A SOLUTION APPROACH TO DISTRIBUTIONALLY ROBUST
CHANCE-CONSTRAINED ASSIGNMENT PROBLEMS∗

SHANSHAN WANG †, JINLIN LI ‡, AND SANJAY MEHROTRA §

Abstract. We study an assignment problem with chance constraints (CAP) and its distributionally robust counterpart (DR-CAP). We present a technique for big-M estimation, applicable for general distributionally robust chance constraint problems, that takes advantage of the ambiguity set in the big-M estimations. Next, for a bilinear formulation of CAP, we develop a class of valid inequalities for a 0-1 bilinear knapsack set, as well as multiple bilinear knapsack sets with a binary linear knapsack constraint. These are subsequently extended to DR-CAP. Branch-and-cut approaches combined with the valid inequalities are developed to solve CAP. This is used within a probability cut framework to solve DR-CAP. A computational study for CAP and DR-CAP using data from a hospital operating room problem is conducted. We find that the incorporation of big-M calculations and the proposed cuts allow us to solve certain model instances, and reduce the computational time for others. The instances requiring a high probability of chance satisfaction remain challenging. We also find that the use of the Wasserstein ambiguity set improves the out-of-sample performance of satisfying the chance constraints. This improvement is more significant than the one possible by increasing the sample size. The DR-CAP model instances can be solved in approximately four times the time required to solve CAP instances.

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

AMS subject classifications. 90C10, 90C15, 90C34, 90C90

1. Introduction. In the chance-constrained assignment problem, we assign items with random weights to available bins and minimize the assignment cost while satisfying the bin capacity constraints with probability at least 1 − ε. 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:

\[
\begin{align*}
\text{(1.1a) } \quad \text{minimize} & \quad \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
\text{(1.1b) } & \quad \text{subject to} \quad \sum_{j \in J} y_{ij} = 1, \quad \forall i \in I, \\
\text{(1.1c) } & \quad \sum_{i \in I} y_{ij} \leq \rho_j, \quad \forall j \in J, \\
\text{(1.1d) } & \quad \mathbb{P} \left\{ \sum_{i \in I} \xi_i y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, \quad \forall j \in J,
\end{align*}
\]

where \( I := \{1, \ldots, |I|\} \) is the set of items, \( J := \{1, \ldots, |J|\} \) 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 \( y_{ij} \)

∗Submitted to the editors DATE.
†Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 60208. Department of Management Science, Beijing Institute of Technology, China, 100081 (shshwang_bit@163.com).
‡Department of Management Science, Beijing Institute of Technology, Beijing China, 100081 (jinlinli@bit.edu.cn).
§Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 60208 (mehrotra@iems.northwestern.edu).

This manuscript is for review purposes only.
indicates if item $i$ is assigned to bin $j$. Let $y_j := (y_{1j}, \ldots, y_{|I|j})^\top$ for $j \in J$, and $y := (y_1, \ldots, y_{|J|})^\top$. The objective (1.1a) minimizes the total cost of assigning the items to the bins. Constraints (1.1b) ensure that item $i$ is assigned to only one bin. Constraints (1.1c) ensure that at most $p_{ij}$ items are assigned to bin $j$. Constraints (1.1d) 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. 2018), and cloud computing (Cohen et al. 2019), among others.

There are several challenges in solving the chance-constrained assignment problem. First, (CAP) is not a convex optimization problem, given that the variables in (CAP) are binary and chance constraints (1.1d) 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 (1.1d) are satisfied over all probability distributions within the ambiguity set $\mathcal{P}$, resulting in the formulation:

$$
\begin{align*}
(1.2a) & \quad \text{minimize} & & \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
& \quad \text{subject to} & & (1.1b), (1.1c), \\
(1.2b) & \quad \inf_{P \in \mathcal{P}} \mathbb{P} \left\{ \sum_{i \in I} \xi_i y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, \quad \forall j \in J.
\end{align*}
$$

In this paper we assume that the probability distribution $\mathbb{P}$ has finite support $\xi := (\xi_1^\top, \ldots, \xi_N^\top)^\top$, where $\xi^\top := (\xi_1^\top, \ldots, \xi_N^\top)^\top$ 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 $i \in I, j \in J, \omega \in \Omega$.

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), K"{u}ç{"u}kyavuz (2012), Abdi and Fukasawa (2016), Liu et al. (2019), Zhao et al. (2017), Peng et al. (2018)).

1.1. Chance-Constrained Programs with Random Technology Matrices. The model in (1.1) 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. In a companion paper, Wang et al. (2019) studied the chance-constrained bin-packing problem and used the lifting technique to develop a family of valid inequalities for the set obtained from only one binary bilinear knapsack constraint (see Example 2). The difference in the structures of the bin-packing problems and the assignment problems motivate us to generalize the applicability of the approach developed in Wang et al. (2019) and obtain stronger valid inequalities for (CAP).

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), Zhao and Guan (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.

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 (1.1c). 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 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 gives proofs of some of the propositions and theorems. Appendix B provides pseudo-code of some of the algorithms implemented in our computations. Appendix C gives additional tables containing computational results.

2. Model Reformulation. We formulate (CAP) as a binary linear program in Section 2.1, and present a dynamic programming based approach to estimate the big-M coefficients in Appendix B.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 \( j \in J \) and \( \omega \in \Omega \). Namely, \( z_{j\omega} = 1 \) if the constraint \( \sum_{i \in I} \xi_i^\omega y_{ij} \leq t_j \) is satisfied, and \( z_{j\omega} = 0 \), otherwise. For \( j \in J \), let \( z_j := (z_{j1}, \ldots, z_{jN})^T \) and \( z := (z_1, \ldots, z_{|J|})^T \). The constraints (1.1d) can be formulated as

\[
\begin{align*}
(2.1a) & \quad \sum_{i \in I} \xi_i^\omega y_{ij} + (M_j^\omega - t_j)z_{j\omega} \leq M_j^\omega, \quad \forall j \in J, \omega \in \Omega, \\
(2.1b) & \quad \sum_{\omega \in \Omega} p_{\omega} z_{j\omega} \geq 1 - \varepsilon, \quad \forall j \in J,
\end{align*}
\]  

where \( M_j^\omega \) is a constant that ensures that the constraints (2.1a) hold when \( z_{j\omega} = 0 \). Computation of a small valid value of big-M gives a tighter formulation in (2.1). 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 \( j \in J \) and \( \omega \in \Omega \):

\[
(2.2) \quad M_j^\omega \geq M_j^\omega := \max_{y_j \in \{0,1\}^{|I|}} \left\{ \sum_{i \in I} \xi_i^\omega y_{ij} \right\}, \quad \forall j \in J, \omega \in \Omega,
\]

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

\[
(2.3) \quad m_j^\omega(k) := \max_{y_j, \varepsilon \in \{0,1\}^{|I|}} \left\{ \sum_{i \in I} \xi_i^\omega y_{ij} \right\}, \quad \forall j \in J, \omega, k \in \Omega,
\]

We sort \( m_j^\omega(k) \) in a non-decreasing order such that \( m_j^\omega(k_1) \leq \ldots \leq m_j^\omega(k_N) \). An upper bound for \( M_j^\omega \) is provided in Proposition 2.1. A proof of this proposition is given in Appendix A.1.

PROPOSITION 2.1. \( m_j^\omega(k_q) \) is an upper bound for \( M_j^\omega \), where \( q := \min \left\{ l \mid \sum_{j=1}^l p_{kj} > \varepsilon \right\} \), and (CAP) can be equivalently reformulated as the following binary integer program

\[
\begin{align*}
(2.4a) & \quad \text{(IP)} \quad \minimize_{y,z \in \{0,1\}^{|I|}} \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
& \quad \text{subject to} \quad (1.1b), (1.1c), (2.1b), \\
(2.4b) & \quad \sum_{i \in I} \xi_i^\omega y_{ij} + (m_j^\omega(k_q) - m_j^\omega(\omega)) z_{j\omega} \leq m_j^\omega(k_q), \quad \forall j \in J, \omega \in \Omega. \quad \Box
\end{align*}
\]
Remark. (2.3) has a knapsack constraint and a cardinality constraint. We describe a dynamic programming method for solving (2.3) in Appendix B.1. 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 (2.3) to optimality.

Remark. The constraints (1.1b), and (1.1c) represent the assignment structure of model (1.1). In this model, and in the corresponding statement of Proposition 2.1, 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:

\[
P_W = \{ p \in \mathbb{R}_+^N | \sum_{\omega \in \Omega} p_\omega = 1, \sum_{\omega \in \Omega} \sum_{k \in \Omega} \| \xi^\omega - \xi^k \| \nu_{\omega k} \leq \eta, \sum_{k \in \Omega} \nu_{\omega k} = p_\omega, \forall \omega \in \Omega, \sum_{\omega \in \Omega} \nu_{\omega k} = p^*_k, \forall k \in \Omega, \nu_{\omega k} \geq 0, \forall \omega, k \in \Omega \},
\]

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 \( 1(\cdot) \) denote an indicator function. Using this notation the constraint (1.2b) using the Wasserstein ambiguity set is given as follows:

\[
\inf \left\{ \sum_{\omega \in \Omega} p_\omega \cdot \left( \sum_{i \in I} \xi^\omega_{ij} y_{ij} \leq t_j \right) \right\} p \in P_W \geq 1 - \varepsilon, \quad \forall j \in \mathcal{J}.
\]

Let \( z_{j\omega} \) and \( m_j^\omega(\cdot) \) be defined as in Section 2.1. The following theorem gives a semi-infinite reformulation of (DR-CAP) with a general ambiguity set \( P \). The proof is given in Appendix A.2.

**Theorem 2.2.** 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:

\[
(2.6a) \quad \text{minimize} \quad \sum_{i \in I} \sum_{j \in \mathcal{J}} c_{ij} y_{ij}
\]

subject to (1.1b), (1.1c),

\[
(2.6b) \quad \inf_{p \in P} \sum_{\omega \in \Omega} p_\omega z_{j\omega} \geq 1 - \varepsilon, \quad \forall j \in \mathcal{J},
\]

\[
(2.6c) \quad \sum_{i \in I} \xi^\omega_{ij} y_{ij} + \left( m_j^\omega(k_\bar{q}) - m_j^\omega(\omega) \right) z_{j\omega} \leq m_j^\omega(k_\bar{q}), \quad \forall j \in \mathcal{J}, \omega \in \Omega,
\]

where \( \bar{q} := \min \{ l | \sup_{p \in P} \sum_{j=1}^l p_{kj} > \varepsilon \} \). □

Note that when \( P := P_W \), \( \bar{q} \) in Theorem 2.2 is obtained by solving a sequence of linear programs. Moreover, the left-hand side of (2.6b) 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 \( P \) is sufficient for the big-M estimation. Such distributions are available as the probability cut algorithm given in Section 4.3 progresses.

This manuscript is for review purposes only.
Corollary 2.3. Let \( \hat{p}_\omega \) \( \in \mathcal{P} \), and \( \hat{q} = \min \{ l | \sum_{j=1}^l \hat{p}_{k_j} > \varepsilon \} \). Then, \( \hat{q} \leq \hat{q} \) and \( m^\omega_j(k_{\hat{q}}) \leq m^\omega_j(k_{\hat{q}}) \).

Proof. Since \( \sup_{p \in \mathcal{P}} \sum_{j=1}^l p_{k_j} \geq \sum_{j=1}^l \hat{p}_{k_j} \geq \varepsilon \), we have \( \hat{q} \geq \hat{q} \) and \( m^\omega_j(k_{\hat{q}}) \leq m^\omega_j(k_{\hat{q}}) \).

Remark. Theorem 2.2 and Corollary 2.3 remain valid for the case where the cardinality constraint in (2.2) 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 (2.4b) and (2.6c) can also be rewritten as

\[
(2.7) \quad \sum_{i \in \mathcal{I}} \xi_{i\omega}^j y_{ij} z_{j\omega} \leq m^\omega_j(\omega) z_{j\omega}, \quad \forall j \in \mathcal{J}, \omega \in \Omega.
\]

Thus, we can use (2.7) to obtain a binary bilinear reformulation and bilinear semi-infinite reformulation for (CAP) and (DR-CAP), respectively. The following proposition shows a relationship between the bilinear reformulations with the formulations (2.6) and (2.4), respectively. A proof is given in Appendix A.3.

Proposition 2.4. The relaxation of the binary bilinear reformulation for (CAP) obtain from relaxing the binary variables is stronger than the linear relaxation of (2.4). Similarly, the relaxation of the binary bilinear reformulation for (DR-CAP) obtained from relaxing the binary variables is stronger than the linear relaxation of (2.6).

Note that constraints (1.1c), (2.1b) and (2.7) give a key substructure of the binary bilinear reformulation of (CAP). Let

\[
\mathcal{H} := \left\{ (y, z) \in \{0, 1\}^{\mathcal{I} \times \mathcal{J} \times \Omega} \times \{0, 1\}^{\mathcal{J} \times \mathcal{N}} | (1.1c), (2.1b), (2.7) \right\}.
\]

For \( j \in \mathcal{J} \), let

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

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

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 2.5. 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 \( \text{conv}(\mathcal{H}) \).

Proposition 2.5 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 \( j \in \mathcal{J} \), let

\[
\mathcal{G}_j' := \{ (y_j, z_j) \in \{0, 1\}^{\mathcal{I}} \times \{0, 1\}^{\mathcal{N}} \bigg| \sum_{i \in \mathcal{I}} \xi_{i\omega}^j y_{ij} z_{j\omega} \leq m^\omega_j(\omega) z_{j\omega}, \forall \omega \in \Omega, \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega} p_{\omega} z_{j\omega} \geq 1 - \varepsilon, y_j \in \mathcal{Y}_j \}.
\]
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. Due to the flexibility of our approach, Section 3.2 further presents a family of valid inequalities for $G_j$ and $G'_j$.

3.1. Lifted Cover Inequalities. We assume that $j \in J$ and $\omega \in \Omega$ are fixed in this section. Let us consider the binary bilinear knapsack set $F_{j\omega} := \left\{ (y_j, z_{j\omega}) \in \{0, 1\}^{|I|} \times \{0, 1\} \mid \sum_{i \in I} \xi_{\omega}^i y_{ij} z_{j\omega} \leq m_{j\omega}^\omega(\omega), y_j \in Y_j \right\}$. Note that the inequalities valid for $\text{conv}(F_{j\omega})$ 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 $F_{i\omega}$. When $z_{j\omega} = 1$, the set $F_{j\omega}$ becomes the two-constraint 0-1 knapsack set $Q_{j\omega} := \left\{ y_j \in \{0, 1\}^{|I|} \mid \sum_{i \in I} \xi_{\omega}^i y_{ij} \leq m_{j\omega}^\omega(\omega), y_j \in Y_j \right\}$.

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 $Q_{j\omega}$. We also provide a lifted cover inequality that is valid for $\text{conv}(F_{j\omega})$ 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 3.1.** Set $C \subseteq I$ is a cover for $\sum_{i \in I} \xi_{\omega}^i y_{ij} \leq m_{j\omega}^\omega(\omega)$ if $\sum_{i \in C} \xi_{\omega}^i > m_{j\omega}^\omega(\omega)$. The cover $C$ is minimal if no subset of $C$ is a cover for $\sum_{i \in I} \xi_{\omega}^i y_{ij} \leq m_{j\omega}^\omega(\omega)$. □

In this section, we assume that $C$ is a minimal cover for $\sum_{i \in I} \xi_{\omega}^i y_{ij} \leq m_{j\omega}^\omega(\omega)$. Let $D \subseteq C$. 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.

**Proposition 3.2.** The inequality

$$\sum_{i \in C \setminus D} y_{ij} \leq |C \setminus D| - 1$$

is valid for (3.1). If $|C| \leq \rho_j + 1$, the inequality (3.2) is facet-defining for (3.1). □

3.1.1. Up-Lifting. In general, a cover inequality (3.2) 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 $I \setminus C$. This procedure is called up-lifting. By using the up-lifting technique, we obtain an inequality of the form

$$\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} \leq |C \setminus D| - 1,$$
where \( \alpha_i \) is called an up-lifting coefficient. We now provide such an uplifting approach for our problem.

Let \( \{ \pi_k \}_{k=1}^{|I| - |C|} \) be a sequence of the set \( I \setminus C \) and \( \pi(k) = \{ \pi_1, \ldots, \pi_k \} \). For \( k = 1, \ldots, |I| - |C| \), let

\[
\text{(3.4a)} \quad \theta_i^{\pi_k} := \max_{y_j \in \{0, 1\}^{|I| - |C|}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \pi(k-1)} \alpha_i y_{ij}
\]

\[
\text{(3.4b)} \quad \text{subject to } \sum_{i \in C \setminus D} \xi_i y_{ij} + \sum_{i \in \pi(k-1)} \xi_i^{\pi_k} y_{ij} \leq m_j^\omega(\omega) - \sum_{i \in D} h_i
\]

\[
\text{(3.4c)} \quad \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \pi(k-1)} y_{ij} \leq \rho_j - 1 - |D|.
\]

Note that different sequences of \( I \setminus C \) might lead to different valid inequalities (Kaparis and Letchford 2008). The following lemma gives a sufficient condition under which the inequality (3.3) is facet-defining for the convex hull of \( Q_j \) when \( y_{ij} = 1 \), \( i \in D \).

**Lemma 3.3.** For \( k = 1, \ldots, |I| - |C| \), let \( \alpha_{\pi_k} = |C \setminus D| - 1 - \theta_i^{\pi_k} \), where \( \theta_i^{\pi_k} \) is defined in (3.4). The inequality (3.3) is valid for

\[
\text{(3.5)} \quad \text{conv} \left( \left\{ y_j \in \{0, 1\}^{|I| - |C|} \left| \sum_{i \in C \setminus D} \xi_i y_{ij} \leq m_j^\omega(\omega), y_j \in \mathcal{Y}_j, y_{ij} = 1, \forall i \in D \right. \right\} \right).
\]

If \( |C| \leq \rho_j + 1 \), the inequality (3.3) is facet-defining for (3.5).

**Proof** See Appendix A.6. \( \square \)

We use a dynamic programming 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.2.

Remark. Note that the sufficient condition in Lemma 3.3 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 even when this condition is not satisfied. However, it suggests a preference for identifying low cardinality covers when possible.

**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

\[
\text{(3.6)} \quad \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} \leq |C \setminus D| + \sum_{i \in D} \beta_i - 1,
\]

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 \( \{ \kappa_i \}_{l=1}^{l=|D|} \) be a sequence of the set \( D \) and \( \kappa(l) = \{ \kappa_1, \ldots, \kappa_l \} \). For \( l = 1, \ldots, |D| \), let

\[
\text{(3.7a)} \quad \text{obj}_{\kappa_l} := \max_{y_j \in \{0, 1\}^{(I \setminus C) \setminus \kappa(l-1)}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} + \sum_{i \in \kappa(l-1)} \beta_i y_{ij}
\]

\[
\text{(3.7b)} \quad \text{subject to } \sum_{i \in I \setminus D} \xi_i y_{ij} + \sum_{i \in \kappa(l-1)} \xi_i^{\kappa_l} y_{ij} \leq m_j^\omega(\omega) - \sum_{i \in D} \xi_i
\]

\[
\text{(3.7c)} \quad \sum_{i \in I \setminus D} y_{ij} + \sum_{i \in \kappa(l-1)} y_{ij} \leq \rho_j - |D| + l.
\]
LEMA 3.4. For \( l = 1, \ldots, |\mathcal{D}| \), let \( \beta_{n_l} = \text{obj}_{r_{n_l}} - \sum_{i \in \mathcal{K}(l-1)} \beta_i - |\mathcal{C}| + 1 \), where \( \text{obj}_{r_{n_l}} \) is defined in (3.7). The inequality (3.6) is valid for \( \text{conv}(\mathcal{Q}_{j, \omega}) \). If \( |\mathcal{C}| \leq \rho_j + 1 \), (3.6) is facet-defining for \( \text{conv}(\mathcal{Q}_{j, \omega}) \).

**Proof** See Appendix A.7. \( \square \)

### 3.1.3. Lifted Cover Inequality

In the following, we provide coefficient calculations for a lifted cover inequality that is valid for \( \text{conv}(\mathcal{F}_{j, \omega}) \).

**Theorem 3.5.** The lifted cover inequality

\[
(3.8) \quad \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \alpha_i y_{ij} + \sum_{i \in \mathcal{D}} \beta_i y_{ij} + \gamma(z_{\omega} - 1) \leq |\mathcal{C} \setminus \mathcal{D}| + \sum_{i \in \mathcal{D}} \beta_i - 1
\]

is valid for \( \text{conv}(\mathcal{F}_{j, \omega}) \) if

\[
(3.9a) \quad \gamma = \max_{y_{j} \in (0, 1)^{|\mathcal{I} \setminus \mathcal{C}|}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \alpha_i y_{ij} + \sum_{i \in \mathcal{D}} \beta_i y_{ij} - |\mathcal{C} \setminus \mathcal{D}| - \sum_{i \in \mathcal{D}} \beta_i + 1.
\]

Furthermore, if \( |\mathcal{C}| \leq \rho_j + 1 \), (3.8) is facet-defining for \( \text{conv}(\mathcal{F}_{j, \omega}) \).

**Proof** See Appendix A.8. \( \square \)

By restricting the feasible region of \( y_j \) in (3.9) using the chance constraints (1.1d), we obtain a stronger valid inequality for (CAP) in Theorem 3.6.

**Theorem 3.6.** For \( k \in \Omega \setminus \{\omega\} \), let

\[
(3.10a) \quad \delta_k = \max_{y_{j} \in (0, 1)^{|\mathcal{I} \setminus \mathcal{C}|}} \sum_{i \in \mathcal{C} \setminus \mathcal{D}} y_{ij} + \sum_{i \in \mathcal{I} \setminus \mathcal{C}} \alpha_i y_{ij} + \sum_{i \in \mathcal{D}} \beta_i y_{ij} - |\mathcal{C} \setminus \mathcal{D}| - \sum_{i \in \mathcal{D}} \beta_i + 1
\]

subject to \( \sum_{i \in \mathcal{I}} \xi_i y_{ij} \leq m_j^k(k) \).

Sort \( \delta_k \) such that \( \delta_{k_1} \leq \ldots \leq \delta_{k_{|\Omega|-1}} \). Let \( q^1 := \min \{l \mid \sum_{j=1}^l p_{k_j} > \varepsilon\} \), then the inequality (3.8) is valid for (CAP), where \( \gamma = \delta_{k_q^1} \).

**Proof** See Appendix A.9. \( \square \)

We further restrict the feasible region of \( y_j \) in (3.9) by using the distributionally robust chance constraints (1.2d) to obtain a stronger valid inequality for (DR-CAP) in the following theorem.

**Theorem 3.7.** For \( k \in \Omega \setminus \{\omega\} \), let \( \delta_k \) be defined as in Theorem 3.6, and sort \( \delta_k \) such that \( \delta_{k_1} \leq \ldots \leq \delta_{k_{|\Omega|-1}} \). Let \( \bar{q}^1 := \min \{l \mid \sup_{p \in \mathcal{P}} \sum_{j=1}^l p_{k_j} > \varepsilon\} \). Then, the inequality (3.8) is valid for (DR-CAP) when \( \gamma = \delta_{k_{q^1}} \). Moreover, if \( \bar{p}_{\omega} \in \mathcal{P} \), let \( \hat{q}^1 := \min \{l \mid \sum_{j=1}^l \hat{p}_{k_j} > \varepsilon\} \). Then, \( \hat{q}^1 \geq \bar{q}^1 \) and the inequality (3.8) is valid for (DR-CAP) when \( \gamma = \delta_{k_{q^1}} \).

**Proof** See Appendix A.10. \( \square \)

We use the dynamic programming approach similar to the one described in Appendix B.1 to solve (3.7) and (3.10).
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 from Wang et al. (2019) (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 $\rho_j = 3$, $m_j^j(\omega) = 40$, and $\xi_\omega = (7, 8, 11, 9, 14, 23)^T$. Then the set $C = \{1, 2, 3, 4, 5\}$ is a minimal cover. Let $D = \{5\}$, suppose $N = 5$, $\epsilon = 0.6$, and the other scenarios in the computation of lifted cover inequalities are $(8, 11, 7, 10, 7, 17, 23)^T$, $(14, 7, 10, 11, 8, 13, 26)^T$, $(21, 10, 7, 29, 16, 12, 23)^T$, and $(15, 7, 8, 23, 12, 10, 5)^T$, with $p_\omega = 1/N$ for all $\omega \in \Omega$. We get a lifted cover inequality by Theorem 3.6 as:

\[
y_{1j} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + 2y_{6j} + 2y_{7j} + z_{j\omega} \leq 5.
\] (3.11)

If $p_\omega^* = 1/N$ for all $\omega \in \Omega$ and $\eta = 0.5$ in the Wasserstein set $\mathcal{P}_W$ in (2.5), then a lifted cover inequality for (DR-CAP) obtained from Theorem 3.7 is given as follows:

\[
y_{1j} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + 2y_{6j} + 2y_{7j} \leq 4.
\] (3.12)

**Example 2.** (Continued from Example 1) Suppose that the cardinality constraint $\sum_{i\in I} y_{ij} \leq \rho_j$ is removed from $F_{j\omega}$. Following a computation procedure similar to the one for the lifted cover inequality, we obtain a valid inequality of the following form:

\[
y_{1j} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + 2y_{6j} + z_{j\omega} \leq 5.
\] (3.13)

We call the inequality (3.13) single lifted cover inequality. Obviously, the lifted cover inequality (3.11) is stronger than the single lifted cover inequality (3.13). The single lifted cover inequality for (DR-CAP) is

\[
y_{1j} + y_{2j} + y_{3j} + y_{4j} + y_{5j} + y_{6j} + 2y_{7j} \leq 4.
\] (3.14)

(3.12) is also stronger than (3.14). 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 $\tilde{\Omega}$ be a set where each element $\Omega_k \in \tilde{\Omega}$ is a subset of $\Omega$ such that $\sum_{\omega \in \Omega_k} p_\omega \geq 1 - \epsilon$, for $k = 1, \ldots, |\Omega|$. Without loss of generality, we reuse the notation set $\Omega$ and $\Omega_k$ for (DR-CAP). For (DR-CAP), let $\bar{\Omega}$ be a set where each element $\Omega_k \in \bar{\Omega}$ is a subset of $\Omega$ such that $\inf_{\mathcal{P} \in \mathcal{P}_W} \sum_{\omega \in \Omega_k} p_\omega \geq 1 - \epsilon$, for $k = 1, \ldots, |\bar{\Omega}|$. $\bar{\Omega}$ is maximal if it is not a proper subset of any other sets that satisfy the above condition.

For maximal $\bar{\Omega}$, let the global lifted cover inequalities be of the form

\[
\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} + \sum_{\omega \in \Omega_k} \zeta_\omega (z_{j\omega} - 1) \leq |C \setminus D| + \sum_{i \in D} \tilde{\beta}_i - 1, \quad k = 1, \ldots, |\bar{\Omega}|,
\] (3.15)

where $C$ is a cover for the set $Q_{j\omega}$ for some $\omega \in \Omega$, and $D \subseteq C$. For $k \in \{1, \ldots, |\bar{\Omega}|\}$, when $z_{j\omega} = 1$, $\omega \in \Omega_k$, (3.15) becomes

\[
\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \tilde{\beta}_i y_{ij} \leq |C \setminus D| + \sum_{i \in D} \tilde{\beta}_i - 1.
\] (3.16)
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 \( \bar{\bar{\alpha}}_i \) and \( \bar{\bar{\beta}}_i \) in (3.16).

3.2.1. Up-Lifting. Let \( \{ \bar{\pi}_l \}_{l=1}^{[Z,C]} \) be a sequence of \( \mathcal{I} \backslash \mathcal{C} \) and \( \bar{\pi}(l) = \{ \bar{\pi}_1, \ldots, \bar{\pi}_{l} \} \). For \( l = 1, \ldots, |\mathcal{I} \backslash \mathcal{C}| \), the up-lifting problem is as follows:

\[
\begin{align*}
(3.17a) \quad \text{obj}_{\bar{\pi}_l} &= \max_{\bar{y}_j \in \{0,1\}^{(C \backslash D) \cup \pi(l-1)}} \sum_{i \in C \backslash D} y_{ij} + \sum_{i \in \pi(l-1)} \bar{\bar{\alpha}}_i y_{ij} \\
& \quad \text{subject to} \sum_{i \in C \backslash D} \xi^\omega_i y_{ij} + \sum_{i \in \pi(l-1)} \xi^\omega_i y_{ij} \leq m^\omega_i(\omega) - \xi^\omega_i - \sum_{i \in D} \xi^\omega_i, \quad \forall \omega \in \Omega_k, \\
& \quad \sum_{i \in C \backslash D} y_{ij} + \sum_{i \in \pi(l-1)} y_{ij} \leq \rho_j - 1 - |D|. 
\end{align*}
\]

Then \( \bar{\bar{\alpha}}_{\bar{\pi}_l} = |\mathcal{C} \backslash \mathcal{D}| - 1 - \text{obj}_{\bar{\pi}_l} \). 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 (3.17). Kaparis and Letchford (2008) suggest relaxing \( y_j \in \{0,1\}^{[Z]} \) 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 2 in Appendix B.2, we propose a heuristic to calculate \( \bar{\bar{\alpha}}_{\bar{\pi}_l} \) as follows. For each \( \omega \in \Omega_k \), let

\[
\text{obj}_{\bar{\pi}_l}(\omega) := \max_{\bar{y}_j \in \{0,1\}^{(C \backslash D) \cup \pi(l-1)}} \sum_{i \in C \backslash D} y_{ij} + \sum_{i \in \pi(l-1)} \bar{\bar{\alpha}}_i y_{ij} \\
\text{subject to} \sum_{i \in C \backslash D} \xi^\omega_i y_{ij} + \sum_{i \in \pi(l-1)} \xi^\omega_i y_{ij} \leq m^\omega_i(\omega) - \xi^\omega_i - \sum_{i \in D} \xi^\omega_i, \\
\sum_{i \in C \backslash D} y_{ij} + \sum_{i \in \pi(l-1)} y_{ij} \leq \rho_j - 1 - |D|. 
\]

Then, \( \text{obj}_{\bar{\pi}_l}(\omega) \) is an upper bound for \( \text{obj}_{\bar{\pi}_l} \). Algorithm 2 in Appendix B.2 is used to compute \( \text{obj}_{\bar{\pi}_l}(\omega) \), for \( \omega \in \Omega_k \). 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 \( \bar{\bar{\alpha}}_{\bar{\pi}_l} = |\mathcal{C} \backslash \mathcal{D}| - 1 - \min_{\omega \in \Omega_k} \text{obj}_{\bar{\pi}_l}(\omega) \), which implies \( \bar{\bar{\alpha}}_{\bar{\pi}_l} \leq |\mathcal{C} \backslash \mathcal{D}| - 1 - \text{obj}_{\bar{\pi}_l} \). Thus, \( \bar{\bar{\alpha}}_{\bar{\pi}_l} \) is a valid lifting coefficient.

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

\[
\begin{align*}
(3.18a) \quad \text{obj}_{\bar{\kappa}_l} &= \max_{\bar{y}_j \in \{0,1\}^{(I \backslash D) \cup \kappa(l-1)}} \sum_{i \in I \backslash D} y_{ij} + \sum_{i \in I \cap L(l-1)} \bar{\bar{\beta}}_i y_{ij} \\
& \quad \text{subject to} \sum_{i \in I \backslash D} \xi^\omega_i y_{ij} + \sum_{i \in I \cap L(l-1)} \xi^\omega_i y_{ij} \leq m^\omega_i(\omega) - \sum_{i = \bar{\kappa}_{l+1}^{[D]}} \xi^\omega_i, \quad \forall \omega \in \Omega_k, \\
& \quad \sum_{i \in I \backslash D} y_{ij} + \sum_{i \in I \cap L(l-1)} y_{ij} \leq \rho_j - |D| + 1. 
\end{align*}
\]
Instead of computing \( \text{obj}_{\hat{\xi}_1} \), we use the method proposed in Section 3.2.1 to obtain an upper bound for \( \text{obj}_{\hat{\xi}_1} \). Let \( \text{obj}_{\hat{\xi}_1}(\omega) \) be the optimal objective value of the maximization problem that takes a single row \( \omega \) of problem (3.18) for \( \omega \in \Omega_k \). We use a dynamic programming approach similar to the one proposed in Appendix B.1 to compute \( \text{obj}_{\hat{\xi}_1}(\omega) \), and let \( \bar{\beta}_{\hat{\xi}_1} = \min_{\omega \in \Omega_k} \text{obj}_{\hat{\xi}_1}(\omega) - \sum_{i \in \hat{\xi}_1(l-1)} \bar{\beta}_i - |C \setminus D| + 1 \).

### 3.2.3. Global Lifted Cover Inequalities

Finally, to calculate \( \bar{\gamma}_\omega \) in sequence \( \{\tau_1, \ldots, \tau_{|\Omega_k|}\} \), we consider the following problem for \( G_j \), for \( l = 1, \ldots, |\Omega_k| \):

\[
\text{obj}_r = \max_{(y_j, z_j) \in \{(0,1)^{|I|} \times (0,1)^{|\Omega_k|\cup |\tau|}\}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \bar{\alpha}_iy_{ij} + \sum_{i \in D} \bar{\beta}_iy_{ij} + \sum_{\omega \in \tau(l-1)} \gamma_\omega z_{j\omega}
\]

subject to \( \sum_{\omega \in \Omega_k} \bar{\alpha}_i y_{ij} + \sum_{\omega \in \tau(l-1)} \gamma_\omega z_{j\omega} \geq 1 - \varepsilon - \sum_{\omega = \tau_{l+1}} \bar{\beta}_i y_{ij} + \sum_{\omega \in \tau(l-1)} \gamma_\omega z_{j\omega} \)

where \( \tau(l-1) = \{\tau_1, \ldots, \tau_{l-1}\} \). The calculation of \( \text{obj}_r \) is a reformulation of a chance-constrained problem where some variables \( z_{j\omega} \) are given. Instead of solving (3.19) exactly, we provide a heuristic to obtain an upper bound for \( \text{obj}_r \). We relax \( y_j \in [0,1]^{|I|} \) and \( z_j \in [0,1]^{|\Omega_k|\cup |\tau|} \), and solve the LP relaxation of (3.19) to obtain the optimal solution \((y_j', z_j')\) and objective value \( \text{obj}_r' \) of the relaxed problem. Then, \( \text{obj}_r' \) gives an upper bound for \( \text{obj}_r \).

For \( G_j' \), for \( l = 1, \ldots, |\Omega_k| \), let

\[
\text{obj}_r' := \max_{(y_j, z_j) \in \{(0,1)^{|I|} \times (0,1)^{|\Omega_k|\cup |\tau|}\}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \bar{\alpha}_iy_{ij} + \sum_{i \in D} \bar{\beta}_iy_{ij} + \sum_{\omega \in \tau(l-1)} \gamma_\omega z_{j\omega}
\]

subject to (3.19d),

**Theorem 3.8.** Let \( \{\bar{\beta}_i\}_{i \in I \cup C} \) and \( \{\bar{\beta}\}_{i \in D} \) be defined as in Section 3.2.1 and 3.2.2, respectively. For \( l = 1, \ldots, |\Omega_k| \), we set \( \bar{\gamma}_r = \text{obj}_r' - |C \setminus D| + 1 - \sum_{i \in D} \bar{\beta}_i - \sum_{\omega \in \tau(l-1)} \bar{\gamma}_\omega \), where \( \text{obj}_r' \) is the objective value of the LP relaxation of (3.19). Then, (3.15) is valid for \( \text{conv}(G_j) \). For \( l = 1, \ldots, |\Omega_k| \), we set \( \bar{\gamma}_r = |\text{obj}_r'| - |C \setminus D| + 1 - \sum_{i \in D} \bar{\beta}_i - \sum_{\omega \in \tau(l-1)} \bar{\gamma}_\omega \), where \( \text{obj}_r' \) is the objective value of the LP relaxation of (3.20). Then, (3.15) is valid for \( \text{conv}(G_j') \).
The following example gives a global lifted cover inequality.

**Example 3. (Continued from Example 1)** We let $C = \{1, 2, 3, 4, 5\}$ and $D = \{5\}$ as before. Let $\Omega_k = \{1, 2\}$. Then we can obtain a global lifted cover inequality (3.15) 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 $\inf_{p \in \rho, \omega \in \Omega_k} \sum p_\omega \geq 1 - \varepsilon$. A valid inequality (3.15) 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 knapsack problem to separate (3.8) and (3.15), respectively.

**4.1.1. Separation Problem for (3.8).** To obtain the violated inequalities (3.8), we use a heuristic similar to the one in Gu et al. (1998) for the knapsack problem. This heuristic is provided in Algorithm 4 in Appendix B.4. Let $(\hat{y}, \hat{z})$ be a LP relaxation optimal solution, when $\hat{z}_{j\omega} = 1$

$$\sum_{i \in C \setminus D} \hat{y}_{ij} + \sum_{i \in I \setminus C} \alpha_i \hat{y}_{ij} + \sum_{i \in D} \beta_i (\hat{y}_{ij} - 1) + \gamma (\hat{z}_{j\omega} - 1) = \sum_{i \in C \setminus D} \hat{y}_{ij} + \sum_{i \in \Omega \cup (\Omega \setminus \Omega_0)} \alpha_i \hat{y}_{ij} > |C \setminus D| - 1.$$  

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

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

**4.1.2. Separation Problem for (3.15).** We develop a heuristic procedure similar to the one in Kaparis and Letchford (2008) for the multidimensional knapsack problem, to obtain the violated inequalities (3.15). Assume that we are given a LP relaxation solution ($\hat{y}, \hat{z}$) of (IP), for $j \in J$. Let $\Omega_1 = \{\omega \in \Omega | \hat{z}_{j\omega} = 1\}$ such that $\sum_{\omega \in \Omega_1} p_\omega \hat{z}_{j\omega} \geq 1 - \varepsilon$ for (CAP) and $\inf_{p \in \rho, \omega \in \Omega_1} \sum p_\omega \hat{z}_{j\omega} \geq 1 - \varepsilon$ for (DR-CAP), which indicates that $\Omega_1 \in \Omega$. Then we sort $\hat{y}_{ij}$ in a nonincreasing order: $\hat{y}_{1ij} \geq \ldots \geq \hat{y}_{|I|ij}$. Let $C = \{i_1, \ldots, i_{|C|}\}$ be such that $C$ is a cover for some $\omega \in \Omega_1$, and we delete elements from the tail of the set $C$ until the cover $C$ is minimal. Let $D = \{i \in C | \hat{y}_{ij} = 1\}$ and $I_0 = \{i \in I \setminus C | \hat{y}_{ij} = 0\}$. We first use the up-lifting procedure on the variables $y_{ij}$ in $I \setminus (C \cup I_0)$. If the inequality $\sum_{i \in C \setminus D} \hat{y}_{ij} + \sum_{i \in I \setminus (C \cup I_0)} \alpha_i \hat{y}_{ij} > |C \setminus D| - 1$, then we use the down-lifting procedure for variables $y_{ij}$ in $D$ and the up-lifting procedure for variables $y_{ij}$ in $I_0$. Finally, we compute the coefficients of $z_{j\omega}$ for $\omega \in \Omega_1$. Algorithm 5 in Appendix B.5 gives an overview of this heuristic.

Similar to Section 4.1.1, if $|D| > \rho_j - 1$ or $m_j^{\omega} (\omega) - \sum_{i \in D} \xi_{i\omega} - \max_{i \in I \setminus C} \xi_{i\omega} < 0$ for some $\omega \in \Omega$, we remove the items from $D$ until $|D| \leq \rho_j - 1$ and $m_j^{\omega} (\omega) - \sum_{i \in D} \xi_{i\omega} - \max_{i \in I \setminus C} \xi_{i\omega} \geq 0$ for all $\omega \in \Omega$.
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). Let LB and UB denote the current lower and upper bound for the optimal objective value of (CAP), and \( \mathcal{N} \) denote the set of remaining nodes in the branch-and-cut search tree. An overview of the branch-and-cut framework is given in Algorithm 6 (Appendix B.6). 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).

4.3. Branch-and-Cut Algorithm with Probability Cuts for (DR-CAP). We now investigate probability cuts within a branch-and-cut framework for solving (DR-CAP). We define the master problem as follows:

\[
\text{(MP)} \quad \begin{array}{ll}
\text{minimize} & \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \\
\text{subject to} & (1.1b), (1.1c), (2.6c),
\end{array}
\]

where the set \( \mathcal{X} \) is a complementary set that defines the feasible region of (2.6). 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 \( j \in \mathcal{J} \), a distribution separation problem is given by:

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

The problem (SP\(_j\)) is used to verify the feasibility of \((\hat{y}_j, \hat{z}_j)\) to (DR-CAP). If \( S_j(\hat{z}) \geq 1 - \varepsilon \), \((\hat{y}_j, \hat{z}_j)\) 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

\[
(4.2) \quad \sum_{\omega \in \Omega} \hat{p}_\omega \hat{z}_{j\omega} \geq 1 - \varepsilon.
\]

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

\[
(4.3) \quad \sum_{i \in I^1_j} y_{ij} \leq |I^1_j| - 1.
\]

Algorithm 1 gives a pseudocode of the branch-and-cut algorithm with probability and feasibility cuts.

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 \( \mathcal{N} \) 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 \((\hat{y}^k, \hat{z}^k)\) and the objective value \( \text{lobj}^k \) (line 4-6). If the objective value \( \text{lobj}^k \) is smaller than the current upper bound, then we check whether \((\hat{y}^k, \hat{z}^k)\) is binary (line 7). If \((\hat{y}^k, \hat{z}^k)\) is binary, we solve the distribution separation problem (SP\(_j\)) with the ambiguity set \( \mathcal{P} \) for all \( j \in \mathcal{J} \), and obtain the optimal solution \( \{\hat{p}_{\omega}^k\}_{\omega \in \Omega} \) and the objective value \( \text{uobj}^k \). We add probability and feasibility cuts to (LMP) if \( \text{uobj}^k \) 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, \((\hat{y}^k, \hat{z}^k)\) is a feasible solution to (DR-CAP), we update the upper bound and record the corresponding solution \((\hat{y}^k, \hat{z}^k)\) (line 15-20). If \((\hat{y}^k, \hat{z}^k)\) 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 \((\hat{y}^*, \hat{z}^*)\) (line 33).

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

This manuscript is for review purposes only.
Algorithm 1: Branch-and-Cut Algorithm with Probability Cuts

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

while ($N$ is nonempty) do

3. Select a node $o \in N$, $N \leftarrow N - \{o\}$.

4. Solve (LMP) at the node $o$. $k = k + 1$.

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

6. if $lobj^k < UB$ then

7. if $(y^k, z^k)$ is an integer then

8. for $j \in \mathcal{J}$ do

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

10. if $uobj^k < 1 - \varepsilon$ then

11. Add the cuts (4.2) and (4.3) to (LMP).

end

if Cuts (4.2) and (4.3) are found then

14. Go to step 5.

else

16. $UB = lobj^k$, $(y^*, z^*) = (y^k, z^k)$.

end

end

if $(y^k, z^k)$ is fractional then

18. Use Algorithm 4 and 5 to find the violated inequalities.

19. if Violated inequalities are found then

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

else

23. Branch, resulting in nodes $o^*$ and $o^{**}$, $N \leftarrow N \cup \{o^*, o^{**}\}$.

end

end

end

end

end

end

end

end

end

end

end

end

end

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

Theorem 4.1. If there exists an oracle that solves (SP$_j$) to optimality, then Algorithm 1 terminates in finitely many iterations. If $UB < +\infty$, $UB$ is the optimal value of (DR-CAP) and Algorithm 1 obtains an optimal solution $(y^*, z^*)$ at termination.

Proof See Appendix A.12. □

Recall that $\mathcal{P}_W$ in (2.5) is a polyhedral set with a finite number of extreme points, thus (SP$_j$) can be solved to optimality.

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. Performance of the branch-and-cut algorithm (Algorithm 6) 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 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. Table 5 in Appendix C.1 provides the mean and standard deviation of the surgery
duration, and the percentage for each surgery type. For the problem instances, the log-normal distribution
with the mean, and the standard deviation provided in Table 5 was used to generate surgery duration
samples (see 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 (|J| = 8) ORs are available to serve |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, \( p_j \), is limited to
[3, 5], \( \forall j \in J \). We used the number of surgeries and the percentage for each surgery type (given in Table
5) 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 (3.8) 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 (3.15). Therefore, we added the violated
inequalities from (3.15) 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
(3.8), 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 this 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 dynamic programming approach described in
Appendix B.1 to compute \( m^\omega_j(k) \) for \( j \in J \) and \( \omega, k \in \Omega \) (defined in problem (2.3)). An easier way
to compute \( m^\omega_j(k) \) is to let \( m^\omega_j(k) = \text{maximize} \{ \sum_{i \in I} \xi_i^\omega y_{ij} \mid \sum_{i \in I} \xi_i^k y_{ij} \leq t_j \} \), i.e., ignoring the cardinality
constraint in (2.3). 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 C.2. Comparing results from Table 6 in Appendix C.2 with those in Table 1, we
see the computational trade-offs resulting from using this 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.
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>ε</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>0.12</td>
<td>CPX 165.0</td>
<td>52.8</td>
<td>-</td>
<td>1.725</td>
<td>-</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-1 165.0</td>
<td>33.3</td>
<td>1.7</td>
<td>1.446</td>
<td>283</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-2 165.0</td>
<td>47.1</td>
<td>14.7</td>
<td>1.076</td>
<td>300</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-G 165.0</td>
<td>65.5</td>
<td>8.6</td>
<td>1.959</td>
<td>9</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>CPX 641.1</td>
<td>781.4</td>
<td>-</td>
<td>3.715</td>
<td>-</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-1 641.1</td>
<td>659.4</td>
<td>10.9</td>
<td>10.788</td>
<td>563</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-2 641.1</td>
<td>502.4</td>
<td>76.0</td>
<td>3.756</td>
<td>561</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-G 641.1</td>
<td>739.5</td>
<td>101.0</td>
<td>3.715</td>
<td>12</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.08</td>
<td>CPX 165.0</td>
<td>125.5</td>
<td>-</td>
<td>3.477</td>
<td>-</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-1 165.0</td>
<td>122.5</td>
<td>1.2</td>
<td>4.688</td>
<td>210</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-2 165.0</td>
<td>136.0</td>
<td>10.6</td>
<td>3.641</td>
<td>224</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-G 165.0</td>
<td>136.7</td>
<td>10.6</td>
<td>2.926</td>
<td>12</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.06</td>
<td>CPX 641.1</td>
<td>102.5</td>
<td>-</td>
<td>6.492</td>
<td>-</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-1 641.1</td>
<td>329.2</td>
<td>4.3</td>
<td>5.919</td>
<td>346</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-2 641.1</td>
<td>305.3</td>
<td>29.2</td>
<td>5.832</td>
<td>320</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cover-G 641.1</td>
<td>481.0</td>
<td>42.8</td>
<td>5.669</td>
<td>12</td>
<td>5/5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

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 ε = 0.08 and N = 500, 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 ε = 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 \( \varepsilon = 0.06 \) is infeasible. It makes is hard to decide how
many and which surgeries are assigned to the pseudo OR while satisfying the chance constraint with
\( \varepsilon = 0.06 \), and minimizing the total cost. Nevertheless, for these problems, 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 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 versus 69,798) when compared
to Cover-2 variant, this reduction did not translate into a reduction in the overall solution time (28,248
versus 28,014 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 (3.8) of (DR-CAP). Using the sample average distribution, we let \( \hat{q} := \hat{q} \) (Corollary
2.3) for the big-M calculations in (2.6). For (3.8), we set the coefficient \( \gamma \) as \( \delta_{\hat{k} \delta} \) (Theorem 3.7). 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 (3.8) to the branch-and-cut algorithm with prob-
  ability cuts (Algorithm 1).

We solved the instances generated in Section 5.3 with the Wasserstein set \( \mathcal{P}_W \) 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 (\( \eta = 0.1, \eta = 1, N = 1500 \)). 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

This manuscript is for review purposes only.
Table 2: The average CPU time (in seconds) for 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>0.08</td>
<td>500</td>
<td>CPX</td>
<td>272.7</td>
<td>-</td>
<td>-</td>
<td>61.9</td>
<td>6,277</td>
<td>3</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>144.0</td>
<td>2.8</td>
<td>54.4</td>
<td>3,589</td>
<td>410</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>147.0</td>
<td>11.5</td>
<td>45.9</td>
<td>3,379</td>
<td>250</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>147.0</td>
<td>0.19</td>
<td>-</td>
<td>588,891</td>
<td>-</td>
<td>0/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>272.7</td>
<td>-</td>
<td>-</td>
<td>273.9</td>
<td>9,476</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>144.0</td>
<td>2.8</td>
<td>54.4</td>
<td>3,589</td>
<td>410</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>147.0</td>
<td>11.5</td>
<td>45.9</td>
<td>3,379</td>
<td>250</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>147.0</td>
<td>0.19</td>
<td>-</td>
<td>588,891</td>
<td>-</td>
<td>0/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>272.7</td>
<td>-</td>
<td>-</td>
<td>273.9</td>
<td>9,476</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>144.0</td>
<td>2.8</td>
<td>54.4</td>
<td>3,589</td>
<td>410</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>147.0</td>
<td>11.5</td>
<td>45.9</td>
<td>3,379</td>
<td>250</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>147.0</td>
<td>0.19</td>
<td>-</td>
<td>588,891</td>
<td>-</td>
<td>0/5</td>
<td></td>
</tr>
<tr>
<td>0.06</td>
<td>500</td>
<td>CPX</td>
<td>648.6</td>
<td>-</td>
<td>-</td>
<td>239.7</td>
<td>9,476</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-1</td>
<td>144.0</td>
<td>2.8</td>
<td>54.4</td>
<td>3,589</td>
<td>410</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>147.0</td>
<td>11.5</td>
<td>45.9</td>
<td>3,379</td>
<td>250</td>
<td>2</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>147.0</td>
<td>0.19</td>
<td>-</td>
<td>588,891</td>
<td>-</td>
<td>0/5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>CPX</td>
<td>1,390.0</td>
<td>-</td>
<td>-</td>
<td>1,089.9</td>
<td>12,089</td>
<td>-</td>
<td>5/5</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>12,148</td>
<td>447</td>
<td>8</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>884.0</td>
<td>33.0</td>
<td>397.5</td>
<td>10,720</td>
<td>373</td>
<td>9</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,390.0</td>
<td>-</td>
<td>-</td>
<td>1,089.9</td>
<td>12,089</td>
<td>-</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>CPX</td>
<td>1,390.0</td>
<td>-</td>
<td>-</td>
<td>1,089.9</td>
<td>12,089</td>
<td>-</td>
<td>5/5</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>12,148</td>
<td>447</td>
<td>8</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-2</td>
<td>884.0</td>
<td>33.0</td>
<td>397.5</td>
<td>10,720</td>
<td>373</td>
<td>9</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cover-G</td>
<td>1,390.0</td>
<td>-</td>
<td>-</td>
<td>1,089.9</td>
<td>12,089</td>
<td>-</td>
<td>5/5</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&CP 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.

---

This manuscript is for review purposes only.
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 incumbent problem is used to warm-start the solution of the new problem after a probability cut is added.

5.5. Performance of Tightening Big-M in (SIP) Using the 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 2.2 and Corollary 2.3. While we found that the solution time required by the linear programs in Theorem 2.2 is not justified, we did find computational value in using Corollary 2.3 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 \( \bar{q} \) as valid inequalities and adding these inequalities to CPX. For CPX-UM, we update \( \bar{q} \) defined in Corollary 2.3 as new \( \{p_{\omega}\}_{\omega \in \Omega} \) becomes available in the probability cuts. We set \( \bar{q} = \hat{q} \) and add constraints (2.6c) 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.

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>888.2</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>

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 for other instances it 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. 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 (3.8) to improve the performance of Cover-2 described 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.

Table 4: The average total cost (Avg-cost), the average overtime probability (Avg-prob), the worst-case overtime probability (Worst-prob), the average overtime (Avg-over) (in minutes), and 85%, 95%, 99% quantiles (in minutes) for (CAP) and (DR-CAP) are reported.

<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></td>
<td>1000</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></td>
<td>1500</td>
<td>(CAP)</td>
<td>70.1</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></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></td>
<td>1000</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></td>
<td>1500</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></td>
<td>1000</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></td>
<td>1500</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>

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 0.088. 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, 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 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 an increasing sample size. However, the solution time for the distributionally robust chance-constrained models appear 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 and the ONR grant N00014-18-1-2097. This paper also benefited from a discussion with Professor Simge Küçükyavuz and Hamed Rahimian.

References.

Yan Deng, Siqian Shen, and Brian Denton. Chance-constrained surgery planning under conditions of limited and ambiguous data. *Available at SSRN 2432775*, 2016.


*This manuscript is for review purposes only.*
A SOLUTION APPROACH TO DISTRIBUTIONALLY ROBUST CHANCE-CONSTRAINED ASSIGNMENT


Appendix A. Proof of Propositions and Theorems.

A.1. Proof of Proposition 2.1. Let $y_j^*$ be an optimal solution of (2.2). Then, there exists at least one $k' \in \{k_1, \ldots, k_q\}$ such that $\sum_{i \in I} \xi_{i}^{k'} y_{ij}^* \leq t_j$. Otherwise, we have $\sum_{i \in I} \xi_{i}^{k} y_{ij}^* > t_j$, for $k \in \{k_1, \ldots, k_q\}$.

Since $\sum_{j=1}^{q} p_{kj} > \varepsilon$, the inequality $\mathbb{P}\left\{ \sum_{i \in I} \xi_i y_{ij}^* \leq t_j \right\} \geq 1 - \varepsilon$ is violated. This is a contradiction. Therefore, $y_j^*$ is a feasible solution of (2.3) with $k = k'$. We have $m_{ij}^w(k+1) \geq m_{ij}^w(k') \geq \sum_{i \in I} \xi_i y_{ij}^* = \bar{M}_{ij}^w$. Thus, $m_{ij}^w(k+1)$ is an upper bound for $\bar{M}_{ij}^w$. □

A.2. Proof of Theorem 2.2. We first show that $m_{ij}^w(k_q)$ is an upper bound for $\bar{M}_{ij}^w$, where

\[(A.1a) \quad \bar{M}_{ij}^w := \max_{y} \sum_{i \in I} \xi_i^w y_{ij}, \]
\[(A.1b) \quad \text{subject to } \sum_{i \in I} y_{ij} \leq \rho_j, \quad \forall j \in J, \]
\[(A.1c) \quad \inf_{\mathcal{P} \in \mathcal{P}} \mathbb{P}\left\{ \sum_{i \in I} \xi_i y_{ij} \leq t_j \right\} \geq 1 - \varepsilon, \quad \forall j \in J. \]
Let $y_{ij}^*$ be an optimal solution of (A.1), then there exist at least one $k' \in \bar{\Omega} := \{1, \cdots, \bar{q}\}$ such that 
\[
\sum_{i \in I} \xi_i^k y_{ij}^* \leq t_j.
\]
Otherwise, $\sum_{i \in I} \xi_i^k y_{ij}^* > t_j$ for all $k \in \{1, \cdots, \bar{q}\}$. We have 
\[
\inf_{p \in \mathcal{P}} \mathbb{P} \left\{ \sum_{i \in I} \xi_i^k y_{ij}^* \leq t_j \right\} = \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega} p_\omega \mathbb{I} \left( \sum_{i \in I} \xi_i^k y_{ij}^* \leq t_j \right) 
\[
\leq \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega \setminus \bar{\Omega}} p_\omega = \inf_{p \in \mathcal{P}} \left( 1 - \sum_{\omega \in \Omega} p_\omega \right) 
\[
= 1 - \sup_{p \in \mathcal{P}} \sum_{\omega \in \bar{\Omega}} p_\omega < 1 - \varepsilon,
\]
which is a contradiction. Thus, $m_{ij}^\omega (k_q) \geq m_{ij}^\omega (k') \geq M_{ij}^\omega$, which implies that (2.6c) hold. Therefore, (DR-CAP) can be rewritten as (2.6).

A.3. Proof of Proposition 2.4. Let $(y, z)$ be a feasible solution of the relaxation problem of the binary bilinear reformulation of (DR-CAP). We have
\[
\sum_{i \in I} \xi_i^\omega y_{ij} z_j^\omega - \sum_{i \in I} \xi_i^\omega y_{ij} - m_{ij}^\omega (k_q) (z_j^\omega - 1) = (z_j^\omega - 1) \left( \sum_{i \in I} \xi_i^\omega y_{ij} - m_{ij}^\omega (k_q) \right) \geq 0.
\]
Consequently, the following inequality
\[
\sum_{i \in I} \xi_i^\omega y_{ij} + m_{ij}^\omega (k_q) (z_j^\omega - 1) \leq \sum_{i \in I} \xi_i^\omega y_{ij} z_j^\omega \leq m_{ij}^\omega (\omega) z_j^\omega
\]
holds. Therefore, $(y, z)$ is a feasible solution of the relaxation problem of (2.6). The proof can be similarly extend to (CAP). □

A.4. Proof of Proposition 2.5. The set $\mathcal{H} = \bigcap_{j \in \mathcal{J}} \{ (y, z) | (y_j, z_j) \in \mathcal{G}_j \}$ 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 $|\mathcal{I}| + N$ affinely independent points that satisfy this inequality at equality. Because this inequality does not have coefficients with respect to a pair of $(y_{ji}, z_{ji})$ for $j_1 \in \mathcal{J}$ and $j_1 \neq j$, we can extend the $|\mathcal{I}| + N$ affinely independent points to a set of $|\mathcal{I}| \times |\mathcal{J}| \times N$ affinely independent points by appropriately setting the values of $(y_{ji}, z_{ji})$ for each $j_1 \in \mathcal{J}$ and $j_1 \neq j$. □

A.5. Proof of Proposition 3.2. The inequality (3.2) is valid for (3.1) based on the definition of $\mathcal{C}$.

Consider the following $|\mathcal{C} \setminus \mathcal{D}|$ feasible points of (3.1): for $k \in \mathcal{C} \setminus \mathcal{D}$, set $y_{ij} = 1, \forall i \in \mathcal{C} \setminus \mathcal{D}$, $y_{ij} = 0, \forall i \in k \cup (\mathcal{I} \setminus \mathcal{C})$, and $y_{ij} = 1, \forall i \in \mathcal{D}$; These $|\mathcal{C} \setminus \mathcal{D}|$ points are affinely independent and satisfy (3.2) at equality. When $|\mathcal{C}| \leq \rho_j + 1$, these $|\mathcal{C} \setminus \mathcal{D}|$ points are feasible. □

A.6. Proof of Lemma 3.3. Suppose that there exists $\hat{y}_j$ that serves as a member of the set $\{y_j \in \{0, 1\}^{\mathcal{I}} | \sum_{i \in I} \xi_i^\omega y_{ij} \leq m_{ij}^\omega (\omega), y_{ij} \in \mathcal{Y}_j, y_{ij} = 1, \forall i \in \mathcal{D} \}$ such that $\sum_{i \in \mathcal{C} \setminus \mathcal{D}} \hat{y}_{ij} \leq |\mathcal{C} \setminus \mathcal{D}| - 1$ and

This manuscript is for review purposes only.
\[
\sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \pi(k)} \alpha_i \hat{y}_{ij} > |\mathcal{C}\setminus \mathcal{D}| - 1. \quad \text{Let}\ r := \max\{k| \sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \pi(k)} \alpha_i \hat{y}_{ij} \leq |\mathcal{C}\setminus \mathcal{D}| - 1\}. \quad \text{We have}
\]

\[
\sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \pi(r+1)} \alpha_i \hat{y}_{ij} = \sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \pi(r)} \alpha_i \hat{y}_{ij} + \alpha_{\pi_{r+1}} \hat{y}_{\pi_{r+1}, j}
\]

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

which is a contradiction. Thus, (3.3) is valid for (3.5).

A.7. Proof of Lemma 3.4. Suppose that we have \(\hat{y}_{ij} \in \mathcal{Q}_{j\omega}\) that violates (3.6). \(\kappa\) can be partitioned into \(\mathcal{D}^0 := \{i \in \kappa| \hat{y}_{ij} = 0\}\) and \(\mathcal{D}^1 := \{i \in \kappa| \hat{y}_{ij} = 1\}\). We assume that the last element in the set \(\mathcal{D}^0\) is \(\kappa_h\) where \(h \leq |\mathcal{D}|\). Then, we have

\[
\sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{C}\setminus \mathcal{C}} \alpha_i \hat{y}_{ij} > |\mathcal{C}\setminus \mathcal{D}| + \sum_{i \in \mathcal{D}^0} \beta_i - 1.
\]

Note that

\[
|\mathcal{C}\setminus \mathcal{D}| + \sum_{i \in \mathcal{D}^0} \beta_i - 1 = |\mathcal{C}\setminus \mathcal{D}| + \text{obj}_{\kappa_h} - \sum_{i \in \kappa(h-1)} \beta_i - |\mathcal{C}\setminus \mathcal{D}| + 1 + \sum_{i \in \mathcal{D}^0\setminus \kappa_h} \beta_i - 1
\]

\[
= \text{obj}_{\kappa_h} - \sum_{i \in \kappa(h-1)} \beta_i + \sum_{i \in \mathcal{D}^0\setminus \kappa_h} \beta_i.
\]

Based on the definition of \(\text{obj}_{\kappa_h}\), we have that \(\hat{y}_{ij}\) is a feasible solution of (3.7) with \(l = h\). Then,

\[
\text{obj}_{\kappa_h} - \sum_{i \in \kappa(h-1)} \beta_i + \sum_{i \in \mathcal{D}^0\setminus \kappa_h} \beta_i \geq \sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{C}\setminus \mathcal{C}} \alpha_i \hat{y}_{ij} + \sum_{i \in \kappa(h-1)} \beta_i \hat{y}_{ij} - \sum_{i \in \kappa(h-1)} \beta_i + \sum_{i \in \mathcal{D}^0\setminus \kappa_h} \beta_i
\]

\[
= \sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{C}\setminus \mathcal{C}} \alpha_i \hat{y}_{ij}.
\]

This is a contradiction. Thus, (3.6) is valid for \(\text{conv}(\mathcal{Q}_{j\omega})\).

A.8. Proof of Theorem 3.5. When \(z_{\omega} = 1\), (3.8) is valid for \(\text{conv}(\mathcal{F}_{j\omega})\) because of Lemma 3.4. When \(z_{\omega} = 0\), due to the definition of \(\gamma\), (3.8) is also valid for \(\text{conv}(\mathcal{F}_{j\omega})\). Thus, (3.8) is valid for \(\text{conv}(\mathcal{F}_{j\omega})\).

Consider the following \(|\mathcal{I}| + 1\) feasible points of \(\text{conv}(\mathcal{F}_{j\omega})\): when \(z_{\omega} = 1\), there exists \(|\mathcal{I}|\) feasible points of \(\text{conv}(\mathcal{F}_{j\omega})\) that are affinely independent and satisfy (3.8) at equality based on the Lemma 3.4; when \(z_{\omega} = 0\), let \(\mathcal{y}_{j}\) be the optimal solution of (3.9). These \(|\mathcal{I}| + 1\) feasible points satisfy (3.8) at equality and are affinely independent. Thus, (3.8) is facet-defining for \(\text{conv}(\mathcal{F}_{j\omega})\).

\[(A.2a) \quad \gamma = \max_{y_j \in \{0,1\}^{|\Omega|}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} - |C \setminus D| - \sum_i \beta_i + 1 \]

subject to \[
\sum_{i \in \Omega \setminus \{\omega\}} p_k \mathbb{I} \left( \sum_{i \in I} \xi_i^k y_{ij} \leq t_j \right) \geq 1 - \varepsilon,
\]

\[(A.2c) \quad \sum_{i \in I} y_{ij} \leq \rho_j. \]

Because \(y_j\) satisfies the chance constraint (1.1d) and \(z_\omega = 0\) for computing \(\gamma\), the inequality (3.8) is valid for (CAP).

Let \(\hat{y}_j\) be an optimal solution of (A.2). Then, there exists at least one \(k' \in \{k_1, \ldots, k_q\}\) such that \(\sum_{i \in I} \xi_i^k \hat{y}_{ij} \leq t_j\). Otherwise, if \(\sum_{i \in I} \xi_i^k \hat{y}_{ij} > t_j\) for all \(k \in \{k_1, \ldots, k_q\}\), then \(\sum_{k \in \{k_1, \ldots, k_q\}} p_k \mathbb{I} \left( \sum_{i \in I} \xi_i^k \hat{y}_{ij} > t \right) > \varepsilon\), which indicates that (A.2b) is violated by \(\hat{y}_j\). Therefore, \(\hat{y}_j\) is a feasible solution of (3.10) for \(k = k'\). We have \(\delta_{k_1} \geq \delta_{k'} \geq \gamma\), and (3.8) is a valid inequality for (CAP) when \(\gamma = \delta_{k_1}\).

A.10. Proof of Theorem 3.7. Let

\[(A.3a) \quad \gamma = \max_{y_j \in \{0,1\}^{|\Omega|}} \sum_{i \in C \setminus D} y_{ij} + \sum_{i \in I \setminus C} \alpha_i y_{ij} + \sum_{i \in D} \beta_i y_{ij} - |C \setminus D| - \sum_i \beta_i + 1 \]

subject to \[
\inf_{\rho \in \mathcal{G}} \sum_{k \in \Omega \setminus \{\omega\}} p_k \mathbb{I} \left( \sum_{i \in I} \xi_i^k \hat{y}_{ij} \leq t_j \right) \geq 1 - \varepsilon,
\]

\[(A.3c) \quad \sum_{i \in I} y_{ij} \leq \rho_j. \]

Because \(y_j\) satisfies the chance constraint (1.2b) and \(z_\omega = 0\) for computing \(\gamma\), the inequality (3.8) is valid for (DR-CAP).

Let \(\hat{y}_j\) be an optimal solution of (A.3). Then, \(\sum_{i \in I} \xi_i^k \hat{y}_{ij} \leq t_j\) for at least one \(k' \in \{k_1, \ldots, k_q\}\). Otherwise, if \(\sum_{i \in I} \xi_i^k \hat{y}_{ij} > t_j\) for all \(k \in \{k_1, \ldots, k_q\}\), we have \(\sup_{\rho \in \mathcal{G}} \sum_{k \in \{k_1, \ldots, k_q\}} p_k \mathbb{I} \left( \sum_{i \in I} \xi_i^k \hat{y}_{ij} > t \right) > \varepsilon\), which indicates that (A.3b) is violated by \(\hat{y}_j\). Therefore, \(\hat{y}_j\) is a feasible solution of (3.10) for \(k = k'\). We have \(\delta_{k_q} \geq \delta_{k'} \geq \gamma\), and (3.8) is a valid inequality for (DR-CAP) when \(\gamma = \delta_{k_q}\).

A.11. Proof of Theorem 3.8. We first prove that for (CAP) if the coefficients are described in Theorem 3.8, then, (3.15) is valid for \(\text{conv}(\mathcal{G}_j)\). For \(k \in \{1, \ldots, |\Omega|\}\), let \((\hat{y}_j, \hat{z}_j) \in \mathcal{G}_j\). If \(\hat{z}_{j\omega} = 1\) for \(\omega \in \Omega_k\), then (3.15) is valid for \(\text{conv}(\mathcal{G}_j)\). Otherwise, let \(\tau\) be partitioned into \(\Omega_k^0 = \{\omega \in \tau | \hat{z}_{j\omega} = 0\}\) and \(\Omega_k^1 = \{\omega \in \tau | \hat{z}_{j\omega} = 1\}\). We assume that the last element of \(\Omega_k^0\) is \(\tau_h\) where \(h \leq |\Omega_k|\). (3.15) becomes

\[\sum_{i \in C \setminus D} \hat{y}_{ij} + \sum_{i \in I \setminus C} \bar{\alpha}_i \hat{y}_{ij} + \sum_{i \in D} \bar{\beta}_i \hat{y}_{ij} \leq |C \setminus D| + \sum_i \bar{\beta}_i - 1 + \sum_{\omega \in \Omega_k^0} \bar{\gamma}_\omega.\]
Note that
\[ |\mathcal{C}\setminus \mathcal{D}| + \sum_{i \in \mathcal{D}} \bar{\beta}_i - 1 + \sum_{\omega \in \Omega_k^p} \bar{\gamma}_\omega = \text{obj}_{\mathcal{T}_h} - \sum_{\omega \in \mathcal{T}(h-1)} \bar{\gamma}_\omega + \sum_{\omega \in \Omega_k \setminus \mathcal{T}_h} \bar{\gamma}_\omega. \]

Since \((\hat{y}_j, \hat{z}_j)\) satisfies (3.19) with \(k = h\), we have
\[ \text{obj}_{\mathcal{T}_h} - \sum_{\omega \in \mathcal{T}(h-1)} \bar{\gamma}_\omega + \sum_{\omega \in \Omega_k \setminus \mathcal{T}_h} \bar{\gamma}_\omega \]
\[ \geq \sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \alpha_i \hat{y}_{ij} + \sum_{i \in \mathcal{D}} \bar{\alpha}_i \hat{y}_{ij} + \sum_{\omega \in \mathcal{T}(h-1)} \bar{\gamma}_\omega \hat{z}_{j\omega} - \sum_{\omega \in \mathcal{T}(h-1)} \bar{\gamma}_\omega + \sum_{\omega \in \Omega_k \setminus \mathcal{T}_h} \bar{\gamma}_\omega \]
\[ = \sum_{i \in \mathcal{C}\setminus \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{C} \setminus \mathcal{D}} \alpha_i \hat{y}_{ij} + \sum_{i \in \mathcal{D}} \bar{\beta}_i \hat{y}_{ij}. \]

Thus, (3.15) is valid for \(\text{conv}(\mathcal{G}_j)\) when \(\bar{\gamma}_{\mathcal{T}_1} = \text{obj}_{\mathcal{T}_h} - |\mathcal{C}\setminus \mathcal{D}| - 1 - \sum_{i \in \mathcal{D}} \bar{\beta}_i - \sum_{\omega \in \mathcal{T}(h-1)} \bar{\gamma}_\omega\) for \(l = 1, \ldots, |\Omega_k|\).

Since \(\text{obj}_{\mathcal{T}_1}^p\) is an upper bound on \(\text{obj}_{\mathcal{T}_h}\) and all the coefficients in (3.15) are integers, \(\text{obj}_{\mathcal{T}_1}^p\) is also an upper bound on \(\text{obj}_{\mathcal{T}_h}\). Therefore, (3.15) is valid for \(\text{conv}(\mathcal{G}_j)\) when \(\bar{\gamma}_{\mathcal{T}_1} = |\mathcal{C}\setminus \mathcal{D}| + 1 - \sum_{i \in \mathcal{D}} \bar{\beta}_i - \sum_{\omega \in \mathcal{T}(h-1)} \bar{\gamma}_\omega\) for \(l = 1, \ldots, |\Omega_k|\). The proof can be similarly extended to \(\mathcal{G}_j^p\). This completes the proof.

A.12. Proof of Theorem 4.1. The algorithm processes a finite number of nodes as it is based on branching on a finite number of binary variables, and given that \(z^k\) is obtained from (MP), we solve \(|\mathcal{J}|\) distribution separation problems with \(\mathcal{P}\). 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 only 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 (4.2) and (4.3) can remove the current infeasible solution and never cut off any feasible solutions of (DR-CAP). It can be verified that (4.2) and (4.3) can remove the current infeasible solution. Also,
\[ \sum_{\omega \in \Omega} p^k_{\omega} z_{\omega} \geq \inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega} p_{\omega} z_{\omega} \geq 1 - \varepsilon. \]

Thus, (4.2) never cuts off any feasible solutions of (DR-CAP). We assume that \(\hat{y}\) is a new future solution from (MP) and the corresponding set \(\hat{\mathcal{I}}_j^1\). Let \(\hat{y}_{ij} = \hat{y}_{ij}\), for \(i \in \mathcal{I}\). Then the feasibility cut (4.3) becomes
\[ \sum_{i \in \mathcal{I}^1_j} \hat{y}_{ij} \leq |\mathcal{I}^1_j| - 1, \]
which is decomposed to
\[ \sum_{i \in \mathcal{I}^1_j \cap \hat{\mathcal{I}}_j^1} \hat{y}_{ij} + \sum_{i \in \mathcal{I}^1_j \setminus \hat{\mathcal{I}}_j^1} \hat{y}_{ij} \leq |\mathcal{I}^1_j \cap \hat{\mathcal{I}}_j^1| + |\mathcal{I}^1_j \setminus \hat{\mathcal{I}}_j^1| - 1 \iff \sum_{i \in \mathcal{I}^1_j \setminus \hat{\mathcal{I}}_j^1} \hat{y}_{ij} \leq |\mathcal{I}^1_j \setminus \hat{\mathcal{I}}_j^1| - 1, \]

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

Appendix B. Algorithm Details.

This manuscript is for review purposes only.
B.1. Dynamic Programming for Estimating big-M. Let \( D(I, t_j, \rho_j) \) represents (2.3), where \( I \) denotes the \( |I| \) variables of \( y_j \). Let us consider the subproblem \( D(n, t_j^0, \rho_j^0) \) of \( D(I, t_j, \rho_j) \) which includes the first \( n \) variables of \( y_j \) and the right-hand side values of constraints in (2.3) respectively. Let \( S(n, t_j^0, \rho_j^0) \) be the optimal objective value of \( D(n, t_j^0, \rho_j^0) \). If \( D(n, t_j^0, \rho_j^0) \) is infeasible, we set \( S(n, t_j^0, \rho_j^0) = -\infty \). Since \( y_{nj} \) is binary, if \( y_{nj} = 0 \), \( S(n, t_j^0, \rho_j^0) \) is equal to \( S(n - 1, t_j^0, \rho_j^0) \), which is the optimal objective value of the subproblem \( D(n - 1, t_j^0, \rho_j^0) \). If \( y_{nj} = 1 \), \( S(n, t_j^0, \rho_j^0) \) is equal to \( S(n - 1, t_j^0 - \xi_k, \rho_j^0 - 1) + \xi_n \), which is the optimal objective value of the subproblem \( D(n - 1, t_j^0 - \xi_k, \rho_j^0 - 1) \) plus \( \xi_n \). Thus, we have

\[
S(n, t_j^0, \rho_j^0) = \max\{S(n - 1, t_j^0, \rho_j^0), S(n - 1, t_j^0 - \xi_k, \rho_j^0 - 1) + \xi_n\},
\]

where \( n = 2, \ldots, |I|, \) and \( t_j^0 \leq t_j, \rho_j^0 \leq \rho_j \). It is easy to verify that the dynamic programming procedure has \( O(|I| (\text{max}\{t_j, \rho_j\})^2) \) time complexity.

B.2. Dynamic Programming for Up-lifting Coefficient. Dynamic programming has been used to calculate the up-lifting coefficients in the binary single knapsack problem (see, Zemel (1989)). We use this technique to obtain the lifting coefficient \( \alpha_i \). For each \( k = 1, \ldots, |C\setminus D| \), we solve the following problem for \( \lambda_1 = 0, \ldots, |C\setminus D| - 1 \), and \( \lambda_2 = 0, \ldots, \rho_j - 1 - |D| \):

\[
A_{\pi_k}(\lambda_1, \lambda_2) = \min_{y_j \in \{0,1\}^{(C \setminus D) \cup \pi(k-1)}} \sum_{i \in C \setminus D} \xi^\omega_{ij} y_{ij} + \sum_{i \in \pi(k-1)} \xi^\omega_i y_{ij},
\]

subject to

\[
\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \pi(k-1)} \alpha_i y_{ij} \geq \lambda_1,
\]

\[
\sum_{i \in C \setminus D} y_{ij} + \sum_{i \in \pi(k-1)} y_{ij} \leq \lambda_2.
\]

Let \( l_t, \ t = 0, \ldots, |C\setminus D| - 1 \) be the
A SOLUTION APPROACH TO DISTRIBUTIONALLY ROBUST CHANCE-CONSTRAINED ASSIGNMENT

sum of the $t$ smallest $\xi^i_{\omega}$, $i \in C\setminus D$. Algorithm 2 gives an outline of our dynamic programming framework.

Algorithm 2: Dynamic Programming for the Lifting Coefficients

1. for $\lambda_2 = 0, \ldots, \rho_j - 1 - |D|$ do
2. for $\lambda_1 = 0, \ldots, |C\setminus D| - 1$ do
3. if $\lambda_1 \leq \lambda_2$ then
4. $A_{\pi_1}(\lambda_1, \lambda_2) = l_{\lambda_1}$,
5. end
6. else
7. $A_{\pi_1}(\lambda_1, \lambda_2) = +\infty$,
8. end
9. end
10. end
11. for $k = 1, \ldots, |I\setminus C|$ do
12. $\text{obj}_{\pi_k} = \max \left\{ 1 : A_{\pi_k}(\lambda_1, \rho_j - 1 - |D|) \leq m^r_j(\omega) - \xi^i_{\omega} - \sum_{i \in D} \xi^i_{\omega} \right\}$.
13. $\alpha_{\pi_k} = |C\setminus D| - 1 - \text{obj}_{\pi_k}$
14. for $\lambda_2 = 0, \ldots, \rho_j - 1 - |D|$ do
15. for $\lambda_1 = 0, \ldots, |C\setminus D| - 1$ do
16. if $\lambda_1 \geq \alpha_{\pi_k}$ and $\lambda_2 \geq 1$ then
17. $A_{\pi_{k+1}}(\lambda_1, \lambda_2) = \min \left\{ A_{\pi_k}(\lambda_1, \lambda_2), A_{\pi_k}(\lambda_1 - \alpha_{\pi_k}, \lambda_2 - 1) + \zeta^i_{\omega} \right\}$.
18. end
19. end
20. $A_{\pi_{k+1}}(\lambda_1, \lambda_2) = A_{\pi_k}(\lambda_1, \lambda_2)$.
21. end
22. end
23. end
24. end

B.3. Probability Cut Algorithm. Algorithm 3 gives an outline of the probability cut algorithm for obtaining the optimal objective value $\text{obj}_{\pi_l}^r$ defined in Section 3.2.3

Algorithm 3: Probability Cut Algorithm

1. Initialize $\text{obj} = 0$, the number of iterations $k = 1$, and the maximal number of iterations $K = 100$.
2. while $(\text{obj} < 1 - \varepsilon \& \& k < K)$ do
3. Solve the following problem.
4. \[
\text{maximize } \sum_{i \in C\setminus D} y_{ij} + \sum_{i \in \bar{C}} \bar{\alpha}_i y_{ij} + \sum_{i \in D} \bar{\beta}_i y_{ij} + \sum_{\omega = r_1, \tau_l} \gamma_{\omega} z_{j\omega}
\]
5. \[
\text{subject to (3.19d), (3.20c)},
\]
6. \[
\sum_{\omega \in \Omega_l \Omega_k} p_{ij} z_{j\omega} + \sum_{\omega \in \pi_{l+1}} p_{ij} z_{j\omega} \geq 1 - \varepsilon - \sum_{\omega = r_{l+1}} p_{ij}, \quad l = 1, \ldots, k - 1.
\]
7. Record the optimal solution $(y^k, z^k)$ and objective value $\text{obj}^k$.
8. Fix $z$ to be $z^k$, solve the distribution separation problem.
9. Obtain the optimal solution $p^k$ and objective value $\text{obj}^k$.
10. $k = k + 1$;
11. return $\text{obj}^{k-1}$.

This manuscript is for review purposes only.
B.4. Separation Heuristic for (3.8). Algorithm 4 gives an overview of the separation heuristic for (3.8).

Algorithm 4: Separation Heuristic for (3.8)

1. Given the LP relaxation optimal solution \((\hat{y}, \hat{z})\).
2. for \(j = 1, \ldots, |J|\) do
   3. for \(\omega = 1, \ldots, N\) do
      4. if \(\hat{z}_{ij\omega} = 1\) then
         5. Sort \(\hat{y}_j\) in non-increasing order: \(\hat{y}_{i1j} \geq \ldots \geq \hat{y}_{i|I|j}\).
         6. Let \(C = \{i_1, \ldots, i_o\}\) where \(o \leq |I|\) is a smallest number such that \(C\) is a cover.
         7. Delete elements from the tail of \(C\) to get a minimal cover \(C\).
         8. Let \(D = \{i \in C : \hat{y}_{ij} = 1\}\) and \(I_0 = \{i \in I \setminus C : \hat{y}_{ij} = 0\}\).
         9. Calculate the up-lifting coefficient \(\alpha_i\) for \(i \in I \setminus (C \cup I_0)\).
        10. if \(\sum_{i \in C \setminus D} \hat{y}_{ij} + \sum_{i \in I \setminus (C \cup I_0)} \alpha_i \hat{y}_{ij} > |C \setminus D| - 1\) then
            11. Calculate the down-lifting coefficient \(\beta_i\) for \(i \in D\).
            12. Calculate the up-lifting coefficient \(\alpha_i\) for \(i \in I_0\).
            13. Calculate \(\delta_k\) for \(k \in \Omega \setminus \omega\), set \(\gamma = \delta_{k_1}\) for (CAP) and \(\gamma = \delta_{k_2}\) for (DR-CAP).
            14. Obtain the violated inequality (3.8).
        15. end
      6. end
    5. end
  4. end
3. end


Algorithm 5: Separation Heuristic for (3.15)

1. Given the LP relaxation optimal solution \((\hat{y}, \hat{z})\).
2. for \(j = 1, \ldots, |J|\) do
   3. for \(\omega = 1, \ldots, N\) do
      4. if \(\sum_{\omega \in \Omega} p_{\omega} \hat{z}_{ij\omega} \geq 1 - \varepsilon\) (for (CAP)) or \(\inf_{p \in \mathcal{P}} \sum_{\omega \in \Omega} p_{\omega} \hat{z}_{ij\omega} \geq 1 - \varepsilon\) (for (DR-CAP)) then
         5. Sort \(\hat{y}_j\) in non-increasing order: \(\hat{y}_{i1j} \geq \ldots \geq \hat{y}_{i|I|j}\).
         6. Let \(C = \{i_1, \ldots, i_o\}\) where \(o \leq |I|\) is a smallest number such that \(C\) is a cover for \(\omega\).
         7. Delete elements from the tail of \(C\) to get a minimal cover \(C\).
         8. Let \(D = \{i \in C : \hat{y}_{ij} = 1\}\) and \(I_0 = \{i \in I \setminus C : \hat{y}_{ij} = 0\}\).
         9. Calculate the up-lifting coefficient \(\alpha_i\) for \(i \in I \setminus (C \cup I_0)\).
        10. if \(\sum_{i \in C \setminus D} \hat{y}_{ij} + \sum_{i \in I \setminus (C \cup I_0)} \alpha_i \hat{y}_{ij} > |C \setminus D| - 1\) then
            11. Calculate the down-lifting coefficient \(\beta_i\) for \(i \in D\).
            12. Calculate the up-lifting coefficient \(\alpha_i\) for \(i \in I_0\).
            13. Calculate \(\gamma_{\omega}\) for \(\omega \in \Omega \setminus \omega\), set \(\gamma = \gamma_{k_1}\) for (CAP) and \(\gamma = \gamma_{k_2}\) for (DR-CAP).
            14. Obtain the violated inequality (3.15).
        15. end
      6. end
    5. end
  4. end
3. end

This manuscript is for review purposes only.
Algorithm 6: Branch-and-Cut Implementation

1. Initialize $UB = +\infty$, $LB = -\infty$, the iteration number $k = 0$.
2. Initialize Node list $\mathcal{N} = \{o\}$, where $o$ is a branching node without constraints.
3. while ($\mathcal{N}$ is nonempty) do
   4. Select a node $o \in \mathcal{N}$, $\mathcal{N} \leftarrow \mathcal{N}/\{o\}$.
   5. At the node $o$, solve the LP relaxation problem of (IP). $k = k + 1$.
   6. Obtain the optimal solution $(y^k, z^k)$ and the objective value $obj^k$.
   7. if $obj^k < UB$ then
      8. if $(y^k, z^k)$ is fractional then
         9. Use Algorithm 4 and Algorithm 5 to find the violated inequalities.
      10. if Violated inequalities are found then
          11. Add the violated inequalities to the LP relaxation problem.
          12. Go to line 5.
      13. end
   14. else
      15. Branch, resulting in nodes $o^*$ and $o^{**}$, $\mathcal{N} \leftarrow \mathcal{N} \cup \{o^*, o^{**}\}$.
      16. end
   17. end
   18. else
      19. Update $UB$, $UB = obj^k$, $(y^*, z^*) = (y^k, z^k)$.
   20. end
   21. end
22. return $UB$ and its corresponding optimal solution $(y^*, z^*)$.

Appendix C. Tables.

C.1. Statistics about Surgery Data. Table 5 gives the mean and standard deviation of the surgery duration, and the percentage for each surgery type.

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>Gyneacology</td>
<td>1.1</td>
<td>1.3</td>
<td>0.29</td>
</tr>
<tr>
<td>Galactophore</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>Orthopeadic</td>
<td>3.2</td>
<td>1.8</td>
<td>0.03</td>
</tr>
</tbody>
</table>

C.2. Computational Results for Loose Big-M of (CAP). Table 6 reports the average time for the big-M coefficient computations, the branch-and-cut algorithm time, the average number of nodes, and the number of instances solved to optimality for the five generated instances for the weaker big-M coefficients described in Section 5.2.

This manuscript is for review purposes only.
Table 6: The average CPU time (in seconds) for big-M coefficients (AvT-M), and branch-and-cut algorithm (AvT-B&C), the average number of nodes (# of nodes), and the number of solved instances from the five instances (solved) for (CAP) are reported.

<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>8,459.90</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>