INFORMS JOURNAL ON COMPUTING
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0899-1499 | EISSN 1526-5528 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/ijoc.2014.0595
© 0000 INFORMS

Authors are encouraged to submit new papers to INFORMS journals by means of a style file template, which includes the journal title. However, use of a template does not certify that the paper has been accepted for publication in the named journal. INFORMS journal templates are for the exclusive purpose of submitting to an INFORMS journal and should not be used to distribute the papers in print or online or to submit the papers to another publication.

Chance-Constrained Binary Packing Problems

Yongjia Song, James R. Luedtke
University of Wisconsin-Madison, jrluedt1@wisc.edu

Simge Küçükyavuz
The Ohio State University, kucukyavuz.2@osu.edu

We consider a class of packing problems with uncertain data, which we refer to as the *chance-constrained binary packing* problem. In this problem, a subset of items is selected that maximizes the total profit so that a generic packing constraint is satisfied with high probability. Interesting special cases of our problem include chance-constrained knapsack and set packing problems with random coefficients. We propose a problem formulation in its original space based on the so-called *probabilistic covers*. We focus our solution approaches on the special case in which the uncertainty is represented by a finite number of scenarios. In this case, the problem can be formulated as an integer program by introducing a binary decision variable to represent feasibility of each scenario. We derive a computationally efficient coefficient strengthening procedure for this formulation, and demonstrate how the scenario variables can be efficiently projected out of the linear programming relaxation. We also study how methods for lifting deterministic cover inequalities can be leveraged to perform approximate lifting of probabilistic cover inequalities. We conduct an extensive computational study to illustrate the potential benefits of our proposed techniques on various problem classes.

Key words: Chance constrained stochastic programming; Integer programming

1. Introduction

The chance-constrained binary packing problem is to select the items with random weights that maximize the total profit and satisfy the packing constraints *jointly* with probability at least $1-\epsilon$, where ϵ is a given reliability threshold. Specifically, given a set N of n items, $N=\{1,\dots,n\}$, with profit vector $c \in \mathbb{Q}_{+}^{n}$, nonnegative m-dimensional random capacity vector $\tilde{b}$, and nonnegative random $m \times n$ weight matrix $\tilde{A}$, the chance-constrained binary packing problem is given by

$$\max_{x \in \{0,1\}^n} \{cx \mid \mathbb{P}(\tilde{A}x \leq \tilde{b}) \geq 1-\epsilon\}. \tag{1}$$

When $m=1$ we call (1) a *chance-constrained individual knapsack problem.* When $\tilde{A}$ is a random 0-1 matrix, we call (1) a *chance-constrained set packing problem.*

We begin by proposing a formulation based on an exponential family of inequalities corresponding to *probabilistic covers*, which is a probabilistic generalization of knapsack covers for deterministic problems. This formulation can be solved using delayed constraint generation within a branch-and-cut algorithm provided that the probability $\mathbb{P}(\tilde{A}x \leq \tilde{b})$ in (1) can be calculated for any $x \in \{0,1\}^n$.

By itself, the cover-based formulation is likely to have a weak linear programming (LP) relaxation, and hence is likely to be impractical for all but the smallest instances. We therefore extensively study the special case in which the random variables $(\tilde{A}, \tilde{b})$ have finite support. In this finite scenario case, (1) can be formulated as an explicit binary integer program by introducing binary variables z_k, where $z_k = 1$ implies that the packing constraints are satisfied in scenario k. This "extended formulation" requires the use of "big-M" coefficients to prevent the constraints of scenario k from being binding if $z_k = 0$. Qiu et al. (2013) recently demonstrate that by improving these coefficients with an iterative scheme, the LP relaxation of the extended formulation of a chance-constrained *continuous* covering problem can be significantly improved. The disadvantage of their approach is that it can be very time consuming when the number of scenarios is large. We propose a coefficient strengthening procedure that is significantly more efficient than this iterative scheme, but which yields coefficients of similar quality in our experiments. While this procedure may improve the LP relaxation of the extended formulation, the size of this formulation can become too large as the number of scenarios increases, because it requires introducing a binary variable and a set of packing constraints for every scenario. To overcome this drawback, we characterize the projection of the LP relaxation of this formulation into the space of the original x variables, and demonstrate that separating inequalities defining this projected set can be done efficiently. These inequalities can therefore be used as cuts to strengthen our cover-based formulation, which does not introduce the scenario variables z. We also study how the cover inequalities defining our formulation can be approximately sequentially lifted, leveraging techniques for sequential lifting of deterministic knapsack problems. We perform an extensive computational study and find that our formulation scales much better than the extended formulation in terms of the number of branch-and-bound nodes explored and the solution time as the number of scenarios increases.

In general, chance-constrained stochastic programs (CCSPs) are hard to solve, primarily because the feasible region characterized by chance constraints is non-convex in general (Nemirovski and Shapiro 2006), although promising results have been obtained for some special cases. For example, under certain assumptions on the random vector $\tilde{a}$, the individual chance constraint $\mathbb{P}(\tilde{a}x \leq \tilde{b}) \geq 1-\epsilon$ can either be exactly represented (Charnes and Cooper 1963) or approximated by a conic quadratic knapsack constraint (Atamtürk and Narayanan 2009). In particular, when the random vector $\tilde{a}$ is normally distributed with a known mean a and correlation matrix Σ, the knapsack capacity $\tilde{b} = b$ is deterministic and $\epsilon < 1/2$, the individual chance constraint can be exactly reformulated as:

$$X_{CQ} := \{x \in \{0,1\}^n : ax + \Phi^{-1}(1-\epsilon)\sqrt{x^T \Sigma x} \leq b\}, \tag{2}$$

where $\Phi^{-1}(\cdot)$ is the quantile function of the standard normal distribution. In our experience, this binary convex quadratic program can be solved very efficiently by commercial solvers, e.g. CPLEX. Unfortunately, such a reformulation is not obtainable when considering a joint constraint with multiple rows (i.e., $m > 1$), or with distributions other than joint normal. For example, this formulation is not possible for chance-constrained set packing problems.

If only some partial information on the random vector $\tilde{a}$ is available, such as the first two moments of $\tilde{a}$, a conservative approximation of (1) with a similar structure to (2) can be constructed (Ben-Tal et al. 2009). Such an approximation yields feasible solutions to (1), but unfortunately does not have any guarantee on solution quality. Conservative approximations can also be obtained by sampling the constraints and solving a deterministic problem in which all of the sampled constraints are enforced (Calafiore and Campi 2005, 2006, Campi and Garatti 2011). In this paper, we aim to solve (1) exactly or solve a finite scenario sample approximation of (1) to obtain statistical lower and upper bounds.

Luedtke and Ahmed (2008) and Pagnoncelli et al. (2009) have studied sample average approximation (SAA) of CCSPs in which only a fraction of the sampled constraints are required to be satisfied. Such an SAA problem preserves the structure of a CCSP, but replaces the distribution with a finite scenario approximation. While this SAA problem is computationally more challenging to solve, it has the advantage that the optimal value of such an SAA problem converges to the true optimal value as the sample size increases. The SAA problem can also be used to derive statistical confidence intervals on the optimal value of the true problem (Nemirovski and Shapiro 2005). These SAA results motivate the problem of solving CCSPs in which the underlying distribution has a finite number of scenarios. When randomness appears only in the right-hand side $\tilde{b}$ of the constraints, approaches based on the p-efficient points of the distribution have been studied in Dentcheva et al. (2000), Beraldi and Ruszczyński (2002), Saxena et al. (2009), Dentcheva and Martinez (2012), Lejeune (2012). An approach based on studying the corresponding mixed-integer programming formulation has been effective for this case

(Luedtke et al. 2010, Küçükyavuz 2012). Methods for more general finite scenario CCSPs, where the constraint matrix $\tilde{A}$ may also be random, have been studied in Ruszczyński (2002), Beraldi and Bruni (2010), Tanner and Ntaimo (2010), Luedtke (2010), Beraldi et al. (2012), Luedtke (2013). None of these methods exploit the integrality of the discrete decision variables if they exist, in contrast to our method for lifting probabilistic cover inequalities.

This paper is organized as follows. In Section 2, we give a formulation and a solution method for the problem with any probability distribution for which the violation probability of any given solution can be calculated. Next, in Section 3, we present our results for finite scenario models. In Section 4 we present implementation details and results of our computational experiments with the proposed methods.

2. A cover-based formulation

In this section, we consider problem (1), with the assumption that the probability $\mathbb{P}(\tilde{A}x \leq \tilde{b})$ in (1) can be calculated for any $x \in \{0,1\}^n$. In general, exactly calculating $\mathbb{P}(\tilde{A}x \leq \tilde{b})$ is difficult, but this is possible when $(\tilde{A}, \tilde{b})$ has finite support of modest size, which is the case when solving a sample average approximation problem, and is the case that we focus most of this paper on. See the Online Supplement and Song (2013) for other examples where $\mathbb{P}(\tilde{A}x \leq \tilde{b})$ can be calculated efficiently.

We first give a reformulation of the problem (1). Given a set of items C, define $\phi(C) = \mathbb{P}\{\sum_{j \in C} \tilde{A}_{\cdot j} \nleq \tilde{b}\}$, where $\tilde{A}_{\cdot j}$ denotes column j of $\tilde{A}$. Thus, $\phi(C)$ is the probability that the packing constraint is violated by selecting all the items in C. If the probability of violation exceeds the threshold ϵ, then we cannot select all items in set C, which motivates the following definition.

Definition 1. C is a *probabilistic cover* if $\phi(C) > \epsilon$. Moreover, a probabilistic cover is minimal, if $\phi(C \setminus \{j\}) \leq \epsilon, \forall j \in C$.

By definition, if C is a probabilistic cover, then $\sum_{j\in C} x_j \leq |C|-1$ is a valid inequality for (1). We call it the *probabilistic cover inequality.* Any probabilistic cover defines a valid probabilistic cover inequality, but inequalities defined by non-minimal covers are dominated by those defined by minimal covers. Let $\mathcal{C}$ be the set of all probabilistic covers. Then we have the following formulation for (1):

$$\max_{x\in\{0,1\}^n} \quad cx \tag{3a}$$

$$\text{s.t.} \quad \sum_{j\in C} x_j \leq |C|-1, \quad \forall C \in \mathcal{C}. \tag{3b}$$

Although there are exponentially many constraints (3b), we can solve formulation (3) using delayed constraint generation within a branch-and-cut framework. In this approach we solve a relaxed master problem with a subset of constraints (3b) with branch-and-bound by branching on the x variables. Given an integer feasible solution $\hat{x} \in \{0,1\}^n$ at a node in the branch-and-bound tree, let $C = \{j \in N \mid \hat{x}_j = 1\}$. If $\phi(C) > \epsilon$, then C is a probabilistic cover. We add the corresponding inequality (3b) as a feasibility cut, resolve that node LP relaxation, and continue the branch-and-bound algorithm. Because the number of probabilistic covers is finite and branching is done on a finite set of binary decision variables, this algorithm converges to the optimal solution finitely.

The next definition is useful for deriving additional valid inequalities for (1).

Definition 2. A set of items P is a *probabilistic pack*, if $\phi(P) \leq \epsilon$. Moreover, a probabilistic pack is maximal, if $\phi(P \cup \{j\}) > \epsilon, \forall j \in N \setminus P$.

A probabilistic pack P is a set of items that can be feasibly selected if no other items are selected. If a probabilistic pack P is not maximal, some additional items from the set $N \setminus P$ can also be selected. Let $Q(P) = \{j \in N \setminus P \mid \phi(P \cup \{j\}) > \epsilon\}$ be the set of items that cannot be selected if all items in P are selected. A probabilistic pack P induces a valid inequality on the restricted space with x_j fixed to 1, $\forall j \in P$:

$$\sum_{j\in Q(P)} x_j \leq 0, \quad \text{if } x_j = 1, \forall j \in P. \tag{4}$$

We call this family of inequalities the *probabilistic pack inequalities*. Inequalities (4) are not valid for the formulation (3). However, we can perform lifting on variables in set P to make them valid.

2.1. Lifting

Lifting is a well-known technique for deriving strong valid inequalities for a closed set from inequalities that are valid for its lower dimensional restrictions (Atamtürk 2004, Padberg 1975). We leverage lifting techniques for deterministic binary knapsack problems in Zemel (1989), Gu et al. (1998a,b, 2000) and Kaparis and Letchford (2008) to strengthen the minimal probabilistic cover inequalities (3b) and to obtain valid inequalities from the probabilistic pack inequalities (4). An inequality (3b) defined by minimal probabilistic cover C is strong only with respect to a restricted feasible set where variables $x_j, j \notin C$ are fixed to 0. Let X be the feasible set of (3), and $X_C := \{x \in X \mid x_j = 0, \forall j \in N \setminus C\}$ be the restricted feasible set.

Proposition 1. *The probabilistic cover inequality* (3b) *defined by a minimal probabilistic cover C is facet-defining for* $\mathrm{conv}(X_C)$*.*

To illustrate lifting of minimal probabilistic cover inequalities, we consider a sequential uplifting procedure, where we sequentially lift a known valid inequality with a set of L variables fixed to 0. We start with a minimal probabilistic cover C, and follow a lifting sequence $\{\pi_k\}_{k=1}^{L}$. We define $T(i) = \{\pi_1, \pi_2, \ldots, \pi_i\}$, and let the corresponding lifting coefficient for each variable x_{π_j} be $\beta_{\pi_j}, j = 1, 2, \ldots, i-1$. The exact lifting problem for the next variable x_{π_i} in the sequence is:

$$\zeta^*_{\pi_i} = \max_{x \in \{0,1\}^{|C \cup T(i-1)|}} \quad \sum_{j \in C} x_j + \sum_{j \in T(i-1)} \beta_j x_j \tag{5a}$$

$$\text{s.t.} \quad \mathbb{P}\Big(\sum_{j \in C \cup T(i-1)} \tilde{A}_{\cdot j} x_j \le \tilde{b} - \tilde{A}_{\cdot \pi_i} \Big) \ge 1 - \epsilon. \tag{5b}$$

The exact lifting coefficient for variable x_{π_i} is given by $\beta^*_{\pi_i} = |C| - 1 - \zeta^*_{\pi_i}$. We also use a downlifting procedure (Gu et al. 1998b) to unfix variables x_j that have been fixed to value 1. This is equivalent to uplifting the complement of the variable $(1-x_j)$, which is fixed to 0.

After lifting the entire sequence of fixed variables, the lifted probabilistic cover inequalities are *facet-defining* for conv(X). (See Padberg (1973, 1975), Balas and Zemel (1978), etc.) However, solving even a single lifting problem (5) is hard, since it is another chance-constrained binary packing problem. Therefore, we look for an upper bound for the exact lifting problem to obtain a valid lifting coefficient. In Section 3 we describe a general technique for the finite scenario distribution case.

2.2. Local cuts

A general technique for deriving strong valid inequalities for integer programs is *local cuts* (Applegate et al. 2006, Chvatal et al. 2009). The idea of local cuts is to consider a restricted feasible set where many variables are fixed at one of their bounds. Exact separation of a valid inequality in this restricted region is accomplished by solving a polar LP that obtains a most violated valid inequality for the restricted set, where validity is imposed by directly considering all feasible solutions of the restricted feasible set. In our implementation, we explicitly enumerate the maximal packs in the restricted feasible set. Although we did not explore this option, another approach is to solve the dual of the polar LP via column generation, in which case the column generation subproblem would have the form of a (smaller) chance-constrained packing problem (Applegate et al. 2006). In a packing problem, the variables that are fixed to one can then be downlifted to obtain a valid inequality for the original set. Variables fixed to zero can be uplifted to strengthen the inequality.

The motivation for using local cuts in the chance-constrained binary packing problem is that, the probabilistic cover C in (3b) is only heuristically chosen, since the separation problems for cover inequalities and lifted cover inequalities are NP-hard even for deterministic knapsack problems. (See Kaparis and Letchford (2010) and references therein.) Moreover, probabilistic cover inequalities may not be the most violated inequalities at a given point $\hat{x}$. On the other hand, a local cut corresponds to the most violated valid inequality by $\hat{x}$ for a selected restricted set. In Section 4, we report our computational findings on when and how to generate local cuts in a branch-and-cut framework. We describe more details on local cuts in Section 2 of the Online Supplement.

3. Finite scenario approximation

In this section, we assume that we are able to sample from the known distribution of random weight matrix $\tilde{A}$ and capacity vector $\tilde{b}$, so that we can approximate the distribution using a finite set of scenarios S, where the probability of scenario k is $p_k, k \in S$. When the scenarios are obtained from a Monte Carlo sample, we have $p_k = 1/|S|$, $k \in S$.

Let A^k and b^k be the weight matrix and capacity vector, respectively, in scenario $k \in S$. We introduce a binary variable z_k for each scenario $k \in S$: if $z_k = 1$, the constraint $A^k x \leq b^k$ is enforced, otherwise it can be violated. The finite scenario approximation of (1) can be formulated as the following binary integer program:

$$\max \quad cx \tag{6a}$$

$$\text{s.t.} \quad \sum_{k \in S} p_k z_k \geq 1 - \epsilon \tag{6b}$$

$$A^k x \leq b^k + M^k(1 - z_k), \forall k \in S \tag{6c}$$

$$z \in \{0,1\}^{|S|}, x \in \{0,1\}^n, \tag{6d}$$

where M^k is a big-M coefficient vector ensuring that when $z_k = 0$, constraint (6c) does not cut off any feasible solution. A naive choice of M^k is given by $A^k e - b^k$, where e is a vector of all ones. The chance constraint is now represented by the knapsack constraint (6b).

3.1. Big-M coefficients strengthening

The extended formulation (6) with naively chosen big-M coefficients may have a poor LP relaxation bound. We consider the problem of strengthening big-M coefficients. In fact, a big-M coefficient M_i^k for scenario k and row i is valid if:

$$M_i^k \geq \bar{M}_i^k := \max_{x\in\{0,1\}^n} \sum_{j\in N} A_{ij}^k x_j - b_i^k \tag{7a}$$

$$\text{s.t.} \quad \sum_{k'\in S} p_{k'} \mathbf{1}(A^{k'}x \leq b^{k'}) \geq 1-\epsilon, \tag{7b}$$

where $\mathbf{1}(\cdot)$ is an indicator function that takes value 1 if the condition is satisfied, and 0 otherwise. Problem (7) is a chance-constrained binary packing problem with exactly the same set of constraints as the original problem. We do not expect to solve (7) exactly to get the "tightest" big-M coefficient $\bar{M}_i^k$. Instead, we look for efficient heuristic routines to get an upper bound. Qiu et al. (2013) propose a strengthening procedure by solving the LP relaxation of (7) iteratively to get better big-M coefficients. In each iteration t of the procedure, for each scenario $k \in S$ and row $i \in \{1,\ldots,m\}$ a coefficient $M_i^k(t+1)$ is calculated by solving an LP using big-M coefficients $M^k(t)$:

$$M_i^k(t+1) = \max_{x\in[0,1]^n, z\in[0,1]^{|S|}} \sum_{j\in N} A_{ij}^k x_j - b_i^k \tag{8a}$$

$$\text{s.t.} \quad \sum_{k\in S} p_k z_k \geq 1-\epsilon \tag{8b}$$

$$A^k x - M^k(t)(1-z_k) \leq b^k, \tag{8c}$$

where initial valid big-M coefficients $M^k(1)$ are given (e.g., the naive choice). When the number of scenarios is large, iteratively solving the LP (8) is time-consuming. We propose a computationally more efficient procedure for obtaining an upper bound for (7). Consider a fixed scenario $k \in S$ and row $i \in \{1,\ldots,m\}$. For each scenario $k' \in S$, we calculate:

$$\eta_i^k(k') := \max\Big\{\sum_{j\in N} A_{ij}^k x_j - b_i^k : A^{k'}x \leq b^{k'},\ x \in \{0,1\}^n\Big\}. \tag{9}$$

Then we sort $\{\eta_i^k(k')\}_{k'\in S}$ in a nondecreasing order: $\eta_i^k(\sigma_1) \le \eta_i^k(\sigma_2) \le \cdots \le \eta_i^k(\sigma_{|S|})$. Let $q = \max\{l \mid \sum_{j=1}^{l} p_{\sigma_j} \le \epsilon\}$. Then $\eta_i^k(\sigma_{q+1})$ gives an upper bound for (7). Indeed, because $\sum_{j=1}^{q+1} p_{\sigma_j} > \epsilon$, if x is any feasible solution to (7), then $A^{k'}x \le b^{k'}$ must hold for at least one $k' \in \{\sigma_1, \sigma_2, \ldots, \sigma_{q+1}\}$. But then x is a feasible solution to (9) for this k', and hence $\eta_i^k(\sigma_{q+1}) \ge \eta_i^k(k') \ge \sum_{j\in N} A_{ij}^k x_j - b_i^k$.

When performing coefficient strengthening, it is possible to obtain a negative big-M coefficient, $M_i^k < 0$. This means that the inequality

$$\sum_{j\in N} A_{ij}^k x_j \le b_i^k + M_i^k \tag{10}$$

is implied by the constraints in the problem, and because $M_i^k < 0$ this dominates the inequality $\sum_{j\in N} A_{ij}^k x_j \le b_i^k$ that is enforced when $z_k = 1$. On the other hand, because $M_i^k < 0$, the corresponding inequality in (6c), $\sum_{j\in N} A_{ij}^k x_j \le b_i^k + (1-z_k)M_i^k$ is *not* necessarily valid. In this case, we redefine the value b_i^k to be $b_i^k + M_i^k$, which does not change the problem because (10) is satisfied by any feasible solution. With this redefinition, $M_i^k = 0$ is now a valid big-M coefficient. We always perform this transformation if a negative big-M coefficient is encountered, and therefore we can assume $M_i^k \ge 0$ holds for all $k \in S$, $i = 1, \ldots, m$.

This approach may still be time-consuming, especially when $m > 1$ so that (9) is a multi-dimensional knapsack problem for each $k' \in S$. To obtain valid big-M coefficients more efficiently, we may further relax problem (9), for example by solving the LP relaxation of (9). We may even solve the LP relaxation of (9) with just a single row at a time, and then use the minimum of these single-row objective values as the relaxation bound. This approach is especially efficient because each single row knapsack LP can be solved by a simple sorting procedure. In Section 4, we show that our method and the iterative LP method both yield significant improvements in the big-M coefficients, and that our method is more efficient for our test instances.

3.2. Projection cuts

Although the relaxation bound can be improved substantially by the big-M coefficient strengthening procedure, the big-M formulation (6) still has a drawback of being large when we have a large number of scenarios, since it introduces one additional variable z_k and constraint set (6c) for each scenario k. We study the projection the LP relaxation of (6) onto the space of x variables, so that our proposed probabilistic cover formulation (3) can be augmented with the inequalities defining the projection. We note that this projection result does not depend on the packing problem structure, so it could be applied in other chance-constrained stochastic programs.

Let Z be the LP relaxation of the big-M extended formulation:

$$Z = \{x \in [0,1]^n, z \in [0,1]^{|S|} \mid (x,z) \text{ satisfy (6b), (6c)}\},$$

and $\text{proj}_x(Z)$ be the projection of Z onto x-space. For a fixed $\bar{S} \subseteq S$, let $\Psi(\bar{S})$ be the set of all mappings of the form $\psi : \bar{S} \to \{1,2,\ldots,m\}$ such that $M^k_{\psi(k)} > 0$ for $k \in \bar{S}$. Also, for $i \in \{1,\ldots,m\}$, let $A^k_{i\cdot}$ represent row i of the matrix A^k.

Theorem 1. $\text{proj}_x(Z)$ *is described by* $x \in [0,1]^n$ *and the inequalities:*

$$A^k x \le b^k + M^k, \quad \forall k \in S \tag{11}$$

$$\sum_{k\in\bar{S}} \frac{p_k}{M^k_{\psi(k)}}(A^k_{\psi(k)\cdot}x - b^k_{\psi(k)}) \le \epsilon, \qquad \forall \bar{S} \subseteq S, \psi \in \Psi(\bar{S}). \tag{12}$$

The proof of Theorem 1 is in Section 3 of the Online Supplement. We call inequalities (11) and (12) *projection cuts*. There are exponentially many inequalities in (12). However, given $\hat{x} \in [0,1]^n$, the most violated inequality from (12), if any, can be found as follows. First, let $\bar{S} = \{k \in S \mid A^k_{i\cdot}\hat{x} > b^k_i \text{ for some } i \in \{1,\ldots,m\} \text{ with } M^k_i > 0\}$. For each scenario $k \in \bar{S}$, choose $\psi(k) \in \arg\max_{i=1,\ldots,m}\{(A^k_{i\cdot}\hat{x} - b^k_i)/M^k_i \mid M^k_i > 0\}$. Therefore, the complexity of exact separation of inequality (12) is dominated by performing m matrix-vector multiplications (where the matrix has $|S|$ rows and n columns) to determine the set $\bar{S}$.

3.3. Approximate lifting

A simple idea for approximately lifting probabilistic cover inequalities is to adapt the idea of the extended cover inequalities used for the deterministic knapsack problem (Nemhauser and Wolsey 1988) to the probabilistic setting. We present this idea in Section 4 of the Online Supplement. In this section, we propose an efficient heuristic sequential lifting strategy that approximates the exact lifting problem (5) when the number of scenarios is finite. In our computational experience, we found that this strategy yields better performance than the extended cover inequalities.

We first find the optimal value $\zeta_{\pi_i}^k$ of a lifting problem similar to (5), except that we consider a separate lifting problem for each *individual* scenario $k \in S$:

$$\zeta_{\pi_i}(k) := \max_{x \in \{0,1\}^{|C|+|T(i-1)|}} \Big\{ \sum_{j \in C} x_j + \sum_{j \in T(i-1)} \beta_j x_j \mid \sum_{j \in C \cup T(i-1)} A^k_{\cdot j} x_j \leq b^k - A^k_{\cdot \pi_i} \Big\}. \tag{13}$$

We then sort $\{\zeta_{\pi_i}(k)\}_{k \in S}$ in a nondecreasing order: $\zeta_{\pi_i}(\sigma_1) \leq \zeta_{\pi_i}(\sigma_2) \leq \cdots \leq \zeta_{\pi_i}(\sigma_{|S|})$. Then, using an argument identical to that used for deriving an upper bound on the big-M coefficient problem (7), we have that $\zeta_{\pi_i}^* \leq \zeta_{\pi_i}(\sigma_{q+1})$, where $q = \max\{k \mid \sum_{j=1}^k p_{\sigma_j} \leq \epsilon\}$.

Since $\zeta_{\pi_i}(\sigma_{q+1})$ gives an upper bound on $\zeta_{\pi_i}^*$, $\beta'_{\pi_i} = |C| - 1 - \zeta_{\pi_i}(\sigma_{q+1})$ is a lower bound on the exact lifting coefficient $\beta_{\pi_i}^*$, and so β'_{π_i} is a valid lifting coefficient. Because this lifting coefficient is based on an approximate solution to (5), it may be negative, even though we know that zero is a valid coefficient. In Section 5 of the Online Supplement we provide an example where this occurs. We therefore use $\beta_{\pi_i} = \max\{\beta'_{\pi_i}, 0\}$ as the approximate lifting coefficient. Although this heuristic lifting method only ensures a valid lifting coefficient, we provide a testable sufficient condition for the heuristic lifting to be exact.

PROPOSITION 2. *Let $x_{\pi_i}(\sigma_{q+1}) \in \{0,1\}^n$ be an optimal solution to the individual lifting problem* (13) *corresponding to scenario σ_{q+1}. If $x_{\pi_i}(\sigma_{q+1})$ is feasible to* (5), *then the optimal value of this lifting problem $\zeta_{\pi_i}(\sigma_{q+1})$ equals the optimal value of exact lifting problem $\zeta_{\pi_i}^*$.*

In our computational experience, this sufficient condition is frequently met. For example, it is met for 86% of the calculated coefficients for instance 1-7-1 with 100 scenarios and for 54% of the coefficients for instance weish26-1 with 100 scenarios.

The lifting procedure is also applied to make probabilistic pack inequalities (4) valid. Recall that unlike probabilistic cover inequalities (3b), probabilistic pack inequalities (4) are only valid for a restricted set, thus lifting is necessary.

For each variable x_{π_i} in the lifting sequence, the heuristic lifting procedure involves solving a deterministic (multidimensional if $m>1$) knapsack problem to calculate $\zeta_{\pi_i}(k)$ for each scenario k, which can be a computational bottleneck if it is not implemented appropriately. We next present some details on efficient implementation for the heuristic sequential lifting procedure in different cases.

Individual knapsack: We present an efficient warm start strategy for heuristic sequential lifting in the case where the packing constraint is an individual knapsack constraint ($m=1$). We adopt the idea in Zemel (1989), which is a dynamic programming algorithm for calculating the lifting coefficients when solving deterministic binary knapsack problems via sequential lifting of minimal cover inequalities. To emphasize that this section focuses on the single row case, we use the notation a^k to represent the single row in the matrix A^k.

For calculating the heuristic lifting coefficient for variable x_{π_i} in the sequence $\{\pi_1, \pi_2, \dots, \pi_L\}$, we solve a binary knapsack problem for each scenario $k \in S$:

$$\begin{aligned} \zeta_{\pi_i}(k) = \max\nolimits_{x\in\{0,1\}^{|C\cup T(i-1)|}} \quad & \textstyle\sum_{j\in C} x_j + \sum_{j\in T(i-1)} \beta_j x_j \\ \text{s.t.} \quad & \textstyle\sum_{j\in C\cup T(i-1)} a^k_j x_j \le b^k - a^k_{\pi_i}. \end{aligned} \tag{14}$$

Equivalently, we solve the following problem:

$$\zeta_{\pi_i}(k) = \max\{y : \eta^k_{\pi_i}(y) \ge b^k - a^k_{\pi_i}\}$$

where

$$\begin{aligned} \eta^k_{\pi_i}(y) := \min\nolimits_{x\in\{0,1\}^{|C\cup T(i-1)|}} \quad & \textstyle\sum_{j\in C\cup T(i-1)} a^k_j x_j \\ \text{s.t.} \quad & \textstyle\sum_{j\in C} x_j + \sum_{j\in T(i-1)} \beta_j x_j \ge y. \end{aligned}$$

The calculation of the lifting coefficient on variable $x_{\pi_{i+1}}$ can be warm-started by using information calculated in the previous step. This idea is summarized in Algorithm 1.

Algorithm 1 A variant of Zemel's algorithm for approximate sequential lifting of probabilistic cover inequalities.

$\forall k \in S : l_y^k =$ sum of y smallest values in $\{a_j^k\}_{j\in C}, y = 1, 2, \ldots, |C|-1.$
$\forall k \in S : \eta_{\pi_1}^k(0) = 0$, $\eta_{\pi_1}^k(y) = l_y^k, y = 1, 2, \ldots, |C|-1$
for $i = 1, 2, \ldots, L$ **do**
  $\forall k \in S : \zeta_{\pi_i}(k) = \max\{y : \eta_{\pi_i}^k(y) \le b^k - a_{\pi_i}^k\}$
  sort $\{\zeta_{\pi_i}(k)\}_{k\in S}$ in a nondecreasing order: $\zeta_{\pi_i}(\sigma_1) \le \zeta_{\pi_i}(\sigma_2) \le \cdots \le \zeta_{\pi_i}(\sigma_{|S|})$,
  and set $\beta_{\pi_i} = \max\{|C| - 1 - \zeta_{\pi_i}(\sigma_{q+1}), 0\}$
  if $i < L$ **then**
    for $y = 0$ to $|C| + \sum_{l=1}^{i} \beta_{\pi_l}$ **do**
      if $y < \beta_{\pi_i}$ **then**
        $\eta_{\pi_{i+1}}^k(y) = \min\{\eta_{\pi_i}^k(y), a_{\pi_i}^k\}$
      else
        if $y > |C| + \sum_{l=1}^{i-1} \beta_{\pi_l}$ **then**
          $\eta_{\pi_{i+1}}^k(y) = \eta_{\pi_i}^k(y - \beta_{\pi_i}) + a_{\pi_i}^k$
        else
          $\eta_{\pi_{i+1}}^k(y) = \min\{\eta_{\pi_i}^k(y), \eta_{\pi_i}^k(y - \beta_{\pi_i}) + a_{\pi_i}^k\}$
        end if
      end if
    end for
  end if
end for

There are several differences between Zemel's lifting algorithm for the deterministic knapsack problem and this variant. First, for deterministic binary knapsack problem, starting with a minimal cover, Zemel's algorithm only needs to calculate $\eta_{\pi_i}(y)$ for $y = 0$ up to $y = |C| - 1$. However, in our case the coefficients are not necessarily bounded by $|C| - 1$, and so we also must calculate $\eta_{\pi_i}(y)$ for values of $y > |C| - 1$. (We provide an example where this is necessary in Section 5 of the Online Supplement.) Also, although not mentioned in Zemel (1989), $\eta_{\pi_{i+1}}(y) = \eta_{\pi_i}(y)$ always holds when $y < \beta_{\pi_i}$, since $\eta_{\pi_i}(y) \le a_{\pi_i}$ in deterministic sequential lifting of binary knapsack problems. However, in our setting $\eta_{\pi_i}^k(y) \le a_{\pi_i}^k$ does not always hold (again, see example in Online Supplement), and so in the case that

$y < \beta_{\pi_i}$ we must set $\eta^k_{\pi_{i+1}}(y) = \min\{\eta^k_{\pi_i}(y), a^k_{\pi_i}\}$ as opposed to just $\eta^k_{\pi_i}(y)$. The complexity of our algorithm is slightly higher than $|S|$ times the complexity of original Zemel's algorithm for a single scenario knapsack problem ($\eta^k_{\pi_i}(y)$ is calculated for $y = 0, 1, \ldots, |C| + \sum_{l=1}^{i-1} \beta_{\pi_l}$.) However, the key computational efficiency of Zemel's algorithm, the ability to reuse significant information from calculation of one coefficient to the next, is preserved.

Multi-dimensional knapsack: When $m > 1$, the lifting problem (13) for each scenario k is a deterministic multi-dimensional knapsack problem, and so Algorithm 1 cannot be directly used in this case. Kaparis and Letchford (2008) propose a heuristic lifting procedure for the deterministic multi-dimensional knapsack problem based on solving the LP relaxation of the lifting problem. In order to take advantage of the efficiency of Algorithm 1, we instead obtain an upper bound on (13) by taking the minimum of the single-row problems corresponding to each row. Specifically, to obtain an upper bound on $\zeta_{\pi_j}(k)$, for each row $i = 1, 2, \ldots, m$ we apply one step of Algorithm 1 to obtain the optimal value, $\zeta^i_{\pi_j}(k)$, of the single-row problem (14) with a^k replaced by $A^k_{i\cdot}$. Then, $\min_{i=1,\ldots,m} \zeta^i_{\pi_j}(k)$ is an upper bound for $\zeta_{\pi_j}(k)$. In our computational study we found that this heuristic lifting scheme is more efficient than solving consecutive LP relaxations.

Set packing: In the case of the chance-constrained set packing problem, the lifting problem (13) for each individual scenario k is a deterministic set packing problem. To obtain an upper bound in this case we use the same strategy as multi-dimensional knapsack case. However, in the special case of set packing constraints in which the coefficients are zero or one, the single-row lifting problem can be solved more efficiently by a simple sorting procedure.

4. Computational Experiments

In this section we study the computational performance of our proposed ideas for solving chance-constrained binary packing problems with finite scenario distribution. We compare our big-M strengthening procedure to the previously proposed iterative strengthening procedure in terms of both time and effectiveness. We also study the value of using lifted probabilistic cover inequalities and pack inequalities (4), local cuts, and projection cuts in the probabilistic cover formulation (3). Finally, we compare the best option for the probabilistic cover formulation with directly solving the extended formulation with strengthened big-M coefficients. We test on three types of packing constraints: individual knapsack constraints, multi-dimensional knapsack constraints, and set packing constraints.

4.1. Implementation Details

Violation thresholds: We search for violated lifted probabilistic cover inequalities and pack inequalities throughout the branch-and-bound tree for each round of cut generation. If a relaxation solution $\hat{x}$ is integral, we add any violated inequality, as this is required to ensure correctness of the algorithm. Otherwise, at the root node we add a candidate inequality $\alpha x \leq \beta$ if the violation, $\alpha\hat{x} - \beta$, is at least the violation threshold of 10^{-5}. After the root node, we use a violation threshold of $\max\{10^{-5}, 0.7 \times \theta\}$ for fractional solutions, where θ is the average violation value for the last five infeasible integral relaxation solutions. θ is initialized as 0. Since the separation procedures for lifted probabilistic cover and pack inequalities are heuristic, valid inequalities that are not violated by the current relaxation solution may be violated by relaxation solutions in later solves. As a result, for multi-dimensional knapsack problems we found it beneficial to save a lifted probabilistic cover or pack inequality for five consecutive relaxation solutions, and discard it only if none of these five relaxation solutions violates the inequality. For individual knapsack instances, we found that this strategy does not yield significant improvement, and so we do not use it for these instances.

Lifted probabilistic cover inequalities: In our heuristic sequential lifting procedure, we adopt computational strategies that have been successful for deterministic binary knapsack problems (Zemel 1989, Gu et al. 1998b, Kaparis and

Letchford 2008) for cover initialization, cover reduction, and sequence selection. We provide the main points here, but further details are available in Section 6 of the Online Supplement. Given a relaxation solution $\hat{x}$, we choose a cover C by sorting the values $\{\hat{x}_j\}_{j\in N}$ in a nonincreasing order: $\hat{x}_{\sigma_1} \geq \hat{x}_{\sigma_2} \geq \cdots \geq \hat{x}_{\sigma_n}$ and letting $C = \{\sigma_1, \ldots, \sigma_r\}$ where r is the smallest integer such that $\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ is a probabilistic cover. Items are then removed from C until it is minimal. We then let $C_2 = \{j \in C \mid \hat{x}_j = 1\}$, $C_1 = \{j \in C \mid \hat{x}_j < 1\}$, $F = \{j \notin C \mid 0 < \hat{x}_j < 1\}$, and $W = \{j \notin C \mid \hat{x}_j = 0\}$. Starting with inequality $\sum_{j\in C_1} x_j \leq |C_1| - 1$, we first perform uplifting on variables in F with variables in C_2 fixed to one, then perform down-lifting on variables in C_2. Finally, we optionally perform uplifting on variables in W. This is called the "default" sequence of lifting by Gu et al. (1998b).

It is possible for our heuristic sequential lifting procedure to encounter a *lifting coefficient failure* when calculating a coefficient for some variable x_t. Specifically, suppose we have lifted variables in set $V \subseteq F$, and we lift on variable x_t in the current step. The single-row lifting problem for each scenario $k \in S$ has the form:

$$\begin{aligned} \max_{x\in\{0,1\}^{|C_1\cup V|}} \quad & \textstyle\sum_{j\in C_1} x_j + \sum_{j'\in V} \beta_{j'} x_{j'} \\ \text{s.t.} \quad & \textstyle\sum_{j\in C_1\cup V} a_j^k x_j \leq b^k - a_t^k - \sum_{j\in C_2} a_j^k. \end{aligned} \tag{15}$$

Some of these problems may be infeasible since the right-hand-side of the knapsack constraint may be negative. Let G be the set of scenarios for which the lifting problem is infeasible. If $\sum_{k\in G} p_k > \epsilon$ then we cannot obtain a valid lifting coefficient (this occurs because $C_2 \cup \{t\}$ is a probabilistic cover for some row). In this case, we remove items from the set C_2 and put them into set C_1 (i.e., fewer variables are fixed to one) in lexicographic order until $\sum_{k\in G} p_k \leq \epsilon$.

Lifted probabilistic pack inequalities: We apply the heuristic sequential lifting procedure on pack inequalities. Given a relaxation solution $\hat{x}$, we obtain a maximal pack by setting $P = \{j \in N \mid \hat{x}_j = 1\}$ and then sequentially adding items $j \notin P$ to P in a nonincreasing order of $\hat{x}$ values until P is maximal. We then apply downlifting on variables in P sequentially in a nondecreasing order of the $\hat{x}_j$ values, and obtain: $\sum_{j\in N\setminus P} x_j + \sum_{j\in P} \beta_j x_j \leq \sum_{j\in P} \beta_j$. While we may look for lifted probabilistic pack inequalities at any fractional relaxation solution $\hat{x}$, we found that it is most effective to add lifted probabilistic pack inequalities when we encounter a lifting coefficient failure when performing uplifting on probabilistic cover inequalities. Specifically, we found that the set C_2 which caused the lifting coefficient failure is a good candidate for using as the probabilistic pack to derive the base probabilistic pack inequality from.

Local cuts: We used local cuts only for chance-constrained individual knapsack problems, as we did not find a setting for them that improved performance for multi-dimensional problems. At each round of cut generation, we apply local cuts only when we do not find any lifted probabilistic cover inequalities and pack inequalities. For these instances, the restricted set is obtained by fixing all but 10 variables to a bound. After enumerating all the maximal packs in the restricted set, we solve the cut generating polar LP (as described in Section 2 of the Online Supplement) to get the coefficients. The coefficients obtained directly from the polar LP might be fractional. We follow a continued fraction method as in Applegate et al. (2006) to compute a close integer approximation. Given the revised coefficients, we then recompute a valid right-hand side by evaluating the left-hand side for all maximal packs. This procedure ensures that the base inequality has integer coefficients, so that the heuristic lifting can still be done by Algorithm 1.

Projection cuts: Projection cut generation follows a different cut selection rule, as we observed that projection cuts are frequently very nearly parallel to each other. We therefore conduct a check for parallelism before adding the projection cuts following the ideas in Andreello et al. (2007). We keep a pool of projection cuts added so far $\{(\alpha^k, \beta^k) \mid k \in K\}$, and every time a new projection cut $\alpha x \leq \beta$ is generated, we compute a measure of parallelism between this cut and every cut in the pool, and do not add the cut if this measure exceeds a threshold $\mathrm{P}^{\max}$ for any previously added projection cut. For cuts with coefficients α^1, α^2, the measure we use is $\frac{<\alpha^1,\alpha^2>}{\|\alpha^1\|\times\|\alpha^2\|}$. We used $\mathrm{P}^{\max} = 0.9999$ for individual knapsack instances, $\mathrm{P}^{\max} = 0.999$ for multi-dimensional knapsack instances, and $\mathrm{P}^{\max} = 0.9$ for set packing instances. We apply a two-phase procedure for adding the projection cuts (11) and (12). In the first phase, we apply a cutting plane method for solving the linear programming relaxation defined by (11) and (12). (Although the number of inequalities (11) is not exponential, we still add these as cuts to avoid adding all $|S|m$ of these constraints.) The first phase is terminated when no violated inequalities that pass the parallel check are found. In the second phase, the inequalities found in the first phase are then added to the initial formulation for solving the probabilistic cover-based formulation (3), and the branch-and-cut algorithm begins. Within the branch-and-cut procedure, at each round of cutting plane generation we search for inequalities (11) and (12) that are violated by the current relaxation solution, and add them as cuts if they pass the parallel check.

Dominance inequalities: We also use a preprocessing procedure that uses dominance information from the problem instance. Ruszczyński (2002) has proposed the concept of dominance between different *scenarios*, so that precedence constraints of the form $z_k \geq z_{k'}$ can be added to strengthen the formulation. In contrast, we consider dominance of *items*. We say that item i dominates item j if $\forall k \in S$, $A^k_{\cdot i} \leq A^k_{\cdot j}$ component-wise and $c_i \geq c_j$. If item i dominates item j, then the optimal solution does not change if we add the dominance inequality $x_i \geq x_j$ to the formulation. Dominance between items is rare in the multi-dimensional knapsack instances, but we found the addition of dominance inequalities to be useful in the chance-constrained individual knapsack instances. The time to identify dominance relationships was less than 0.1 seconds in all our tests.

4.2. Test Instances

We generate individual chance-constrained knapsack problems and chance-constrained multi-dimensional knapsack problems based on available deterministic knapsack instances. We generate chance-constrained set packing instances based on deterministic set partition instances by changing the equality constraints into inequality constraints. All deterministic instances are from Beasley (1990). Each item $j \in N$ appears or not according to a Bernoulli distribution. An item that does not appear has weight zero in all rows. As a result, the item weights in different rows are not independent. For individual and multi-dimensional knapsack instances, if an item appears, then its weight in each row is normally distributed with mean equal to its weight in the deterministic instance and standard deviation equal to 0.1 times the mean. Given this distribution, we take independent samples of size 100, 500, 1000 and 3000, and for each sample size we take five different replications. For each instance and sample size, we report the average results over the five instances at that sample size. Section 7 of the Online Supplement contains complete details of the instance generation procedure.

4.3. Results

We use the following abbreviations throughout this section:

- AvT: Average time to solve a formulation with given big-M coefficients.
- AvN: Average number of branch-and-bound nodes processed.
- AvG: Average optimality gap, where optimality gap for an instance is calculated as $(UB-LB)/LB$ and UB and LB are the best upper and lower bounds, respectively, obtained by the algorithm within the time limit.
- AvS: Average time spent strenthening big-M coefficients.

Unless stated otherwise, all experiments were performed on a Linux workstation with eight 2.93GHz processors and 11.7Gb memory. We used the commercial integer programming solver IBM Ilog CPLEX, version 12.2, to implement the branch-and-cut algorithms for solving (3) and to solve the extended formulation (6). We set the number of threads to one. CPLEX presolve is turned off for the branch-and-cut algorithms for solving (3). For all experiments, we use a time limit of 3600 seconds. If some instance is not solved to optimality within the time limit, we show the number of instances out of five replications that are solved to optimality in parentheses, instead of the computational time. For calculating the average number of nodes, we use the number of nodes that have been processed up to the time limit (and denote this average with $>$ to indicate that it is a lower bound). We calculate the average optimality gap for instances that are not solved to optimality, and "-" to denote that all five instances are solved to optimality within the time limit.

Coefficient strengthening for the extended formulation: First, we present computational results for solving the extended formulation (6) directly. We focus here only on the individual knapsack instances to illustrate how different coefficient strengthening methods perform. We consider the following options:

- Simple: Big-M coefficients are chosen in the naive way.
- IterLP: Strengthen big-M coefficients by iteratively solving LP (8). We report results using only one iteration. We also experimented with more iterations, but we found that additional iterations had very little incremental benefit in terms of the big-M coefficient reduction, but required significantly more time. (See Section 8 of the Online Supplement for a table with these results.)
- Scen: Strengthen big-M coefficients by the scenario-based upper bound approximation using (9).

Table 1 presents the average time (AvT) and number of nodes (AvN) to solve the four different instances, with three sample sizes each, using the big-M coefficients obtained using these three different methods. We also report the time spent strengthening the big-M coefficients (AvS) using the iterative LP approach, and using our scenario-based strengthening approach. Thus, the average total time spent to solve these instances is the sum of the numbers in the AvT and AvS

Table 1 Average big-M strengthening time, and average time and number of nodes for simple extended formulation, extended formulation strengthened by iterative LP, and strengthened by scenario-based strengthening. K represents thousand.

Instances		(6)-Simple		(6)-IterLP			(6)-Scen		
Instance	$\|S\|$	AvT	AvN	AvS	AvT	AvN	AvS	AvT	AvN
1-7-1	100	0.4	153	1.3	0.2	91	0.0	0.2	60
n=50	500	40.1	7766	79.4	15.3	3032	1.3	8.0	1636
	1000	(4)	>154K	573.5	135.9	23K	5.2	109.0	16K
1-7-5	100	0.6	510	1.9	0.3	169	0.0	0.2	159
n=50	500	167.0	61K	74.9	71.7	18K	1.2	9.7	2978
	1000	(1)	>294K	526.6	630.3	69K	4.9	45.8	8617
weish26-1	100	0.5	598	2.4	0.4	345	0.1	0.4	288
n=90	500	612.6	257K	127.2	101.7	41K	2.3	72.9	34K
	1000	(1)	>562K	898.6	2215.9	555K	9.2	2046.4	543K
weish26-5	100	0.6	579	2.5	0.4	284	0.1	0.3	240
n=90	500	307.5	135K	130.7	84.0	37K	2.4	55.6	27K
	1000	(1)	>522K	728.6	1955.6	671K	9.7	1645.6	482K

columns. This table indicates that the big-M strengthening procedures perform comparably in terms of time and nodes to solve the strengthened formulation, and much better than using naively chosen coefficients. However, the scenario-based approach requires significantly less time to calculate the big-M coefficients than the iterative LP approach. We therefore use the scenario-based strengthening procedure to obtain improved big-M coefficients in all the rest of our computational experiments. For multi-dimensional instances, we use the procedure described at the end of Section 3.1, based on solving single-row knapsack LPs, to obtain the improved big-M coefficients.

Probabilistic Cover Formulation: We now consider variants of a branch-and-cut algorithm for solving the formulation based on probabilistic covers (3). In all variants we use *lifted* probabilistic cover inequalities, as we found any variant that uses probabilistic cover inequalities without lifting to be uncompetitive.

We begin by studying the benefit of using local cuts and projection cuts for solving individual knapsack instances. Table 2 compares the performance of a variant that uses only lifted cover and pack inequalities (Basic), a variant that adds local cuts to Basic (Local), and a variant that adds the projection cuts to Basic (Proj).

Table 2 Average time, number of nodes, and optimality gap for three methods for solving the probabilistic cover-based formulation for individual knapsack instances. K represents thousand.

Instances		Basic			Local			Proj		
Instance	$\|S\|$	AvT	AvN	AvG	AvT	AvN	AvG	AvT	AvN	AvG
1-7-1	100	215.5	4034	-	90.6	1285	-	28.3	3503	-
n=50	1000	316.6	3094	-	279.4	1658	-	72.4	2970	-
	3000	875.8	4799	-	838.1	2149	-	368.2	5631	-
1-7-5	100	20.5	1155	-	16.2	713	-	11.6	1796	-
n=50	1000	87.8	1178	-	67.3	443	-	49.8	1825	-
	3000	275.6	1267	-	243.1	570	-	154.9	1943	-
weish26-1	100	(0)	>15K	1.0%	(4)	>5898	1.6%	129.3	5091	-
n=90	1000	(0)	>6907	1.8%	(2)	>4932	1.0%	301.4	4730	-
	3000	(0)	>3856	2.4%	(0)	>2149	1.6%	919.5	5383	-
weish26-5	100	(0)	>11K	2.5%	(0)	>6739	1.5%	493.5	14K	-
n=90	1000	(0)	>5341	3.1%	(0)	>3189	2.7%	1648.7	18K	-
	3000	(0)	>3384	3.5%	(0)	>1556	2.9%	2793.1	12K	-

We see from Table 2 that the number of processed nodes is reduced significantly by adding local cuts. This translates to improvements in number of instances solved and ending optimality gaps for the more difficult instances (weish26-1 and weish26-5) but the improvement in computational time for the easier instances (1-7-1 and 1-7-5) is modest. We also see from Table 2 that adding the projection cuts (11) and (12) to the probabilistic cover formulation (3) yields great improvements in computational time, as well as the number of instances solved and ending optimality gaps, especially for harder instances (weish26-1 and weish26-5).

Next, we compare the best version of the probabilistic cover-based formulation with two approaches for solving the extended formulation (6) on this set of individual knapsack instances. Based on the results from Table 2 we found that for the individual knapsack instances the best variant for solving the probabilistic cover-based formulation (Best prob cover), uses local cuts and projection cuts, (11) and (12). The first approach for solving extended formulation (6) is to directly solve it using strengthened big-M coefficients. The second approach was to solve formulation (6) with the addition of mixing inequalities as in Luedtke (2013). We follow the implementation described in Luedtke (2013) for generating mixing inequalities.

The results of these three methods are shown in Table 3. The column AvS reports the average big-M strengthening time, which is required for all the formulations. Section 9 of the Online Supplement provides details on the time spent generating the different types of cuts in the probabilistic cover-based formulation. We also tried a specialized branch-and-bound algorithm proposed in Beraldi et al. (2012), but found that although it worked well for an instance with $n = 20$ items, it was unable to solve any of these larger instances.

Table 3 **Average big-M strengthening time, and average time, number of nodes and optimality gap (when positive) for the best option of the probabilistic cover formulation, the extended formulation (6), and formulation (6) with mixing inequalities as in Luedtke (2013), for individual knapsack instances. K represents thousand.**

Instances			Best prob cover		(6)-Scen			Luedtke (2013)		
Instance	$\|S\|$	AvS	AvT	AvN	AvT	AvN	AvG	AvT	AvN	AvG
1-7-1	100	0.0	19.2	1544	0.2	60	-	0.2	49	-
n=50	1000	5.2	81.2	1423	109.0	16K	-	235.7	27K	-
	3000	48.3	350.6	1889	(1)	>216K	0.5%	(0)	>67K	0.6%
1-7-5	100	0.0	8.8	869	0.2	159	-	0.4	165	-
n=50	1000	4.9	47.3	602	45.8	8617	-	919.3	54K	-
	3000	45.0	152.0	696	(4)	>177K	0.1%	(0)	>95K	0.7%
weish26-1	100	0.1	20.8	404	0.4	288	-	0.7	329	-
n=90	1000	9.2	148.7	431	2046.4	543K	-	(3)	>143K	0.2%
	3000	84.4	651.9	616	(0)	>208K	0.4%	(0)	>79K	0.5%
weish26-5	100	0.1	95.7	1410	0.3	240	-	0.6	255	-
n=90	1000	9.7	696.0	2236	1645.6	482K	-	(1)	>159K	0.2%
	3000	87.1	1612.3	1883	(0)	>227K	0.4%	(0)	>85K	0.6%

We see from Table 3 that when solving the probabilistic cover-based formulation of the individual knapsack instances, the combination of local cuts and projection cuts yields improved performance over using either set of cuts individually (shown in Table 2), now allowing all instances to be solved within the time limit. Furthermore, we observe that for instances with $|S| \geq 1000$, the best probabilistic cover-based formulation performs significantly better than solving the extended formulation with or without the mixing inequalities. In particular, in either case, the number of processed nodes for the extended formulation grows quickly as the number of scenarios increases. On the other hand, the number of nodes required by the best probabilistic cover-based formulation does not vary considerably as the number of scenarios increases. We also find that when solving formulation (6) with the strengthened big-M coefficients, using the mixing inequalities as in Luedtke (2013) yields worse performance. Thus, it appears that for these instances, the additional bound improvement of the mixing inequalities beyond the big-M coefficient strengthening is not sufficient to reduce the number of branch-and-bound nodes. Furthermore, because these instances have no recourse variables and only a single constraint for each scenario, there is little benefit from using decomposition as in Luedtke (2013).

Next, we consider a set of instances that includes three chance-constrained multi-dimensional knapsack instances and two chance-constrained set packing instances[1]. We compare the results of three methods: the basic method for solving the probabilistic cover-based formulation (Basic prob cover), the best variant for solving the probabilistic cover-based formulation (Best prob cover), and the extended formulation using strengthened big-M parameters. For these multi-dimensional packing instances, we found that the best option for the probabilistic cover-based formulation was to used the projection cuts, but *not* the local cuts. Since Table 3 indicated that the mixing inequalities did not improve the performance of the extended formulation, we exclude those results in this table for brevity.

We observe from Table 4 that once again the addition of the projection cuts to the multi-dimensional knapsack and set packing instances yields significant improvement over the basic probabilistic cover formulation. These results are particularly strong for the set packing instances. Similar to the individual knapsack instances, the number of processed nodes for the extended formulation grows much faster than the best probabilistic cover-based formulation as the number of scenarios increases.

[1] These experiments were performed on a Linux workstation with eight 2.93GHz processors and 2.9Gb memory. The machine memory is different because these experiments were done in an earlier revision of the manuscript.

Table 4 Average big-M strengthening time, and average time, number of nodes and optimality gap for basic probabilistic cover formulation, the best option of the probabilistic cover formulation, and extended formulation (6) for multi-dimensional knapsack and set packing instances. K represents thousand.

Instances			Basic prob cover			Best prob cover			(6)-Scen		
Instance	$\lvert S\rvert$	AvS	AvT	AvN	AvG	AvT	AvN	AvG	AvT	AvN	AvG
pb-2	100	0.5	221.4	6997	-	42.2	4456	-	8.1	2726	-
n=34	1000	46.4	1254.3	7969	-	550.2	6604	-	(0)	>126K	1.3%
m=4	3000	421.4	(4)	>7584	0.4%	2290.7	8855	-	(0)	>26K	2.7%
weish26	100	2.3	(0)	>10K	2.0%	69.7	4512	-	7.9	1038	-
n=90	1000	234.8	(0)	>4108	2.6%	955.5	6522	-	(0)	>28K	0.6%
m=5	3000	2134.9	(0)	>1287	3.7%	(4)	>7100	0.3%	(0)	>12K	1.5%
1-7	100	1.2	(0)	>9427	1.9%	(4)	>19K	1.0%	5.3	877	-
n=50	1000	120.2	(0)	>3069	3.1%	(0)	>5980	1.7%	(0)	>41K	1.0%
m=5	3000	1077.9	(0)	>1081	3.4%	(0)	>1775	2.0%	(0)	>12K	2.2%
sppnw15	100	2.0	1062.5	47K	-	3.9	445	-	18.6	1022	-
n=200	1000	202.2	(0)	>7436	52.9%	27.8	641	-	(0)	>11K	4.4%
m=31	3000	1987.4	(0)	>631	65.4%	120.6	671	-	(0)	>152	6.1%
sppnw41	100	0.7	1144.3	74K	-	0.3	27	-	2.7	117	-
n=197	1000	72.8	(0)	>17K	51.2%	2.0	18	-	(0)	>27K	1.1%
m=17	3000	737.9	(0)	>3357	66.4%	18.7	138	-	(0)	>3340	4.4%

We also observe that multi-dimensional knapsack instances are still challenging, for both formulations, especially on instance 1-7. Finally, Table 4 indicates that strengthening the big-M coefficients takes significantly longer for these instances, which is expected as there are many more big-M coefficients to calculate.

5. Concluding Remarks

We studied solution approaches for solving chance-constrained binary packing problems. We proposed a formulation that is based on probabilistic cover inequalities, introduced the probabilistic lifting problem and other valid inequalities. In the case of finite scenarios, we proposed an effective coefficient strengthening procedure for a natural extended formulation with scenario variables. We introduced projection cuts that enable the strength of the extended formulation to be used in the cover-based formulation. This projection characterization does not depend on the packing structure of this problem, and hence may be useful for more general finite scenario CCSPs. Extensive computational tests are conducted on three types of chance-constrained packing constraints in the finite scenario case: individual knapsack, multi-dimensional knapsack, and set packing instances. We find that the probabilistic cover-based formulation is more competitive when there is a large number of scenarios. As shown in the computational results, multi-dimensional knapsack instances are still challenging and deserve further study.

Acknowledgments

Yongjia Song and James Luedtke are supported in part by NSF-CMMI grant 0952907. Simge Küçükyavuz is supported in part by NSF-CMMI grant 1055668.

References

Andreello, G., A. Caprara, M. Fischetti. 2007. Embedding cuts in a branch and cut framework: a computational study with $\{0,1/2\}$-cuts. *INFORMS J. Comput.* **19(2)** 229–238.

Applegate, D.L., R.E. Bixby, V. Chvátal, W.J. Cook. 2006. *The Traveling Salesman Problem*. Princeton University Press, Princeton, NJ.

Atamtürk, A. 2004. Sequence independent lifting for mixed-integer programming. *Oper. Res.* **52** 487–490.

Atamtürk, A., V. Narayanan. 2009. The submodular 0-1 knapsack polytope. *Discrete Optim.* **6** 333–344.

Balas, E., E. Zemel. 1978. Facets of the knapsack polytope from minimal covers. *SIAM J. Appli. Math.* **34** 119–148.

Beasley, J.E. 1990. OR-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** 1069–1072.

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

Beraldi, P., M.E. Bruni. 2010. An exact approach for solving integer problems under probabilistic constraints with random technology matrix. *Ann. Oper. Res.* **177** 127–137.

Beraldi, P., M.E. Bruni, A. Violi. 2012. Capital rationing problems under uncertainty and risk. *Comp. Opt. and Appl.* **51(3)** 1375–1396.

Beraldi, P., A. Ruszczyński. 2002. The probabilistic set-covering problem. *Oper. Res.* **50** 956–967.

Calafiore, G.C., M.C. Campi. 2005. Uncertain convex programs: randomized solutions and confidence levels. *Math. Program.* **102** 25–46.

Calafiore, G.C., M.C. Campi. 2006. The scenario approach to robust control design. *IEEE Trans. Automat. Control* **51** 742–753.

Campi, M.C., S. Garatti. 2011. A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *J. Optim. Theory Appl.* **148** 257–280.

Charnes, A., W.W. Cooper. 1963. Deterministic equivalents for optimizing and satisficing under chance constraints. *Oper. Res.* **11** 18–39.

Chvatal, V., W. Cook, D. Espinoza. 2009. Local cuts for mixed-integer programming. *Submitted* .

Dentcheva, D., G. Martinez. 2012. Regularization methods for optimization problems with probabilistic constraints. *Math. Program.* **138(1-2)** 223–251.

Dentcheva, D., A. Prékopa, A. Ruszczyński. 2000. Concavity and efficient points of discrete distributions in probabilistic programming. *Math. Program.* **89** 55–77.

Gu, Z., G.L. Nemhauser, M.W.P. Savelsbergh. 1998a. Lifted cover inequalities for 0-1 integer programs: Complexity. *INFORMS J. Comput.* **11** 117–123.

Gu, Z., G.L. Nemhauser, M.W.P. Savelsbergh. 1998b. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS J. Comput.* **10** 427–437.

Gu, Z., G.L. Nemhauser, M.W.P. Savelsbergh. 2000. Sequence independent lifting in mixed integer programming. *J. Comb. Opt.* **4** 109–129.

Kaparis, K., A.N. Letchford. 2008. Local and global lifted cover inequalities for the multidimensional knapsack problem. *European J. Oper. Res.* **186** 91–103.

Kaparis, K., A.N. Letchford. 2010. Separation algorithms for 0-1 knapsack polytopes. *Math. Program.* **124** 69–91.

Küçükyavuz, S. 2012. On mixing sets arising in chance-constrained programming. *Math. Program.* **132** 31–56.

Lejeune, M. 2012. Pattern-based modeling and solution of probabilistically constrained optimization problems. *Oper. Res.* **60(6)** 1356–1372.

Luedtke, J. 2010. An integer programming and decomposition approach to general chance-constrained mathematical programs. F. Eisenbrand, F.B. Shepherd, eds., *IPCO 2010*. Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 271–284.

Luedtke, J., S. Ahmed. 2008. A sample approximation approach for optimization with probabilistic constraints. *SIAM J. Optim.* **19** 674–699.

Luedtke, J., S. Ahmed, G.L. Nemhauser. 2010. An integer programming approach for linear programs with probabilistic constraints. *Math. Program.* **12** 247–272.

Luedtke, James. 2013. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Math. Program.* Http://dx.doi.org/10.1007/s10107-013-0684-6.

Nemhauser, George L., Laurence A. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley.

Nemirovski, A., A. Shapiro. 2005. Scenario approximation of chance constraints. G. Calafiore, F. Dabbene, eds., *Probabilistic and Randomized Methods for Design Under Uncertainty*. Springer, London, 3–48.

Nemirovski, A., A. Shapiro. 2006. Convex approximations of chance constrained programs. *SIAM J. Optim.* **17** 969–996.

Padberg, M.W. 1973. On the facial structure of set packing polyhedra. *Math. Program.* **5** 198–216.

Padberg, M.W. 1975. A note on zero-one programming. *Oper. Res.* **3** 833–837.

Pagnoncelli, B., S. Ahmed, A. Shapiro. 2009. The sample average approximation method for chance constrained programming:theory and applications. *J. Optim. Theory Appl.* **142** 399–416.

Qiu, F., S. Ahmed, S. Dey, L. Wolsey. 2013. Covering linear programs with violations. *INFORMS J. Comput.* Accepted for publication.

Ruszczyński, A. 2002. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Math. Program.* **93** 195–215.

Saxena, A., V. Goyal, M. Lejeune. 2009. MIP reformulations of the probabilistic set covering problem. *Math. Program.* **121** 1–31.

Song, Y. 2013. Structure-exploiting algorithms for chance-constrained and integer stochastic programs. Ph.D. thesis, University of Wisconsin-Madison.

Tanner, M.W., L. Ntaimo. 2010. IIS branch-and-cut for joint chance-constrained programs and application to optimal vaccine allocation. *European J. Oper. Res.* **207** 290–296.

Zemel, E. 1989. Easily computable facets of the knapsack polytope. *Math. Meth. Oper. Res.* **14** 760–764.