Chance-Constrained Multiple Bin Packing Problem with an Application to Operating Room Planning

Shanshan Wang
Department of Management Science and Engineering, Beijing Institute of Technology, Beijing, China 100081
shshwang_bit@163.com

Jinlin Li
Department of Management Science and Engineering, Beijing Institute of Technology, Beijing, China 100081
jinlinli@bit.edu.cn

Sanjay Mehrotra
Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 60208
mehrotra@iems.northwestern.edu

We study the chance-constrained bin packing (CBP) problem, with an application to hospital operating room planning. The bin packing problem allocates items of random size that follow a discrete distribution to a set of bins with limited capacity, while minimizing the total cost. The bin capacity constraints are satisfied with a given probability. We investigate a big-M and a 0-1 bilinear formulation of this problem. We analyze the bilinear structure of the formulation and use the lifting techniques to identify cover, clique and projection inequalities to strengthen the formulation. We show that in certain cases these inequalities are facet defining for a bilinear knapsack constraint that arises in the reformulation. An extensive computational study is conducted for the operating room planning problem that minimizes the number of open operating rooms. The computational tests are performed using problems generated based on real data from a hospital. A lower bound improvement heuristic is combined with the cuts proposed in this paper in a branch-and-cut framework. The computations illustrate that the techniques developed in this paper can significantly improve the performance of the branch-and-cut method. Problems with up to 1,000 scenarios are solved to optimality in less than an hour. A safe-approximation based on conditional value at risk (CVaR) is also solved. The computations show that the CVaR approximation typically leaves a gap of one operating room (e.g., six instead of five) to satisfy the chance constraint.

Key words: chance-constrained stochastic programming, bin packing, bilinear integer program, branch-and-cut, valid inequalities, operating room planning

1. Introduction

The bin packing problem is to assign a set of items with positive size to bins so as to minimize the total cost, while satisfying the bin capacity constraints. The bin packing problem has been applied in various fields. Application examples include healthcare (Denton et al. 2010, Deng and Shen 2016), scheduling (Reich et al. 2016, Song et al. 2018), transportation and logistics (Crainic et al. 2016, Perboli et al. 2014), etc. For many practical bin packing problems, the item sizes are uncertain. For instance, surgery duration is uncertain in healthcare operations management, as planners often do not know the exact duration of a surgery in advance. Disregarding the uncertainty in item size might provide a solution that violates the bin capacity constraints with an undesirable probability. In a stochastic programming framework, the chance constraint paradigm can be employed to overcome the above concern. More specifically, the chance-constrained bin packing problem requires that the bin capacity constraints are satisfied with a prespecified probability. For instance, the chance constraints provide a probabilistic guarantee for each operating room to finish the assigned surgeries without overtime. Ensuring that operating room shifts end at a specified time is desirable to achieve a work-life balance of the service providers. In this article we study a chance-constrained bin packing problem to handle the uncertainty in the item size for the case where the item size follows a discrete distribution.

In the following, Section 1.1 and 1.2 review the literature on chance-constrained programs including convex conservative approximations and mixed-integer formulation of these problems, and bin packing problem.

1.1. Literature Review on Chance-Constrained Programs

Chance-constrained programs (CCPs) were introduced by Charnes and Cooper (1959). Since then, CCP has been extensively studied in terms of new methodological developments and its applications. For more details about CCP, readers are referred to Ahmed and Shapiro (2008), Nemirovski (2012), Birge and Louveaux (2011) and references therein. CCP problems are generally very difficult to solve because of their non-convex feasible region (Ahmed and Shapiro 2008). Moreover, the chance constraint does not necessarily preserve the smoothness of the original constraints (Hu et al. 2013). Only in the case of normally distributed random variate, they admit a second-order cone program formulation (Song et al. 2014). In many situations, however, the probability distributions are not normally distributed. This is the case when considering surgery times in the operating room, which are observed to follow a log-normal distribution.

1.1.1. Convex Conservative Approximations of Chance-Constrained Programs A number of approaches have been developed to obtain a solution of CCP. One possible approach is to use a convex conservative approximation of CCP whose feasible set is contained in the CCP, and it is tractable (Nemirovski and Shapiro 2006). This includes the use of Bernstein approximation proposed by (Nemirovski and Shapiro 2006), and CVaR approximation studied in (Rockafellar et al. 2000, Wang and Ahmed 2008). The Bernstein approximation is applicable when the components of the random vector are independent and moment-generating functions are computable. This approximation is efficient to solve. Unfortunately, however, the Bernstein approximation can be conservative. The CVaR approximation is the best conservative convex approximation (Nemirovski 2012). Despite being convex, CVAR approximations remain computationally challenging to solve (see Appendix B regarding CVaR approximation of (CBP)). Nevertheless, it is a worth considering such approximations for a difficult problem.

1.1.2. Mixed-Integer Formulation of Chance-Constrained Programs Under this framework, one assumes that the true probability distribution of $\boldsymbol{\xi}$ is replaced by a finite number of samples. In order to satisfy the chance constraint, Luedtke and Ahmed (2008) used a mixed-integer formulation. The formulation ensures that a correct number of sampled constraints are satisfied. This has motivated a number of studies to model chance-constrained programs under the assumption of finite support, and using its formulation as a mixed-integer linear program (MILP) (Luedtke et al. 2010, Küçükyavuz 2012, Luedtke 2014, Zhao et al. 2017, Peng et al. 2019, Ahmed and Xie 2018). The justification for using a finite sample-based approximation is motivated from the fact that, as the sample size increases, the approximation may represent the true constraint (Pagnoncelli et al. 2009). Nevertheless, solving such approximations present a major challenge. In particular, the problems are very challenging when technology matrices are random, i.e., problems where randomness appears in the coefficients of the model, as is the case with the chance constrained bin packing problems. Therefore, cutting plane methods with enhanced strategies for CCP have been proposed in the literature (Ruszczyński 2002, Tanner and Ntaimo 2010, Luedtke 2014, Qiu et al. 2014, Xie and Ahmed 2018). Recently, computational methods using Lagrangian relaxation and scenario decomposition are also developed for obtaining a lower bound for these problems (Watson et al. 2010, Ahmed et al. 2017, Deng et al. 2017). The aforementioned papers only study the generic CCP and none of them exploit the constraint structure of CCP.

1.2. Literature Review on Bin Packing Problem

The classical bin packing problem involves packing of a set of items of given sizes in a minimum number of bins. The deterministic problem has been extensively studied, and exact and approximation algorithms have been developed for this problem (Wei et al. 2020, Scheithauer 2017, Kucukyilmaz and Kiziloz 2018). Well known variations and generalizations of the classical bin packing problem include multidimensional bin packing problem (Lodi et al. 1999, Martello et al. 2000), variable size bin packing problem (Kang and Park 2003, Correia et al. 2008), and the dynamic bin packing (Berndt et al. 2020, Coffman et al. 1983). We refer the interested readers to the review papers by Lodi et al. (2002)

and Coffman et al. (2013) and the references therein for more details on this topic. In terms of chance-constrained bin packing problem, Kleinberg et al. (2000) applied stochastic bin packing to bandwidth allocation for bursty connections in high-speed networks. The authors developed an approximation algorithm to solve the chance-constrained model. Shylo et al. (2012) modeled the stochastic operating room scheduling problem by embedding probabilistic capacity constraints to restrict the overtime. The authors assumed that surgery durations follow a multivariate normal distribution, and formulated the model as a MILP problem. Deng and Shen (2016) developed a decomposition method with acceleration strategies for solving a chance-constrained appointment scheduling problem. Zhang et al. (2020) considered the two-stage stochastic and distributionally robust chance-constrained bin packing problems with binary assignment and continuous bin extension decisions. Problems with 500 scenarios and a probability tolerance of 0.1 for chance constraints on total item size within the bins were solved by a column generation based branch-and-price method. Zhang et al. (2018) studied the distributionally robust chance-constrained bin packing problem with mean-covariance information. The authors formulated the problem as a second order cone program, and developed valid inequalities to improve the algorithmic performance in a branch-and-cut framework.

The problem studied in this paper is a generalization of Song et al. (2014) in that it has multiple bins and chance constraints. Song et al. (2014) considered the single bin case. The authors proposed a probabilistic cover and a delayed constraint generation approach within a big-M framework. A coefficient strengthening procedure was provided for the big-M formulation, and a lifting technique proposed by Zemel (1989) was used to obtain the probabilistic cover inequalities. The problems were solved using a branch-and-cut algorithm, and several types of cutting planes were tested to compare the algorithm's performance.

1.3. Contributions of this Paper

This paper makes the following contributions. Specifically,

- We first formulate (CBP) as a binary bilinear program. We show that this formulation provides a stronger relaxation than a strengthened big-M reformulation of (CBP).
- We use the binary bilinear formulation to obtain several new valid inequalities for (CBP). In particular, we consider the binary bilinear knapsack set obtained from a single row and scenario in the bilinear constraints. We propose cover and clique inequalities for the convex hull of the set by using a lifting technique. We show that these inequalities are facet-defining. We also linearize the bilinear formulation using additional binary variables and project the relaxation of the linearized formulation onto the space of the original variables to generate projection inequalities.
- We incorporate the valid inequalities within a branch-and-cut framework to solve the strengthened big-M reformulation of (CBP). The big-M strengthening generalizes the concepts introduced in

Song et al. (2014) for our setting. Computational experiments for operating room scheduling problem using real data from a hospital are conducted to demonstrate the improvement from the proposed techniques. For the problem that minimizes the number of bins, we present a lower bound generation technique based on relaxing the scenario variables. We find that this technique significantly improves the algorithmic performance in our computational experiments. The performance is further improved by a systematic inclusion of the developed inequalities.

- Using the techniques developed in this paper, we solved instances with up to 1,000 scenarios for $\varepsilon = 0.05, 0.15$ within two hours and $\varepsilon = 0.1$ within an hour. A comparison also shows that on the tested problems the approach outperforms the algorithm proposed in Song et al. (2014). Several models remained unsolved (gap in the objective value ≥ 1) when using a commercial solver without enhancements. We also compared the results with a CVaR approximation, and find that CVaR formulation typically leaves a gap of one room to the optimal number of rooms required to satisfy the chance constraint. Moreover, it does not seem to provide any computational benefit.

1.4. Organization

The remainder of this paper is organized as follows. Section 2 formulates (CBP) as a binary program and adapts the technique of Song et al. (2014) to strengthen its big-M coefficients. We then present an alternative binary bilinear formulation for (CBP). We show that this bilinear formulation has a tighter relaxation than the big-M formulation. We explore the structure of the bilinear formulation to develop three classes of valid inequalities in Section 3. Specifically, we show in Section 3.1 how the sequential lifting technique can be used to generate the facet-defining cover and clique inequalities. We linearize the bilinear formulation and study the projection inequalities for (CBP) in Section 3.2. In Section 4 we incorporate the valid inequalities within a branch-and-cut solution scheme to solve the strengthened big-M reformulation of (CBP), and propose a lower bound improvement heuristic to accelerate the computation for the problem that minimizes the number of bins. Section 5 reports computational results on (CBP) formulation with application to operating room planning, and shows the efficiency of the techniques developed in this paper. Section 6 concludes the paper with a summary of the important findings. Appendix A provides CVaR approximation of (CBP). Appendix B gives a performance comparison of (CBP) solutions with CVaR approximation. Appendix C gives proofs of some propositions and theorems. An algorithm based on the probabilistic cover approach, generalized to our problem, is given in Appendix D. Appendix E gives details about dynamic programming approach for computing big-M values in the model formulation.

2. Model Reformulations

In this paper we study the problem (CBP):

$$\text{(CBP)} \qquad \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} c_{ij}^b y_{ij} \tag{1a}$$

$$\text{subject to } y_{ij} \le x_j, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, \tag{1b}$$

$$\sum_{j \in \mathcal{J}} y_{ij} = 1, \qquad \forall i \in \mathcal{I}, \tag{1c}$$

$$\mathbb{P}\left\{\sum_{i \in \mathcal{I}} \xi_i y_{ij} \le t_j\right\} \ge 1 - \varepsilon, \qquad \forall j \in \mathcal{J}, \tag{1d}$$

$$x_j \in \{0,1\}, y_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, \tag{1e}$$

where $\mathcal{I} := \{1,\dots,|\mathcal{I}|\}$ is a collection of items and $\mathcal{J} := \{1,\dots,|\mathcal{J}|\}$ is a collection of bins, and $|\cdot|$ is the cardinality of a set. c_j^a is the nonnegative cost for opening bin j, and c_{ij}^b is the nonnegative cost for assigning item i to bin j. $\boldsymbol{\xi} = (\xi_1,\dots,\xi_{|\mathcal{I}|})^\top$ is a vector of random sizes with a joint probability distribution $\mathbb{P}$. We assume that the random vector $\boldsymbol{\xi}$ are drawn from a finite support of N scenarios $\{\boldsymbol{\xi}^\omega\}_{\omega \in \Omega}$, where $\Omega = \{1,\dots,N\}$. Hence, the distribution $\mathbb{P}$ can be characterized using a probability vector $(p_1,\dots,p_N)^\top$ such that $p_\omega \ge 0$ and $\sum_{\omega \in \Omega} p_\omega = 1$. Let ξ_i^ω denote the size of item i for the scenario $\omega \in \Omega$, let $\varepsilon \in [0,1]$ be the level of chance satisfaction, and t_j be the capacity of bin j. Without loss of generality, we assume that $\xi_i^\omega < t_j$, for all $i \in \mathcal{I}, j \in \mathcal{J}, \omega \in \Omega$. The binary variable x_j for all $j \in \mathcal{J}$ is such that $x_j = 1$ if bin j is open, and $x_j = 0$ otherwise. The binary variable y_{ij} for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$ is such that $y_{ij} = 1$ if item i is assigned to bin j, $y_{ij} = 1$ otherwise. Let $\boldsymbol{x} = (x_1,\dots,x_{|\mathcal{I}|})^\top$, $\boldsymbol{y}_j = (y_{1j},\dots,y_{|\mathcal{I}|j})^\top$, $\boldsymbol{y} = (\boldsymbol{y}_1,\dots,\boldsymbol{y}_{|\mathcal{J}|})^\top$. The objective (1a) is to minimize the total cost for opening the bins, and the cost of assigning items to the opened bins. Constraints (1b) guarantee that item i can be assigned to bin j only if bin j is open. Constraints (1c) enforce that each item i is assigned to exactly one bin. Constraints (1d) restrict the bin capacity for bin j with probability $1-\varepsilon$. Constraints (1e) define binary variables x_j and y_{ij}. In a special case all c_j^a are equal, $c_{ij}^b = 0$, $\forall i \in \mathcal{I}, j \in \mathcal{J}$; and the problem reduces to that of finding the minimum number of bins to pack the items.

We first formulate (CBP) as a binary integer program and use the big-M coefficient strengthening procedure from Song et al. (2014) for (CBP) in Section 2.1. By applying the techniques in Luedtke (2014), we obtain mixing set inequalities for (CBP) in Section 2.1.2. We then present an alternative binary bilinear reformulation for (CBP) in Section 2.2. A result on the strength of this binary bilinear formulation is given in this section (see Proposition 4).

2.1. Big-M Reformulation for (CBP)

Let us introduce a binary variable z_j^ω to indicate if bin j satisfies the bin capacity constraint for each scenario $\omega \in \Omega$, namely,

$$z_j^\omega = \begin{cases} 1, & \text{if } \sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} \le t_j, \\ 0, & \text{otherwise.} \end{cases}$$

For $j \in \mathcal{J}$, let $\boldsymbol{z}_j = (z_j^1,\dots,z_j^N)^\top$, $\boldsymbol{z} = (\boldsymbol{z}_1,\dots,\boldsymbol{z}_{|\mathcal{J}|})^\top$.

Using the big-M approach, the chance constraints (1d) can be written as

$$\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} + (M_j^\omega - t_j) z_j^\omega \le M_j^\omega, \qquad \forall j \in \mathcal{J}, \omega \in \Omega, \tag{2a}$$

$$\sum_{\omega\in\Omega} p_\omega z_j^\omega \ge 1-\varepsilon, \qquad \forall j \in \mathcal{J}, \tag{2b}$$

where M_j^ω is a constant with sufficiently large value, guaranteeing that constraints (2a) are inactive when $z_j^\omega = 0$. Note that $z_j^\omega = 1$ ensures that $\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \le t_j$. (CBP) with new constraints (2a) and (2b) may provide a weak linear programming (LP) relaxation bound if M_j^ω is chosen naively. For example, a choice is possible by taking $M_j^\omega := \sum_{i\in\mathcal{I}} \xi_i^\omega$. Note that for $j \in \mathcal{J}, \omega \in \Omega$, M_j^ω is valid for constraints (2a) if

$$M_j^\omega \ge \bar{M}_j^\omega := \underset{\boldsymbol{y}_j}{\text{maximize}} \left\{ \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \middle| \mathbb{P}\left\{ \sum_{i\in\mathcal{I}} \xi_i y_{ij} \le t_j \right\} \ge 1-\varepsilon,\, \boldsymbol{y}_j \in \{0,1\}^{|\mathcal{I}|} \right\}. \tag{3}$$

We now describe a *coefficient strengthening procedure* for strengthening this big-M formulation. The procedure borrows ideas from Qiu et al. (2014) and Song et al. (2014). We then develop the *mixing set inequalities* for (CBP) by applying the techniques in Luedtke (2014).

2.1.1. Coefficient Strengthening Procedure Given $j \in \mathcal{J}$, and $\omega \in \Omega$, let us consider the following problem (4) for each $k \in \Omega$,

$$m_j^\omega(k) = \underset{\boldsymbol{y}_j}{\text{maximize}} \left\{ \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \middle| \sum_{i\in\mathcal{I}} \xi_i^k y_{ij} \le t_j,\, \boldsymbol{y}_j \in \{0,1\}^{|\mathcal{I}|} \right\}. \tag{4}$$

Now sort $m_j^\omega(k)$ in a non-decreasing order such that $m_j^\omega(k_1) \le \ldots \le m_j^\omega(k_N)$. The following proposition gives an upper bound for $\bar{M}_j^\omega$.

PROPOSITION 1. *For $j \in \mathcal{J}$ and $\omega \in \Omega$, $m_j^\omega(k_{q+1})$ is an upper bound for $\bar{M}_j^\omega$, where $q := \max\left\{ l : \sum_{j=1}^{l} p_{k_j} \le \varepsilon \right\}$. Furthermore, (2a) may be replaced by*

$$\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} + m_j^\omega(k_{q+1})(z_j^\omega - 1) \le m_j^\omega(\omega) z_j^\omega. \tag{5}$$

Proof See Appendix C.1. □

Hence, (CBP) can be formulated as the binary integer program (6):

$$\text{(IP)} \qquad \underset{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}{\text{minimize}} \sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} c_{ij}^b y_{ij} \tag{6a}$$

$$\text{subject to } (1b), (1c), (1e), (2b), (5)$$

$$z_j^\omega \in \{0,1\} \qquad \forall j \in \mathcal{J}, \omega \in \Omega. \tag{6b}$$

Solving (IP) with the above big-M coefficient strengthening strategy may be time-consuming as the number of problems in (4) significantly increases with $|\mathcal{J}|$ and N. For $i \in \mathcal{I}, \omega \in \Omega$, we assume that ξ_i^ω is non-negative integer. Note that (4) is a standard 0/1 Knapsack Problem (KP). We used a dynamic programming based implementation to solve the knapsack problem (4). The details of this implementation are given in Appendix E.

2.1.2. Mixing Set Inequalities Mixing set inequalities were introduced by Atamtürk et al. (2000) and Günlük and Pochet (2001). Recently, Luedtke et al. (2010) and Luedtke (2014) used them in strengthening the mixed integer programming formulation of chance-constrained programs. By applying the techniques in Luedtke (2014), we obtain the mixing set inequalities (7) for (CBP) in Proposition 2.

PROPOSITION 2. *Given $j \in \mathcal{J}$, and $\omega \in \Omega$, let $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_l\} \subseteq \{k_1, \ldots, k_q\}$ with $m_j^\omega(\tau_1) \le \ldots \le m_j^\omega(\tau_l)$. Then the inequality*

$$\sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} + \sum_{n=1}^{l} \left(m_j^\omega(\tau_{n+1}) - m_j^\omega(\tau_n) \right) z_j^{\tau_n} \le m_j^\omega(k_{q+1}) \tag{7}$$

is valid for (CBP), where $m_j^\omega(\tau_{l+1}) = m_j^\omega(k_{q+1})$ and q is determined from Proposition 1.

Proof See Appendix C.2. □

2.2. Binary Bilinear Integer Reformulation for (CBP)

The problem (IP) in Section 2.1 is derived by using the big-M approach to guarantee that constraints (2a) hold when $z_j^\omega = 0$. We now provide an alternative formulation for (CBP).

Let binary variables z_j^ω be defined as in Section 2.1, and consider the following formulation (8):

$$\text{(BIP)} \qquad \underset{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}{\text{minimize}} \sum_{j \in \mathcal{J}} c_j^a x_j + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij}^b y_{ij} \tag{8a}$$

$$\text{subject to } (1b), (1c), (1e), (2b), (6b)$$

$$\sum_{i \in \mathcal{I}} \xi_i^\omega y_{ij} z_j^\omega \le m_j^\omega(\omega) z_j^\omega, \qquad \forall j \in \mathcal{J}, \omega \in \Omega. \tag{8b}$$

The following proposition shows the equivalence of (BIP) and (IP).

PROPOSITION 3. *Let $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ be an optimal solution of (CBP). Then there exists $\boldsymbol{z}^*$ such that $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP). Conversely, if $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP), then $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an optimal solution of (CBP).*

Proof See Appendix C.3. □

Let (RIP) and (RBIP) be the relaxation problems of (IP) and (BIP) respectively, where the integrality restriction on the variables $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{z}$ is relaxed. The feasible solution sets in terms of $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ to (RIP) and (RBIP) are denoted by $\mathcal{X}_{RIP}$ and $\mathcal{X}_{RBIP}$, respectively. The following relationship between (RIP) and (RBIP) motivates us to explore the binary bilinear formulation.

PROPOSITION 4. $\mathcal{X}_{RBIP} \subseteq \mathcal{X}_{RIP}$.

Proof Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathcal{X}_{RBIP}$. We have

$$\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} z_j^\omega - \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} - m_j^\omega(k_{q+1})(z_j^\omega - 1) = (z_j^\omega - 1)\left(\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} - m_j^\omega(k_{q+1})\right) \geq 0.$$

Consequently, the following inequality holds,

$$\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} + m_j^\omega(k_{q+1})(z_j^\omega - 1) \leq \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} z_j^\omega \leq m_j^\omega(\omega) z_j^\omega.$$

Therefore, $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathcal{X}_{RIP}$, proving that $\mathcal{X}_{RBIP} \subseteq \mathcal{X}_{RIP}$. □

Proposition 4 shows that (BIP) provides a stronger relaxation than the relaxation possible from the strengthened big-M approach.

3. Valid Inequalities Using the Bilinear Formulation

We now show how the formulation proposed in Section 2.2 can be used to generate valid inequalities for (CBP). Our analysis relies on investigating a binary bilinear knapsack set.

3.1. Strong Inequalities for Single Binary Bilinear Knapsack

We assume that $j \in \mathcal{J}, \omega \in \Omega$ are fixed in this section. Let us consider the following binary bilinear knapsack set,

$$\mathcal{F}_{j\omega} = \left\{ (\boldsymbol{y}_j, z_j^\omega) \in \{0,1\}^{|\mathcal{I}|} \times \{0,1\} \,\middle|\, \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} z_j^\omega \leq m_j^\omega(\omega) z_j^\omega \right\}.$$

We use $conv(\cdot)$ to denote the convex hull of a set. The inequalities valid for $conv(\mathcal{F}_{j\omega})$ are also valid for (CBP). We now develop a binary bilinear lifting technique to derive valid inequalities for the set $conv(\mathcal{F}_{j\omega})$. More specifically, we develop two different types of valid inequalities. The first inequality is obtained using a general form of cover inequalities. The second is obtained using clique inequalities as the seed inequalities, and computing the lifting coefficients for the variable z_j^ω.

3.1.1. Lifted Cover Inequalities Lifting techniques have been used to develop valid inequalities for the binary linear knapsack problem (see, for example Zemel (1989), Gu et al. (1998, 2000)). We now show its applicability to the binary bilinear knapsack set $\mathcal{F}_{j\omega}$. Let us first consider a 0-1 knapsack constraint $\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \leq m_j^\omega(\omega)$.

Let

$$\mathcal{Q}_{j\omega} = \left\{ \boldsymbol{y}_j \in \{0,1\}^{|\mathcal{I}|} \,\middle|\, \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \leq m_j^\omega(\omega) \right\}.$$

Note that the set $\mathcal{Q}_{j\omega}$ is obtained from $\mathcal{F}_{j\omega}$ for $z_j^\omega = 1$. If $\sum_{i\in\mathcal{C}} \xi_i^\omega > m_j^\omega(\omega)$, the set $\mathcal{C} \subseteq \mathcal{I}$ is called a cover. The cover $\mathcal{C}$ is minimal if no subset of $\mathcal{C}$ is a cover. It is straightforward to see that the cover inequality $\sum_{i\in\mathcal{C}} y_{ij} \leq |\mathcal{C}| - 1$ is valid for $conv(\mathcal{Q}_{j\omega})$. A stronger cover inequality is obtained when the cover is minimal. In this paper, let $\mathcal{C}$ be a minimal cover. The cover inequality is obtained from a restricted set of variables. The coefficients for the remaining variables, as given in

$$\sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} \le |\mathcal{C}| - 1, \tag{9}$$

are obtained from a lifting procedure, where the lifting coefficient α_i is computed sequentially (Zemel 1989). Let $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_{|\mathcal{I}\backslash\mathcal{C}|}\}$ be a sequence of set $\mathcal{I}\backslash\mathcal{C}$. For $k = 1, \dots, |\mathcal{I}\backslash\mathcal{C}|$, the lifting problem is as follows:

$$\begin{aligned} obj_{\pi_k} := \underset{\boldsymbol{y}_j}{\text{maximize}} & \sum_{i\in\mathcal{C}} y_{ij} + \sum_{i=\pi_1}^{\pi_{k-1}} \alpha_i y_{ij} \\ \text{subject to } & \sum_{i\in\mathcal{C}} \xi_i^\omega y_{ij} + \sum_{i=\pi_1}^{\pi_{k-1}} \xi_i^\omega y_{ij} \le m_j^\omega(\omega) - \xi_{\pi_k}^\omega, \\ & y_{ij} \in \{0,1\}, \qquad\qquad i \in \mathcal{C} \bigcup \{\pi_1, \dots, \pi_{k-1}\}. \end{aligned}$$

The following result from Padberg (1973) shows that the inequalities (9) are facet-defining for $conv(\mathcal{Q}_{j\omega})$.

Lemma 1 (**Padberg (1973)**). *For $k = 1, \dots, |\mathcal{I}\backslash\mathcal{C}|$, let $\alpha_{\pi_k} = |\mathcal{C}| - 1 - obj_{\pi_k}$. The inequality (9) is facet-defining for $conv(\mathcal{Q}_{j\omega})$.* □

Facet-defining inequalities for $conv(\mathcal{F}_{j\omega})$ can be obtained from the facet-defining inequalities in Lemma 1.

Proposition 5. *The inequality*

$$\sum_{i\in\mathcal{C}} y_{ij} + z_j^\omega \le |\mathcal{C}| \tag{10}$$

is valid for $conv(\mathcal{F}_{j\omega})$.

Proof For $z_j^\omega = 1$ and $z_j^\omega = 0$, it is easy to verify that inequality (10) is valid for $conv(\mathcal{F}_{j\omega})$. □

Theorem 1. *The lifted cover inequality*

$$\sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} + \gamma(z_j^\omega - 1) \le |\mathcal{C}| - 1 \tag{11}$$

is facet-defining for $conv(\mathcal{F}_{j\omega})$, where $\gamma = \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i + 1$.

Proof Let

$$\left.\begin{aligned} \gamma = \underset{\boldsymbol{y}_j, z_j^\omega}{\text{maximize}} & \ \frac{\sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} - |\mathcal{C}| + 1}{1 - z_j^\omega} \\ \text{subject to } & (\boldsymbol{y}_j, z_j^\omega) \in \mathcal{F}_{j\omega}, \ z_j^\omega = 0. \end{aligned}\right\} \iff \left\{\begin{aligned} \gamma = \underset{\boldsymbol{y}_j}{\text{maximize}} & \ \sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} - |\mathcal{C}| + 1 \\ \text{subject to } & y_{ij} \in \{0,1\}, \ \forall i \in \mathcal{I}. \end{aligned}\right.$$

We have $\gamma = \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i + 1$. Hence, (11) is valid for $conv(\mathcal{F}_{j\omega})$ when $z_j^\omega = 0$. Because of the validity of inequality (9), (11) is valid for $conv(\mathcal{F}_{j\omega})$ when $z_j^\omega = 1$.

When $z_j^\omega = 1$, there exists n feasible points of variables $\boldsymbol{y}_j$ that are affinely independent and satisfy inequality (11) at equality as the facet-defining inequalities in Lemma 1. Similarly, when $z_j^\omega = 0$, $\boldsymbol{y}_j = 1_{|\mathcal{I}|}$, where $1_{|\mathcal{I}|}$ is a $1 \times |\mathcal{I}|$ vector of all ones. Thus, the $|\mathcal{I}|+1$ feasible points are affinely independent and satisfy inequality (11) at equality. Therefore, we conclude that the inequality (11) is facet-defining for $conv(\mathcal{F}_{j\omega})$. □

We can restrict the feasible region of $\boldsymbol{y}_j$ using the additional constraints in (CBP) to compute a better value of the coefficient γ in (11), by considering additional constraint in (1d) using $k \in \Omega\setminus\{\omega\}$.

THEOREM 2. *For $k \in \Omega\setminus\{\omega\}$, let*

$$\delta_k = \underset{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}}{\text{maximize}} \sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i y_{ij} - |\mathcal{C}| + 1 \tag{12a}$$

$$\text{subject to } \sum_{i\in\mathcal{I}} \xi_i^k y_{ij} \le m_j^k(k). \tag{12b}$$

Sort δ_k in a nondecreasing order such that $\delta_{k_1} \le \ldots \le \delta_{k_{N-1}}$. Let q be defined as in Proposition 1. Then, $\delta_{k_{q+1}}$ is an upper bound on γ, and (11) is a valid inequality for (CBP) when $\gamma = \delta_{k_{q+1}}$.

Proof Since $\boldsymbol{y}$ satisfies constraints (1d), the inequality (11) is valid for (CBP) when

$$\gamma = \underset{\boldsymbol{y}_j, z_j^\omega}{\text{maximize}} \frac{\sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i y_{ij} - |\mathcal{C}| + 1}{1 - z_j^\omega} \tag{13a}$$

$$\text{subject to } (\boldsymbol{y}_j, z_j^\omega) \in \mathcal{F}_{j\omega}, \ z_j^\omega = 0 \tag{13b}$$

$$\mathbb{P}\left\{\sum_{i\in\mathcal{I}} \xi_i y_{ij} \le t_j\right\} \ge 1 - \varepsilon. \tag{13c}$$

Since $z_j^\omega = 0$, (13) can be rewritten as

$$\gamma = \underset{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}}{\text{maximize}} \sum_{i\in\mathcal{C}} y_{ij} + \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i y_{ij} - |\mathcal{C}| + 1 \tag{14a}$$

$$\text{subject to } \sum_{k\in\Omega\setminus\{\omega\}} p_k \mathbf{1}\left\{\sum_{i\in\mathcal{I}} \xi_i^k y_{ij} \le t\right\} \ge 1 - \varepsilon. \tag{14b}$$

Let $\boldsymbol{y}_j^*$ be an optimal solution of (14). Then, there exists at least one $k' \in \{k_1, \ldots, k_{q+1}\}$ such that $\sum_{i\in\mathcal{I}} \xi_i^{k'} y_{ij}^* \le t_j$, otherwise, (14b) is violated by $\boldsymbol{y}_j^*$. Therefore, $\boldsymbol{y}_j^*$ is a feasible solution of (12) for $k = k'$. We have $\delta_{k_{q+1}} \ge \delta_{k'} \ge \gamma$, and (11) is a valid inequality for (CBP) when $\gamma = \delta_{k_{q+1}}$. □

3.1.2. General Lifted Cover Inequality As noted by Gu et al. (1998), a more general form of the cover inequality in the binary linear knapsack problem is as follows:

$$\sum_{i\in\mathcal{C}\backslash\mathcal{D}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} + \sum_{i\in\mathcal{D}} \beta_i y_{ij} \le |\mathcal{C}\backslash\mathcal{D}| + \sum_{i\in\mathcal{D}} \beta_i - 1, \tag{15}$$

where set $\mathcal{D}\subseteq\mathcal{C}$. Computing the coefficients α and β is called up-lifting and down-lifting, respectively. When $\mathcal{D}=\varnothing$, inequality (15) is same as the inequality (9). Gu et al. (1998) argued that inequality (15) resulted in a more effective branch-and-cut algorithm.

The following sequence of problems are solved to obtain the down-lifting coefficients. Let $\boldsymbol{\kappa} = \{\kappa_1,...,\kappa_{|\mathcal{D}|}\}$ be a sequence in the set $\mathcal{D}$. For $k=1,\cdots,|\mathcal{D}|$, let

$$\begin{aligned}
\text{obj}_{\kappa_k} = \underset{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}}{\text{maximize}} & \sum_{i\in\mathcal{C}\backslash\mathcal{D}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} + \sum_{i=\kappa_1}^{\kappa_{k-1}} \beta_i y_{ij}\\
\text{subject to } & \sum_{i\in\mathcal{I}} \xi_i^{\omega} y_{ij} \le m_j^{\omega}(\omega),\\
& y_{\kappa_k j}=0,\ \ y_{ij}=1, \qquad\qquad i\in\{\kappa_{k+1},...,\kappa_{|\mathcal{D}|}\}.
\end{aligned}$$

Lemma 2 (**Gu et al. (1998)**). *For $k=1,\dots,|\mathcal{D}|$, let $\beta_{\kappa_k} = obj_{\kappa_k} - \sum_{i=\kappa_1}^{\kappa_{k-1}}\beta_i - |\mathcal{C}\backslash\mathcal{D}| + 1$. The inequality (15) is facet-defining for $conv(\mathcal{Q}_{j\omega})$.* □

Next, we consider the general lifted cover inequality for $conv(\mathcal{F}_{j\omega})$.

Theorem 3. *The general lifted cover inequality*

$$\sum_{i\in\mathcal{C}\backslash\mathcal{D}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} + \sum_{i\in\mathcal{D}} \beta_i y_{ij} + \gamma(z_j^{\omega}-1) \le |\mathcal{C}\backslash\mathcal{D}| + \sum_{i\in\mathcal{D}} \beta_i - 1 \tag{16}$$

is facet-defining for $conv(\mathcal{F}_{j\omega})$, where $\gamma = \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i + 1$.

Proof The proof is given in Appendix C.4. □

By applying the coefficient strengthening procedure to the coefficient γ we have the following result.

Theorem 4. *For $k\in\Omega\backslash\{\omega\}$, let*

$$\begin{aligned}
\delta_k^1 = \underset{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}}{\text{maximize}} & \sum_{i\in\mathcal{C}\backslash\mathcal{D}} y_{ij} + \sum_{i\in\mathcal{I}\backslash\mathcal{C}} \alpha_i y_{ij} + \sum_{i\in\mathcal{D}} \beta_i y_{ij} - |\mathcal{C}\backslash\mathcal{D}| - \sum_{i\in\mathcal{D}} \beta_i + 1\\
\text{subject to } & \sum_{i\in\mathcal{I}} \xi_i^{k} y_{ij} \le m_j^{k}(k).
\end{aligned}$$

Sort δ_k^1 in a nondecreasing order such that $\delta_{k_1}^1 \le \ldots \le \delta_{k_{N-1}}^1$. Let q be defined as in Proposition 1, then $\delta_{k_{q+1}}^1$ is an upper bound on γ, and the inequality (16) is valid for (CBP) when $\gamma = \delta_{k_{q+1}}^1$.

Proof The proof is similar to the proof of Theorem 2 in Section 3.1.1. It is given in Appendix C.5. □

Given a linear programming relaxation solution, the separation problem is to find a valid inequality that is violated by this solution. We use a heuristic procedure similar to the one in Gu et al. (1998) and Kaparis and Letchford (2008) for the binary bilinear knapsack problem. This heuristic is given in Algorithm 1.

Algorithm 1: General Lifted Cover Inequality Separation Heuristic

Given the current relaxation optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$, Let $\mathcal{I}_0 = \{i \in \mathcal{I} : \hat{y}_{ij} = 0\}$.
Sort $\hat{\boldsymbol{y}}_j$ in non-increasing order such that $\hat{y}_{i_1 j} \geq ... \geq \hat{y}_{i_{|\mathcal{I}|} j}$, let $\mathcal{S} = \{i_1, ..., i_{|\mathcal{I}|}\}$.
for $\omega = 1, \ldots, N$ **do**
    if $\hat{z}_j^\omega = 1$ **then**
        Insert an item from the head of $\mathcal{S}$, until obtain a cover $\mathcal{C}$.
        Delete elements from the cover to get a minimal cover $\mathcal{C}$.
        Let set $\mathcal{D} = \{i \in \mathcal{C} : \hat{y}_{ij} = 1\}$.
        Calculate up-lifting coefficient α_i for $i \in \mathcal{I} \backslash \{\mathcal{C} \bigcup \mathcal{I}_0\}$.
        if $\sum_{i \in \mathcal{C} \backslash \mathcal{D}} \hat{y}_{ij} + \sum_{i \in \mathcal{I} \backslash \{\mathcal{C} \bigcup \mathcal{I}_0\}} \alpha_i \hat{y}_{ij} > |\mathcal{C} \backslash \mathcal{D}| - 1$ **then**
            Calculate down-lifting coefficient β_i for $i \in \mathcal{D}$.
            Calculate up-lifting coefficient α_i for $i \in \mathcal{I}_0 \backslash \mathcal{C}$.
            For $k \in \Omega \backslash \{\omega\}$, calculate δ_k^1.
            Let $\gamma = \delta_{k_{q+1}}^1$.
            Obtain the violated general lifted cover inequality (16).
        end
    end
end

The separation heuristic needs to compute up-lifting and down-lifting coefficients α_i, β_i, $\forall i \in \mathcal{I}$, and γ. Several previous papers have given methods for lifting coefficients' computation. Balas (1975) showed that one can compute the upper and lower bound of the lifting coefficients in linear time. Zemel (1989) proposed a dynamic programming algorithm for calculating the lifting coefficients exactly. Gu et al. (2000) used valid superadditive lifting functions to get lower and upper bounds for the lifting coefficients. In our computational experiments, we used the dynamic programming approach to calculate these coefficients.

3.1.3. 2-Clique Inequalities If $\xi_i^\omega + \xi_k^\omega > m_j^\omega(\omega)$ for all $i, k \in \mathcal{K}$ and $i \neq k$, the set $\mathcal{K} \subseteq \mathcal{I}$ is called a 2-clique. A clique is called maximal if it is not a proper subset of any other cliques. Let $\mathcal{K}$ be maximal clique. For each maximal 2-clique set $\mathcal{K}$, the following inequality is valid for $conv(\mathcal{Q}_{j\omega})$

$$\sum_{i \in \mathcal{K}} y_{ij} \leq 1. \tag{17}$$

To obtain valid inequalities for $conv(\mathcal{F}_{j\omega})$, we use (17) as a seed and calculate the lifting coefficient for the variable z_j^ω.

Theorem 5. *Let $\mathcal{K}$ be a maximal clique for $\mathcal{F}_{j\omega}$. Then the following inequality is facet-defining for conv($\mathcal{F}_{j\omega}$):*

$$\sum_{i\in\mathcal{K}} y_{ij} + \mu(z_j^\omega - 1) \le 1, \tag{18}$$

where $\mu = |\mathcal{K}| - 1$.

Proof The lifting coefficient μ is given by

$$\mu = \underset{\boldsymbol{y}_j, z_j^\omega}{\text{maximize}} \ \frac{\sum_{i\in\mathcal{K}} y_{ij} - 1}{1 - z_j^\omega} \tag{19a}$$

$$\text{subject to } (\boldsymbol{y}_j, z_j^\omega) \in \mathcal{F}_{j\omega}, \ \ z_j^\omega = 0. \tag{19b}$$

It is easy to verify that the optimal solution of the problem sets $y_{ij}^* = 1$, $\forall i \in \mathcal{K}$. Therefore, the best lifting coefficient is $\mu = |\mathcal{K}| - 1$. Consider the points $z_j^\omega = 0$, $\boldsymbol{y}_j = 1_{|\mathcal{I}|}$; for $z_j^\omega = 1$, $|\mathcal{K}|$ feasible point: $i \in \mathcal{K}$, $y_{ij} = 1$, $y_{kj} = 0$, $\forall k \in \mathcal{I}\backslash i$; and $|\mathcal{I}\backslash\mathcal{K}|$ feasible point: $i \in \mathcal{I}\backslash\mathcal{K}$, $y_{ij} = 1$, $\exists l \in \mathcal{K}$ such that $\zeta_i^\omega + \zeta_l^\omega \le m_j^\omega(\omega)$, let $y_{lj} = 1$, $y_{kj} = 0$, $\forall k \in \mathcal{I}\backslash\{l \cup i\}$. It is easy to verify that these $|\mathcal{I}| + 1$ points are affinely independent and satisfy inequality (18) at equality. Therefore, inequality (18) is facet-defining for *conv*($\mathcal{F}_{j\omega}$). □

We can use constraints (1d) to restrict the feasible region of $\boldsymbol{y}_j$ in (19). This yields a strengthened lifted coefficient μ. Instead of solving a chance-constrained problem. The following proposition gives an upper bound on μ.

Theorem 6. *For $k \in \Omega\backslash\{\omega\}$, let*

$$\lambda_k = \underset{\boldsymbol{y}_j \in \{0,1\}^{|\mathcal{I}|}}{\text{maximize}} \ \sum_{i\in\mathcal{K}} y_{ij} - 1$$

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

Sort λ_k in a nondecreasing order such that $\lambda_{k_1} \le \ldots \le \lambda_{k_{N-1}}$. Let q be defined as in Proposition 1. Then $\lambda_{k_{q+1}}$ is an upper bound on μ, and (18) is valid for (CBP) when $\mu = \lambda_{k_{q+1}}$.

Proof The proof is similar to that of Theorem 2 in Section 3.1.1. It is given in Appendix C.6. □

Note that the problem in Theorem 6 can be solved in polynomial time to compute the value of λ_k since the coefficients in the objective function of this problem are all one. It is sufficient to sort the items in the set $\mathcal{K}$, and keep putting them in the bin till the capacity is exceeded. We use the heuristic given in Algorithm 2, similar to the one in Nemhauser and Sigismondi (1992), to solve the separation problem for obtaining the clique inequalities.

Algorithm 2: 2-Clique Inequalities Separation Heuristic

Given the current relaxation optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$.

Sort $\hat{\boldsymbol{y}}$ in non-increasing order such that $\hat{y}_{i_1 j} \geq ... \geq \hat{y}_{i_{|\mathcal{I}|} j}$, let $\mathcal{S} = \{i_1, ..., i_{|\mathcal{I}|}\}$.

for $\omega = 1, \ldots, N$ **do**
    if $\hat{z}_j^\omega = 1$ **then**
        Insert an item from the head of $\mathcal{S}$ to obtain a clique set $\mathcal{K}$.
        if $\sum_{i \in \mathcal{K}} \hat{y}_{ij} > 1$ **then**
            Calculate λ_k, for $k \in \Omega \backslash \{\omega\}$.
            Let $\mu = \lambda_{k_{q+1}}$.
            Obtain the 2-clique inequality (18).
        end
    end
end

3.2. Projection Inequalities

We now reformulate (BIP) as MILP using additional binary variable u_{ij}^ω, for $i \in \mathcal{I}, j \in \mathcal{J}, \omega \in \Omega$. Let $\boldsymbol{u}_j^\omega = (u_{1j}^\omega, \ldots, u_{|\mathcal{I}|j}^\omega)^\top$ and $\boldsymbol{u} = \{u_{11}^1, \ldots, u_{|\mathcal{I}||\mathcal{J}|}^N\}$. We derive valid inequalities for (CBP) based on this formulation. The basic idea of deriving the inequalities is from Benders feasibility cuts. The following proposition gives a MILP formulation for (CBP).

PROPOSITION 6. *Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be an optimal solution of (BIP). Then, there exists $\boldsymbol{u}^*$ such that $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{u}^*)$ is an optimal solution of*

$$\underset{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{u}}{\text{minimize}} \sum_{j \in \mathcal{J}} c_j^a x_j + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij}^b y_{ij} \tag{20a}$$

$$\text{subject to (1b), (1c), (1e), (2b), (6b)}$$

$$\sum_{i \in \mathcal{I}} \xi_i^\omega u_{ij}^\omega \leq m_j^\omega(\omega) z_j^\omega, \qquad \forall j \in \mathcal{J}, \omega \in \Omega, \tag{20b}$$

$$u_{ij}^\omega \leq y_{ij}, u_{ij}^\omega \leq z_j^\omega, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, \omega \in \Omega, \tag{20c}$$

$$y_{ij} + z_j^\omega - u_{ij}^\omega \leq 1, u_{ij}^\omega \geq 0, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, \omega \in \Omega. \tag{20d}$$

Conversely, if $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{u}^*)$ is an optimal solution of (20), then $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP).

Proof See Appendix C.7. □

We now describe an approach for generating valid inequalities from the formulation (20). For $j \in \mathcal{J}$, and $\omega \in \Omega$, let us consider the subproblem with variable $\boldsymbol{u}_j^\omega$ as follows:

$$\underset{\boldsymbol{u}_j^\omega \geq 0}{\text{minimize}} \; 0 \tag{21a}$$

$$\text{subject to} \sum_{i \in \mathcal{I}} \xi_i^\omega u_{ij}^\omega \leq m_j^\omega(\omega) z_j^\omega, \tag{21b}$$

$$u_{ij}^\omega \leq y_{ij}, u_{ij}^\omega \leq z_j^\omega, \qquad \forall i \in \mathcal{I}, \tag{21c}$$

$$y_{ij} + z_j^\omega - u_{ij}^\omega \leq 1, \qquad \forall i \in \mathcal{I}. \tag{21d}$$

Given $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in \mathcal{X}_{RIP}$, if $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ violates constraints (8b), it is possible to identify a supporting hyperplane at $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ by solving the dual of (21):

$$\underset{\mu^1,\boldsymbol{\mu^2},\boldsymbol{\mu^3},\boldsymbol{\mu^4}}{\text{maximize}} \; -m_j^{\omega}(\omega)\hat{z}_j^{\omega}\mu^1 - \sum_{i\in\mathcal{I}} \hat{y}_{ij}\mu_i^2 - \hat{z}_j^{\omega}\sum_{i\in\mathcal{I}}\mu_i^3 + \sum_{i\in\mathcal{I}}(\hat{y}_{ij} + \hat{z}_j^{\omega} - 1)\mu_i^4 \tag{22a}$$

$$\text{subject to } \xi_i^{\omega}\mu^1 + \mu_i^2 + \mu_i^3 - \mu_i^4 \geq 0, \qquad \forall i \in \mathcal{I}, \tag{22b}$$

$$\mu^1, \boldsymbol{\mu^2}, \boldsymbol{\mu^3}, \boldsymbol{\mu^4} \geq 0, \tag{22c}$$

where μ^1, $\boldsymbol{\mu}^2$, $\boldsymbol{\mu}^3$, and $\boldsymbol{\mu}^4$ are dual variables for constraints (21b)-(21d), respectively.

THEOREM 7. *The projection inequality*

$$\sum_{i\in\mathcal{I}}(\hat{\mu}_i^4 - \hat{\mu}_i^2)y_{ij} + (\sum_{i\in\mathcal{I}}\hat{\mu}_i^4 - \sum_{i\in\mathcal{I}}\hat{\mu}_i^3 - m_j^{\omega}(\omega)\hat{\mu}^1)z_j^{\omega} \leq \sum_{i\in\mathcal{I}}\hat{\mu}_i^4, \tag{23}$$

where $\hat{\mu}^1$, $\hat{\boldsymbol{\mu}}^2$, $\hat{\boldsymbol{\mu}}^3$, *and* $\hat{\boldsymbol{\mu}}^4$ *is an extreme ray of* (22), *is valid for (CBP).*

Proof If $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ violates constraints (8b), then problem (21) is infeasible. Thus, problem (22) is either unbounded or infeasible. If the feasible region defined by (22b) and (22c) is not empty, then we can obtain an extreme ray $\hat{\mu}^1$, $\hat{\boldsymbol{\mu}}^2$, $\hat{\boldsymbol{\mu}}^3$, and $\hat{\boldsymbol{\mu}}^4$. Therefore, we have

$$\sum_{i\in\mathcal{I}}(\hat{\mu}_i^4 - \hat{\mu}_i^2)\hat{y}_{ij} + (\sum_{i\in\mathcal{I}}\hat{\mu}_i^4 - \sum_{i\in\mathcal{I}}\hat{\mu}_i^3 - m_j^{\omega}(\omega)\hat{\mu}^1)\hat{z}_j^{\omega} - \sum_{i\in\mathcal{I}}\hat{\mu}_i^4 > 0.$$

Hence, the theorem follows. □

Note that the inequalities in (23) are obtained by considering the dual problem (22) for each $j \in \mathcal{J}$, and $\omega \in \Omega$. It is possible to combine multiple j and ω (possibly all) in generating Benders-type inequalities. It can be achieved by considering the following problem:

$$\begin{aligned} &\underset{\boldsymbol{u}}{\text{minimize}} \;\; 0 \\ &\text{subject to } \;\; (20b) - (20d). \end{aligned} \tag{24}$$

Let $\boldsymbol{v}^1$, $\boldsymbol{v}^2$, $\boldsymbol{v}^3$, and $\boldsymbol{v}^4$ be dual variables of constraints (20b)-(20d), respectively.

THEOREM 8. *The combined projection cut is given by:*

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{\omega\in\Omega}(\hat{v}_{ij\omega}^4 - \hat{v}_{ij\omega}^2)y_{ij} + \sum_{j\in\mathcal{J}}\sum_{\omega\in\Omega}(\sum_{i\in\mathcal{I}}\hat{v}_{ij\omega}^4 - \sum_{i\in\mathcal{I}}\hat{v}_{ij\omega}^3 - m_j^{\omega}(\omega)\hat{v}_{j\omega}^1)z_j^{\omega} \leq \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{\omega\in\Omega}\hat{v}_{ij\omega}^4, \tag{25}$$

where $\hat{\boldsymbol{v}}^1$, $\hat{\boldsymbol{v}}^2$, $\hat{\boldsymbol{v}}^3$, *and* $\hat{\boldsymbol{v}}^4$ *is an extreme ray of the dual of* (24).

Proof The proof is similar to the proof of Theorem 7. □

Algorithm 3: Branch-and-Cut Implementation

1 **Initialize** $\mathrm{UB} = +\infty$, $\mathrm{LB} = -\infty$ and $\mathcal{N} = \varnothing$.
2 **Initialize** Nodelist $\mathcal{N} = \{o\}$, where o is a branching node without constraints.
3 **while** *($\mathcal{N}$ is nonempty)* **do**
4   Select a node $o \in \mathcal{N}$.
5   Update, $\mathcal{N} \leftarrow \mathcal{N}/\{o\}$.
6   Optimize the LP relaxation problem of (IP) at the node o.
7   **if** *the generated an optimal solution* $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ *with objective* $obj^* < UB$ **then**
8   **if** $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ *is fractional* **then**
9   **if** *the violated inequalities* (16), (18) *or* (23) *are found* **then**
10   Add the violated inequalities to LP relaxation problem.
11   Go back to line 6.
12   **end**
13   **else**
14   Branch, resulting in nodes o^* and o^{**}.
15                 $\mathcal{N} \leftarrow \mathcal{N} \cup \{o^*, o^{**}\}$.
16   **end**
17   **end**
18   **else**
19   Update UB, $\mathrm{UB} = obj^*$.
20   **end**
21   **end**
22 **end**
23 **return** UB and its corresponding optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$.

4. Branch-and-Cut Solution Scheme

We illustrate the use of valid inequalities presented in the previous sections within a branch-and-cut framework. We make use of Algorithm 3 to solve the strengthened big-M reformulation (IP) of (CBP), and study the value of using cover, clique, and projection inequalities in the branch-and-cut method. Let LB and UB denote the current lower and upper bound of (CBP), and $\mathcal{N}$ denote the set of remaining nodes in the branch-and-cut search tree. Algorithm 3 provides an outline of the branch-and-cut framework.

At each node, we solve a relaxation problem to obtain an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ and objective value obj^*. If $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is fractional, we solve the corresponding problems to find violated inequalities. If valid inequalities are found, we add the violated inequalities to the LP relaxation problems and resolve the LP relaxation problems. Otherwise, we continue branching. If $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is integral, we update the upper bound, if possible.

We implemented our algorithm within the branch-and-cut scheme of the CPLEX MILP solver. We used the default branching strategy of the CPLEX MILP solver in this implementation. The CPLEX solver is allowed to add its proprietary cuts.

In addition to adding cuts, efficiently exploring the branch-and-cut tree is also an important consideration in solving our problem. Next, we present a strategy that has helped in significantly reducing the size of the branch-and-cut tree. Specifically, we solve integer programs to obtain an improved lower bound for the optimal objective value. We report the number of valid cuts from our proposed ideas in Section 5.2 after incorporating improvements from this heuristic.

4.1. Calculating the Lower Bound

A standard method to compute a lower bound for (CBP) is to relax all the integer variables and solve the relaxation LP problem. Note that in our model variables $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are binary. Let v^* be the optimal value of (CBP). We first solve the relaxation of (IP) called (RIP$_z$) in which only the integrality restriction on variables $\boldsymbol{z}$ is relaxed. We obtain the optimal objective value v_r^* and a solution $(\boldsymbol{x}_r^*, \boldsymbol{y}_r^*)$ of this problem. In our experiments, we observe that the lower bound generated in this way generally satisfies $v_r^* < v^*$. To improve the lower bound, we further solve (IP) with the given objective value v_r^*. If the problem is feasible, the lower bound v_r^* is the optimal value of (CBP), and we have an optimal solution. Otherwise, we update the lower bound by letting the lower bound be equal to $v_r^* + \delta$, where δ is an appropriate value. Since $\boldsymbol{x}$ and $\boldsymbol{y}$ are binary, when c_j^a and c_{ij}^b are integer valued, all possible values of $c_j^a x_j + c_{ij}^b y_{ij}$ are integer. Then $v_1 = \min\{\sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}, j\in\mathcal{J}} c_{ij}^b y_{ij} : \sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}, j\in\mathcal{J}} c_{ij}^b y_{ij} > v_r^*, \boldsymbol{x} \in \{0,1\}^{|\mathcal{J}|}, \boldsymbol{y} \in \{0,1\}^{|\mathcal{I}|\times|\mathcal{J}|}\}$ provides an improved lower bound, and we can choose $\delta = v_1 - v_r^*$. The approach is effective, specially for the problem that minimizes the number of bins to pack items in (CBP). In this special case $\delta = 1$. We now continue to solve the feasibility problem, with this improved lower bound. Algorithm 4 provides formal description of this lower bound improvement heuristic for the problem that minimizes the number of bins. A finite number (K) of updates to the lower bound is performed, when possible.

Algorithm 4: The Lower Bound Improvement Heuristic

```
1  Initialize: Let lower bound of (CBP) LB = −∞.
2  Initialize: Let κ = 1, and K, T represent the maximum number of iteration and time limit respectively.
3  Optimize the relaxation problem (RIP_z) with the time limit T.
4  Obtain the optimal number of opening bins n_r^{κ*}, and corresponding lower bound LB.
5  while (κ ≤ K) do
6      Fix the variable x in (CBP) with n_r^{κ*}.
7      if n_r^{κ*} is the optimal number of opening bins of (CBP) then
8          Obtain an optimal solution of (CBP), and go to line 14.
9      end
10     else
11         Update n_r^{κ*} = n_r^{κ*} + 1, and the lower bound LB. κ = κ + 1.
12     end
13 end
14 return LB and the optimal solution of (CBP) if found.
```

5. Computational Experiments

In this section we present our computation experience with the developed ideas on an operating room (OR) scheduling problem. The problem assigns a set $\mathcal{I}$ of surgeries with random duration ζ_i, $\forall i \in \mathcal{I}$, to a set $\mathcal{J}$ of ORs, so as to minimize the number of opened ORs. An OR j has time limit t_j, $\forall j \in \mathcal{J}$. The overtime constraints for ORs are given by chance constraints which ensure that the probability of overtime is no more than a given parameter ε.

In what follows, in Section 5.1 we provide additional implementation details. Section 5.2 discusses the performance of valid inequalities (described in Section 3) and the lower bound improvement heuristic (described in Section 4.1). A comparison with CVaR approximation is given in Appendix B.

5.1. Implementation Details

We used real data from a large public hospital in Beijing, China to show the performance of the proposed algorithm. The collected data set has 5,721 surgical durations for nine major surgery types from 2015/01 to 2015/10. This data is used to specify the surgery probability distribution for each surgery type. The department has eight ORs where nine major types of surgeries are performed. There is no restriction on the OR where a surgery can be performed. In our test examples, we take the OR time limit to be 10 hours, that is $t_j := 10$ hours for $j \in \mathcal{J}$. Table 1 gives the mean and standard deviation of the surgery duration, and percentage for each surgery type. In our problem generation we assume that 18 surgeries (mean number of surgeries) are performed in a day. We use this number and the percentage of surgeries of a given type to calculate the number of surgeries for each surgery type to be performed in a day. The calculations are rounded to the nearest integer while ensuring that 18 surgeries are performed each day. Note that the surgery duration is non-negative. In the literature, it is found that this duration can be characterized by a log-normal distribution (Spangler et al. 2004, Gul et al. 2011). Therefore, we chose log-normal distribution for generating problem instances in this paper. The log-normal distribution was parameterized to ensure that its mean and standard deviation was equal to that given in Table 1. We rounded-up the samples generated from this log-normal distribution to the nearest 15 minutes. Therefore, the weights and bin capacity are expressed in time duration of 15 minutes in our problem generation. We generated five instances for each sample size. Note that for the problem that minimizes the number of opened ORs $c^a_j := 1$, and $c^b_{ij} := 0$, $\forall i \in \mathcal{I}, j \in \mathcal{J}$.

In our implementation of the valid inequality finding procedure (described in Section 3), we add the identified valid inequalities that are violated by the current solution by a minimum violation threshold. The inequalities (18) and (23) are added if they have violation at least 10^{-4}, and (16) is added if it has a relative violation of at least 0.3. The relative violation is defined as the absolute violation of the cut divided by $|\mathcal{C} \backslash \mathcal{D}|$. The valid inequalities in Section 3 are generated repeatedly

Table 1 For each surgery type, the mean (mean), standard deviation (std) in hours, and the percentage for each surgery type (percentage) are reported

surgery type	mean (hrs)	std (hrs)	percentage
Gynaecology	1.1	1.3	0.29
Galactophore	1.6	1.0	0.15
Lymphatic	3.2	1.1	0.14
Ear	2.8	1.7	0.13
Urology	2.3	1.7	0.07
Vascular	2.6	1.5	0.07
Obstetrics	1.5	0.5	0.06
Joint	2.8	1.3	0.06
Orthopeadic	3.2	1.8	0.03

until one of the following stopping criteria is met: no cut is available with the violation threshold, or the improvement on the objective value of LP relaxation is less than 0.2 at the node of the branch-and-cut tree. We add the violated inequalities (23) and (18) only at the root node of the branch-and-bound tree when the gap is no more than 1, where the gap is given by UB − LB. At each round of generating cuts of type (23) and (18), for each $j \in \mathcal{J}$, we only use one pair of (j, ω), $\forall \omega \in \Omega$ such that the corresponding valid inequality is the most violated inequality. We add the valid inequalities (16) at all the nodes that are at a depth less than 3. We keep only the efficient cuts, which are identified by the optimization solver, in the branch-and-cut tree at the end of this procedure.

All experiments are coded in the programming language C using IBM CPLEX version 12.71 callable libraries. A laptop with Intel(R) 2.80 GHz processor and 16 GB RAM is used for computation on a 64-bit computer using Windows operating system. Only one thread is used for all computations. We turned off CPLEX presolve procedure when implementing the branch-and-cut algorithm because we needed to use CPLEX callback function to work on the original problem in our testing. A prioritized node selection strategy is used in the branch-and-cut algorithm. Specifically, branching on x is preferred over y, and y is preferred over z. For all instances, we use the time limit of 10 hours. For instances that could not be solved to optimality, we give the number of ORs opened in the sub-optimal solution and the optimal number of ORs when it is known from the computations performed in a different algorithm. We report the solution time (in seconds) for the instances solved to optimality within the time limit.

5.2. Discussion on the Algorithmic Performance

In Section 5.2.1, we presents the performance of different variants of the lower bound improvement heuristic (Algorithm 4) for (CBP). The performance of the branch-and-cut algorithm (Algorithm 3) with the proposed lower bound improvement heuristic and valid inequalities is discussed in Section 5.2.2. A comparison with a generalization of the probabilistic cover approach for our problem is given in Section 5.2.3. We also performed an out-of-sample analysis. These results are given in Section 5.2.4. Additional computational results on the performance of the lower bound improvement heuristic and valid inequalities on larger problem instances are given in Section 5.2.5.

5.2.1. Performance of Lower Bound Improvement Heuristic for (CBP) We now discuss our results on the lower bound improvement heuristic presented in Algorithm 4 for (CBP). The level of chance satisfaction $\varepsilon \in \{0.05, 0.1, 0.15\}$ and $N \in \{100, 500, 1000\}$ are used in problem generation. Valid inequalities were not added when performing computations for results discussed in this section. We compare the following three different variants to illustrate the performance of the lower bound improvement heuristic. Note that all the variants used the strengthened big-M reformulation (IP) of (CBP).

- CPX: refers to using the branch-and-cut algorithm as implemented in CPLEX to solve (IP) of (CBP).
- LBH0: refers to using the optimal objective value of (RIP$_z$), i.e., $K = 0$ in Algorithm 4, as an initial lower bound of the branch-and-cut algorithm for (CBP).
- LBH1: refers to using Algorithm 4 with $K = 1$ as an initial lower bound of branch-and-cut algorithm for (CBP).

Table 2 presents the solution details, including the average time for the lower bounding heuristic and the branch-and-cut algorithm, the average and geometric mean of total time spent to solve (CBP), the average number of nodes for the branch-and-cut algorithm and the number of opened ORs, the number of solved instances from the five instances, and the proportion of instances where the lower bound is equal to the optimal objective value. We found that using Algorithm 4 with $K = 2$ for computing an initial lower bound for the branch-and-cut algorithm for (CBP), (LBH2) and LBH1 have comparable performance for most of the instances. Thus, the results of LBH2 are not presented in Table 2. The time required for the strengthened big-M computation in the reformulation is typically less than 5 seconds, and therefore not included in the table.

From Table 2 we observe that when solving (CBP), initialization of the lower bound using Algorithm 4 significantly outperforms the one without an initial lower bound computation for most of the instances. For $\varepsilon = 0.1$, the lower bound obtained using Algorithm 4 with $K = 1$ gives the optimal objective value for almost all the instances, indicating that $K = 1$ provides a lower bound with reasonably good quality. However, it does increase the average time of calculating the lower bound by almost a factor of 10. Recall that one more binary program is being solved. However, LBH1 is still more effective than LBH0 in terms of the average total time spent to solve (CBP). In particular,

Table 2 The average CPU time (seconds) for the lower bounding heuristic (LBH-AvT) and the branch-and-cut (B&C-AvT), the average total time (seconds) spent to solve (CBP) (AvT), the geometric mean of total time (seconds) spent to solve (CBP) (G-AvT), the average number of nodes for the branch-and-cut algorithm (# of nodes), the number of opened ORs (# of ORs) and the number of solved instances from the five instances (solved), the proportion that the lower bound is equal to optimal objective value (Δ).

ε	N	approach	LBH-AvT	B&C-AvT	AvT	G-AvT	# of nodes	# of ORs	solved	Δ
0.05	100	CPX	N/A	372.4	372.4	325.1	67,454	[6, 6, 6, 6, 6]	5/5	0
		LBH0	2.1	152.0	154.1	125.8	24,503	[6, 6, 6, 6, 6]	5/5	0
		LBH1	84.9	0.7	85.6	62.9	364	[6, 6, 6, 6, 6]	5/5	1
	500	CPX	N/A	5,287.0	5,287.0	4,869.9	43,503	[6, 6, 6, 6, 6]	5/5	0
		LBH0	30.5	3,156.3	3,186.8	3,018.1	26,919	[6, 6, 6, 6, 6]	5/5	0
		LBH1	3,183.8	1.7	3,185.5	3,109.8	140	[6, 6, 6, 6, 6]	5/5	1
	1000	CPX	N/A	19,900.6	19,900.6	19,159.8	70,726	[6, (5,6), 6, 6, 6]	4/5	0
		LBH0	71.3	8,824.3	8,895.7	8,290.5	20,042	[6, 6, 6, 6, 6]	5/5	0
		LBH1	7,062.0	10.1	7,072.1	7,021.3	78	[6, 6, 6, 6, 6]	5/5	1
0.1	100	CPX	N/A	1,744.5	1,744.5	265.5	547,776	[6, 5, 5, 5, 5]	5/5	0
		LBH0	1.2	1,499.0	1,500.2	172.3	433,573	[6, 5, 5, 5, 5]	5/5	0
		LBH1	5.7	523.3	528.9	45.8	116,292	[6, 5, 5, 5, 5]	5/5	0.8
	500	CPX	N/A	2,182.0	2,182.0	805.7	26,392	[5, 5, 5, 5, 5]	5/5	0
		LBH0	14.3	1,581.7	1,596.0	863.6	22,962	[5, 5, 5, 5, 5]	5/5	0
		LBH1	142.1	479.0	621.2	511.2	11,163	[5, 5, 5, 5, 5]	5/5	1
	1000	CPX	N/A	13,101.2	13,101.2	11,301.4	59,711	[5, (6,4), 5, 5, 5]	4/5	0
		LBH0	33.5	15,498.7	15,533.1	14,019.4	89,513	[(6,4), 5, 5, 5, 5]	4/5	0
		LBH1	474.9	1,401.8	1,876.6	1,347.5	9,876	[5, 5, 5, 5, 5]	5/5	1
0.15	100	CPX	N/A	154.5	154.5	150.5	13,232	[5, 5, 5, 5, 5]	5/5	0
		LBH0	0.9	120.9	121.8	112.8	8,077	[5, 5, 5, 5, 5]	5/5	0
		LBH1	87.6	0.4	88.0	71.4	103	[5, 5, 5, 5, 5]	5/5	1
	500	CPX	N/A	1,460.0	1,460.0	1,307.1	6,993	[5, 5, 5, 5, 5]	5/5	0
		LBH0	14.3	1,282.9	1,297.2	1,175.1	6,345	[5, 5, 5, 5, 5]	5/5	0
		LBH1	1,441.0	3.8	1,444.8	1,424.0	78	[5, 5, 5, 5, 5]	5/5	1
	1000	CPX	N/A	5,353.2	5,353.2	4,139.5	7,511	[5, 5, 5, 5, 5]	5/5	0
		LBH0	25.2	4,983.4	4,948.6	4,990.2	6,669	[5, 5, 5, 5, 5]	5/5	0
		LBH1	5,126.4	10.4	5,139.7	4,856.9	64	[5, 5, 5, 5, 5]	5/5	1

LBH0 reduces the average total time by an average of more than 7%, LBH1 further reduce the time by 68%. For harder instances ($N = 1000$), LBH1 solves all the five instances within 1 hour. The total CPU time spent to solve (CBP) changes across different sample sizes and ε values. Thus, we also provide the geometric mean of CPU time over the instances. From Table 2 we observe that the geometric mean of CPU time is similar to the average time for $\varepsilon = 0.05$ and 0.15. For $\varepsilon = 0.1$, the geometric mean of CPU time is significantly smaller than the average time for $N = 100$ and 500, and LBH0 has a comparable performance with CPX. Compared with LBH0, LBH1's geometric mean is 68% lower. This improvement can be explained by the fact that the extra restriction on $\boldsymbol{x}$ reduces the feasible region, and consequently decreases the number of nodes explored to prove optimality. For $\varepsilon = 0.05$ and 0.15, we see from Table 2 that the time taken by LBH1 increases significantly. This yields a comparable performance with LBH0 in terms of the total solution time and the proportion of instances that are solved to optimality. We note from the results in Table 2 that in one 100 scenario instance, the solution opened an additional operating room when compared to solutions for the 500 and 1000 scenario instances. This is possible because the samples generated from the sample average approximation method only approximate the true underlying distribution, and the solution to an approximate problem may have a more conservative higher cost solution. In view of this, we have also provided results on the out-of-sample performance of the generated solutions in Section 5.2.4.

5.2.2. Performance of Lower Bound Improvement Heuristic and Valid inequalities

In this section, we discuss the usefulness of adding inequalities, while also using the lower bound improvement heuristic. We use Algorithm 4 with $K = 1$ (LBH1) to obtain an initial lower bound for the branch-and-cut algorithm. We consider the sample size $N \in \{100, 500, 1000\}$. Since for the level of chance satisfaction $\varepsilon = 0.05, 0.15$, the average time for the branch-and-cut algorithm is less than 11 seconds after completing LBH1, we only consider problems with $\varepsilon = 0.1$ in this section. We consider the following five variants:

- Cover: refers to adding the general lifted cover inequalities (16) to LBH1.
- C&C: refers to adding the general lifted cover inequalities (16) and 2-clique inequalities (18) to LBH1.
- Proj: refers to adding the projection inequalities (23) to LBH1.
- P&C: refers to adding the projection inequalities (23) and 2-clique inequalities (18) to LBH1.
- B&C: refers to adding the projection inequalities (23), lifted cover (16) and 2-clique inequalities (18) to LBH1.

For C&C, we only added the violated clique inequalities when we could not find any lifted cover inequality at the root node. For P&C, we only added the violated clique inequalities when we could not find any projection inequality at the root node. For B&C, we added the 2-clique inequalities (18) when we could not find any the projection inequality (23), and added the lifted cover when there is no violated inequality (23) and (18). We could not find a setting for the mixing set inequalities (7) that improved the performance. For several harder instances ($N = 500, 1000$), the use of mixing set inequalities resulted in a worse performance. This might be due to the default search mechanism in CPLEX. However, it is unclear if a modification to this search mechanism will provide improved result. Table 3 reports the average and geometric mean of total time spent to solve (CBP), the average number of nodes for the branch-and-cut algorithm, the number of opened ORs, the number of solved instances from the five generated instances, and the average number of cuts for (CBP).

Table 3 The average total time (seconds) spent to solve (CBP) (AvT), the geometric mean of total time (seconds) spent to solve (CBP) (G-AvT), the average number of nodes for the branch-and-cut algorithm (# of nodes), the number of opened ORs (# of ORs), the number of solved instances from the five instances (solved), and the average number of cuts (# of cuts) for (CBP) are reported, and $K = 1$ in Algorithm 4 is used for these computations

N	approach	AvT	G-AvT	# of nodes	# of ORs	solved	# of cuts
100	Cover	251.5	37.0	62,737	[6, 5, 5, 5, 5]	5/5	10
	C&C	252.8	43.6	63,076	[6, 5, 5, 5, 5]	5/5	23
	Proj	514.0	52.0	115,418	[6, 5, 5, 5, 5]	5/5	11
	P&C	514.0	52.0	115,418	[6, 5, 5, 5, 5]	5/5	11
	B&C	251.4	44.8	65,789	[6, 5, 5, 5, 5]	5/5	17
500	Cover	410.9	373.4	5,166	[5, 5, 5, 5, 5]	5/5	14
	C&C	244.2	201.5	2,740	[5, 5, 5, 5, 5]	5/5	22
	Proj	250.5	198.4	1,789	[5, 5, 5, 5, 5]	5/5	8
	P&C	250.5	198.4	1,789	[5, 5, 5, 5, 5]	5/5	8
	B&C	250.5	198.4	1,789	[5, 5, 5, 5, 5]	5/5	8
1000	Cover	1,073.3	934.8	3,876	[5, 5, 5, 5, 5]	5/5	11
	C&C	1,028.9	809.4	3,536	[5, 5, 5, 5, 5]	5/5	13
	Proj	1,130.4	974.2	5,122	[5, 5, 5, 5, 5]	5/5	5
	P&C	1,011.6	811.1	4,483	[5, 5, 5, 5, 5]	5/5	8
	B&C	818.6	736.2	2,134	[5, 5, 5, 5, 5]	5/5	9

The results in Tables 2 and 3 show that adding the general lifted cover and projection inequalities provide significant improvements in the solution time and the number of processed nodes for the harder instances ($N = \{500, 1000\}$). For problems with $N = 1000$ scenarios, the average solution time is decreased by more than 40% on average by using the inequalities. However, for the instances ($N = 100$) the improvement from adding the projection inequalities is modest. Moreover, adding the version using lifted cover and 2-clique inequalities performs better than the version that uses only the lifted cover inequalities except for the 100 scenario instances. In Table 3, we also observe that for the harder instances ($N = \{500, 1000\}$) adding the projection inequalities and clique inequalities performs compared to the version that uses the cover and clique inequalities.

5.2.3. Comparison with the Probability Cover Approach The results in this section are for the harder problems that are generated for $\varepsilon = 0.1$ and $N \in \{500, 1000\}$. We compare the performance of the following approaches:

- B&C: is described in Section 5.2.2.
- BPC: is the probability cover approach from Song et al. (2014) adapted for the (CBP) problem.

The implementation details are presented in Appendix D.

In order to compare the proposed methods, for each setting, ten instances are considered. These are labeled as $N - \#$, where $\#$ denotes the instance number. Table 4 reports the total time spent to solve (CBP), the number of nodes, the number of cuts and ORs for these approaches.

Table 4 **Algorithmic 3 comparison with four exact approaches, where we report the total time spent to solve (CBP) (time) in seconds, the number of nodes (nodes), and the number of cuts (cuts) for each instance**

instance	time		# of nodes		# of cuts		# of ORs	
	B&C	BPC	B&C	BPC	B&C	BPC	B&C	BPC
500-1	98.4	1,028.4	218	869,176	2	6,538	5	5
500-2	227.7	894.4	580	811,260	0	5,568	5	5
500-3	110.8	1,008.2	218	869,176	2	6,538	5	5
500-4	613.8	1,158.2	7,658	1,501,493	2	4,853	5	5
500-5	201.6	635.9	270	855,766	2	4,826	5	5
500-6	103.3	1,411.4	1,547	1,599,748	0	4,848	5	5
500-7	226.3	2,340.0	3,054	1,594,015	2	7,831	5	5
500-8	250.2	1,579.7	420	1,950,550	1	5,166	5	5
500-9	599.9	1,132.9	7,658	1,501,493	2	4,853	5	5
500-10	223.9	2,595.0	1,567	1,484,011	5	8,923	5	5
Average	265.6	1,378.4	2,319	1,303,669	2	5,994	5	5
1000-1	598.4	762.7	1,020	958,747	0	4,929	5	5
1000-2	882.0	1,701.9	2,584	1,068,529	2	8,054	5	5
1000-3	396.0	1,593.2	810	1,220,706	3	6,807	5	5
1000-4	668.0	1,733.0	1,070	1,571,896	3	5,969	5	5
1000-5	1,548.7	1,110.3	5,186	819,237	1	7,068	5	5
1000-6	1,014.9	1,567.5	1,999	1,482,643	1	5,812	5	5
1000-7	931.8	1,103.1	3,306	1,730,061	4	4,623	5	5
1000-8	998.3	1,485.5	1,825	1,255,048	2	5,933	5	5
1000-9	1,386.7	1,474.7	2,619	1,554,822	7	5,480	5	5
1000-10	926.4	1,246.2	1,617	1,178,122	4	5,949	5	5
Average	935.1	1,377.8	2,204	1,283,981	3	6,062	5	5

The results in Table 4 indicate that BPC is also able to solve the large-scale instances but it takes longer than our implementation of B&C, especially for the instances with $N = 500$. The solution time saved by B&C is up to 90%, and the search tree size is reduced by over 99%. On average, solution times for 500 scenario models is reduced by a factor of approximately 5, and for 1000 scenario models it is reduced by a factor of approximately 1.5. Song et al. (2014) also added a type of projection cut to improve the performance of the BPC algorithm for the single chance constraint model. In our computations the projection cuts introduced by Song et al. (2014) did not benefit the multiple chance constraints setting of the (CBP) problem.

5.2.4. Out-of-Sample Performance The solutions obtained in the previous section are based on a finite sample of scenarios. We performed an experiment to evaluate the quality of these solutions out-of-sample. In this experiment, using the underlying log-normal distribution we generated 1,500,000 samples and used the integer solutions obtained from model (CBP) for the sample sizes $N \in \{100, 500, 1000\}$ and $\varepsilon \in \{0.05, 0.1, 0.15\}$. Table 5 gives the average of overtime probability, and the worst-case overtime probability for the solutions.

Table 5 The average overtime probability (AvT-prob), the worst-case overtime probability (Worst-prob) are reported.

ε	0.05			0.1			0.15		
N	100	500	1000	100	500	1000	100	500	1000
AvT-prob	0.039	0.037	0.037	0.094	0.104	0.105	0.107	0.109	0.107
Worst-prob	0.069	0.058	0.062	0.134	0.131	0.121	0.157	0.164	0.148

The results in Table 5 show that the average out-of-sample overtime probability is nearly equal to ε for all cases. However, we observe that the worst-case out-of-sample chance constraint satisfaction probability improves with the larger sample sizes ($N = 500, 1000$). This is expected as we have improved approximations of the underlying probability distribution with an increased sample size. Interestingly, for $N = 1000$ the worst probability of 0.148 also met the desired $\varepsilon = 0.15$.

5.2.5. Performance of Lower Bound Improvement Heuristic and Valid inequalities on Larger Problems In this section, we use the same problem settings as in Section 5.1, and generate problems for $|\mathcal{I}| = 25$ and $|\mathcal{J}| = 15$ to test the performance of the lower bound improvement heuristic and valid inequalities. In these tests we could only consider problems with sample size $N = 100$ and $\varepsilon = 0.1$, as problems with a smaller ε and larger sample size were too difficult to solve to optimality by any of the methods. Table 6 presents the solution details, including the average time for the lower bounding heuristic and the branch-and-cut algorithm, the average total time and the geometric mean of total time spent to solve (CBP), the average number of nodes for the branch-and-cut algorithm and the number of opened ORs, the number of solved instances from the five instances, the proportion of instances where the lower bound is equal to the optimal objective value, and the average number of cuts.

Table 6 The average CPU time (seconds) for the lower bounding heuristic (LBH-AvT) and the branch-and-cut (B&C-AvT), the average total time (seconds) spent to solve (CBP) (AvT), the geometric mean of total time (seconds) spent to solve (CBP) (G-AvT), the average number of nodes for the branch-and-cut algorithm (# of nodes), the number of opened ORs (# of ORs) and the number of solved instances from the five instances (solved), the proportion that the lower bound is equal to optimal objective value (△), and the average number of cuts (# of cuts).

Approach	CPX	LBH0	LBH1	B&C
LBH-AvT	N/A	2.4	2,087.2	2,087.2
B&C-AvT	–	–	17,373.7	7,500.8
AvT	–	–	19,461.0	9,588.0
G-AvT	–	–	14,616.4	6,331.6
# of nodes	4,535,731	4,365,748	2,531,293	1,088,997
# of ORs	[(7,6),(7,6),(7,6),(7,6),(7,6)]	[(7,6),(7,6),(7,6),(7,6),(7,6)]	[7,7,7,7,7]	[7,7,7,7,7]
solved	0/5	0/5	5/5	5/5
Δ	0	0	1	1
# of cuts	N/A	N/A	N/A	8

"–" means that no instance can be solved to optimality within the runtime limit.

Table 6 provides further evidence on the effectiveness of the lower bound generation heuristic and the use of the valid inequalities developed in this paper. From Table 6, we observe that CPX and LBH0 cannot solve any instance to optimality within the runtime limit, while LBH1 and B&C can solve all of the five instances. Compared with LBH1, adding the valid inequalities saves 51% of the average solution time, 57% of the geometric mean solution time, while generating 57% fewer nodes. This implies that the problems become more tractable with the incorporation of lower bound generation heuristic, and adding the valid inequalities leads to an additional time saving for these harder problems.

6. Concluding Remarks

This paper investigated the chance-constrained bin packing problem. We formulated the model as a 0-1 bilinear program and developed three classes of valid inequalities from the bilinear formulation. Computational results showed that the three valid inequalities combined with a lower bound computation heuristic allow us to solve models with up to 1,000 scenarios for the chance constraints specified at 0.95, 0.90 and 0.85 satisfaction of the bins needing to pack items with random sizes. The data for our computational tests was generated based on a real data set for a hospital operating room surgery assignment problem. We also observed that the CVaR approximation for the test problems was generally not tight. Our attempt to solve larger problems (e.g., with 1,500 scenarios) met with partial success. Specifically, for these problem B&C and BPC discussed in Section 5.2.3 could solve only 1 out of the 5 problem instances with a 10 hour CPU time limit. It is unclear if the generalization of the probabilistic cover approach Song et al. (2014) can be combined with the approach developed in the current paper. It remains a topic of future research.

Acknowledgments

This research of the first two authors was 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 grants CMMI-1361942 and CMMI-1763035.

References

Ahmed, S., Luedtke, J., Song, Y., Xie, W., 2017. Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. Mathematical Programming 162 (1-2), 51–81.

Ahmed, S., Shapiro, A., 2008. Solving chance-constrained stochastic programs via sampling and integer programming. INFORMS Tutorials in Operations Research 10, 261–269.

Ahmed, S., Xie, W., 2018. Relaxations and approximations of chance constraints under finite distributions. Mathematical Programming 170 (1), 43–65.

Atamtürk, A., Nemhauser, G. L., Savelsbergh, M. W., 2000. The mixed vertex packing problem. Mathematical Programming 89 (1), 35–53.

Balas, E., 1975. Facets of the knapsack polytope. Mathematical Programming 8 (1), 146–164.

Berndt, S., Jansen, K., Klein, K.-M., 2020. Fully dynamic bin packing revisited. Mathematical Programming 179, 109–155.

Birge, J. R., Louveaux, F., 2011. Introduction to stochastic programming. Springer Science & Business Media.

Charnes, A., Cooper, W. W., 1959. Chance-constrained programming. Management Science 6 (1), 73–79.

Coffman, E. G., Csirik, J., Galambos, G., Martello, S., Vigo, D., 2013. Bin packing approximation algorithms: survey and classification. In: Handbook of Combinatorial Optimization. Springer New York, pp. 455–531.

Coffman, Jr, E. G., Garey, M. R., Johnson, D. S., 1983. Dynamic bin packing. SIAM Journal on Computing 12 (2), 227–258.

Correia, I., Gouveia, L., Saldanha-da Gama, F., 2008. Solving the variable size bin packing problem with discretized formulations. Computers & Operations Research 35 (6), 2103–2113.

Crainic, T. G., Gobbato, L., Perboli, G., Rei, W., 2016. Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. European Journal of Operational Research 253 (2), 404–417.

Deng, Y., Ahmed, S., Lee, J., Shen, S., 2017. Scenario grouping and decomposition algorithms for chance-constrained programs. Available on Optimization Online http://www. optimizationonline. org/DB HTML/2017/02/5853. html.

Deng, Y., Shen, S., 2016. Decomposition algorithms for optimizing multi-server appointment scheduling with chance constraints. Mathematical Programming 157 (1), 245–276.

Denton, B. T., Miller, A. J., Balasubramanian, H. J., Huschka, T. R., 2010. Optimal allocation of surgery blocks to operating rooms under uncertainty. Operations Research 58 (4-part-1), 802–816.

Gu, Z., Nemhauser, G. L., Savelsbergh, M. W., 1998. Lifted cover inequalities for 0-1 integer programs: Computation. INFORMS Journal on Computing 10 (4), 427–437.

Gu, Z., Nemhauser, G. L., Savelsbergh, M. W., 2000. Sequence independent lifting in mixed integer programming. Journal of Combinatorial Optimization 4 (1), 109–129.

Gul, S., Denton, B. T., Fowler, J. W., Huschka, T., 2011. Bi-criteria scheduling of surgical services for an outpatient procedure center. Production and Operations Management 20 (3), 406–417.

Günlük, O., Pochet, Y., 2001. Mixing mixed-integer inequalities. Mathematical Programming 90 (3), 429–457.

Hu, Z., Hong, L. J., Zhang, L., 2013. A smooth monte carlo approach to joint chance-constrained programs. IIE Transactions 45 (7), 716–735.

Kang, J., Park, S., 2003. Algorithms for the variable sized bin packing problem. European Journal of Operational Research 147 (2), 365–372.

Kaparis, K., Letchford, A. N., 2008. Local and global lifted cover inequalities for the 0–1 multidimensional knapsack problem. European Journal of Operational Research 186 (1), 91–103.

Kleinberg, J., Rabani, Y., Tardos, É., 2000. Allocating bandwidth for bursty connections. SIAM Journal on Computing 30 (1), 191–217.

Küçükyavuz, S., 2012. On mixing sets arising in chance-constrained programming. Mathematical Programming 132 (1-2), 31–56.

Kucukyilmaz, T., Kiziloz, H. E., 2018. Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. Computers & Industrial Engineering 125, 157–170.

Lodi, A., Martello, S., Monaci, M., 2002. Two-dimensional packing problems: A survey. European Journal of Operational Research 141 (2), 241–252.

Lodi, A., Martello, S., Vigo, D., 1999. Approximation algorithms for the oriented two-dimensional bin packing problem. European Journal of Operational Research 112 (1), 158–166.

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

Luedtke, J., Ahmed, S., 2008. A sample approximation approach for optimization with probabilistic constraints. SIAM Journal on Optimization 19 (2), 674–699.

Luedtke, J., Ahmed, S., Nemhauser, G. L., 2010. An integer programming approach for linear programs with probabilistic constraints. Mathematical Programming 122 (2), 247–272.

Martello, S., Pisinger, D., Vigo, D., 2000. The three-dimensional bin packing problem. Operations Research 48 (2), 256–267.

Nemhauser, G. L., Sigismondi, G., 1992. A strong cutting plane/branch-and-bound algorithm for node packing. Journal of the Operational Research Society 43 (5), 443–457.

Nemirovski, A., 2012. On safe tractable approximations of chance constraints. European Journal of Operational Research 219 (3), 707–718.

Nemirovski, A., Shapiro, A., 2006. Convex approximations of chance constrained programs. SIAM Journal on Optimization 17 (4), 969–996.

Padberg, M. W., 1973. On the facial structure of set packing polyhedra. Mathematical Programming 5 (1), 199–215.

Pagnoncelli, B., Ahmed, S., Shapiro, A., 2009. Sample average approximation method for chance constrained programming: theory and applications. Journal of Optimization Theory and Applications 142 (2), 399–416.

Peng, C., Delage, E., Li, J., 2019. Probabilistic envelope constrained multiperiod stochastic emergency medical services location model and decomposition scheme. forthcoming in Transportation Science.

Perboli, G., Gobbato, L., Perfetti, F., 2014. Packing problems in transportation and supply chain: new problems and trends. Procedia-Social and Behavioral Sciences 111 (0), 672–681.

Qiu, F., Ahmed, S., Dey, S. S., Wolsey, L. A., 2014. Covering linear programming with violations. INFORMS Journal on Computing 26 (3), 531–546.

Reich, D., Shi, Y., Epelman, M., Cohn, A., Barnes, E., Arthurs, K., Klampfl, E., 2016. Scheduling crash tests at ford motor company. Interfaces 46 (5), 409–423.

Rockafellar, R. T., Uryasev, S., et al., 2000. Optimization of conditional value-at-risk. Journal of Risk 2, 21–42.

Ruszczyński, A., 2002. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. Mathematical Programming 93 (2), 195–215.

Scheithauer, G., 2017. Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods. Vol. 263. Springer.

Shylo, O. V., Prokopyev, O. A., Schaefer, A. J., 2012. Stochastic operating room scheduling for high-volume specialties under block booking. INFORMS Journal on Computing 25 (4), 682–692.

Song, G., Kowalczyk, D., Leus, R., 2018. The robust machine availability problem–bin packing under uncertainty. IISE Transactions 50 (11), 997–1012.

Song, Y., Luedtke, J. R., Küçükyavuz, S., 2014. Chance-constrained binary packing problems. INFORMS Journal on Computing 26 (4), 735–747.

Spangler, W. E., Strum, D. P., Vargas, L. G., May, J. H., 2004. Estimating procedure times for surgeries by determining location parameters for the lognormal model. Health Care Management Science 7 (2), 97–104.

Tanner, M. W., Ntaimo, L., 2010. Iis branch-and-cut for joint chance-constrained stochastic programs and application to optimal vaccine allocation. European Journal of Operational Research 207 (1), 290–296.

Wang, W., Ahmed, S., 2008. Sample average approximation of expected value constrained stochastic programs. Operations Research Letters 36 (5), 515–519.

Watson, J.-P., Wets, R. J., Woodruff, D. L., 2010. Scalable heuristics for a class of chance-constrained stochastic programs. INFORMS Journal on Computing 22 (4), 543–554.

Wei, L., Luo, Z., Baldacci, R., Lim, A., 2020. A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. INFORMS Journal on Computing 32 (2), 428–443.

Xie, W., Ahmed, S., 2018. On quantile cuts and their closure for chance constrained optimization problems. Mathematical Programming 172 (1-2), 621–646.

Zemel, E., 1989. Easily computable facets of the knapsack polytope. Mathematics of Operations Research 14 (4), 760–764.

Zhang, Y., Jiang, R., Shen, S., 2018. Ambiguous chance-constrained binary programs under mean-covariance information. SIAM Journal on Optimization 28 (4), 2922–2944.

Zhang, Z., Denton, B. T., Xie, X., 2020. Branch and price for chance-constrained bin packing. INFORMS Journal on Computing.

Zhao, M., Huang, K., Zeng, B., 2017. A polyhedral study on chance constrained program with random right-hand side. Mathematical Programming 166 (1-2), 19–64.

Appendix A: CVaR Approximation

In this section, we briefly review the CVaR approximation. We first consider the case where ξ is a N-dimensional continuous random vector. Note that $p(y) := \mathbb{P}\left\{\sum_{i\in\mathcal{I}} \xi_i y_{ij} > t_j\right\} = \mathbb{E}\left[\mathbf{1}_{(0,+\infty)}(\sum_{i\in\mathcal{I}} \xi_i y_{ij} - t_j)\right]$. Because $1_{(0,+\infty)}(\cdot)$ is a step function, let $\phi(\cdot)$ be a convex approximation of $\mathbf{1}_{(0,+\infty)}(\cdot)$ such that $\phi(\cdot) \geq \mathbf{1}_{(0,+\infty)}(\cdot)$. Clearly, a $\phi(\cdot)$ with smaller value gives a better approximation of $\mathbf{1}_{(0,+\infty)}(\cdot)$. CVaR approximation uses $\phi(x,\tau) = \frac{1}{\tau}[\tau + x]^+$ to approximate $\mathbf{1}_{(0,+\infty)}(x)$, where $[\cdot]^+ = \max\{0,\cdot\}$. The CVaR approximation of the chance constraint is given as:

$$\inf_{\tau>0} \mathbb{E}\left[\frac{1}{\tau}\left[\tau + \sum_{i\in\mathcal{I}} \xi_i y_{ij} - t_j\right]^+\right] \leq \varepsilon$$

$$\Leftrightarrow \text{CVaR}_\varepsilon\left\{\sum_{i\in\mathcal{I}} \xi_i y_{ij} - t_j\right\} = \inf_{\eta\in\mathbb{R}}\left\{\eta + \frac{1}{\varepsilon}\mathbb{E}\left[\left[\sum_{i\in\mathcal{I}} \xi_i y_{ij} - t_j - \eta\right]^+\right]\right\} \leq 0.$$

When ξ is N-dimensional discrete random vector, according to Ahmed and Xie (2018), the CVaR approximation is also valid.

Proposition 7. *The CVaR approximation of (CBP) can be reformulated as*

$$\text{(CVaR)} \qquad \underset{\boldsymbol{x},\boldsymbol{y},\eta,\boldsymbol{\rho}}{\text{minimize}} \sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^b y_{ij} \tag{26a}$$

subject to (1a), (1b), (1d)

$$\eta + \frac{1}{\varepsilon}\sum_{\omega\in\Omega} p_\omega \rho_j^\omega \leq 0 \qquad \forall j \in \mathcal{J} \tag{26b}$$

$$\eta + \rho_j^\omega \geq \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} - t_j \qquad \forall j \in \mathcal{J}, \omega\in\Omega \tag{26c}$$

$$\boldsymbol{\rho} \geq 0. \tag{26d}$$

Proof Let $(\boldsymbol{x},\boldsymbol{y},\eta,\boldsymbol{\rho})$ be a solution of (CVaR). We now prove that $(\boldsymbol{x},\boldsymbol{y})$ is a feasible solution of (CBP). For all $j\in\mathcal{J}$, let $\Omega_j^0 = \{\omega\in\Omega : \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} - t_j > 0\}$. If $\sum_{\omega\in\Omega_j^0} p_\omega \leq \varepsilon$ holds, it implies $(\boldsymbol{x},\boldsymbol{y})$ is a feasible solution of (CBP). According to constraints (26b) and (26c),

$$\eta + \frac{1}{\varepsilon}\sum_{\omega\in\Omega_j^0} p_\omega \left(\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} - t_j - \eta\right) \leq 0.$$

Let $H_j = \min_{\omega\in\Omega_j^0} \xi_i^\omega y_{ij} - t_j$, then we have

$$\sum_{\omega\in\Omega_j^0} p_\omega \leq \frac{-\varepsilon\eta}{H_j - \eta} \leq \varepsilon,$$

where the second inequality in the above expression is because $\eta \leq 0$. Hence, $(\boldsymbol{x},\boldsymbol{y})$ is a feasible solution of (CBP). □

Appendix B: Approximation Comparison

In this section, we compare the computational results for (CBP) with the CVaR approximation formulation (26) (denoted by (CVaR)), which is presented in Appendix A. We set the runtime limit to 2 hours. We

Table 7 The average (AvT), maximum (max), minimum (min) CPU solution time (seconds), the number of opened ORs (# of ORs) and the number of solved instances from the five generated instances (solved), for the B&C of (CBP) and CVaR approximation

N	model	AvT	max	min	# of ORs	solved
100	CBP	528.9	2,424.7	5.2	[6, 5, 5, 5, 5]	5/5
	CVaR	88.1	221.3	2.5	[7, 6, 6, 6, 7]	5/5
500	CBP	621.2	905.6	220.5	[5, 5, 5, 5, 5]	5/5
	CVaR	398.8	600.1	35.1	[6, (6,7), 6, (6,7), 6]	3/5
1000	CBP	1,876.6	4,637.3	595.3	[5, 5, 5, 5, 5]	5/5
	CVaR	–	–	–	[(6,7), (6,7), (6,7), (6,7), (6,7)]	0/5

"–" means that no instance can be solved to optimality within the runtime limit.

use B&C described in Section 5.2.2 to solve (CBP). We report the average, maximum, minimum time, the number of opened ORs, the number of solved instances from the five instances for $\varepsilon = 0.1$ in Table 7.

We can see from Table 7 that (CBP) has a better performance than the CVaR approximation formulation in terms of the number of solved instances. We notice that the CVaR approximation can only solve 3 out of 5 instances within the runtime limit when $N = 500$, and cannot solve any instance to optimality when $N = 1000$. The CVaR approximation solutions open more ORs. For example, for the 100 scenario instances, the CVaR approximation opens 6 or 7 ORs, while (CBP) only opens 5 or 6 ORs. Therefore, the CVaR approximation formulation is more conservative than (CBP).

Appendix C: Proof

C.1. Proof of Proposition 1

Let $\boldsymbol{y}_j^*$ be an optimal solution of (3). Then, there exists at least one $k' \in \{k_1, \ldots, k_{q+1}\}$ such that $\sum_{i\in\mathcal{I}} \xi_i^{k'} y_{ij}^* \le t_j$. Otherwise, we have $\sum_{i\in\mathcal{I}} \xi_i^k y_{ij}^* > t_j$, for $k \in \{k_1, \ldots, k_{q+1}\}$. Since $\sum_{j=1}^{q+1} p_{k_j} > \varepsilon$, the inequality $\mathbb{P}\left\{\sum_{i\in\mathcal{I}} \xi_i y_{ij}^* \le t_j\right\} \ge 1-\varepsilon$ is violated. This is a contradiction. Therefore, $\boldsymbol{y}_j^*$ is a feasible solution of (4) with $k = k'$. We have $m_j^\omega(k_{q+1}) \ge m_j^\omega(k') \ge \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij}^* = \bar{M}_j^\omega$. Thus, $m_j^\omega(k_{q+1})$ is an upper bound for $\bar{M}_j^\omega$.

Based on the definition of $m_j^\omega(\omega)$, we have $\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \le m_j^\omega(\omega)$. Let $M_j^\omega = m_j^\omega(k_{q+1})$. By replacing t_j with $m_j^\omega(\omega)$, constraints (2a) are reformulated as (5). $\square$

C.2. Proof of Proposition 2

In order to prove that the inequality (7) is valid, let $(\boldsymbol{y}_j, z_j^\omega)$ be a feasible solution of (CBP), and $n^* = \min\left\{n \in \{1, \ldots, l\} : z_j^{\tau_n} = 1\right\}$. Then we have $\sum_{i\in\mathcal{I}} \xi_i^{\tau_{n^*}} y_{ij} \le t_j$ and $z_j^{\tau_n} = 0$, for $n \in \{1, \ldots, n^*-1\}$. Thus, $\boldsymbol{y}_j$ is a feasible solution of (4) for $k = \tau_{n^*}$, which indicates $\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} \le m_j^\omega(\tau_{n^*})$. Therefore,

$$
\begin{aligned}
&\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij} + \sum_{n=1}^{l} \left(m_j^\omega(\tau_{n+1}) - m_j^\omega(\tau_n)\right) z_j^{\tau_n} \le m_j^\omega(\tau_{n^*}) + \sum_{n=n^*}^{l} \left(m_j^\omega(\tau_{n+1}) - m_j^\omega(\tau_n)\right) z_j^{\tau_n} \\
&\le m_j^\omega(\tau_{n^*}) + \sum_{n=n^*}^{l} \left(m_j^\omega(\tau_{n+1}) - m_j^\omega(\tau_n)\right) = m_j^\omega(k_{q+1}).
\end{aligned}
$$

This completes our proof. $\square$

C.3. Proof of Proposition 3

We first prove that $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP). According to constraints (1d), we have $\sum_{\omega\in\Omega} p_\omega \mathbf{1}\left\{\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij}^* \le t_j\right\} \ge 1-\varepsilon$, where $\mathbf{1}\{\cdot\}$ is an indicator function, which returns 1 if the expression in $\{\cdot\}$ is true. Since $\mathbf{1}\left\{\sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij}^* \le t_j\right\} = z_j^{\omega*}$, $z_j^{\omega*}$ satisfies constraints (2b) based on the definition of $m_j^\omega(\omega)$. Therefore, $z_j^{\omega*}$ satisfies constraints (2b) and (8b), proving that $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is a feasible solution of (BIP).

On the other hand, suppose that $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ is an optimal solution of (BIP). We now show that $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is a feasible solution of (CBP). When $\hat{z}_j^\omega = 1$, we have $\sum_{i\in\mathcal{I}} \xi_i^\omega \hat{y}_{ij} \le m_j^\omega(\omega)$. Hence, constraints (2b) imply $\mathbb{P}\left\{\sum_{i\in\mathcal{I}} \xi_i \hat{y}_{ij} \le t_j\right\} \ge 1-\varepsilon$. We have that $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is also a solution of (CBP). Since $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an optimal solution of (CBP), $\sum_{j\in\mathcal{J}} c_j^a \hat{x}_j + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^b \hat{y}_{ij} \ge \sum_{j\in\mathcal{J}} c_j^a x_j^* + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^b y_{ij}^*$. Hence, $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP). Conversely, it is easy to verify that if $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP), then $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an optimal solution of (CBP). □

C.4. Proof of Theorem 3

We first prove that (16) is valid for $conv(\mathcal{F}_{j\omega})$. When $z_j^\omega = 1$, (16) is valid for $conv(\mathcal{F}_{j\omega})$ due to the valid of (15). When $z_j^\omega = 0$, since $\gamma = \operatorname{maximize}_{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}} \sum_{i\in\mathcal{C}\setminus\mathcal{D}} y_{ij} + \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i y_{ij} + \sum_{i\in\mathcal{D}} \beta_i y_{ij} - |\mathcal{C}\setminus\mathcal{D}| - \sum_{i\in\mathcal{D}} \beta_i + 1 = \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i + 1$, indicating (16) is also valid for $conv(\mathcal{F}_{j\omega})$.

When $z_j^\omega = 1$, there exists n feasible points of variables $\boldsymbol{y}_j$ that are affinely independent and satisfy inequality (16) at equality as the facet defining of (15). Similarly, when $z_j^\omega = 0$, $\boldsymbol{y}_j = 1_{|\mathcal{I}|}$, where $1_{|\mathcal{I}|}$ is a $1\times|\mathcal{I}|$ vector of all ones. Thus, the $|\mathcal{I}|+1$ feasible points are affinely independent and satisfy inequality (11) at equality. Therefore, we conclude that the inequality (11) is facet-defining for $conv(\mathcal{F}_{j\omega})$. □

C.5. Proof of Theorem 4

Let

$$\gamma = \operatorname*{maximize}_{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}} \sum_{i\in\mathcal{C}\setminus\mathcal{D}} y_{ij} + \sum_{i\in\mathcal{I}\setminus\mathcal{C}} \alpha_i y_{ij} + \sum_{i\in\mathcal{D}} \beta_i y_{ij} - |\mathcal{C}\setminus\mathcal{D}| - \sum_{i\in\mathcal{D}} \beta_i + 1 \tag{27a}$$

$$\text{subject to} \quad \sum_{k\in\Omega\setminus\{\omega\}} p_k \mathbf{1}\left\{\sum_{i\in\mathcal{I}} \xi_i^k y_{ij} \le t\right\} \ge 1-\varepsilon. \tag{27b}$$

Then (16) is valid for $conv(\mathcal{F}_{j\omega})$.

Let $\boldsymbol{y}_j^*$ be an optimal solution of (27), then, there exists at least one $k' \in \{k_1,\dots,k_{q+1}\} \subseteq \{\Omega\setminus\{\omega\}\}$ such that $\sum_{i\in\mathcal{I}} \xi_i^{k'} y_{ij}^* \le t_j$. Therefore, $\boldsymbol{y}_j^*$ is a feasible solution of $\delta_{k'}^1$. We have $\delta_{k_{q+1}}^1 \ge \delta_{k'}^1 \ge \gamma$. Hence, (16) is a valid inequality for (CBP) when $\gamma = \delta_{k_{q+1}}^1$. □

C.6. Proof of Theorem 6

Let

$$\mu = \operatorname*{maximize}_{\boldsymbol{y}_j\in\{0,1\}^{|\mathcal{I}|}} \sum_{i\in\mathcal{K}} y_{ij} - 1 \tag{28a}$$

$$\text{subject to} \quad \sum_{k\in\Omega\setminus\{\omega\}} p_k \mathbf{1}\left\{\sum_{i\in\mathcal{I}} \xi_i^k y_{ij} \le t\right\} \ge 1-\varepsilon. \tag{28b}$$

Then (18) is valid for $conv(\mathcal{F}_{j\omega})$.

It is straightforward to verify that $\lambda_{k_{q+1}} \ge \mu$. Hence, (18) is a valid inequality for (CBP) when $\mu = \lambda_{k_{q+1}}$. □

C.7. Proof of Proposition 6

Let $\boldsymbol{u}^* = \boldsymbol{y}^*\boldsymbol{z}^*$. For $j \in \mathcal{J}$, and $\omega \in \Omega$, we have

$$m_j^\omega(\omega) z_j^{\omega *} \geq \sum_{i\in\mathcal{I}} \xi_i^\omega y_{ij}^* z_j^{\omega *} = \sum_{i\in\mathcal{I}} \xi_i^\omega u_{ij}^{\omega *}.$$

Since $\boldsymbol{y}^*, \boldsymbol{z}^*$ are binary variables, constraints (20b)-(20d) hold. Therefore, $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{u}^*)$ is a solution of (20). Now suppose $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{u}})$ is an optimal solution of (20). If $\hat{z}_j^\omega = 0$,

$$m_j^\omega(\omega) \hat{z}_j^\omega \geq \sum_{i\in\mathcal{I}} \xi_i^\omega \hat{u}_{ij}^\omega = \sum_{i\in\mathcal{I}} \xi_i^\omega \hat{y}_{ij} \hat{z}_j^\omega.$$

Otherwise,

$$m_j^\omega(\omega) \hat{z}_j^\omega \geq \sum_{i\in\mathcal{I}} \xi_i^\omega \hat{u}_{ij}^\omega = \sum_{i\in\mathcal{I}} \xi_i^\omega \hat{y}_{ij} = \sum_{i\in\mathcal{I}} \xi_i^\omega \hat{y}_{ij} \hat{z}_j^\omega.$$

Hence, $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ is a solution of (BIP), which implies $\sum_{j\in\mathcal{J}} c_j^a \hat{x}_j + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^b \hat{y}_{ij} \geq \sum_{j\in\mathcal{J}} c_j^a x_j^* + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^b y_{ij}^*$. Therefore, $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{u}^*)$ is an optimal solution of (20). In a similar way, we can prove that if $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{u}^*)$ is an optimal solution of (20), then $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is an optimal solution of (BIP). The proposition follows. □

Appendix D: Implementation Details for BPC

We formulate (CBP) as a probability cover problem:

$$\text{(BPC)} \qquad \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^b y_{ij} \tag{29a}$$

$$\text{subject to } y_{ij} \leq x_j, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, \tag{29b}$$

$$\sum_{j\in\mathcal{J}} y_{ij} = 1, \qquad \forall i \in \mathcal{I}, \tag{29c}$$

$$\sum_{i\in C_j} y_{ij} \leq |C_j| - 1, \qquad \forall j \in \mathcal{J}, C_j \in \mathscr{P}, \tag{29d}$$

$$x_j \in \{0,1\}, y_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, \tag{29e}$$

where C_j is a minimal probability cover such that $\mathbb{P}\left\{\sum_{i\in C_j} \xi_i \leq t_j\right\} < 1-\varepsilon$, for $j \in \mathcal{J}$. Then we lift (29d) to derive a strong valid inequality based on the method proposed in Song et al. (2014). For $j \in \mathcal{J}$, let $\bar{\boldsymbol{\pi}}_j = \{\bar{\pi}_{1j}, \dots, \bar{\pi}_{|\mathcal{I}\setminus C_j|j}\}$ be a sequence of $\mathcal{I}\setminus C_j$, and the coefficients for y_{ij} be $\bar{\alpha}_{ij}$. The lifting problem is as follows: for $j \in \mathcal{J}$ and $k = \{1, \dots, |\mathcal{I}\setminus C_j|\}$

$$obj_{\bar{\pi}_{kj}} := \underset{\boldsymbol{y}_j}{\text{maximize}} \sum_{i\in C_j} y_{ij} + \sum_{i=\bar{\pi}_{1j}}^{\bar{\pi}_{k-1,j}} \bar{\alpha}_{ij} y_{ij}$$

$$\text{subject to } \mathbb{P}\left\{\sum_{i\in C_j} \xi_i y_{ij} + \sum_{i=\bar{\pi}_{1j}}^{\bar{\pi}_{k-1,j}} \xi_i y_{ij} \leq t_j - \xi_{\pi_k}\right\} \geq 1-\varepsilon,$$

$$y_{ij} \in \{0,1\}, \qquad i \in C_j \bigcup \{\bar{\pi}_{1j}, ..., \bar{\pi}_{k-1,j}\}.$$

The lifting coefficient $\bar{\alpha}_{\bar{\pi}_{kj},j} = |C_j| - 1 - obj_{\bar{\pi}_{kj}}$. A sequential lifting strategy to approximate the value of the lifting coefficients was given in Algorithm 1 in Song et al. (2014). We used the same algorithm to lift the coefficients in our case. We then strengthen the lifted probabilistic cover inequalities by multiplying the right-hand side with the variable x_j.

Let $(\hat{\text{BPC}})$ be (BPC) without constraints (29d). At each round of cut generation, we search for violated lifted probabilistic cover inequalities. If the solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the relaxation problem of $(\hat{\text{BPC}})$ is integral, we add an available violated lifted probabilistic cover inequality. We found that adding a violated lifted probabilistic cover inequality at fractional solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ was less efficient. Hence, the implementation adds these inequalities after a binary solution of the problem generated after each fractional solution is obtained. Algorithm 5 gives an overview of the implementation.

Algorithm 5: (BPC) Implementation

1 **Initialize** $\text{UB} = +\infty$, $\text{LB} = -\infty$ and $\mathcal{N} = \varnothing$.
2 **Initialize** Nodelist $\mathcal{N} = \{o\}$, 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   Optimize the LP relaxation problem of $(\hat{\text{BPC}})$ in the node o.

$$(\hat{\text{BPC}}) \qquad \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \sum_{j\in\mathcal{J}} c_j^a x_j + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} c_{ij}^b y_{ij}$$
$$\text{subject to } (29b), (29c)(29e)$$

6   Obtain the optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and objective value $\hat{obj}$.
7   **if** $\hat{obj} < UB$ **then**
8   **if** $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ *is integral* **then**
9   **if** $\exists C_j \in \mathscr{P}$ *such that* $\hat{\boldsymbol{y}}_j$ *violates* (29d)*, for* $j \in \mathcal{J}$ **then**
10   Add the violated lifted probabilistic cover inequalities to $(\hat{\text{BPC}})$. Go to line 5.
11   **end**
12   **else**
13   Update UB, $\text{UB} = \hat{obj}$.
14   **end**
15   **end**
16   **else**
17   Branch, resulting in nodes o^* and o^{**}, $\mathcal{N} \leftarrow \mathcal{N} \cup \{o^*, o^{**}\}$.
18   **end**
19   **end**
20 **end**
21 **return** UB and its corresponding optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$.

Appendix E: Dynamic Programming Approach for Computing Big-M values

In this appendix, we give the details about using *dynamic programming* approach to compute the Big-M values in the model reformulation.

For $j \in J$, if t_j is moderate, dynamic programming is an efficient approach for solving (4) to optimality. For $j \in J$, let $D(|\mathcal{I}|, t_j)$ represents (4), where $|\mathcal{I}|$ denotes the $|\mathcal{I}|$ variables of $\boldsymbol{y}_j$. Let us consider the subproblem

$D(n,t)$ of $D(|\mathcal{I}|,t_j)$ which includes the first n variables of $\boldsymbol{y}_j$ and the right-hand side values of constraints in (4) respectively. Let $S(n,t)$ be the optimal objective value of $D(n,t)$. If $D(n,t)$ is infeasible, we set $S(n,t) = -\infty$. Since y_{nj} is binary, if $y_{nj} = 0$, $S(n,t)$ is equal to $S(n-1,t)$, which is the optimal objective value of the subproblem $D(n-1,t)$. If $y_{nj} = 1$, $S(n,t)$ is equal to $S(n-1,t-\xi_n^k)+\xi_n^\omega$, which is the optimal objective value of the subproblem $D(n-1,t-\xi_n^k)$ plus ξ_n^ω. Thus, for $0 \le t \le t_j$ and $n = 1,\dots,|\mathcal{I}|$, if $t \ge \xi_n^k$

$$S(n,t) = \max\{S(n-1,t), S(n-1,t-\xi_n^k)+\xi_n^\omega\},$$

else $S(n,t) = S(n-1,t)$, with an initial condition $S(0,t) = 0$ and $S(n,0) = 0$. Hence,

$$m_j^\omega(k) = S(|\mathcal{I}|,t_j).$$