequalities for the binary bilinear knapsack set and show that these inequalities are facet-defining under certain conditions. We then present stronger lifted cover inequalities for (CAP) and (DR-CAP) by restricting the feasible region of $y$. We further analyze the multiple binary bilinear knapsack sets with a general 0-1 knapsack constraint and develop a class of valid inequalities in Section 3.2. In Section 4, we describe a branch-and-cut solution scheme for (CAP) and propose separation heuristics to obtain the violated valid inequalities. A branch-and-cut algorithm with probability cuts for solving (DR-CAP), and its convergence proof is provided in this section. Section 5 reports computational results on (CAP) and (DR-CAP) formulations of the operating room assignment problem. Section 6 concludes the paper with a summary of the important findings. Appendix A presents the proofs of the propositions and theorems. Appendix B describes the pseudo-code of the algorithms implemented in our computations. Appendix C gives a dynamic programming approach to compute the big-M coefficients. Appendix D presents the statistics of surgery duration for the real-life data. Appendix E provides the additional computational results.

2. Model Reformulation

We formulate (CAP) as a binary linear program in Section 2.1. A semi-infinite reformulation for (DR-CAP) is presented in Section 2.2. We then present binary bilinear reformulations for (CAP) and (DR-CAP) in Section 2.3.
2.1. Binary Integer Reformulation for (CAP)

Let the binary variable \( z_{j\omega} \) indicate if the capacity constraint is violated for \( j \in \mathcal{J} \) and \( \omega \in \Omega \). Namely, \( z_{j\omega} = 1 \) if the constraint \( \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} \leq t_j \) is satisfied, and \( z_{j\omega} = 0 \), otherwise. For \( j \in \mathcal{J} \), let \( z_j := (z_{j1}, \ldots, z_{jN})^T \) and \( z := (z_1, \ldots, z_{|\mathcal{J}|})^T \). The constraints (1d) can be formulated as

\[
\begin{align*}
\sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} + (M_j^\omega - t_j)z_{j\omega} &\leq M_j^\omega, & \forall j \in \mathcal{J}, \omega \in \Omega, \\
\sum_{\omega \in \Omega} p_{\omega} z_{j\omega} &\geq 1 - \varepsilon, & \forall j \in \mathcal{J},
\end{align*}
\]

where \( M_j^\omega \) is a constant that ensures that the constraints (3a) hold when \( z_{j\omega} = 0 \). Computation of a small valid value of big-M gives a tighter formulation in (3). We develop a big-M coefficient strengthening procedure inspired from Song et al. (2014) to obtain a value of \( M_j^\omega \). Note that for \( j \in \mathcal{J} \) and \( \omega \in \Omega \):

\[
M_j^\omega \geq \bar{M}_j^\omega := \max_{y_j \in \{0,1\}^{2|\mathcal{I}|}} \left\{ \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} \mid \mathbb{P} \left\{ \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, y_j \in \mathcal{Y}_j \right\}
\]

where \( \mathcal{Y}_j := \{y_j \mid \sum_{i \in \mathcal{I}} y_{ij} \leq \rho_j \} \). For \( j \in \mathcal{J} \) and \( \omega, k \in \Omega \), let

\[
m_j^\omega(k) := \max_{y_j \in \{0,1\}^{2|\mathcal{I}|}} \left\{ \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} \left\{ \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} \leq t_j, y_j \in \mathcal{Y}_j \right\} \right. \]

We sort \( m_j^\omega(k) \) such that \( m_j^\omega(k_1) \leq \ldots \leq m_j^\omega(k_N) \). An upper bound for \( \bar{M}_j^\omega \) is given in Proposition 1. A proof of this proposition is given in Appendix A.1.

**Proposition 1.** \( m_j^\omega(k_q) \) is an upper bound for \( \bar{M}_j^\omega \), where \( q := \min \{ l \mid \sum_{j=1}^l p_{kj} > \varepsilon \} \), and (CAP) can be equivalently reformulated as the following binary integer program

\[
\text{(IP)} \quad \begin{array}{ll}
\text{minimize} & \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} y_{ij} \\
\text{subject to} & (1b), (1c), (3b), \\
& \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} + (m_j^\omega(k_q) - m_j^\omega(\omega)) z_{j\omega} \leq m_j^\omega(k_q), \forall j \in \mathcal{J}, \omega \in \Omega.
\end{array}
\]

**Remark.** (5) has a knapsack constraint and a cardinality constraint. We use a dynamic programming method for solving (5) in Section 5. The procedure uses the methodology in Bertsimas and Demir (2002). For \( j \in \mathcal{J} \), if \( t_j \) and \( \rho_j \) are moderate, dynamic programming is an efficient approach for solving (5) to optimality (see more details about the dynamic programming in Appendix C).

**Remark.** The constraints (1b) and (1c) represent the assignment structure of model (1). In the corresponding statement of Proposition 1, they can be replaced with a general constraint set \( \mathcal{Y}_j \).
2.2. Semi-Infinite Programming Reformulation for (DR-CAP)

In this section, we study the chance-constrained models, where the distribution of random weights belongs to an ambiguity set. The results in this section are stated for any ambiguity set defined on a finite support (see Bansal et al. (2018)). However, in the computational results of this paper, we used the $l_1$-Wasserstein ambiguity set:

$$
\mathcal{P}_W = \{ p \in \mathbb{R}_+^N | \sum_{\omega \in \Omega} p_\omega = 1, \sum_{\omega \in \Omega} \sum_{k \in \Omega} \| \xi^\omega - \xi^k \| \nu_{\omega k} \leq \eta, \sum_{k \in \Omega} \nu_{\omega k} = p_\omega, \forall \omega \in \Omega, \sum_{\omega \in \Omega} \nu_{\omega k} = p_k, \forall k \in \Omega, \nu_{\omega k} \geq 0, \forall \omega, k \in \Omega \},
$$

where $\eta \geq 0$ is the Wasserstein radius and $\{p_k^*\}_{k \in \Omega}$ is an empirical probability distribution of $\xi$. Note that if $\eta = 0$, then $p_\omega = p_\omega^*$ for all $\omega \in \Omega$ and (DR-CAP) reduces to (CAP). Let $\mathbb{1}(\cdot)$ denote an indicator function. Using this notation the constraint (2b) using the Wasserstein ambiguity set is given as follows:

$$
\inf \left\{ \sum_{\omega \in \Omega} p_\omega \mathbb{1} \left( \sum_{i \in I} \xi_i^\omega y_{ij} \leq t_j \right) \left| p \in \mathcal{P}_W \right. \right\} \geq 1 - \varepsilon, \quad \forall j \in J.
$$

Let $z_{j\omega}$ and $m_j^\omega(\cdot)$ be defined as in Section 2.1. The following theorem gives a reformulation of (DR-CAP) with a general ambiguity set $\mathcal{P}$. Note that this formulation is a semi-infinite program because of constraints (8b). The proof is given in Appendix A.2.

**Theorem 1.** We sort $m_j^\omega(\cdot)$ in a non-decreasing order such that $m_j^\omega(k_1) \leq \cdots \leq m_j^\omega(k_N)$. Then, (DR-CAP) can be represented as the following semi-infinite program

$$(SIP) \begin{align*}
\min_{(y,z) \in \{0,1\}^I \times \{0,1\}^J} & \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
\text{subject to} & \quad (1b), (1c), \\
& \quad \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega} p_\omega z_{j\omega} \geq 1 - \varepsilon, \quad \forall j \in J, \\
& \quad \sum_{i \in I} \xi_i^\omega y_{ij} + (m_j^\omega(k_\bar{q}) - m_j^\omega(\omega)) z_{j\omega} \leq m_j^\omega(k_\bar{q}), \quad \forall j \in J, \omega \in \Omega,
\end{align*}$$

where $\bar{q} := \min \{ l | \sup_{p \in \mathcal{P}} \sum_{j=1}^l p_k > \varepsilon \}$. □

Note that when $\mathcal{P} := \mathcal{P}_W$, (SIP) can be rewritten as

$$(DSIP) \begin{align*}
\min_{(y,z) \in \{0,1\}^I \times \{0,1\}^J} & \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
\text{subject to} & \quad (1b), (1c), (8c) \\
& \quad \mu_j^4 - \eta \mu_j^2 + \sum_{k \in \Omega} p_k^* \mu_j^4 \geq 1 - \varepsilon, \quad \forall j \in J,
\end{align*}$$

\(-\mu_j^1 + \mu_j^3 + z_{j\omega} \geq 0, \quad \forall j \in \mathcal{J}, \omega \in \Omega, \)  
\(9c\)

\[ \|\xi^\omega - \xi^k\| \mu_j^2 - \mu_j^3 - \mu_j^4 \geq 0, \quad \forall j \in \mathcal{J}, \omega, k \in \Omega, \]  
\(9d\)

\[ \mu_j^2 \geq 0, \quad \forall j \in \mathcal{J}, \]  
\(9e\)

where \(\mu^1, \mu^2, \mu^3,\) and \(\mu^4\) are dual variables of the constraints in \((7)\). Similar dual formulations are possible when \(\mathcal{P}\) for other ambiguity sets for which strong duality holds and the dual formulations are explicitly available (see e.g., Rahimian and Mehrotra (2019)).

Note also that when \(\mathcal{P} := \mathcal{P}_W, \bar{q}\) in Theorem 1 is obtained by solving a sequence of linear programs. Moreover, the left-hand side of \((8b)\) is a linear program for a fixed \(z_{j\omega}.\) The use of an optimization problem in identifying \(\bar{q}\) may provide a smaller value of \(m_j^\omega(\cdot)\) used in the big-M formulation. Since solving linear programs for computing the index \(\bar{q}\) for all \(j \in \mathcal{J}\) and \(\omega \in \Omega\) can be time-consuming, the following corollary shows that the use of any distribution in the set \(\mathcal{P}\) is sufficient for the big-M estimation. Such distributions are available as the probability cut algorithm given in Section 4.3 progresses.

**Corollary 1.** Let \(\{\hat{p}_\omega\}_{\omega \in \Omega} \in \mathcal{P},\) and \(\hat{q} = \min\{l | \sum_{j=1}^l \hat{p}_k > \varepsilon\}.\) Then, \(\hat{q} \leq \bar{q}\) and \(m_j^\omega(k_{\bar{q}}) \leq m_j^\omega(k_{\hat{q}}).\)

**Proof.** Since \(\sup_{p \in \mathcal{P}} \sum_{j=1}^\hat{q} p_k \geq \sum_{j=1}^\hat{q} \hat{p}_k > \varepsilon,\) we have \(\hat{q} \geq \bar{q}\) and \(m_j^\omega(k_{\hat{q}}) \leq m_j^\omega(k_{\bar{q}}).\) \(\square\)

**Remark.** Theorem 1 and Corollary 1 remain valid for the case where the cardinality constraint in \((4)\) is replaced by a more general constraint set \(\mathcal{Y}_j\) for \(j \in \mathcal{J}.\)

### 2.3. Binary Bilinear Reformulations

In the previous section, we calculated the strengthened big-M coefficients to formulate the chance constraints as binary linear constraints. In this section, we present an alternative approach following Wang et al. (2019). Let \(z_{j\omega}\) be defined as in Section 2.1. The constraints \((6b)\) and \((8c)\) can also be rewritten as

\[ \sum_{i \in I} \xi^\omega_{ji} y_{ij} z_{j\omega} \leq m_j^\omega(\omega) z_{j\omega}, \quad \forall j \in \mathcal{J}, \omega \in \Omega. \]  
\((10)\)

Thus, we can use \((10)\) to obtain a binary bilinear reformulation and bilinear semi-infinite reformulation for \((\text{CAP})\) and \((\text{DR-CAP}),\) respectively. The following proposition shows a relationship between the bilinear reformulations with the formulations \((8)\) and \((6),\) respectively. A proof is given in Appendix A.3.

**Proposition 2.** The relaxation of the binary bilinear reformulation for \((\text{CAP})\) obtained from relaxing the binary variables is stronger than the linear relaxation of \((6).\) Similarly, the relaxation of the binary bilinear reformulation for \((\text{DR-CAP})\) obtained from relaxing the binary variables is stronger than the linear relaxation of \((8).\) \(\square\)
Note that constraints (1c), (3b) and (10) give a key substructure of the binary bilinear reformulation of (CAP). Let $\mathcal{H} := \{(y, z) \in \{0,1\}^{J} \times \{0,1\}^{N} | (1c), (3b), (10)\}$. For $j \in J$, let

$$
\mathcal{G}_j := \left\{(y_j, z_j) \in \{0,1\}^{J} \times \{0,1\}^{N} \bigg| \sum_{i \in I} \xi_i^{j} y_{ij} z_{j\omega} \leq m^{e}_{j}(\omega) z_{j\omega}, \forall \omega \in \Omega, \sum_{\omega \in \Omega} p_{\omega} z_{j\omega} \geq 1 - \varepsilon, y_j \in \mathcal{Y}_j \right\}.
$$

The set $\mathcal{G}_j$ is the intersection of multiple binary bilinear knapsacks with a general knapsack constraint, and a cardinality constraint. We have $\mathcal{H} = \bigcap_{j \in J} \{(y, z) | (y_j, z_j) \in \mathcal{G}_j\}$.

Let us use $\text{conv}(\cdot)$ to denote the convex hull of a set. The following proposition shows that in order to identify strong valid inequalities for $\text{conv}(\mathcal{H})$, we can develop strong valid inequalities for $\text{conv}(\mathcal{G}_j)$. A proof can be found in Appendix A.4.

**Proposition 3.** If an inequality is valid for $\text{conv}(\mathcal{G}_j)$, this inequality is also valid for $\text{conv}(\mathcal{H})$. Moreover, if an inequality is facet-defining for $\text{conv}(\mathcal{G}_j)$, it is also facet-defining for $\text{conv}(\mathcal{H})$. $\square$

Proposition 3 gives a motivation to investigate the set $\mathcal{G}_j$. Hence, in the following, we develop a class of valid inequalities for $\mathcal{G}_j$. Similarly, we define a key substructure $\mathcal{G}_j'$ of the bilinear reformulation of (DR-CAP) and obtain valid inequalities for $\mathcal{G}_j'$. For $j \in J$, let

$$
\mathcal{G}_j' := \{(y_j, z_j) \in \{0,1\}^{J} \times \{0,1\}^{N} \bigg| \sum_{i \in I} \xi_i^{j} y_{ij} z_{j\omega} \leq m^{e}_{j}(\omega) z_{j\omega}, \forall \omega \in \Omega, \inf_{p \in p'} \sum_{\omega \in \Omega} p_{\omega} z_{j\omega} \geq 1 - \varepsilon, y_j \in \mathcal{Y}_j \}.
$$

### 3. Valid Inequalities for (CAP) and (DR-CAP)

We first apply the lifting technique for the knapsack problem to a binary bilinear knapsack set and develop a family of valid inequalities in Section 3.1. Section 3.2 further presents a family of valid inequalities for $\mathcal{G}_j$ and $\mathcal{G}_j'$.

#### 3.1. Lifted Cover Inequalities

We assume that $j \in J$, $\omega \in \Omega$ are fixed in this section. Let us consider the binary bilinear knapsack set $\mathcal{F}_{j\omega} := \left\{(y_j, z_{j\omega}) \in \{0,1\}^{J} \times \{0,1\} \bigg| \sum_{i \in I} \xi_i^{j} y_{ij} z_{j\omega} \leq m^{e}_{j}(\omega) z_{j\omega}, y_j \in \mathcal{Y}_j \right\}$. Note that the inequalities valid for $\text{conv}(\mathcal{F}_{j\omega})$ are also valid for (CAP) and (DR-CAP). Note also that when compared to the development in Wang et al. (2019), we include the cardinality constraint in addition to the binary bilinear knapsack constraint in the description of $\mathcal{F}_{j\omega}$. When $z_{j\omega} = 1$, the set $\mathcal{F}_{j\omega}$ becomes the two-constraint 0-1 knapsack set $\mathcal{Q}_{j\omega} := \left\{y_j \in \{0,1\}^{J} \bigg| \sum_{i \in I} \xi_i^{j} y_{ij} \leq m^{e}_{j}(\omega), y_j \in \mathcal{Y}_j \right\}$.

We now extend the results for the single binary knapsack set from Zemel (1989) and Gu et al. (1998) to develop a valid inequality that under a condition is facet-defining for the set $\mathcal{Q}_{j\omega}$. We also provide a lifted cover inequality that is valid for $\text{conv}(\mathcal{F}_{j\omega})$ by rotating this valid inequality. Then the restriction of the feasible region of $y$ is used to obtain a stronger valid inequality for (CAP) and (DR-CAP).
Definition 1. Set \( \mathcal{C} \subseteq \mathcal{I} \) is a cover for \( \sum_{i \in \mathcal{I}} \xi^w_i y_{ij} \leq m^w_j(\omega) \) if \( \sum_{i \in \mathcal{C}} \xi^w_i > m^w_j(\omega) \). The cover \( \mathcal{C} \) is minimal if no subset of \( \mathcal{C} \) is a cover for \( \sum_{i \in \mathcal{I}} \xi^w_i y_{ij} \leq m^w_j(\omega) \). \( \square \)

In this section, we assume that \( \mathcal{C} \) is a minimal cover for \( \sum_{i \in \mathcal{I}} \xi^w_i y_{ij} \leq m^w_j(\omega) \). Let \( \mathcal{D} \subseteq \mathcal{C} \). The following proposition gives a valid inequality that is facet-defining under suitable cardinality conditions for the following convex hull. A proof is given in Appendix A.5.

\[
\text{conv} \left( \left\{ y_j \in \{0,1\}^{\mathcal{I}} \left| \sum_{i \in \mathcal{I}} \xi^w_i y_{ij} \leq m^w_j(\omega), y_j \in \mathcal{Y}, y_{ij} = 0, \forall i \in \mathcal{I} \setminus \mathcal{C}, y_{ij} = 1, \forall i \in \mathcal{D} \right. \right\} \right).
\] (11)

Proposition 4. The inequality

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \alpha_i y_{ij} \leq |\mathcal{C} \setminus \mathcal{D}| - 1
\] (12)

is valid for (11). If \( |\mathcal{C}| \leq \rho_j + 1 \), the inequality (12) is facet-defining for (11). \( \square \)

3.1.1. Up-Lifting In general, a cover inequality (12) does not induce a facet of a knapsack set. To obtain a facet-defining inequality of a knapsack set, we compute coefficients of variables in \( \mathcal{I} \setminus \mathcal{C} \). This procedure is called up-lifting. By using the up-lifting technique, we obtain an inequality of the form

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \alpha_i y_{ij} \leq |\mathcal{C} \setminus \mathcal{D}| - 1,
\] (13)

where \( \alpha_i \) is called an up-lifting coefficient. We now provide such an uplifting approach for our problem. Let \( \{\pi_k\}_{k=1}^{\mathcal{I} \setminus \mathcal{C}} \) be a sequence of the set \( \mathcal{I} \setminus \mathcal{C} \) and \( \pi(k) = \{\pi_1, \ldots, \pi_k\} \). For \( k = 1, \ldots, |\mathcal{I} \setminus \mathcal{C}| \), let

\[
\text{obj}_{\pi_k} := \max_{y_j \in \{0,1\}^{(\mathcal{C} \setminus \mathcal{D}) \cup \omega(k-1)}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \pi(k-1)} \alpha_i y_{ij}
\] (14a)

subject to

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \xi^w_i y_{ij} + \sum_{i \in \pi(k-1)} \xi^w_i y_{ij} \leq m^w_j(\omega) - \xi^w_i - \sum_{i \in \mathcal{D}} \xi^w_i,
\] (14b)

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \pi(k-1)} y_{ij} \leq \rho_j - 1 - |\mathcal{D}|.
\] (14c)

Note that different sequences of \( \mathcal{I} \setminus \mathcal{C} \) might lead to different valid inequalities (Kaparis and Letchford 2008). The following lemma gives a sufficient condition under which the inequality (13) is facet-defining for the convex hull of \( \mathcal{Q}_j, \omega \) when \( y_{ij} = 1, i \in \mathcal{D} \). The proof is given in Appendix A.6.

Lemma 1. For \( k = 1, \ldots, |\mathcal{I} \setminus \mathcal{C}| \), let \( \alpha_{\pi_k} = |\mathcal{C} \setminus \mathcal{D}| - 1 - \text{obj}_{\pi_k} \), where \( \text{obj}_{\pi_k} \) is defined in (14). The inequality (13) is valid for

\[
\text{conv} \left( \left\{ y_j \in \{0,1\}^{\mathcal{I}} \left| \sum_{i \in \mathcal{I}} \xi^w_i y_{ij} \leq m^w_j(\omega), y_j \in \mathcal{Y}, y_{ij} = 1, \forall i \in \mathcal{D} \right. \right\} \right).
\] (15)

If \( |\mathcal{C}| \leq \rho_j + 1 \), the inequality (13) is facet-defining for (15). \( \square \)
We use a dynamic programming based approach, which has been used to calculate the up-lifting coefficients in the binary single knapsack problem (see Zemel (1989)), to obtain the lifting coefficient \( \alpha_i \). This algorithm is given in Appendix B.1.

**Remark.** Note that the sufficient condition in Lemma 1 for ensuring that an inequality is facet defining requires us to start with covers with cardinality less than \( \rho_j + 1 \). The inequality remains valid when this condition is not satisfied. However, it suggests a preference for identifying low cardinality covers.

### 3.1.2. Down-Lifting

Similar to the up-lifting, down-lifting computes the coefficients for the variables \( y_{ij} \) in \( D \). We use this technique to obtain a valid inequality for \( \text{conv}(Q_j) \) of the form

\[
\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \mathcal{I} \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} \leq |C \setminus D| + \sum_{i \in D} \beta_i - 1, \tag{16}
\]

where for \( i \in D \), \( \beta_i \) is called a down-lifting coefficient. The coefficient \( \beta_i \) can be obtained by solving the following sequence of problems. Let \( \{\kappa_i\}_{i=1}^{\mathcal{D}} \) be a sequence of the set \( \mathcal{D} \) and \( \kappa(l) = \{\kappa_1, \ldots, \kappa_l\} \).

For \( l = 1, \ldots, |\mathcal{D}| \), let

\[
\text{obj}_{\kappa_i} := \max_{y_j \in \{0,1\}^{(\mathcal{I} \setminus D) \cup \kappa(l-1)}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \mathcal{I} \setminus C} \alpha_i y_{ij} + \sum_{i \in \kappa(l-1)} \beta_i y_{ij} \tag{17a}
\]

subject to

\[
\sum_{i \in \mathcal{I} \setminus D} \xi_i^\omega y_{ij} + \sum_{i \in \kappa(l-1)} \xi_i^\omega y_{ij} \leq m^\omega_j(\omega) - \sum_{i = n_{l+1}}^{n_{\mathcal{D}}} \xi_i^\omega, \tag{17b}
\]

\[
\sum_{i \in \mathcal{I} \setminus D} y_{ij} + \sum_{i \in \kappa(l-1)} y_{ij} \leq \rho_j - |\mathcal{D}| + l. \tag{17c}
\]

**Lemma 2.** For \( l = 1, \ldots, |\mathcal{D}| \), let \( \beta_{\kappa_i} = \text{obj}_{\kappa_i} - \sum_{i \in \kappa(l-1)} \beta_i - |\mathcal{C} \setminus D| + 1 \), where \( \text{obj}_{\kappa_i} \) is defined in (17).

The inequality (16) is valid for \( \text{conv}(Q_j) \). If \( |C| \leq \rho_j + 1 \), (16) is facet-defining for \( \text{conv}(Q_j) \).

**Proof** See Appendix A.7. □

### 3.1.3. Lifted Cover Inequality

In the following, we provide coefficient calculations for a lifted cover inequality that is valid for \( \text{conv}(F_j) \). The proof is provided in Appendix A.8.

**Theorem 2.** The lifted cover inequality

\[
\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \mathcal{I} \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} + \gamma(z^\omega - 1) \leq |C \setminus D| + \sum_{i \in D} \beta_i - 1 \tag{18}
\]

is valid for \( \text{conv}(F_j) \) if

\[
\gamma = \max_{y_j \in \{0,1\}^{(\mathcal{I} \setminus D) \cup \kappa}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \mathcal{I} \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} - |C \setminus D| - \sum_{i \in D} \beta_i + 1. \tag{19a}
\]

Furthermore, if \( |C| \leq \rho_j + 1 \), (18) is facet-defining for \( \text{conv}(F_j) \). □
By restricting the feasible region of $y_j$ in (19) using the chance constraints (1d), we obtain a stronger valid inequality for (CAP) in Theorem 3. The proof is given in Appendix A.9.

**Theorem 3.** For $k \in \Omega \setminus \{\omega\}$, let

$$
\begin{align*}
\delta_k &= \maximize_{y_j \in \{0,1\}^{|E_j|}} \sum_{j \in C} y_{ij} + \sum_{i \in C \setminus D} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} - |C \setminus D| - \sum_{i \in D} \beta_i + 1 \\
\text{subject to } \sum_{i \in I} \xi_i^k y_{ij} &\leq m_j^k(k).
\end{align*}
$$

Sort $\delta_k$ such that $\delta_k \leq \ldots \leq \delta_{k_{|\Omega|-1}}$. Let $q^1 := \min \{l \mid \sum_{j=1}^l p_{kj} > \varepsilon\}$, then the inequality (18) is valid for (CAP), where $\gamma = \delta_{k_{q^1}}$. □

We further restrict the feasible region of $y_j$ in (19) by using (2b) to obtain a stronger valid inequality for (DR-CAP) in the following theorem. The proof is given in Appendix A.10.

**Theorem 4.** For $k \in \Omega \setminus \{\omega\}$, let $\delta_k$ be defined as in Theorem 3, and sort $\delta_k$ such that $\delta_{k_1} \leq \ldots \leq \delta_{k_{|\Omega|-1}}$. Let $q^1 := \min \{l \mid \sup_{p \in \mathcal{P}} \sum_{j=1}^l p_{kj} > \varepsilon\}$. Then, the inequality (18) is valid for (DR-CAP) when $\gamma = \delta_{k_{q^1}}$. Moreover, if $\{\tilde{p}_\omega\}_{\omega \in \Omega} \in \mathcal{P}$, let $\bar{q}^1 := \min \{l \mid \sum_{j=1}^l \tilde{p}_{kj} > \varepsilon\}$. Then, $\bar{q}^1 \geq q^1$ and the inequality (18) is valid for (DR-CAP) when $\gamma = \delta_{k_{\bar{q}^1}}$. □

3.1.4. **Examples of the Lifted Cover Inequalities.** We now provide an example to illustrate the lifted cover inequalities described in the previous sections and the advantage of using the cardinality constraint (i.e., solving a two-constrained dynamic program). In the second example, we use the family of valid inequalities referred to as single lifted cover inequality (obtained by ignoring the cardinality constraint in $F_{j|\omega}$) and show that it gets strengthened in the DR framework.

**Example 1.** Suppose $F_{j|\omega}$ is defined by $\rho_j = 3$, $m^\top_j(\omega) = 40$, and $\xi_j = (7, 8, 10, 11, 14, 23)^\top$. Then the set $C = \{1, 2, 3, 4, 5\}$ is a minimal cover. Let $D = \{5\}$, suppose $N = 5$, $\varepsilon = 0.6$, and the other scenarios in the computation of lifted cover inequalities are $(8, 11, 7, 10, 17, 23)^\top$, $(14, 7, 10, 11, 8, 13, 26)^\top$, $(21, 10, 7, 29, 16, 12, 23)^\top$, and $(15, 7, 8, 23, 12, 10, 5)^\top$, with $p_\omega = 1/N$ for all $\omega \in \Omega$. We get a lifted cover inequality by Theorem 3 as:

$$y_{ij} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + 2y_{6j} + 2y_{7j} + z_{j|\omega} \leq 5. \tag{21}$$

If $p^*_\omega = 1/N$ for all $\omega \in \Omega$ and $\eta = 0.5$ in the Wasserstein set $W$ in (7), then a lifted cover inequality for (DR-CAP) obtained from Theorem 4 is given as follows:

$$y_{ij} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + 2y_{6j} + 2y_{7j} \leq 4. \tag{22}$$
Example 2. (Continued from Example 1) Suppose that the cardinality constraint $\sum_{i \in I} y_{ij} \leq \rho_j$ is removed from $F_j$. Following a computation procedure similar to the one for the lifted cover inequality, we obtain a valid inequality of the following form:

$$y_{1j} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + y_{6j} + 2y_{7j} + z_{j\omega} \leq 5.$$  \hspace{1cm} (23)

We call the inequality (23) single lifted cover inequality. Obviously, the lifted cover inequality (21) is stronger than the single lifted cover inequality (23). The single lifted cover inequality for (DR-CAP) is

$$y_{1j} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + y_{6j} + 2y_{7j} \leq 4.$$  \hspace{1cm} (24)

(22) is also stronger than (24). Thus showing the possible benefit of using the cardinality constraint, and the ambiguity set in the coefficient calculations.

3.2. Global Lifted Cover Inequalities

In this section we develop a class of valid inequalities referred to as global lifted cover inequalities for $G_j$ and $G_j'$, which are valid for (CAP) and (DR-CAP), respectively. For (CAP), let $\Omega$ be a set where each element $\Omega_k \in \tilde{\Omega}$ is a subset of $\Omega$ such that $\sum_{\omega \in \Omega_k} p_\omega \geq 1 - \varepsilon$, for $k = 1, \ldots, |\tilde{\Omega}|$. Without loss of generality, we reuse the notation set $\tilde{\Omega}$ and $\Omega_k$ for (DR-CAP). For (DR-CAP), let $\tilde{\Omega}$ be a set where each element $\Omega_k \in \tilde{\Omega}$ is a subset of $\Omega$ such that $\inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega_k} p_\omega \geq 1 - \varepsilon$, for $k = 1, \ldots, |\tilde{\Omega}|$. $\tilde{\Omega}$ is maximal if it is not a proper subset of any other sets that satisfy the above condition. For maximal $\tilde{\Omega}$, let the global lifted cover inequalities be of the form:

$$\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in T \setminus C} \bar{\alpha}_i y_{ij} + \sum_{i \in D} \bar{\beta}_i y_{ij} + \sum_{\omega \in \Omega_k} \bar{\gamma}_\omega (z_{j\omega} - 1) \leq |C \setminus D| + \sum_{i \in D} \bar{\beta}_i - 1, \; k = 1, \ldots, |\tilde{\Omega}|,$$  \hspace{1cm} (25)

where $C$ is a cover for the set $Q_{j\omega}$ for some $\omega \in \Omega$, and $D \subseteq C$. For $k \in \{1, \ldots, |\tilde{\Omega}|\}$, when $z_{j\omega} = 1$, $\omega \in \Omega_k$, (25) becomes

$$\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in T \setminus C} \bar{\alpha}_i y_{ij} + \sum_{i \in D} \bar{\beta}_i y_{ij} \leq |C \setminus D| + \sum_{i \in D} \bar{\beta}_i - 1.$$  \hspace{1cm} (26)

Kaparis and Letchford (2008) developed a valid inequality for multi-constrained knapsack problems. In Section 3.2.1 and 3.2.2, we use the ideas from Kaparis and Letchford (2008) to calculate the coefficients $\bar{\alpha}_i$ and $\bar{\beta}_i$ in (26).
3.2.1. Up-Lifting Let \( \{\bar{\pi}_l\}_{l=1}^{\vert \mathcal{C} \setminus \mathcal{I} \vert} \) be a sequence of \( \mathcal{I} \setminus \mathcal{C} \) and \( \bar{\pi}(l) = \{\bar{\pi}_1, \ldots, \bar{\pi}_l\} \). For \( l = 1, \ldots, \vert \mathcal{I} \setminus \mathcal{C} \vert \), the up-lifting problem is as follows:

\[
\text{obj}_{\bar{\pi}_l} := \max_{y_j \in \{0, 1\}^{(\mathcal{C} \setminus \mathcal{D}) \cup \mathcal{I}(l-1)}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \pi(l-1)} \bar{\alpha}_i y_{ij}
\]

subject to

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \xi^\omega_i y_{ij} + \sum_{i \in \pi(l-1)} \xi^\omega_i y_{ij} \leq m^\omega_j(\omega) - \xi^\omega_i - \sum_{i \in \mathcal{D}} \xi^\omega_i, \quad \forall \omega \in \Omega_k, \quad (27b)
\]

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \pi(l-1)} y_{ij} \leq \rho_j - 1 - \vert \mathcal{D} \vert. \quad (27c)
\]

Then \( \bar{\alpha}_{\bar{\pi}_l} = \vert \mathcal{C} \setminus \mathcal{D} \vert - 1 - \text{obj}_{\bar{\pi}_l} \). It is time-consuming to solve the up-lifting problem exactly. Dynamic programming is also not an efficient approach since its complexity grows with the number of constraints in (27). Kaparis and Letchford (2008) suggest relaxing \( y_j \in \{0, 1\}^{\vert \mathcal{I} \vert} \) and solving the LP relaxation to compute an upper bound on \( \text{obj}_{\bar{\pi}_l} \). The objective value is then rounded down to the nearest integer. In order to make use of the dynamic programming based algorithm for (14), we propose a heuristic to calculate \( \bar{\alpha}_{\bar{\pi}_l} \) as follows. For each \( \omega \in \Omega_k \), let

\[
\text{obj}_{\bar{\pi}_l}(\omega) := \max_{y_j \in \{0, 1\}^{(\mathcal{C} \setminus \mathcal{D}) \cup \mathcal{I}(l-1)}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \pi(l-1)} \bar{\alpha}_i y_{ij}
\]

subject to

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \xi^\omega_i y_{ij} + \sum_{i \in \pi(l-1)} \xi^\omega_i y_{ij} \leq m^\omega_j(\omega) - \xi^\omega_i - \sum_{i \in \mathcal{D}} \xi^\omega_i, \quad (28b)
\]

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \pi(l-1)} y_{ij} \leq \rho_j - 1 - \vert \mathcal{D} \vert. \quad (28c)
\]

Then, \( \text{obj}_{\bar{\pi}_l}(\omega) \) is an upper bound for \( \text{obj}_{\bar{\pi}_l} \). We use \( \min_{\omega \in \Omega_k} \text{obj}_{\bar{\pi}_l}(\omega) \) to obtain a minimal upper bound for \( \text{obj}_{\bar{\pi}_l} \), from among the values \( \{\text{obj}_{\bar{\pi}_l}(\omega)\}_{\omega \in \Omega_k} \). Let \( \bar{\alpha}_{\bar{\pi}_l} = \vert \mathcal{C} \setminus \mathcal{D} \vert - 1 - \min_{\omega \in \Omega_k} \text{obj}_{\bar{\pi}_l}(\omega) \), which implies \( \bar{\alpha}_{\bar{\pi}_l} \leq \vert \mathcal{C} \setminus \mathcal{D} \vert - 1 - \text{obj}_{\bar{\pi}_l} \). Thus, \( \bar{\alpha}_{\bar{\pi}_l} \) is a valid lifting coefficient.

3.2.2. Down-Lifting Similarly, we can obtain the down-lifting coefficient \( \bar{\beta}_i \) for \( i \in \mathcal{D} \). Let \( \{\bar{k}_l\}_{l=1}^{\vert \mathcal{D} \vert} \) be a sequence of \( \mathcal{D} \) and \( \bar{k}(l) = \{\bar{k}_1, \ldots, \bar{k}_l\} \). For \( l = 1, \ldots, \vert \mathcal{D} \vert \), let

\[
\text{obj}_{\bar{k}_l} := \max_{y_j \in \{0, 1\}^{(\mathcal{D} \setminus \mathcal{I}) \cup \mathcal{I}(l-1)}} \sum_{i \in \mathcal{D} \setminus \mathcal{I}} y_{ij} + \sum_{i \in \mathcal{I}(l-1)} \bar{\alpha}_i y_{ij} + \sum_{i \in \bar{k}(l-1)} \bar{\beta}_i y_{ij}
\]

subject to

\[
\sum_{i \in \mathcal{D} \setminus \mathcal{I}} \xi^\omega_i y_{ij} + \sum_{i \in \mathcal{I}(l-1)} \xi^\omega_i y_{ij} \leq m^\omega_j(\omega) - \sum_{i \in \bar{k}(l+1)} \xi^\omega_i, \quad \forall \omega \in \Omega_k, \quad (28b)
\]

\[
\sum_{i \in \mathcal{D} \setminus \mathcal{I}} y_{ij} + \sum_{i \in \mathcal{I}(l-1)} y_{ij} \leq \rho_j + \vert \mathcal{D} \vert + l. \quad (28c)
\]

Instead of computing \( \text{obj}_{\bar{k}_l} \), we use the method proposed in Section 3.2.1 to obtain an upper bound for \( \text{obj}_{\bar{k}_l} \). Let \( \text{obj}_{\bar{k}_l}(\omega) \) be the optimal objective value of the maximization problem that takes a single row \( \omega \) of problem (28) for \( \omega \in \Omega_k \). Let \( \bar{\beta}_{\bar{k}_l} = \min_{\omega \in \Omega_k} \text{obj}_{\bar{k}_l}(\omega) - \sum_{i \in \bar{k}(l-1)} \bar{\beta}_i - \vert \mathcal{C} \setminus \mathcal{D} \vert + 1. \)
3.2.3. Global Lifted Cover Inequalities Finally, to calculate $\tilde{\gamma}_\omega$ in sequence $\{\tau_1, \ldots, \tau_{\Omega_k}\}$, we consider the following problem for $\mathcal{G}_j$, for $l = 1, \ldots, |\Omega_k|$:

$$ \text{obj}_{l\tau} = \max_{(y_j, z_j) \in \{0, 1\}^{|\mathcal{I}|} \times \{0, 1\}^{(|\Omega_k| \cup |\tau|)}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \tilde{\alpha}_i y_{ij} + \sum_{i \in \mathcal{D}} \tilde{\beta}_i y_{ij} + \sum_{\omega \in \tau(l-1)} \gamma_{\omega} z_{j\omega} \quad (29a) $$

subject to

$$ \sum_{\omega \in \Omega \setminus \Omega_k} p_{\omega} z_{j\omega} + \sum_{\omega \in \tau(l-1)} p_{\omega} z_{j\omega} \geq 1 - \varepsilon - \sum_{\omega = \tau_{l+1}}^{\tau_{\Omega_k}} p_{\omega}, \quad (29b) $$

$$ \sum_{i \in \mathcal{I}} \xi_i \omega y_{ij} + (m_i^\omega(k_q) - m_i^\omega(\omega)) z_j \leq m_i^\omega(k_q), \quad \forall \omega \in \Omega \setminus \Omega_k \cup \tau(l-1), \quad (29c) $$

where $\tau(l-1) = \{\tau_1, \ldots, \tau_{l-1}\}$. The calculation of $\text{obj}_{l\tau}$ is a reformulation of a chance-constrained problem where some variables $z_{j\omega}$ are given. Instead of solving (29) exactly, we provide a heuristic to obtain an upper bound for $\text{obj}_{l\tau}$. We relax $y_j \in [0, 1]^{|\mathcal{I}|}$ and $z_j \in [0, 1]^{|\Omega_k|}$, and solve the LP relaxation of (29) to obtain the optimal solution $(y_j^*, z_j^*)$ and objective value $\text{obj}_{l\tau}^*$ of the relaxed problem. Then, $\text{obj}_{l\tau}^*$ gives an upper bound for $\text{obj}_{l\tau}$.

For $\mathcal{G}_j^i$, for $l = 1, \ldots, |\Omega_k|$, let

$$ \text{obj}_{l\tau}^i := \max_{(y_j, z_j) \in \{0, 1\}^{|\mathcal{I}|} \times \{0, 1\}^{(|\Omega_k| \cup |\tau|)}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \tilde{\alpha}_i y_{ij} + \sum_{i \in \mathcal{D}} \tilde{\beta}_i y_{ij} + \sum_{\omega \in \tau(l-1)} \gamma_{\omega} z_{j\omega} \quad (30a) $$

subject to (29d),

$$ \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega \setminus \Omega_k} p_{\omega} z_{j\omega} + \sum_{\omega \in \tau(l-1)} p_{\omega} z_{j\omega} \geq 1 - \varepsilon - \sum_{\omega = \tau_{l+1}}^{\tau_{\Omega_k}} p_{\omega}, \quad (30b) $$

$$ \sum_{i \in \mathcal{I}} \xi_i \omega y_{ij} + (m_i^\omega(k_q) - m_i^\omega(\omega)) z_j \leq m_i^\omega(k_q), \quad \forall \omega \in \Omega \setminus \Omega_k \cup \tau(l-1). \quad (30c) $$

We relax $y_j \in [0, 1]^{|\mathcal{I}|}$ and $z_j \in [0, 1]^{|\Omega_k|}$, and solve the relaxation problem using the method similar to Algorithm 1 in Section 4.3, and obtain the optimal objective value $\text{obj}_{l\tau}^i$, which is an upper bound on $\text{obj}_{l\tau}^i$.

Theorem 5 gives valid inequalities for $\text{conv}(\mathcal{G}_j)$ and $\text{conv}(\mathcal{G}_j^i)$. A proof is given in Appendix A.11.

**Theorem 5.** Let $\{\tilde{\alpha}_i\}_{i \in \mathcal{I} \setminus \mathcal{C}}$ and $\{\tilde{\beta}_i\}_{i \in \mathcal{D}}$ be defined as in Section 3.2.1 and 3.2.2, respectively. For $l = 1, \ldots, |\Omega_k|$, we set $\tilde{\gamma}_{l\tau} = |\text{obj}_{l\tau}^i| - |\mathcal{C} \setminus \mathcal{D}| + 1 - \sum_{i \in \mathcal{D}} \tilde{\beta}_i - \sum_{\omega \in \tau(l-1)} \tilde{\gamma}_\omega$, where $\text{obj}_{l\tau}^i$ is the objective value of the LP relaxation of (29). Then, (25) is valid for $\text{conv}(\mathcal{G}_j)$. For $l = 1, \ldots, |\Omega_k|$, we set $\tilde{\gamma}_{l\tau} = |\text{obj}_{l\tau}^i| - |\mathcal{C} \setminus \mathcal{D}| + 1 - \sum_{i \in \mathcal{D}} \tilde{\beta}_i - \sum_{\omega \in \tau(l-1)} \tilde{\gamma}_\omega$, where $\text{obj}_{l\tau}^i$ is the objective value of the LP relaxation of (30). Then, (25) is valid for $\text{conv}(\mathcal{G}_j^i)$. $\square$

The following example gives a global lifted cover inequality.
**Example 3. (Continued from Example 1)** We let $\mathcal{C} = \{1, 2, 3, 4, 5\}$ and $\mathcal{D} = \{5\}$ as before. Let $\Omega_k = \{1, 2\}$. Then we can obtain a global lifted cover inequality (25) for (CAP) given as follows:

$$y_{1j} + y_{2j} + y_{3j} + y_{4j} + 2y_{6j} + 2y_{7j} + z_{j1} + z_{j2} \leq 5.$$ 

For (DR-CAP), $\Omega_k = \{1, 2, 3\}$ satisfies $\inf_{\omega \in \Omega_k} \sum_{p \in \mathcal{P}} p \omega \geq 1 - \varepsilon$. A valid inequality (25) is given by

$$y_{1j} + y_{2j} + y_{3j} + y_{4j} + 2y_{6j} + 3y_{7j} + 2z_{j1} + 2z_{j2} + 2z_{j3} \leq 9.$$ 

**4. Solution Scheme**

In Section 4.1, we present a heuristic sequential lifting procedure for separating the valid inequalities developed in Section 3. These valid inequalities are used within a branch-and-cut framework to solve the strengthened big-M binary reformulation (IP) of (CAP) in Section 4.2. A branch-and-cut algorithm with probability cuts to solve the strengthened big-M semi-infinite reformulation (SIP) of (DR-CAP) is given in Section 4.3.

**4.1. Separation Problem**

Separation problem finds valid inequalities that are violated by an LP relaxation solution $(\hat{y}, \hat{z})$. In this section, we adopt the ideas from Gu et al. (1998) and Kaparis and Letchford (2008) for the binary knapsack problem to separate (18) and (25), respectively.

**4.1.1. Separation Problem for (18)** To obtain the violated inequalities (18), we use a heuristic similar to the one in Gu et al. (1998) for the knapsack problem. This heuristic is provided as Algorithm 3 in Appendix B.2. Let $(\hat{y}, \hat{z})$ be a LP relaxation optimal solution, when $\hat{z}_{j\omega} = 1$

$$\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{I} \cap \mathcal{C}} \alpha_i \hat{y}_{ij} + \sum_{i \in \mathcal{D}} \beta_i (\hat{y}_{ij} - 1) + \gamma (\hat{z}_{j\omega} - 1) = \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{I} \setminus (\mathcal{C} \cup \mathcal{I}_0)} \alpha_i \hat{y}_{ij} > |\mathcal{C} \setminus \mathcal{D}| - 1.$$ 

Hence, we obtain an inequality that is violated by the LP relaxation solution.

If $|\mathcal{D}| > \rho_j - 1$ or $m_j^\omega(\omega) - \sum_{i \in \mathcal{D}} \xi_i^\omega - \max_{i \in \mathcal{I} \setminus \mathcal{C}} \xi_i^\omega < 0$ for $\omega \in \Omega$, the down-lifting problems might be infeasible since the right hand side of the down-lifting problems might be negative. In this case, we remove items from $\mathcal{D}$ until $|\mathcal{D}| \leq \rho_j - 1$ and $m_j^\omega(\omega) - \sum_{i \in \mathcal{D}} \xi_i^\omega - \max_{i \in \mathcal{I} \setminus \mathcal{C}} \xi_i^\omega \geq 0$ for $\omega \in \Omega$.

**4.1.2. Separation Problem for (25)** We extended the heuristic in Kaparis and Letchford (2008) for the multidimensional knapsack problem, to obtain the violated inequalities (25) in our case. Algorithm 4 in Appendix B.3 gives an overview of this heuristic.

Similar to the discussion in Section 4.1.1, if $|\mathcal{D}| > \rho_j - 1$ or $m_j^\omega(\omega) - \sum_{i \in \mathcal{D}} \xi_i^\omega - \max_{i \in \mathcal{I} \setminus \mathcal{C}} \xi_i^\omega < 0$ for some $\omega \in \Omega$, we remove the items from $\mathcal{D}$ until $|\mathcal{D}| \leq \rho_j - 1$ and $m_j^\omega(\omega) - \sum_{i \in \mathcal{D}} \xi_i^\omega - \max_{i \in \mathcal{I} \setminus \mathcal{C}} \xi_i^\omega \geq 0$ for all $\omega \in \Omega$. 

Wang, Li and Mehrata: Distributionally Robust Chance-Constrained Assignment
4.2. Branch-and-Cut Algorithm for (CAP)

The valid inequalities in Section 3 are used within a branch-and-cut implementation to solve (CAP). An overview of the branch-and-cut framework is given in Algorithm 5 (Appendix B.4). The algorithm uses the violated inequalities described in Section 4.1.1 and 4.1.2 in line 9 (see Section 5.2 for further discussion). Let LB and UB denote the current lower and upper bound for the optimal objective value of (CAP), and \( N \) denote the set of remaining nodes in the branch-and-cut search tree.

4.3. Branch-and-Cut Algorithm with Probability Cuts for (DR-CAP)

We now investigate the probability cuts within a branch-and-cut framework for solving (DR-CAP). We define the master problem as follows:

\[
\text{(MP)} \quad \min_{(y,z) \in \{(0,1)^{|I||J|\times\{0,1\}^{|J|\times|N|}\}} \cap \mathcal{X}} \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \quad \text{subject to (1b), (1e), (8c)},
\]

where the set \( \mathcal{X} \) is a complementary set that defines the feasible region of (8). Set \( \mathcal{X} \) is defined by a set of probability and feasibility cuts. Let \((\hat{y}, \hat{z})\) be a feasible solution of (MP). For \( j \in J \), a distribution separation problem is given by:

\[
(S_{\mathcal{P}_j}) \quad S_{\mathcal{P}_j}(\hat{z}) := \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega} p_{\omega} \hat{z}_{j\omega}.
\]

The problem (SP \( j \)) is used to verify the feasibility of \((\hat{y}_j, \hat{z}_j)\) to (DR-CAP). If \( S_{\mathcal{P}_j}(\hat{z}) \geq 1 - \varepsilon \), \((\hat{y}_j, \hat{z}_j)\) is feasible to (DR-CAP). Otherwise, probability and feasibility cuts are added to (MP) as follows.

Let \( \{\hat{p}_\omega\}_{\omega \in \Omega} \) be an optimal solution of (SP \( j \)) corresponding to \( \hat{z} \), a probability cut is given by

\[
\sum_{\omega \in \Omega} \hat{p}_\omega z_{j\omega} \geq 1 - \varepsilon. \tag{32}
\]

Let \( I^1_j = \{i \in I | \hat{y}_{ij} = 1\} \). The following feasibility cut in \( y \) variables is added to (MP):

\[
\sum_{i \in I^1_j} y_{ij} \leq |I^1_j| - 1. \tag{33}
\]

In Algorithm 1, UB and LB denote the upper and lower bound, respectively. We initialize the algorithm by setting the iteration number \( k \) to 0, UB to positive infinity, and LB to negative infinity. We add a node \( o \) to the node list \( N \) and use (LMP) to denote the LP relaxation of (MP) (line 1-2). At the selected node \( o \), we solve (LMP) and obtain the corresponding optimal solution \((y^k, z^k)\) and the objective value \( l_{\text{obj}}^k \) (line 4-6). If the objective value \( l_{\text{obj}}^k \) is smaller than the current upper bound, then we check if \((y^k, z^k)\) is binary (line 7). If \((y^k, z^k)\) is binary, we solve the distribution separation problem (SP \( j \)) with \( \mathcal{P} \) for all \( j \in J \), and obtain the optimal solution \( \{p^k_\omega\}_{\omega \in \Omega} \) and the objective value
$\mathbf{uobj}^k$. We add probability and feasibility cuts to (LMP) if $\mathbf{uobj}^k$ is smaller than $1 - \varepsilon$ (line 8-14). If we find probability and feasibility cuts, we go to line 5, and resolve (LMP) at the current node $o$. Otherwise, $(y^k,z^k)$ is a feasible solution to (DR-CAP), we update the upper bound and record the corresponding solution $(y^k,z^k)$ (line 15-20). If $(y^k,z^k)$ is fractional, we add violated inequalities or continue branching (line 22-30). We terminate our algorithm when the node list is empty, and return the optimal value UB and the optimal solution $(y^*,z^*)$ (line 33). Algorithm 1 gives a pseudocode of the branch-and-cut algorithm with probability and feasibility cuts.

The following theorem shows that Algorithm 1 terminates in a finite number of iterations for solving (DR-CAP) to optimality under certain conditions.

**Theorem 6.** If there exists an oracle that solves (SP$_j$) to optimality, then Algorithm 1 terminates in finitely many iterations. If $\mathbf{UB} < +\infty$, $\mathbf{UB}$ is the optimal value of (DR-CAP) and Algorithm 1 obtains an optimal solution $(y^*,z^*)$ at termination.

**Proof** See Appendix A.12. □

**Remark 1.** Recall that $\mathcal{P}_W$ in (7) is a polyhedral set with a finite number of extreme points, thus in this case (SP$_j$) can be solved to optimality.

**Remark 2.** In the case of a polyhedral ambiguity set, such as the set $\mathcal{P}_W$, instead of using the probability cut approach discussed above it is possible to dualize the problem in (8) and write this constraint explicitly (see Rahimian and Mehrotra (2019) and references therein). Such dualization introduces dual variables corresponding to the constraints specifying $\mathcal{P}_W$. While developing the probability cut approach presented above, we also implemented this dualization approach to solve our test problems (see next section). We found that none of the test instances could be solved to optimality with a 10 hour CPU time limit. Specifically, the average optimality gap for the 500 scenario instances remained more than 80%, and for the larger instances ($N = 1500$) this approach could not even find a feasible solution. We conjecture that it is because introduction of continuous (dual) variables in a problem that is otherwise pure binary adds significantly to the difficulty in solving the resulting model.

5. Computational Experiments

We now present computational results for (CAP) and (DR-CAP). Computational experiments were performed using data from an operating room (OR) assignment problem, where a set of surgeries are assigned to operating rooms. Each surgery has a random duration, and each OR has a time limit determined by its work hours. Problem instance generation is discussed in Section 5.1. Section 5.2 provides additional implementation details. The performance of the branch-and-cut algorithm (Algorithm 5) for solving (CAP) is discussed in Section 5.3 and the branch-and-cut algorithm with probability cuts (Algorithm 1) for solving (DR-CAP) is discussed in Section 5.4. Section 5.5 presents
Algorithm 1: Branch-and-Cut Algorithm with Probability Cuts

1. Initialize \( P^0 \in \mathcal{P} \), the number of iteration \( k = 0 \), \( UB = +\infty \), \( LB = -\infty \), \( \mathcal{N} = \{o\} \). \( o \) has no branching constraints.

2. Initialize the root node with the LP relaxation of (MP). Let the LP relaxation of (MP) be denoted by (LMP).

3. while \( \mathcal{N} \) is nonempty do

4. Select a node \( o \in \mathcal{N} \), \( \mathcal{N} \leftarrow \mathcal{N}/\{o\} \).

5. Solve (LMP) at the node \( o \). \( k = k + 1 \).

6. Obtain the optimal solution \((y^k, z^k)\) and the optimal objective \( lobj^k \) of (LMP).

7. if \( lobj^k < UB \) then

8. if \((y^k, z^k)\) is an integer then

9. for \( j \in \mathcal{J} \) do

10. Solve (SP\(_j\)), and obtain an optimal solution \((p^k)\) and objective value \( uobj^k \)

11. if \( uobj^k < 1 - \varepsilon \) then

12. Add the cuts (32) and (33) to (LMP).

13. end

14. end

15. if Cuts (32) and (33) are found then

16. Go to step 5.

17. end

18. else

19. UB = \( lobj^k \), \((y^*, z^*) = (y^k, z^k)\).

20. end

21. end

22. if \((y^k, z^k)\) is fractional then

23. Use Algorithm 3 and 4 to find the violated inequalities.

24. if Violated inequalities are found then

25. Add the violated inequalities to (LMP). Go to line 5.

26. end

27. else

28. Branch, resulting in nodes \( o^* \) and \( o^{**} \), \( \mathcal{N} \leftarrow \mathcal{N} \cup \{o^*, o^{**}\} \).

29. end

30. end

31. end

32. end

33. return UB and its corresponding optimal solution \((y^*, z^*)\).

the performance of strengthening big-M in (SIP). Section 5.6 compares the out-of-sample performance of the solutions generated from the (DR-CAP) instances with the corresponding (CAP) instances.

5.1. Instance Generation

We used historical surgery duration data from a large public hospital in Beijing, China from January 2015 to October 2015. 5,721 surgery durations for the nine major surgery types are available. For the problem instances, the log-normal distribution with the mean and the standard deviation of the surgery duration (see Appendix D) was used to generate surgery duration samples (i.e. Deng and Shen (2016)). The samples generated from the log-normal distribution were rounded to the nearest 15 minutes and assigned equal probabilities as in sample average approximation. Eight (\(|\mathcal{J}| = 8\)) ORs are available to serve \(|\mathcal{I}| = 27\) surgeries (close to the maximum number of surgeries in a day) a day.
The daily time limit $t_j$ is 10 hours, $\forall j \in J$. Following Zhang et al. (2018), we let the assignment cost $c_{ij}$ vary in $[0, 16], \forall i \in I, j \in J$. The number of surgeries in an OR, $\rho_j$, is limited to $[3, 5], \forall j \in J$. We used the number of surgeries and the percentage for each surgery type to calculate the number of surgeries for each surgery type performed in a day. To ensure that (CAP) is always feasible, we added a pseudo OR $j'$ to the set of ORs, which has no quantitative and capacity restrictions. We set the assignment cost $c_{ij'}$ for $i \in I$ as 27. The sample size $N \in \{500, 1000, 1500\}$ and the level of chance satisfaction $\epsilon \in \{0.12, 0.1, 0.08, 0.06\}$ were used in the (CAP) instance generation. Five instances were generated for each sample size.

5.2. Implementation Details

In our implementation of the branch-and-cut algorithm, we add the violated valid inequalities generated from (18) at the nodes that are at a depth no more than 1. No limit was placed on the number of such inequalities added to the formulation. We observed that it is more time-consuming to find a violated inequality of the type (25). Therefore, we added the violated inequalities from (25) at the nodes that are at a depth no more than 2, and the number of violated inequalities of this type was limited to 15. The valid inequalities are generated until one of the following stopping criteria is met: no cut is available with the violation threshold $10^{-2}$, or the number of iterations is up to 100 at the root node of the branch-and-cut tree. At each round of cut generation of the type (18), for each $j \in J$, multiple violated inequalities might be found. We only added the inequality with the most violated value to the branch-and-cut tree.

The algorithm was implemented in the C programming language using IBM CPLEX solver, version 12.71 callable libraries. A laptop with Intel(R) 2.80 GHz processor and 16 GB RAM was used for computations on a 64-bit computer using the Windows operating system. We turned off the CPLEX presolve procedure and set the number of threads to one for all computations. We used CPLEX callback functions for adding the violated valid inequalities proposed in this paper. For all computations, a priority order for the binary variables in the node selection rule was used. The variables $y$ were given a higher priority than $z$. We used a runtime limit of 10 hours or an optimality tolerance of 1% as our stopping criteria. For instances that could not be solved to meet the stopping criteria, we give the average optimality gap, where the optimality gap is calculated as $(UB - LB)/UB$, and UB and LB are the upper and lower bound, respectively. We report the solution time (in seconds) for the instances that are solved to optimality within the runtime limit.

The computational results discussed below use the definition described in Section 2.1 to compute $m^\omega_j(k)$ for $j \in J$ and $\omega, k \in \Omega$. An easier way to compute $m^\omega_j(k)$ is to let $m^\omega_j(k) = \maximize\{\sum_{i \in I} \xi^\omega_i y_{ij} | \sum_{i \in I} \xi^k_i y_{ij} \leq t_j\}$, i.e., ignoring the cardinality constraint in (5). This computation takes less time to compute the big-M coefficients, but lead to larger big-M coefficients. Computational results for this big-M coefficients based implementation of (CAP) are presented in Appendix.
E. Comparing results from this weaker upper bound with those in Table 1, we see the computational trade-offs resulting from using the weaker upper bound. For easier problems, the average total solution times (the sum of the average time for the big-M coefficients and the branch-and-cut algorithm) are less for the model with a weaker big-M. However, solution times for harder problems improve significantly with the strengthened big-M computation.

5.3. Computational Results for the Branch-and-Cut Algorithm for (CAP)
We now discuss the benefits of adding the valid inequalities proposed in Section 3 to the branch-and-cut algorithm when solving (CAP). The performance of the following four variants is compared:

- CPX: refers to using the branch-and-cut algorithm as implemented in CPLEX to solve (IP) of (CAP).
- Cover-1: refers to adding the single lifted cover inequalities to the branch-and-cut algorithm (Algorithm 5) for solving (IP) of (CAP). They are obtained by ignoring the cardinality constraint in the coefficient calculation procedures.
- Cover-2: refers to adding the lifted cover inequalities (18) to the branch-and-cut algorithm (Algorithm 5) for solving (IP) of (CAP).
- Cover-G: refers to adding the global lifted cover inequalities (25) to the branch-and-cut algorithm (Algorithm 5) for solving (IP) of (CAP).

Table 1 reports the average time for the big-M coefficient computations, the cut generation time, the branch-and-cut algorithm time, the average number of nodes, the average number of cuts, and the number of instances solved to optimality for the five generated instances. First we note from Table 1 that the problems become increasingly difficult as the value of $\epsilon$ reduces. A possible reason is that for these problems it is more difficult to find a feasible solution satisfying chance constraint. Note that for $\epsilon = 0.06$ to combinatorially explore chance constraint satisfaction for a 500 scenario problem we can violate 30 out of 500 scenarios. This suggests the possibility of requiring a large number of nodes in the branch-and-bound tree in proving infeasibility.

We see from Table 1 that adding the single cover and lifted cover inequalities reduce the average time for the branch-and-cut algorithm by about 55%. This decrease in the computation time can be associated with the reduction in the number of nodes explored in the branch-and-cut algorithm. For $\epsilon = 0.08$ and $N = 1500$, adding the single and lifted cover inequalities can solve all instances to optimality within the runtime limit, whereas, CPX can only solve four of the five instances to optimality. We also observe that for $\epsilon = 0.06$, most of the instances cannot be solved within the runtime limit by all variants. It seems that this level of chance requirement requires a pseudo OR, i.e., the original model for assigning 27 surgeries to the eight operating rooms with $\epsilon = 0.06$ is infeasible. It makes it hard to decide how many and which surgeries are assigned to the pseudo OR while
Table 1  The average CPU time (in seconds) for strengthened big-M coefficients (AvT-M), branch-and-cut algorithm (AvT-B&C) and valid cut generation (AvT-cut), the average number of nodes (# of nodes) and cuts (# of cuts), and the number of solved instances from the five instances (solved) for (CAP) are reported.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>N</th>
<th>approach</th>
<th>AvT-M</th>
<th>AvT-B&amp;C</th>
<th>AvT-cut</th>
<th># of nodes</th>
<th># of cuts</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>500</td>
<td>CPX</td>
<td>165.0</td>
<td>52.8</td>
<td>-</td>
<td>1,725</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>165.0</td>
<td>33.3</td>
<td>1.7</td>
<td>1,446</td>
<td>283</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>165.0</td>
<td>47.1</td>
<td>14.7</td>
<td>1,076</td>
<td>300</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>165.0</td>
<td>65.5</td>
<td>8.6</td>
<td>1,959</td>
<td>9</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>641.1</td>
<td>135.8</td>
<td>-</td>
<td>1,827</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>641.1</td>
<td>66.5</td>
<td>4.4</td>
<td>1,557</td>
<td>348</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>641.1</td>
<td>109.8</td>
<td>32.6</td>
<td>1,846</td>
<td>345</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>641.1</td>
<td>219.5</td>
<td>37.7</td>
<td>2,847</td>
<td>9</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>1,439.3</td>
<td>781.4</td>
<td>-</td>
<td>5,094</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>659.4</td>
<td>10.9</td>
<td>10,788</td>
<td>563</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>502.4</td>
<td>29.2</td>
<td>5,832</td>
<td>320</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>739.5</td>
<td>42.8</td>
<td>3,715</td>
<td>12</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>CPX</td>
<td>165.0</td>
<td>140.7</td>
<td>-</td>
<td>3,477</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>165.0</td>
<td>122.5</td>
<td>1.2</td>
<td>4,638</td>
<td>210</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>165.0</td>
<td>126.0</td>
<td>10.7</td>
<td>3,641</td>
<td>224</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>165.0</td>
<td>136.7</td>
<td>10.6</td>
<td>2,926</td>
<td>12</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>1,439.3</td>
<td>523.5</td>
<td>-</td>
<td>10,788</td>
<td>563</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>329.2</td>
<td>4.3</td>
<td>5,919</td>
<td>346</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>305.3</td>
<td>29.2</td>
<td>5,832</td>
<td>320</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>481.0</td>
<td>42.8</td>
<td>5,669</td>
<td>12</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>1,439.3</td>
<td>1,888.9</td>
<td>-</td>
<td>10,788</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>995.7</td>
<td>11.7</td>
<td>9,983</td>
<td>657</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>983.6</td>
<td>95.6</td>
<td>7,439</td>
<td>689</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>1,713.0</td>
<td>140.4</td>
<td>8,831</td>
<td>14</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>CPX</td>
<td>165.0</td>
<td>816.6</td>
<td>-</td>
<td>29,710</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>165.0</td>
<td>470.5</td>
<td>1.1</td>
<td>18,226</td>
<td>192</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>165.0</td>
<td>360.2</td>
<td>9.1</td>
<td>14,614</td>
<td>201</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>165.0</td>
<td>809.9</td>
<td>10.1</td>
<td>28,151</td>
<td>10</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>1,439.3</td>
<td>2,375.7</td>
<td>-</td>
<td>31,959</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>2,024.8</td>
<td>4.3</td>
<td>29,594</td>
<td>307</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>1,791.4</td>
<td>25.2</td>
<td>22,455</td>
<td>284</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>2,166.4</td>
<td>37.4</td>
<td>28,596</td>
<td>9</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>1,439.3</td>
<td>4,600.4</td>
<td>[0.03]</td>
<td>65,740</td>
<td>-</td>
<td>4/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>3,095.5</td>
<td>[0.07]</td>
<td>76,573</td>
<td>402</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>7,072.9</td>
<td>[0.44]</td>
<td>69,798</td>
<td>413</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>3,650.7</td>
<td>[0.43]</td>
<td>54,969</td>
<td>10</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>CPX</td>
<td>165.0</td>
<td>32,178.1</td>
<td>[0.11]</td>
<td>1,588,803</td>
<td>-</td>
<td>1/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>165.0</td>
<td>18,923.0</td>
<td>[0.07]</td>
<td>1,296,583</td>
<td>184</td>
<td>1/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>165.0</td>
<td>20,497.6</td>
<td>[0.09]</td>
<td>1,780,324</td>
<td>193</td>
<td>1/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>165.0</td>
<td>16,313.2</td>
<td>[0.11]</td>
<td>1,607,736</td>
<td>3</td>
<td>1/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>1,439.3</td>
<td>0.19</td>
<td>-</td>
<td>558,891</td>
<td>-</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>0.19</td>
<td>32.2</td>
<td>514,576</td>
<td>288</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>0.17</td>
<td>27.4</td>
<td>531,292</td>
<td>313</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>0.19</td>
<td>37.1</td>
<td>562,600</td>
<td>10</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>1,439.3</td>
<td>0.25</td>
<td>-</td>
<td>267,632</td>
<td>-</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,439.3</td>
<td>0.22</td>
<td>21.6</td>
<td>216,724</td>
<td>456</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,439.3</td>
<td>0.25</td>
<td>54.9</td>
<td>247,320</td>
<td>431</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,439.3</td>
<td>0.31</td>
<td>77.5</td>
<td>258,991</td>
<td>6</td>
<td>0/5</td>
</tr>
</tbody>
</table>

"-" in column of AvT-Cut and # of cuts indicates that no valid cut proposed in this paper is added.

"[. . .]" in column of AvT-B&C means the average optimality gap for instances that cannot be solved to optimality within 10 hours time limit.

"*" in column of AvT-B&C means that AvT-B&C is the average time for the solved instances by CPX plus the average time for the other instances.
satisfying the chance constraint with \( \varepsilon = 0.06 \), and minimizing the total cost. Nevertheless, for these problems, the use of Cover-1 and Cover-2 result in a slightly smaller average optimality gap for most instances at termination. The results also show that Cover-2 has a better performance than Cover-1 in terms of the average time for the 1,500 scenario instances \((\varepsilon = 0.1, 0.08, 0.06)\). We find that the big-M computation time is significant for the less difficult instances \((\varepsilon = 0.12, 0.10)\). However, for the difficult instances \((\varepsilon = 0.08)\), the time required in the branch-and-cut algorithm dominates. The benefits of adding Cover-1 and Cover-2 inequalities are more apparent for these instances, and here the use of Cover-2 saves computation time over Cover-1. For the easier problems \((\varepsilon = 0.10, 0.12)\), we observe that typically the number of nodes in the branch-and-cut tree reduces due to the addition of Cover-2 inequalities. However, it does not always translate in a significant reduction of the solution time, and occasionally there is a modest increase in the solution time. Overall, adding Cover-2 inequalities outperforms other variants and yields a more stable performance for most instances.

The use of Cover-G yielded an unfavorable performance for easier instances \((\varepsilon \geq 0.08)\). However, for the hardest instance \((N=500, \varepsilon = 0.06)\) solved in our implementation, the use of Cover-G gives a slightly better performance when compared with Cover-1 and Cover-2. For some instances, it reduced the number of nodes significantly, while for other instances the number of nodes increased. Even for the hardest solved instance \((\varepsilon = 0.08, N = 1,500)\), which took fewer number of nodes \((54,969 \text{ versus } 69,798)\) when compared to Cover-2 variant, this reduction did not translate into a reduction in the overall solution time \((28,248 \text{ versus } 28,014 \text{ seconds})\). It can be surmised that the linear programming relaxation problems resulting from the addition of these cuts are more time consuming to solve, hence offsetting the benefits from the reduction in branch-and-bound nodes. There are several instances where the use of Cover-G increased the number of nodes. This may be because the addition of these inequalities may be yielding a significantly different node selection path within CPLEX.

5.4. Computational Results for (DR-CAP)

We implemented Algorithm 1 to solve the semi-infinite reformulation \((8)\) of (DR-CAP). Using the sample average distribution, we let \( \hat{q} := \hat{q} \) (Corollary 1) for the big-M calculations in \((8)\). For \((18)\), we set the coefficient \( \gamma \) as \( \delta_{\hat{q}1} \) (Theorem 4). The following variants of Algorithm 1 are considered:

- CPX: refers to using the branch-and-cut algorithm with probability cuts (Algorithm 1) to solve (SIP) of (DR-CAP) without any valid inequalities proposed in this paper.
- Cover-1: refers to adding the Cover-1 inequalities from (CAP) to the branch-and-cut algorithm with probability cuts (Algorithm 1).
- Cover-2: refers to adding the valid inequalities \((18)\) to the branch-and-cut algorithm with probability cuts (Algorithm 1).
Table 2 The average CPU time (in seconds) for the branch-and-cut algorithm with probability cuts (AvT-B&CP), valid cut generation (AvT-cut) and distribution separation problem (AvT-SP), the average number of nodes (# of nodes), valid cuts (# of cuts) and probability and feasibility cuts (# of p&f-cuts), and the number of solved instances from the five instances (solved) for (DR-CAP) are reported.

<table>
<thead>
<tr>
<th>η</th>
<th>N</th>
<th>approach</th>
<th>AvT-B&amp;CP</th>
<th>AvT-cut</th>
<th>AvT-SP</th>
<th># of nodes</th>
<th># of cuts</th>
<th># of p&amp;f-cuts</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>500</td>
<td>CPX</td>
<td>272.7</td>
<td>61.9</td>
<td>6,277</td>
<td>–</td>
<td>3</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>144.0</td>
<td>54.4</td>
<td>3,589</td>
<td>410</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>147.0</td>
<td>45.9</td>
<td>3,379</td>
<td>250</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>889.6</td>
<td>273.9</td>
<td>9,476</td>
<td>–</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>728.6</td>
<td>7.2</td>
<td>7,274</td>
<td>553</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>723.5</td>
<td>32.1</td>
<td>7,606</td>
<td>349</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>3,051.7</td>
<td>880.9</td>
<td>14,650</td>
<td>–</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>2,956.2</td>
<td>7.2</td>
<td>7,794</td>
<td>644</td>
<td>4</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>1,658.0</td>
<td>96.0</td>
<td>7,006</td>
<td>716</td>
<td>2</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>500</td>
<td>CPX</td>
<td>648.6</td>
<td>101.0</td>
<td>18,426</td>
<td>–</td>
<td>18</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>290.5</td>
<td>12.4</td>
<td>8,696</td>
<td>226</td>
<td>12</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>319.5</td>
<td>10.8</td>
<td>12,446</td>
<td>250</td>
<td>14</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>1,390.0</td>
<td>403.4</td>
<td>12,095</td>
<td>–</td>
<td>12</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>1,021.5</td>
<td>5.6</td>
<td>346.5</td>
<td>447</td>
<td>8</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>884.0</td>
<td>33.0</td>
<td>397.5</td>
<td>373</td>
<td>9</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>4,957.9</td>
<td>1080.9</td>
<td>37,824</td>
<td>–</td>
<td>12</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>4,003.9</td>
<td>13.6</td>
<td>39,783</td>
<td>706</td>
<td>9</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>4,598.5</td>
<td>100.9</td>
<td>38,227</td>
<td>759</td>
<td>14</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>500</td>
<td>CPX</td>
<td>826.9</td>
<td>104.0</td>
<td>31,989</td>
<td>–</td>
<td>34</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>501.8</td>
<td>1.4</td>
<td>20,067</td>
<td>233</td>
<td>28</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>775.8</td>
<td>11.1</td>
<td>32,536</td>
<td>247</td>
<td>30</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>2,987.3</td>
<td>502.2</td>
<td>47,221</td>
<td>–</td>
<td>30</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>3,173.8</td>
<td>6.2</td>
<td>42,202</td>
<td>482</td>
<td>30</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>2,173.9</td>
<td>36.3</td>
<td>33,143</td>
<td>414</td>
<td>28</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>8,091.2</td>
<td>1,268.2</td>
<td>61,039</td>
<td>–</td>
<td>28</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>8,088.4</td>
<td>12.3</td>
<td>63,029</td>
<td>647</td>
<td>27</td>
<td>5/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>4,962.4</td>
<td>98.9</td>
<td>44,377</td>
<td>716</td>
<td>26</td>
<td>5/5</td>
<td></td>
</tr>
</tbody>
</table>

“–” in column of AvT-cut and # of cuts indicates that no valid cut proposed in this paper is added.

We solved the instances generated in Section 5.3 with the Wasserstein set $P_W$ as the ambiguity set to evaluate the performance of the variants. The sample size $N \in \{500, 1000, 1500\}$, the Wasserstein set radius parameter $\eta \in \{0.1, 0.5, 1\}$, and the level of chance satisfaction $\varepsilon = 0.1$ are used in these instances. Table 2 reports the average time for the branch-and-cut algorithm with probability cuts, the cut generation, the average number of nodes, the average number of cuts, and the number of instances that are solved to optimality from the five generated instances.

Similar to the case of (CAP), the results in Table 2 show that Cover-2 yields a significant improvement over CPX and Cover-1 in two of the three harder instance sets ($\eta = 0.1$, and $\eta = 1$, $N = 1500$). However, the average performance of Cover-1 is better for the ($\eta = 0.5$, $N = 1500$) instances. A comparison of the results in Table 1 and 2 shows that the time required to solve (DR-CAP) is approximately (at most) four times the time required to solve (CAP). Moreover, the average number of probability and feasibility cuts required to solve these models is typically less than 30, though this...
number grows with the Wasserstein radius. This is expected since with increasing radius, the Wasserstein ambiguity set increases in size, resulting in more solutions being generated in the algorithm that are infeasible with respect to the ambiguity set. The average number of nodes required to solve the models also increases with the Wasserstein radius (up to 5 times). Note that the branch-and-cut tree from the incumbent problem is used to warm-start the solution of the new problem after a probability cut is added.

5.5. Performance of Big-M Improvements from Ambiguity Set Information

The results in Table 2 were obtained by using the nominal distribution to compute the big-M coefficients. We now discuss our computational experience with the possibility of big-M tightening due to Theorem 1 and Corollary 1. While we found that the solution time required by the linear programs in Theorem 1 is not justified, we did find computational value in using Corollary 1 as part of our implementation. This is particularly true for the harder problems. In this section, we present the results for the harder problems that are generated for $\eta = 1$ and $N = 1500$. Five instances are considered. These instances are labeled as $N - \#$, where $\#$ denotes the instance number. We compare the performance of the following approaches:

- CPX: is described in Section 5.4.
- CPX-UM: refers to using new $\hat{q}$ as valid inequalities and adding these inequalities to CPX.

For CPX-UM, we update $\hat{q}$ defined in Corollary 1 as new $\{p_\omega\}_{\omega \in \Omega}$ becomes available in the probability cuts. We set $q = \hat{q}$ and add constraints (8c) as valid inequalities. We needed to do this because CPLEX does not allow for changing in the coefficients of the original constraints once a branch-and-bound tree is built. We need to keep the original branch-and-bound tree when solving the problem. For each $j$, multiple violated inequalities might be found. We only added the inequality with the most violated value to the branch-and-cut tree. In the current implementation, it is done only once when a new probability distribution becomes available for each $j$. Table 3 reports the solution time for the branch-and-cut algorithm with probability cuts and the separation problem, the number of nodes, the number of probability and feasible cuts. Note that the time for the valid inequality generation was negligible, and therefore not included in this table.

Specially, for the model with the largest value of $\eta$ ($\eta = 1$), where Algorithm 1 generates many probability cuts, we observe from Table 3 that CPX-UM provides better performance than CPX in the solution time in three of the five instances. The average solution time is decreased by about 800 seconds. The solution time is significantly lower for one instance (1500-1), whereas that for other instances is similar. Compared with CPX, CPX-UM has a reduced total number of nodes for three instances, whereas the number of nodes increases in the other two instances. The increase/decrease in the number of nodes does not necessarily imply a corresponding increase/decrease in solution time.
Table 3 The CPU time (in seconds) for branch-and-cut algorithm with probability cuts (time), and distribution separation problem (time-SP), the number of nodes (# of nodes), and probability and feasibility cuts (# of p&f-cuts) for (DR-CAP) are reported.

<table>
<thead>
<tr>
<th>instance</th>
<th>CPX</th>
<th>CPX-UM</th>
<th>CPX</th>
<th>CPX-UM</th>
<th>CPX</th>
<th>CPX-UM</th>
<th>CPX</th>
<th>CPX-UM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500-1</td>
<td>12,350.4</td>
<td>8,354.3</td>
<td>1,305.1</td>
<td>1,419.1</td>
<td>63,691</td>
<td>93,617</td>
<td>38</td>
<td>44</td>
</tr>
<tr>
<td>1500-2</td>
<td>9,490.7</td>
<td>8,710.0</td>
<td>1,515.7</td>
<td>1,309.2</td>
<td>63,699</td>
<td>85,382</td>
<td>30</td>
<td>24</td>
</tr>
<tr>
<td>1500-3</td>
<td>5,331.2</td>
<td>5,564.0</td>
<td>1,250.1</td>
<td>1,017.6</td>
<td>55,893</td>
<td>36,640</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>1500-4</td>
<td>5,453.2</td>
<td>5,272.6</td>
<td>898.2</td>
<td>885.2</td>
<td>39,918</td>
<td>25,362</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>1500-5</td>
<td>7,830.7</td>
<td>8,627.2</td>
<td>1,372.0</td>
<td>1,560.4</td>
<td>81,992</td>
<td>71,336</td>
<td>38</td>
<td>32</td>
</tr>
<tr>
<td>Average</td>
<td>8,091.2</td>
<td>7,305.6</td>
<td>1,268.2</td>
<td>1,238.3</td>
<td>61,039</td>
<td>62,467</td>
<td>28</td>
<td>25</td>
</tr>
</tbody>
</table>

This may be because the node linear programs may vary in difficulty. We could not find a setting for combining the valid inequalities discussed in this section with (18) to improve the performance of Cover-2 in Section 5.4. We attribute this to the fact that CPLEX does not allow us to change the coefficients of the original data with the progression of the algorithm.

5.6. Out-of-Sample Performance of (DR-CAP) Solutions

The chance constraints used to specify (CAP) and (DR-CAP) are generated using a finite number of samples drawn from a probability distribution. The goal of this section is to evaluate the ‘true chance satisfaction’ of the solution generated from this finite sample approximation. For this purpose, the integer solutions obtained from (CAP) and (DR-CAP) were evaluated using a large number (1,500,000) of scenarios generated from the log-normal distribution. We used five instances each for the sample sizes $N \in \{500,1000,1500\}$ for the (CAP) and (DR-CAP) solutions. The (DR-CAP) solutions were generated using the Wasserstein radius parameter $\eta \in \{0.1,0.5,1\}$. All evaluations were performed for $\varepsilon = 0.1$ in the chance constraint model. Table 4 gives the average total cost, the average overtime probability, the worst-case overtime probability, the average overtime (minutes), and 85%, 95%, 99% overtime quantiles (minutes) for (CAP) and (DR-CAP) solutions.

The results in Table 4 show that the average and worst-case out-of-sample overtime probability decrease with increasing sample size in (CAP) and the radius of the Wasserstein set ($\eta$) in (DR-CAP). The same is observed for the average overtime, and the overtime 85% and 95% quantiles. Consequently, using the largest instance ($N = 1500$) and/or larger $\eta$ solutions are viable alternatives when out-of-sample chance constraint satisfaction is of concern. We observe that the decrease in the worst-case out-of-sample chance constraint satisfaction probability is more modest with increasing sample sizes. For example, the solutions from the instances with $N = 1000$ give a worst probability of 0.122, and the instances with $N = 1500$ have a worst probability of 0.117. However, this worst-case out-of-sample chance constraint satisfaction probability decreases more significantly with increasing $\eta$. For example, the instances with $N = 1000$ and $\eta = 0.1$ have the worst-case out-of-sample probability of 0.122, and the instances with $N = 1000$ and $\eta = 0.5$ have the worst-case probability of 0.088,
### Table 4

<table>
<thead>
<tr>
<th>$\eta$</th>
<th>$N$</th>
<th>model</th>
<th>Avg-cost</th>
<th>Avg-prob</th>
<th>Worst-prob</th>
<th>Avg-over</th>
<th>85%</th>
<th>95%</th>
<th>99%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>500</td>
<td>(CAP)</td>
<td>69.9</td>
<td>0.070</td>
<td>0.122</td>
<td>6.1</td>
<td>0.0</td>
<td>36.4</td>
<td>150.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>70.3</td>
<td>0.068</td>
<td>0.122</td>
<td>6.0</td>
<td>0.0</td>
<td>36.8</td>
<td>147.4</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>(CAP)</td>
<td>70.2</td>
<td>0.069</td>
<td>0.122</td>
<td>6.1</td>
<td>0.0</td>
<td>37.9</td>
<td>150.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>70.7</td>
<td>0.066</td>
<td>0.122</td>
<td>5.8</td>
<td>0.0</td>
<td>33.8</td>
<td>148.5</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>(CAP)</td>
<td>70.7</td>
<td>0.067</td>
<td>0.117</td>
<td>5.8</td>
<td>0.0</td>
<td>34.5</td>
<td>148.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>71.0</td>
<td>0.067</td>
<td>0.117</td>
<td>5.9</td>
<td>0.0</td>
<td>35.6</td>
<td>147.4</td>
</tr>
<tr>
<td>0.5</td>
<td>500</td>
<td>(CAP)</td>
<td>69.9</td>
<td>0.070</td>
<td>0.122</td>
<td>6.1</td>
<td>0.0</td>
<td>36.4</td>
<td>150.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>71.7</td>
<td>0.066</td>
<td>0.121</td>
<td>5.8</td>
<td>0.0</td>
<td>32.3</td>
<td>147.8</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>(CAP)</td>
<td>70.2</td>
<td>0.069</td>
<td>0.122</td>
<td>6.1</td>
<td>0.0</td>
<td>37.9</td>
<td>150.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>71.2</td>
<td>0.065</td>
<td>0.088</td>
<td>5.6</td>
<td>0.0</td>
<td>31.9</td>
<td>149.3</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>(CAP)</td>
<td>70.7</td>
<td>0.067</td>
<td>0.117</td>
<td>5.8</td>
<td>0.0</td>
<td>34.5</td>
<td>148.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>72.0</td>
<td>0.065</td>
<td>0.096</td>
<td>5.6</td>
<td>0.0</td>
<td>29.6</td>
<td>148.1</td>
</tr>
<tr>
<td>1</td>
<td>500</td>
<td>(CAP)</td>
<td>69.9</td>
<td>0.070</td>
<td>0.122</td>
<td>6.1</td>
<td>0.0</td>
<td>36.4</td>
<td>150.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>72.8</td>
<td>0.065</td>
<td>0.121</td>
<td>5.6</td>
<td>0.0</td>
<td>28.9</td>
<td>148.1</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>(CAP)</td>
<td>70.2</td>
<td>0.069</td>
<td>0.122</td>
<td>6.1</td>
<td>0.0</td>
<td>37.9</td>
<td>150.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>73.1</td>
<td>0.064</td>
<td>0.089</td>
<td>5.5</td>
<td>0.0</td>
<td>26.6</td>
<td>149.3</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>(CAP)</td>
<td>70.7</td>
<td>0.067</td>
<td>0.117</td>
<td>5.8</td>
<td>0.0</td>
<td>34.5</td>
<td>148.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DR-CAP)</td>
<td>73.3</td>
<td>0.064</td>
<td>0.082</td>
<td>5.4</td>
<td>0.0</td>
<td>24.4</td>
<td>148.1</td>
</tr>
</tbody>
</table>

i.e., in this case the solutions generated in all the instances satisfy the chance constraint with probability $0.1$. The solutions for the DR-CAP models that satisfy the chance constraint have a modest increase in cost. This cost increases from 70.2 in the (CAP) model to 71.2 in the (DR-CAP) model when using $N = 1000$ and $\eta = 0.5$. Similar observations are made for (CAP) and (DR-CAP) problem instances with $N = 1500$. It is also interesting to observe that the worst-case probability for problem instances with $N = 500$ did not change significantly ($0.122$, $0.121$, $0.121$) for $\eta = 0.1, 0.5$ and $1.0$, despite the solutions becoming costlier. Consequently, increasing both the sample size and the size of the ambiguity set may be important to ensure the worst-case probability satisfaction. However, it is important to note that for the chance constraint problems computational cost increases rapidly with the sample size, while the increase in the computational cost for the (DR-CAP) models is modest (only a constant factor).

### 6. Concluding Remarks

The use of big-M calculations and strong inequalities developed in this paper resulted in the chance-constrained assignment and distributionally robust chance-constrained assignment model solutions with a modest number ($N = 1500$) of scenarios. These models remain difficult to solve when they are infeasible or nearly feasible. The solution time for the models grows rapidly with increasing sample size. However, the solution time for the distributionally robust chance-constrained models appears to be only a constant factor of the time required to solve the chance constraint version. The use of a modest number of samples ($N = 1000$) and an appropriate choice of the radius of the Wasserstein set...
provide a solution that achieves an out-of-sample chance satisfaction. This out-of-sample performance is not possible for the solutions generated from solving the chance constraint problem specified using a modest number of samples. The use of the Wasserstein ambiguity set allows us to have the true probability distribution of the random parameters with a greater probability.

Acknowledgments
This research of the first two authors were partially supported by the National Natural Science Foundation of China (NSFC) grants 71432002, 91746210. The research of the last author was partially supported by the NSF grant CMMI-1763035. This paper also benefited from a discussion with Professor Simge Küçükyavuz and Hamed Rahimian.

References


Shapiro, A., Dentcheva, D., Ruszczyński, A., 2009. Lectures on stochastic programming: modeling and theory. SIAM.


The inequality (it is also valid for conv \(H\). The set \(H\) is finite if and only if \(H\) is a polyhedron. Thus, \(H\) is a polyhedron.

A.2. Proof of Theorem 1

Let \(\bar{M}_j\) be an optimal solution of \((\text{DR-CAP})\). Let \(y_{ij}\) be an optimal solution of the above maximization problem, there exist at least one \(k'\) such that 
\[
\sum_{i \in I} \xi^k_{ji} y_{ij} > t_j, \quad \text{for} \quad k \in \{k_1, \ldots, k_q\}. 
\]

Otherwise, we have \(\sum_{i \in I} \xi^k_{ji} y_{ij} \leq t_j, \) for \(k \in \{k_1, \ldots, k_q\}.\) Since \(\sum_{j=1}^q p_j > \varepsilon,\) the inequality \(\mathbb{P}\left\{\sum_{i \in I} \xi_{ji} y_{ij} \leq t_j\right\} \geq 1 - \varepsilon\) is violated. This is a contradiction. Therefore, \(y_{ij}\) is a feasible solution of \((5)\) with \(k = k'.\) Then \(m^*_{ij}(k_{q+1}) \geq \bar{M}_j, \) \(m^*_{ij}(k_{q+1})\) is an upper bound for \(\bar{M}_j.\) \(\square\)

A.3. Proof of Proposition 2

Let \((y, z)\) be a feasible solution of the relaxation problem of the binary bilinear reformulation of \((\text{DR-CAP})\). We have 
\[
\sum_{i \in I} \xi_{ji} y_{ij} (z^*_{ji} - 1) - m^*_{ij}(k_q)(z^*_{ji} - 1) = (z^*_{ji} - 1)(\sum_{i \in I} \xi_{ji} y_{ij} - m^*_{ij}(k_q)) \geq 0. 
\]

Consequently, 
\[
\sum_{i \in I} \xi_{ji} y_{ij} + m^*_{ij}(k_q)(z^*_{ji} - 1) \leq \sum_{i \in I} \xi_{ji} y_{ij} z^*_{ji} \leq m^*_{ij}(\omega) z^*_{ji} \] holds. Therefore, \((y, z)\) is a feasible solution of the relaxation problem of \((8)\). The proof can be similarly extend to \((\text{CAP})\). \(\square\)

A.4. Proof of Proposition 3

The set \(\mathcal{H} = \bigcap_{j \in J} \{(y, z) \mid (y_j, z_j) \in \mathcal{G}_j\}\) implies that \(\mathcal{H} \subseteq \mathcal{G}_j\). Thus, if an inequality is valid for \(\text{conv}(\mathcal{G}_j),\) then it is also valid for \(\text{conv}(\mathcal{H}).\) If an inequality is facet-defining for \(\text{conv}(\mathcal{G}_j),\) then there exits \(|I| + N\) affinely independent points that satisfy this inequality at equality. Because this inequality does not have coefficients with respect to a pair of \((y_{j_1}, z_{j_1})\) for \(j_1 \in J\) and \(j_1 \neq j,\) we can extend the \(|I| + N\) affinely independent points to a set of \(|I| \times |J| + |J| \times N\) affinely independent points by appropriately setting the values of \((y_{j_1}, z_{j_1})\) for each \(j_1 \in J\) and \(j_1 \neq j.\) \(\square\)

A.5. Proof of Proposition 4

The inequality \((12)\) is valid for \((11)\) based on the definition of \(\mathcal{C}.\)

Consider the following \(|\mathcal{C} \setminus \mathcal{D}|\) feasible points of \((11):\) for \(k \in \mathcal{C} \setminus \mathcal{D},\) set \(y_{ij} = 1, \forall i \in \mathcal{C} \setminus \{\mathcal{D} \cup k\}, \) \(y_{ij} = 0, \forall i \in k \cup (I \cap \mathcal{C}),\) and \(y_{ij} = 1, \forall i \in \mathcal{D} ;\) These \(|\mathcal{C} \setminus \mathcal{D}|\) points are affinely independent and satisfy \((12)\) at equality. When \(|\mathcal{C}| \leq \rho_j + 1,\) these \(|\mathcal{C} \setminus \mathcal{D}|\) points are feasible. \(\square\)
A.6. Proof of Lemma 1

Suppose that there exists \( \tilde{y}_j \) that serves as a member of the set \( \{ y_j \in \{0,1\}^{|\mathcal{I}|} | \sum_{i \in \mathcal{I}} \xi_i^* y_{ij} \leq m_i^* (\omega), y_{ij} = 1, \forall i \in \mathcal{D} \} \) such that \( \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \tilde{y}_{ij} \leq |\mathcal{C} \setminus \mathcal{D}| - 1 \) and \( \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \tilde{y}_{ij} + \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \alpha_i \tilde{y}_{ij} > |\mathcal{C} \setminus \mathcal{D}| - 1 \). Let \( r := \max \{ k | \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \tilde{y}_{ij} + \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \alpha_i \tilde{y}_{ij} \leq |\mathcal{C} \setminus \mathcal{D}| - 1 \} \). We have

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \tilde{y}_{ij} + \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \alpha_i \tilde{y}_{ij} = \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \tilde{y}_{ij} + \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \alpha_i \tilde{y}_{ij} + (|\mathcal{C} \setminus \mathcal{D}| - 1 - \text{obj}_{r+1}) \tilde{y}_{r+1,j} \leq |\mathcal{C} \setminus \mathcal{D}| - 1,
\]

which is a contradiction. Thus, (13) is valid for (15).

Consider the following \(|\mathcal{I} \setminus \mathcal{D}| \) feasible points of (15): for \( k \in \mathcal{C} \setminus \mathcal{D} \), set \( y_{ij} = 1, \forall i \in \mathcal{C} \setminus \mathcal{D} \); for \( k = 1, \ldots, |\mathcal{I} \setminus \mathcal{C}| \), set \( y_{ij} = 0, \forall i \in \{ \pi_{k+1}, \ldots, \pi_{|\mathcal{I} \setminus \mathcal{C}|} \} \), and \( (y_{ij})_{i \in \mathcal{C} \setminus \mathcal{D} \cup \{ \pi_{k+1}, \ldots, \pi_{|\mathcal{I} \setminus \mathcal{C}|} \}} \) are the optimal solutions of (14). All these points have \( y_{ij} = 1, \forall i \in \mathcal{D} \). When \( |\mathcal{C}| \leq \rho_j + 1 \), the above \(|\mathcal{I} \setminus \mathcal{D}| \) points are feasible, satisfy (13) at equality and are affinely independent. \( \square \)

A.7. Proof of Lemma 2

Suppose that we have \( \tilde{y}_j \in \Omega_{\rho_j} \) that violates (16). \( \kappa \) can be partitioned into \( \mathcal{D}^0 := \{ i \in \kappa | \tilde{y}_{ij} = 0 \} \) and \( \mathcal{D}^1 := \{ i \in \kappa | \tilde{y}_{ij} = 1 \} \). We assume that the last element in the set \( \mathcal{D}^0 \) is \( \kappa_h \), where \( h \leq |\mathcal{D}| \). Then, we have

\[
\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \tilde{y}_{ij} + \sum_{i \in \mathcal{D}^0} \alpha_i \tilde{y}_{ij} > |\mathcal{C} \setminus \mathcal{D}| + \sum_{i \in \mathcal{D}^0} \beta_i - 1 - 1 = |\mathcal{C} \setminus \mathcal{D}| + |\mathcal{C} \setminus \mathcal{D}| + \sum_{i \in \mathcal{D}^0 \setminus \kappa_h} \beta_i - 1 = \text{obj}_{\rho_j} \geq \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \tilde{y}_{ij} + \sum_{i \in \mathcal{D}^0 \setminus \kappa_h} \alpha_i \tilde{y}_{ij} + \sum_{i \in \mathcal{D}^0 \setminus \kappa_h} \alpha_i \tilde{y}_{ij}.
\]

This is a contradiction. Thus, (16) is valid for \( \text{conv}(\Omega_{\rho_j}) \).

Consider the following \(|\mathcal{I}| \) feasible points of \( \text{conv}(\Omega_{\rho_j}) \); when \( y_{ij} = 1, \forall i \in \mathcal{D} \), then there exists \(|\mathcal{I} \setminus \mathcal{C}| \) feasible points that are independent and satisfy the inequality (16) at equality based on Lemma 1; for \( l \in \{ 1, \ldots, |\mathcal{D}| \} \), set \( y_j \) is the optimal solution of (17). When \( |\mathcal{C}| \leq \rho_j + 1 \), these \(|\mathcal{I}| \) points are feasible, satisfy the inequality (16) at equality and are affinely independent. \( \square \)

A.8. Proof of Theorem 2

When \( z_\omega = 1 \), (18) is valid for \( \text{conv}(\mathcal{F}_{\rho_j}) \) because of Lemma 2. When \( z_\omega = 0 \), due to the definition of \( \gamma \), (18) is also valid for \( \text{conv}(\mathcal{F}_{\rho_j}) \). Thus, (18) is valid for \( \text{conv}(\mathcal{F}_{\rho_j}) \).

Consider the following \(|\mathcal{I}| + 1 \) feasible points of \( \text{conv}(\mathcal{F}_{\rho_j}) \): when \( z_\omega = 1 \), there exists \(|\mathcal{I}| \) feasible points of \( \text{conv}(\mathcal{F}_{\rho_j}) \) that are affinely independent and satisfy (18) at equality based on the Lemma 2; when \( z_\omega = 0 \), let \( y_j \) be the optimal solution of (19). These \(|\mathcal{I}| + 1 \) feasible points satisfy (18) at equality and are affinely independent. Thus, (18) is facet-defining for \( \text{conv}(\mathcal{F}_{\rho_j}) \). \( \square \)

A.9. Proof of Theorem 3

Let

\[
\gamma = \maximize_{y_j \in \{0,1\}^{|\mathcal{I}| \setminus \mathcal{J}_j}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \alpha_i y_{ij} + \sum_{i \in \mathcal{D}} \beta_i y_{ij} - |\mathcal{C} \setminus \mathcal{D}| - \sum_{i \in \mathcal{D}} \beta_i + 1 \tag{34a}
\]

subject to \( \sum_{k \in \Omega_{l^\omega}} p_k \mathbb{I} \left( \sum_{i \in \mathcal{I}} \xi_i^* y_{ij} \leq t_j \right) \geq 1 - \varepsilon. \tag{34b} \)

\( y_j \) satisfies the chance constraint (1d) and \( z_\omega = 0 \) for computing \( \gamma \), the inequality (18) is valid for (CAP).
Let $\hat{y}_j$ be an optimal solution of (34). Then, there exists at least one $k' \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}$ such that $\sum_{i \in \Omega} \xi_{i}^k \hat{y}_{ij} \leq t_j$. Otherwise, if $\sum_{i \in \Omega} \xi_{i}^k \hat{y}_{ij} > t_j$ for all $k \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}$, then $\sum_{k \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}} p_k \mathbb{1} \left( \sum_{i \in \Omega} \xi_{i}^k \hat{y}_{ij} > t \right) > \varepsilon$, which indicates that (34b) is violated by $\hat{y}_j$. Therefore, $\hat{y}_j$ is a feasible solution of (20) for $k = k'$. We have $\delta_k^q \geq \delta_k \geq \gamma$, and (18) is a valid inequality for (CAP) when $\gamma = \delta_k^q$. \hfill \Box

A.10. Proof of Theorem 4

Let

\[
\begin{align}
\gamma &= \max_{y_j \in \{0,1\}^{|\Omega|} \cap \mathcal{J}} \sum_{i \in \Omega} y_{ij} + \sum_{i \in \Omega} \alpha_i y_{ij} + \sum_{i \in \Omega} \beta_i y_{ij} - |C \setminus D| - \sum_{i \in \Omega} \beta_i + 1 \\
\text{subject to} & \quad \sum_{p \in \mathcal{D}} \sum_{k \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}} p_k \mathbb{1} \left( \sum_{i \in \Omega} \xi_{i}^k y_{ij} \leq t_j \right) \geq 1 - \varepsilon. 
\end{align}
\]

(35a)

(35b)

$y_j$ satisfies the chance constraint (2b) and $z_{\omega} = 0$ for computing $\gamma$. (18) is valid for (DR-CAP).

Let $\hat{y}_j$ be an optimal solution of (35). Then, $\sum_{i \in \Omega} \xi_{i}^k \hat{y}_{ij} \leq t_j$ for at least one $k' \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}$. Otherwise, if $\sum_{i \in \Omega} \xi_{i}^k \hat{y}_{ij} > t_j$ for all $k \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}$, we have $\sum_{p \in \mathcal{D}} \sum_{k \in \{\tilde{k}_1, \ldots, \tilde{k}_q\}} p_k \mathbb{1} \left( \sum_{i \in \Omega} \xi_{i}^k \hat{y}_{ij} > t \right) > \varepsilon$, which indicates that (35b) is violated by $\hat{y}_j$. Therefore, $\hat{y}_j$ is a feasible solution of (20) for $k = k'$. We have $\delta_k^q \geq \delta_k \geq \gamma$, and (18) is a valid inequality for (DR-CAP) when $\gamma = \delta_k^q$. Since $\sum_{j=1}^{q} p_{\omega} \geq \sum_{j=1}^{q} \tilde{p}_{\omega} > \varepsilon$, we have $\tilde{q} \geq q^1$, which implies $\delta_k^q \geq \delta_k \geq \gamma$, and (18) is a valid inequality for (DR-CAP) when $\gamma = \delta_k^q$. \hfill \Box

A.11. Proof of Theorem 5

We first prove that for (CAP) if the coefficients are described in Theorem 5, then (25) is valid for $\text{conv} (\mathcal{G}_j)$.

For $k \in \{1, \ldots, |\Omega_0|\}$, let $(\hat{y}_j, \hat{z}_j) \in \mathcal{G}_j$. If $\hat{z}_{\omega} = 1$ for $\omega \in \Omega_k$, then (25) is valid for $\text{conv} (\mathcal{G}_j)$. Otherwise, let $\tau$ be partitioned into $\Omega_k^0 = \{\omega \in \tau | \hat{z}_{\omega} = 0\}$ and $\Omega_k^1 = \{\omega \in \tau | \hat{z}_{\omega} = 1\}$. We assume that the last element of $\Omega_k^0$ is $\tau_n$ where $h \leq |\Omega_1|$. (25) becomes $\sum_{i \in \Omega_1^0} \hat{y}_{ij} + \sum_{i \in \Omega_0^0} \alpha_i \hat{y}_{ij} + \sum_{i \in \Omega_1^0} \beta_i \hat{y}_{ij} - |C \setminus D| + \sum_{i \in \Omega_1^0} \beta_i - 1 + \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega$. Note that $\sum_{\omega \in \Omega_k^1} \beta_i - 1 + \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega = \text{obj}_{\tau n} - \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega - \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega$. Since $(\hat{y}_j, \hat{z}_j)$ satisfies (29) with $k = h$, we have $\text{obj}_{\tau n - \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega - \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega}$ for $l = 1, \ldots, |\Omega_1|$. Since $\text{obj}_{\tau n}$ is an upper bound on $\text{obj}_{\tau n - \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega - \sum_{\omega \in \Omega_k^1} \tilde{\gamma}_\omega}$ for $l = 1, \ldots, |\Omega_1|$. The proof is similarly extended to $\mathcal{G}_j$. \hfill \Box

A.12. Proof of Theorem 6

The algorithm processes a finite number of nodes as it is based on branching on a finite number of binary variables. When there exists an oracle that solve (SP) to optimality, we can obtain an optimal solution of (SP) and verify the feasibility of $(y^k, z^k)$ from (MP) to (DR-CAP). In addition, since a finite number of integer solutions are obtained from (MP), (SP) is solved finite times and the set of feasibility cuts generated in line 12 is finite. Thus, Algorithm 1 terminates in finitely many iterations. Next, we show that the cuts (32) and (33) can remove the current infeasible solution and never cut off any feasible solutions of (DR-CAP). It can be verified that (32) and (33) can remove the current infeasible solution. Also, $\sum_{\omega \in \Omega_k} p_{\omega}^k z_{\omega} \geq$
\[
\inf_{p \in \mathcal{P}} \sum_{j \in \mathcal{O}} p_j z_{jw} \geq 1 - \varepsilon. \]
Thus, (32) never cuts off any feasible solutions of (DR-CAP). We assume that \(\tilde{y}\) is a new future solution from (MP) and the corresponding set \(\hat{\mathcal{I}}_j\). Let \(y_{ij} = \tilde{y}_{ij}\), for \(i \in \mathcal{I}\). Then the feasibility cut (33) becomes \(\sum_{j \in \hat{\mathcal{I}}_j} \tilde{y}_{ij} \leq |\hat{\mathcal{I}}_j| - 1\), which is decomposed to \(\sum_{i \in \hat{\mathcal{I}}_j \cap \mathcal{I}_j} \tilde{y}_{ij} + \sum_{i \in \hat{\mathcal{I}}_j \setminus \mathcal{I}_j} \tilde{y}_{ij} \leq |\hat{\mathcal{I}}_j \cap \mathcal{I}_j| + |\hat{\mathcal{I}}_j \setminus \mathcal{I}_j| - 1 \iff \sum_{i \in \hat{\mathcal{I}}_j \setminus \mathcal{I}_j} \tilde{y}_{ij} \leq |\hat{\mathcal{I}}_j \setminus \mathcal{I}_j| - 1.

If \(\hat{\mathcal{I}}_j \subseteq \hat{\mathcal{I}}_j\), \(\tilde{y}\) is not a feasible solution, and does not satisfy the feasibility cut. Otherwise, \(\sum_{i \in \hat{\mathcal{I}}_j \setminus \mathcal{I}_j} \tilde{y}_{ij} = 0\) and \(|\hat{\mathcal{I}}_j \setminus \mathcal{I}_j| - 1 \geq 0\.

## Appendix B: Algorithm Details

### B.1. Dynamic Programming for Up-lifting Coefficient

For \(k = 1, \ldots, |\mathcal{I} \setminus \mathcal{C}|\), \(\lambda_1 = 0, \ldots, |\mathcal{C} \setminus \mathcal{D}| - 1\), and \(\lambda_2 = 0, \ldots, \rho_j - 1 - |\mathcal{D}|\), let \(A_{\xi_k}(\lambda_1, \lambda_2) = \min_{y_j \in \{0, 1\}^{\mathcal{C} \setminus \mathcal{D}}} \{ \sum_{j \in \mathcal{C} \setminus \mathcal{D}} \xi_j y_{ij} + \sum_{j \in \mathcal{C} \setminus \mathcal{D}} \xi_j y_{ij} \mid \sum_{j \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{j \in \mathcal{C} \setminus \mathcal{D}} \alpha_i y_{ij} \geq \lambda_1, \sum_{j \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{j \in \mathcal{C} \setminus \mathcal{D}} y_{ij} \leq \lambda_2 \}\) and \(l_i, t = 0, \ldots, |\mathcal{C} \setminus \mathcal{D}| - 1\) be the sum of the \(t\) smallest \(\xi_j\), \(i \in \mathcal{C} \setminus \mathcal{D}\). Algorithm 2 gives an outline of our dynamic programming framework.

### Algorithm 2: Dynamic Programming for the Lifting Coefficients

```plaintext
for \(\lambda_2 = 0, \ldots, \rho_j - 1 - |\mathcal{D}|\) do
  for \(\lambda_1 = 0, \ldots, |\mathcal{C} \setminus \mathcal{D}| - 1\) do
    if \(\lambda_1 \leq \lambda_2\) then
      \(A_{\xi_k}(\lambda_1, \lambda_2) = l_{\lambda_1}\).
    end
  end
else
  \(A_{\xi_k}(\lambda_1, \lambda_2) = +\infty\).
end
end

for \(k = 1, \ldots, |\mathcal{I} \setminus \mathcal{C}|\) do
  \(obj_{\xi_k} = \max \{ \lambda : A_{\xi_k}(\lambda_1, \rho_j - 1 - |\mathcal{D}|) \leq m_{\xi_j}(\omega) - \xi_{\lambda_k} - \sum_{i \in \mathcal{D}} \xi_i \}\), \(\alpha_{\xi_k} = |\mathcal{C} \setminus \mathcal{D}| - 1 - obj_{\xi_k}\).
  for \(\lambda_2 = 0, \ldots, \rho_j - 1 - |\mathcal{D}|\) do
    for \(\lambda_1 = 0, \ldots, |\mathcal{C} \setminus \mathcal{D}| - 1\) do
      if \(\lambda_1 \geq \alpha_{\xi_k}\) and \(\lambda_2 \geq 1\) then
        \(A_{\xi_{k+1}}(\lambda_1, \lambda_2) = \min \{ A_{\xi_k}(\lambda_1, \lambda_2), A_{\xi_k}(\lambda_1 - \alpha_{\xi_k}, \lambda_2 - 1) + \xi_{\lambda_k} \}\).
      end
      else
        \(A_{\xi_{k+1}}(\lambda_1, \lambda_2) = A_{\xi_k}(\lambda_1, \lambda_2)\).
      end
    end
  end
end
```

### B.2. Separation Heuristic for (18)

Algorithm 3 gives an overview of separation heuristic for (18).

### B.3. Separation Heuristic for (25)

Algorithm 4 gives an overview of separation heuristic for (25).
Algorithm 3: Separation Heuristic for (18)

1 Given the LP relaxation optimal solution ($\tilde{y}, \tilde{z}$).

2 for $j = 1, \ldots, |J|$ do

3 for $\omega = 1, \ldots, N$ do

4 if $\tilde{z}_{j\omega} = 1$ then

5 Sort $\tilde{y}_j: \tilde{y}_{i_1j} \geq \ldots \geq \tilde{y}_{i_oj}$. Let $C = \{i_1, \ldots, i_o\}$ where $o \leq |I|$ is a smallest number such that $C$ is a cover.

6 Delete elements from the tail of $C$ to get a minimal cover $C$.

7 Let $D = \{i \in C : \tilde{y}_{ij} = 1\}$ and $\mathcal{I}_0 = \{i \in I \setminus C | \tilde{y}_{ij} = 0\}$. Calculate $\alpha_i$ for $i \in I \setminus (C \cup \mathcal{I}_0)$.

8 if $\sum_{i \in C \setminus D} \tilde{y}_{ij} + \sum_{i \in I \setminus (C \cup \mathcal{I}_0)} \alpha_i \tilde{y}_{ij} > |C \setminus D| - 1$ then

9 Calculate $\beta_i$ for $i \in D$, and $\alpha_i$ for $i \in \mathcal{I}_0$.

10 Calculate $\delta_k, k \in \Omega \setminus \omega$, set $\gamma = \delta_{q_1}$ for (CAP), $\gamma = \delta_{q_1}$ for (DR-CAP). Obtain the inequality (18).

end

end

end

Algorithm 4: Separation Heuristic for (25)

1 Given the LP relaxation optimal solution ($\tilde{y}, \tilde{z}$).

2 for $j = 1, \ldots, |J|$ do

3 Let $\Omega_1 = \{\omega \in \Omega | \tilde{z}_{j\omega} = 1\}$.

4 if $\sum_{\omega \in \Omega_1} p_{\omega} \tilde{z}_{j\omega} \geq 1 - \varepsilon$ (for (CAP)) or $\inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega_1} p_{\omega} \tilde{z}_{j\omega} \geq 1 - \varepsilon$ (for (DR-CAP)) then

5 Sort $\tilde{y}_j$ in non-increasing order: $\tilde{y}_{i_1j} \geq \ldots \geq \tilde{y}_{i_oj}$.

6 for $\omega \in \Omega_1$ do

7 Sort $\tilde{y}_j$ in non-increasing order: $\tilde{y}_{i_1j} \geq \ldots \geq \tilde{y}_{i_oj}$.

8 Delete elements from the tail of $C$ to get a minimal cover $C$.

9 Let set $D = \{i \in C | \tilde{y}_{ij} = 1\}$ and $\mathcal{I}_0 = \{i \in I \setminus C | \tilde{y}_{ij} = 0\}$. Calculate $\alpha_i$ for $i \in I \setminus (C \cup \mathcal{I}_0)$.

10 if $\sum_{i \in C \setminus D} \tilde{y}_{ij} + \sum_{i \in I \setminus (C \cup \mathcal{I}_0)} \alpha_i \tilde{y}_{ij} > |C \setminus D| - 1$ then

11 Calculate $\beta_i$ for $i \in D$, $\alpha_i$ for $i \in \mathcal{I}_0$, and $\gamma$ for $\omega \in \Omega_1$. Obtain the violated inequality (25).

end

end

end

end
B.4. Branch-and-Cut Algorithm

The branch-and-cut algorithm is provided in Algorithm 5.

**Algorithm 5: Branch-and-Cut Implementation**

1. **Initialize** \( UB = +\infty, LB = -\infty, k = 0 \). Node list \( N = \{o\} \), \( o \) is a branching node without constraints.

2. **while** \( (N \) is nonempty) **do**
   
   3. Select a node \( o \in N \), \( N \leftarrow N \setminus \{o\} \).
   
   4. At the node \( o \), solve the LP relaxation problem of (IP). \( k = k + 1 \).
   
   5. Obtain an optimal solution \( (y^k, z^k) \) and objective value \( \text{obj}^k \).

   6. **if** \( \text{obj}^k < UB \) **then**
      
      7. **if** \( (y^k, z^k) \) is fractional **then**
         
         8. **if** Violated inequalities are found **then**
            
            9. Add the violated inequalities to the LP relaxation problem. Go to line 5.
         
         10. **else**
             
             11. Branch, resulting in nodes \( o^* \) and \( o^{**} \), \( N \leftarrow N \cup \{o^*, o^{**}\} \).
             
             12. **end**
      
      13. **else**
          
          14. Update UB, \( UB = \text{obj}^k \), \( (y^*, z^*) = (y^k, z^k) \).
      
      15. **end**
   
   16. **end**

19. **return** UB and its corresponding optimal solution \( (y^*, z^*) \).

**Appendix C: Dynamic Programming Approach for Computing Big-M values**

In this appendix, we use the dynamic programming approach proposed by Bertsimas and Demir (2002) to compute the Big-M values in the model reformulation. For \( j \in J \), let \( D(|I|, t_j, \rho_j) \) represents (5), where \(|I|\) denotes the \(|I|\) variables of \( y_j \). Let \( D(n, t_j, \rho_j) \) be a subproblem of \( D(|I|, t_j, \rho_j) \), where \( n \) denotes the first \( n \) variables of \( y_j \) in (5). Let \( S(n, t_j, \rho_j) \) be the optimal objective value of \( D(n, t_j, \rho_j) \). If \( D(n, t_j, \rho_j) \) is infeasible, we set \( S(n, t_j, \rho_j) = -\infty \). Note that if \( y_{nj} = 0 \), \( S(n, t_j, \rho_j) \) is equal to \( S(n-1, t_j, \rho_j) \). If \( y_{nj} = 1 \), \( S(n, t_j, \rho_j) \) is equal to \( S(n-1, t_j - \xi^k_n, \rho_j - 1) + \xi^\omega_n \). Thus, we have

\[
S(n, t_j, \rho_j) = \max\{S(n-1, t_j, \rho_j), S(n-1, t_j - \xi^k_n, \rho_j - 1) + \xi^\omega_n\},
\]

where \( n = 2, \ldots, |I| \), with an initial condition \( S(1, t_j, \rho_j) \). Hence,

\[
m^\omega_j(k) = S(|I|, t_j, \rho_j).
\]

**Appendix D: Statistics of Surgery Duration**

Table 5 presents the statistics of surgery duration for the real-life data, i.e. mean, standard deviation and the percentage for each surgery type.

**Appendix E: Computational Results for Weaker Big-M of (CAP)**

Table 6 reports computational results for the weaker big-M of (CAP).
### Table 5
For each surgery type, the mean (mean), standard deviation (std) in hours, and the percentage for each surgery type (percentage) are reported.

<table>
<thead>
<tr>
<th>surgery type</th>
<th>mean (hrs)</th>
<th>std (hrs)</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gynaecology</td>
<td>1.1</td>
<td>1.3</td>
<td>0.29</td>
</tr>
<tr>
<td>Galactophile</td>
<td>1.6</td>
<td>1.0</td>
<td>0.15</td>
</tr>
<tr>
<td>Lymphatic</td>
<td>3.2</td>
<td>1.1</td>
<td>0.14</td>
</tr>
<tr>
<td>Ear</td>
<td>2.8</td>
<td>1.7</td>
<td>0.13</td>
</tr>
<tr>
<td>Urology</td>
<td>2.3</td>
<td>1.7</td>
<td>0.07</td>
</tr>
<tr>
<td>Vascular</td>
<td>2.6</td>
<td>1.5</td>
<td>0.07</td>
</tr>
<tr>
<td>Obstetrics</td>
<td>1.5</td>
<td>0.5</td>
<td>0.06</td>
</tr>
<tr>
<td>Joint</td>
<td>2.8</td>
<td>1.3</td>
<td>0.06</td>
</tr>
<tr>
<td>Orthopedic</td>
<td>3.2</td>
<td>1.8</td>
<td>0.03</td>
</tr>
</tbody>
</table>

### Table 6
The average time (in seconds) for the weaker big-M computations (AvT-M), the branch-and-cut algorithm (AvT-B&C), the average number of nodes (# of nodes), and the number of instances solved to optimality (solved).

<table>
<thead>
<tr>
<th>ε</th>
<th>N</th>
<th>AvT-M</th>
<th>AvT-B&amp;C</th>
<th># of nodes</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12</td>
<td>500</td>
<td>11.4</td>
<td>122.6</td>
<td>1,798</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>43.8</td>
<td>219.7</td>
<td>2,088</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>98.7</td>
<td>771.0</td>
<td>5,090</td>
<td>5/5</td>
</tr>
<tr>
<td>0.1</td>
<td>500</td>
<td>11.4</td>
<td>164.9</td>
<td>3,914</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>43.8</td>
<td>604.7</td>
<td>7,192</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>98.7</td>
<td>2,298.8</td>
<td>11,049</td>
<td>5/5</td>
</tr>
<tr>
<td>0.08</td>
<td>500</td>
<td>11.4</td>
<td>1,290.8</td>
<td>42,876</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>43.8</td>
<td>2,777.8</td>
<td>25,874</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>98.7</td>
<td>8,459.9</td>
<td>103,689</td>
<td>4/5</td>
</tr>
<tr>
<td>0.06</td>
<td>500</td>
<td>11.4</td>
<td>[0.11]</td>
<td>2,232,748</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>43.8</td>
<td>[0.21]</td>
<td>632,822</td>
<td>0/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>98.7</td>
<td>[0.28]</td>
<td>362,215</td>
<td>0/5</td>
</tr>
</tbody>
</table>

“[.]” in column of AvT-B&C means the average sub-optimality gap for instances that cannot be solved to optimality within 10 hours time limit.