European Journal of Operational Research 207 (2010) 290–296

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Stochastics and Statistics

IIS branch-and-cut for joint chance-constrained stochastic programs and application to optimal vaccine allocation

Matthew W. Tanner, Lewis Ntaimo *

Department of Industrial and Systems Engineering, Texas A&M University, 3131 TAMU, College Station, TX 77843, USA

ARTICLE INFO

Article history:
Received 10 July 2008
Accepted 14 April 2010
Available online 20 April 2010

Keywords:
Stochastic programming
Chance constraints
Irreducibly infeasible subsystem (IIS)
Branch-and-bound
Branch-and-cut

ABSTRACT

We present a new method for solving stochastic programs with joint chance constraints with random technology matrices and discretely distributed random data. The problem can be reformulated as a large-scale mixed 0–1 integer program. We derive a new class of optimality cuts called IIS cuts and apply them to our problem. The cuts are based on irreducibly infeasible subsystems (IIS) of an LP defined by requiring that all scenarios be satisfied. We propose a method for improving the upper bound of the problem when no cut can be found. We derive and implement a branch-and-cut algorithm based on IIS cuts, and refer to this algorithm as the *IIS branch-and-cut* algorithm. We report on computational results with several test instances from optimal vaccine allocation. The computational results are promising as the IIS branch-and-cut algorithm gives better results than a state-of-the-art commercial solver on one class of problems.

1. Introduction

In stochastic programming, instead of assuming that all parameter values are deterministically known, a subset of the parameters of a mathematical program are given probability distributions. This paper concerns stochastic programs with joint chance constraints, which can be formulated as follows:

$$\begin{aligned} \text{SP}: \ \text{Min} \quad & c^\top x && (1a)\\ \text{s.t.} \quad & Ax \leqslant b && (1b)\\ & \mathbb{P}\{T(\tilde{\omega})x \leqslant r(\tilde{\omega})\} \geqslant \alpha && (1c)\\ & x \geqslant 0. && (1d) \end{aligned}$$

In formulation (1), $x \in \mathbb{R}^{n_1}$ is the decision variable vector, $c \in \mathbb{R}^{n_1}$ is the cost parameter vector, $A \in \mathbb{R}^{n \times m_1}$ is the deterministic constraint matrix, and $b \in \mathbb{R}^{m_1}$ is the deterministic right-hand side vector. In this formulation, uncertainty appears as the multi-dimensional random variables $\tilde{\omega}$, that gives rise to the random technology matrix $T(\tilde{\omega}) \in \mathbb{R}^{n \times m_2}$ and the random right-hand side vector $r(\tilde{\omega}) \in \mathbb{R}^{m_2}$. Individual outcomes (scenarios) of the random variable are represented as realizations $\omega \in \Omega$ of the sample space. The aim of such a formulation is to find a minimum cost strategy while allowing a subset of the constraints to be violated an acceptable amount of time.

Chance constraints are best suited to optimization problems in which satisfying a certain set of constraints is desirable but may not be done almost surely. For example, it may be too expensive to satisfy the constraints almost surely. Thus chance constraints are used to find optimal solutions such that an acceptable level of reliability is achieved. Applications include telecommunications with companies needing to guarantee a given quality-of-service to their customers [13], air-quality management with the requirement that pollution not exceed prescribed limits too often [2], and also inventory control problems where the upper and lower bounds on stock have to be violated with low probability [15].

In general, stochastic programs with joint chance constraints are hard to solve. The main reason for this is that given a general distribution of the parameter values, the feasible space of the problem is nonconvex [7]. Also, in the case where the random parameters have continuous distributions, evaluating the feasibility of a single candidate solution can require an extremely hard integration [27]. Most research into solution methods for SP (1) has focused either on identifying probability distributions for the parameters with the property that the feasible space is convex, or on methods for solving the problem when the parameters have discrete distributions.

In the case when all the randomness appears in the right-hand side of the formulation, results on distributions that allow the chance constraints to be formulated as convex programs are given in [7,25]. In the case of discrete probability distributions, most solution methods use integer programming (IP) or other discrete programming techniques. When only the right-hand side is random, solution methods based on a nonconvex reformulation using p-efficient points are given in [13,12,29]. IP formulations and solution approaches for the discretely distributed case are

* Corresponding author.
E-mail addresses: mtanner@tamu.edu (M.W. Tanner), ntaimo@tamu.edu (L. Ntaimo).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.04.019

addressed in [9,19]. Another branch of research has been methods based on sampling approximations [18,21,22]. These methods use the sample average approximation method [30] to obtain an approximation by replacing the actual probability distribution by an empirical distribution corresponding to a random sample. A different sampling approach, proposed by [8], is to obtain the approximation of a chance-constrained optimization problem by sampling a finite number of its constraints.

When the technology matrix $T(\omega)$ is stochastic, problem SP is significantly harder to solve than for the case in which all the uncertainty appears in the right-hand side. With the matrix $T(\omega)$ allowed to be continuously distributed, Prekopa [26] presents a few parameter distributions which make the problem convex. In the case where the parameters are joint-normally distributed, An and Eheart [2] present a method for finding upper and lower bounds on the optimal objective function of the problem using the extreme cases of correlation between the random variables. However, there are applications for which these types of distribution assumptions are too stringent. Consequently, there has been a lot of interest in formulations with discretely distributed random parameters, which are often created by sampling.

Given problem SP with decision variables that are pure integer, Tayur et al. [32] solve the problem using an algebraic geometry approach and Aringhieri [3] applies a tabu search for finding feasible solutions. For problems with random $T(\omega)$ and possibly continuous decision variables, a deterministic equivalent problem can be formulated as a 'big-M' mixed 0–1 integer program [20]. An advantage to such a formulation is that it may be solved directly by a commercial mixed-integer programming (MIP) solver. However, it often results in extremely weak linear programming (LP) relaxations that hinder finding the optimal solution. To strengthen the LP relaxation, Ruszczyński [28] derives cutting planes based on precedence constraint knapsack polyhedra and gives a specialized branch-and-cut algorithm for the problem.

This paper focuses on theoretical results that can be used for general solution techniques for SP. The main significance of these results is that they are valid for joint chance-constrained problems with discretely distributed random technology matrices and right-hand side vectors. We introduce a new class of optimality cuts, called *irreducibly infeasible subsystem* (IIS) cuts, for strengthening the LP relaxations of the deterministic equivalent MIP reformulation of SP. We also present a method for improving the upper bound found by the algorithm for the case when no IIS cut can be identified. We then derive a branch-and-cut method based on the IIS cuts, termed 'IIS branch-and-cut' algorithm', and discuss its implementation. Finally, we apply the IIS branch-and-cut algorithm to randomly generated large-scale instances arising in optimal vaccine allocation for epidemic prevention.

The rest of the paper is organized as follows: In the next section we give some background on the problem and present the MIP reformulation of the problem that can be solved directly. In Section 3 we derive IIS cuts and an upper bound improvement strategy for the IIS branch-and-cut algorithm. We present and discuss an implementation of the IIS branch-and-cut algorithm in Section 4 and give computational results in Section 5. Finally, we finish with a summary and point further research topics in Section 6.

2. Preliminaries

We make the following assumptions on SP throughout the rest of the paper:

(A1) The random variable $\tilde{\omega}$ is discretely distributed with $|\Omega| < \infty$.

(A2) Bounds $0 \leqslant x \leqslant U$ on x are included in the constraint set $Ax \leqslant b$.

(A3) The polyhedron $P_1 = \{x \in \mathbb{R}^{n_1} | Ax \leqslant b\} \neq \emptyset$ and is compact.

Assumption (A1) makes the problem tractable while assumption (A2) is needed solely to make the implementation of the cut generation LP more clear and does not restrict the application of the results of this paper. Assumption (A3) is mainly needed to keep the problem from being trivially infeasible.

Under assumption (A1), a deterministic equivalent to problem SP can be formulated as a big-M mixed 0–1 IP as follows [20]:

$$\text{Min} \quad c^\top x \tag{2a}$$

$$\text{s.t.} \quad Ax \leqslant b \tag{2b}$$

$$T(\omega)x - Mez_\omega \leqslant r(\omega) \quad \forall \omega \in \Omega \tag{2c}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \leqslant 1 - \alpha \tag{2d}$$

$$z_\omega \in \{0,1\} \quad \forall \omega \in \Omega, \tag{2e}$$

where $M \in \mathbb{R}$ is an appropriate large number, z_ω is a binary decision variable, p_ω is the probability of a scenario $\omega \in \Omega$, and e is an appropriately sized vector of ones. We note that problem (2a) is a mixed 0–1 IP with a binary variable for each scenario and a knapsack constraint (2d) to guarantee that the chance constraint (1c) is satisfied. Each binary variable takes the value of 0 if the corresponding constraints are satisfied, and 1 otherwise. Assumption (A3) guarantees the existence of an M large enough for formulation (2). Also, instead of having just one M in (2c), one can choose an appropriate M_ω for each ω to tighten the formulation.

Based on our computational experience we have seen that often relatively few scenarios are important in the final solution. The rest of the scenarios are either redundant or cannot be satisfied in any nearly optimal solution. Our approach aims at identifying subsets of scenarios that are particularly important for finding optimal solutions to the problem. Using these subsets of scenarios, we are then able to derive cutting planes that can be used to strengthen the LP relaxation of formulation (2). The cutting planes are based on irreducibly (minimally) infeasible subsystems (IISs). An *IIS* is defined as follows:

Definition 2.1. An *IIS* is a set of constraints S of a mathematical programming problem such that S is infeasible but every proper subsystem of S is feasible.

The traditional use of IISs is in the analysis of infeasible LPs to determine the optimal strategy for changing problem parameters to make the system feasible. Several methods for identifying these sets using LP methods have been developed [10,14,17]. In this paper, we use the following basic theorem from Gleeson and Ryan [14] to identify IISs:

Theorem 2.2. *Let $D \in \mathbb{R}^{m \times n}$ be a rational matrix and let $d \in \mathbb{R}^m$ be a rational vector. Then the indices of the minimally infeasible subsystems of the system $Dv \leqslant d$ are exactly the supports of the vertices of the polyhedron $P = \{w \in \mathbb{R}^m | D^\top w = 0, d^\top w \leqslant -1, w \geqslant 0\}$.*

Proof. See Gleeson and Ryan [14] □

Note that the *support* of a vector is the set of indices of its nonzero components. Theorem 2.2 gives a polyhedron with the property that every extreme point of the polyhedron corresponds with an IIS of the system $Dv \leqslant d$. This means we can use LP to identify IISs.

In recent years, IISs have been used to generate valid inequalities for the maximum feasible subsystem problem [1,24]. In IP, IISs have been used to derive combinatorial Benders (CB) cuts for a class of MIP problems [11]. CB cuts are used to solve MIPs with

integer and continuous variables that are linked solely by 'big-M' constraints. They decompose the problem as in Benders decomposition [6] with the master problem having pure integer decision variables and the subproblem having pure continuous decision variables. For the MIP formulation of chance-constrained programs, Benders decomposition is not a good solution method because the master program finds integer feasible solutions without regard to the objective function that is defined by the continuous decision variables. Since every feasible point of the master problem has objective value zero, cutting off an individual point of the master program is often not useful. The IIS cuts are similar to the combinatorial Benders cuts of Codato and Fischetti [11], but they are specifically derived for a problem with the objective function depending on the continuous decision variables rather than on the integer decision variables.

3. IIS cuts

Let us begin by defining some further notation we will use throughout the rest of the paper. At an arbitrary node of a branch-and-bound (BAB) search tree, let $\mathcal{L} \subseteq \Omega$ and $\mathcal{U} \subseteq \Omega$ denote the sets of all scenarios such that z_ω is set to 0 and z_ω is set to 1, respectively. Also, let $u - \epsilon$ denote the current incumbent objective value minus a sufficiently small value. We refer to a scenario ω being *forced into* the problem at a node of the BAB tree if the binary decision variable z_ω is set to 0. This is the case when constraints (2c) for ω are satisfied. Similarly, we refer to a scenario ω being *forced out* of the problem if the binary decision variable z_ω is set to 1. This is the case when constraints (2c) for are ω not satisfied. Since the set $\mathcal{U}$ can be chosen such that $\mathbb{P}\{\Omega \setminus \mathcal{U}\} \geqslant \alpha$, the LP formulation using this polyhedron as a constraint set defines an upper bound on the optimal value of the original problem and can be given as follows:

$$\begin{aligned}
&\text{Min} \quad c^\top x && (3a)\\
&\text{s.t.} \quad Ax \leqslant b && (3b)\\
&\qquad T(\omega)x \leqslant r(\omega) \quad \forall \omega \in \Omega \setminus \mathcal{U} && (3c)\\
&\qquad c^\top x \leqslant u - \epsilon. && (3d)
\end{aligned}$$

We will generate IIS cuts from IISs of the polyhedron defined by having every scenario in $\Omega \setminus \mathcal{U}$ to be satisfied, restricted by an optimality cut generated from an upper bound problem (3). The advantage to formulation (3) is that it can be set up at nodes in the BAB tree that yield different sets $\mathcal{U}$. This means that we can generate optimality cuts early in the BAB tree when they are most effective rather than deep in the tree. To determine IISs that can be used to derive IIS cuts, we propose the following fundamental result:

Theorem 3.1. *IISs of* (3) *are in one-to-one correspondence with the supports of the vertices of the polyhedron*

$$\begin{aligned}
\Pi = \Big\{ & y_1 \in \mathbb{R}^{m_1}, y_2(\omega) \in \mathbb{R}^{m_2} \quad \forall \omega \in \Omega \setminus \mathcal{U},\ y_3 \in \mathbb{R} | y_1^\top A \\
& + \sum_{\omega \in \Omega \setminus \mathcal{U}} y_2(\omega)^\top T(\omega) + y_3^\top c = 0 y_1^\top b \\
& + \sum_{\omega \in \Omega \setminus \mathcal{U}} y_2(\omega)^\top r(\omega) + y_3^\top u \leqslant -1 y_1, y_2, y_3 \geqslant 0. \Big\} \qquad (4)
\end{aligned}$$

Proof. Similar to the proof of Theorem 2.2 in Gleeson and Ryan [14]. □

We omit the proof of Theorem 3.1 since this theorem is a direct application of Theorem 2.2 to (3) to give us a polyhedron with the property that every extreme point of the polyhedron corresponds with an IIS of (3). The theorem is proved by elementary polyhedral results involving the scaled set of extreme points of the dual constraints of problem (3), that is Π, and a version of the well-known Farkas Theorem of the alternative. As with Theorem 2.2, we can simply use LP to identify IISs. We will work with the LP relaxation of problem (2) at each node of the BAB tree and extreme points of (3) to generate IISs. Essentially, IISs are used to identify sets of scenarios $\mathcal{D}$ such that not all of the scenarios in $\mathcal{D}$ can be satisfied in the optimal solution to problem (2). The following result shows how such sets $\mathcal{D}$ can be determined as well as the separation problem that we have to solve.

Theorem 3.2. *Given an IIS S of formulation* (3), *let the subset of scenarios* $\mathcal{D} = \{\omega \in \Omega | T(\omega)x \leqslant r(\omega) \cap S \neq \emptyset\}$. *Then the set* $\mathcal{D} \neq \emptyset$ *defines the IIS cut*

$$\sum_{\omega \in \mathcal{D}} z_\omega \geqslant 1. \qquad (5)$$

The IIS cut is valid in the sense that it does not cut off all optimal solutions to problem (2).

Given a non-integer solution $(\bar{x}, \{\bar{z}_\omega\}_{\omega \in \Omega})$ to the LP relaxation of problem (2), the separation problem for the IIS cut is to find an IIS S of problem (2) and generating a subset $\mathcal{D} \subseteq \Omega$ as in (5) such that $\sum_{\omega \in \mathcal{D}} \bar{z}_\omega < 1$. The separation problem is NP-hard [1] and therefore, heuristics have to be used to find valid inequalities quickly. Pfetsch [24] suggests finding IISs by solving an LP constrained by Π with the objective function coefficients given by a non-integer solution to the LP relaxation of (2) at a given node in the BAB tree.

To find an IIS cut, we propose solving the following LP:

$$\begin{aligned}
&\text{Min} \quad \sum_{\omega \in \Omega \setminus \mathcal{U}} \bar{z}_\omega y_2(\omega) \\
&\text{s.t.} \quad \Pi. && (6)
\end{aligned}$$

Observe that only the values $\bar{z}_\omega$ of the binary variables z_ω are used in (6). The reason for this is that the cardinality $|\mathcal{D}|$ only depends on the constraints defined by the z_ω variables in (3) and therefore, the x variables should not affect the objective function of LP (6). Also, the IIS cuts tend to be stronger when $|\mathcal{D}|$ is small. However, using the objective function given in (6), an IIS cut that separates the current non-integer solution may not be found. Nevertheless, the generated IIS cuts are valid for the entire BAB tree and may cut off some non-integer solution at some later node in the BAB tree.

Since every extreme point of Π defines an IIS, it is possible to generate rounds of cuts using LP. A tempting method to try is to use the extreme points visited by the simplex method as it solves the cut generating LP. Unfortunately, this has been shown to be computationally ineffective [24]. A more effective method is to solve the cut generating LP and then change the objective function coefficients to target specific scenarios. Then the LP can be warm-started with the current solution information. When generating IIS cuts, LP (6) can be infeasible. This happens when $\Pi = \emptyset$, implying that every scenario in $\Omega \setminus \mathcal{U}$ can be satisfied with the current upper bound u. In this case we would like to either improve u, or determine that no such improvement is possible and fathom the current node. The following result shows how to improve u by dropping more scenarios from $\Omega \setminus \mathcal{U}$:

Proposition 3.3. *Let* Π *be as defined in* (4). *If* $\Pi = \emptyset$ *and the original problem SP is bounded, then formulation* (3) *has an optimal solution, denoted* $\bar{x}$. *By setting* $\bar{z}_\omega = 0$ *if* $\omega \in \Omega \setminus \mathcal{U}$ *and setting* $\bar{z}_\omega = 1$ *otherwise, then* $(\bar{x}, \{\bar{z}_\omega\}_{\omega \in \Omega})$ *defines an integer feasible solution to* (2) *with* $c^\top x \leqslant u$. *Furthermore, a possibly improved integer feasible solution can be found by dropping a set of scenarios* $\mathcal{F} \subseteq \Omega$ *from problem (3) for any set* $\mathcal{F}$ *such that* $\mathbb{P}(\Omega \setminus (\mathcal{U} \cup \mathcal{F})) \geqslant \alpha$.

Proof. Since $\Pi = \emptyset$, it implies there are no IISs for formulation (3) and hence it is feasible. Since it cannot be unbounded as it is a

restriction of problem SP, it has an optimal solution $\bar{x}$. The solution $(\bar{x}, \{\bar{z}_\omega\}_{\omega\in\Omega})$ satisfies $\{Ax \leqslant b,\ T(\omega)x - Mez_\omega \leqslant r(\omega)\ \forall\omega \in\ \Omega, x \geqslant 0\}$. Since $\mathbb{P}(\mathcal{U}) \leqslant 1-\alpha$ or else the node would have been fathomed by *infeasibility*, constraint (2d) must also be satisfied, and thus $(\bar{x}, \{\bar{z}_\omega\}_{\omega\in\Omega})$ is an integer feasible point. Also, $c^\top\bar{x} \leqslant u$, otherwise constraint (3c) would have been violated. Finally, dropping the set of scenarios $\mathcal{F}$ from formulation (3) provides a relaxation whose optimal solution will be no worse. □

To improve u it is necessary to carefully choose the set $\mathcal{F}$. If all constraints (3c) for a given ω are redundant, then dropping the constraints will not improve the optimal objective value. The only scenarios whose removal will affect the objective value are those in which at least one constraint is binding. So one way to identify the set $\mathcal{F}$ is to first rank the scenarios $\omega \in \Omega\setminus\mathcal{U}$ in nondecreasing order based on the number of nonzero slack values, and then starting with the lowest ranked scenario, add the scenarios to $\mathcal{F}$ as long as $\mathbb{P}(\Omega\setminus(\mathcal{U}\cup\mathcal{F})) \geqslant \alpha$. A relatively more time consuming method is to remove the lowest ranked scenario from the problem, re-solve the problem, and repeat until no more scenarios can be removed. One can also use a heuristic such as local or tabu search to better identify sets of scenarios to remove. Note that an IIS cut can always be found when an improved upper bound u is used in (6). If all the scenarios in $\Omega\setminus\mathcal{U}$ have all their constraints (3c) redundant, then no upper bound improvement is possible and the node can be fathomed.

4. A branch-and-cut algorithm

We are now in a position to illustrate the use of the IIS ideas developed in the previous section within a branch-and-cut framework. We show how the IIS cuts and the upper bound improvement fit into an exact method for solving problem (2). Let k denote the node index and K denote the total number of nodes in the BAB search tree. The set of all z_ω that are set to 0 or 1 at node k are given by $\mathcal{L}^k$ and $\mathcal{U}^k$, respectively. The set of open nodes in the search tree is given by $\mathcal{N}$, while an individual node is given by $n^k := (\mathcal{L}^k, \mathcal{U}^k)$. Finally, we continue to denote by u the current best upper bound on the optimal solution. Then a basic IIS branch-and-cut algorithm can be stated as follows:

4.1. IIS branch-and-cut algorithm

Step 0. **Initialize:** Set $\mathcal{L}^1 = \emptyset,\ \mathcal{U}^1 = \emptyset,\ n^1 = (\mathcal{L}^1, \mathcal{U}^1),\ \mathcal{N} = \{n^1\}$, $K = 1$, and $u = \infty$.

Step 1. **Node Choice:** Pick some node $n^k \in \mathcal{N}$ according to some search rule.

Step 2. **Solve LP:** Solve the LP relaxation of problem (2) including the constraints of type $z_\omega = 0$ and $z_\omega = 1$ for the ω's that have been set in BAB. This will either find an optimal solution $(\bar{x}, \{\bar{z}_\omega\}_{\omega\in\Omega})$ or that the problem is *infeasible*.

Step 3. **Fathoming rules:** If the node relaxation is *infeasible* or $c^\top\bar{x}^k \geqslant u$ fathom the node and return to step 1. Otherwise, if constraint (2d) is satisfied, set $u = c^\top\bar{x}^k$, fathom the node and return to Step 1. Otherwise, continue to Step 4.

Step 4. **Cut Generation:** Solve problem (6) to get extreme points of (4) that give IIS sets S. Generate and add the IIS cuts (5) implied by these sets and go to Step 2. If (6) is infeasible, improve the upper bound u (if possible) and go to Step 5.

Step 5. **Branching:** Pick a non-integer $\bar{z}^k_\omega$. Create two new nodes $n^{K+1} = (\mathcal{L}^k \cup z_\omega, \mathcal{U}^k)$ and $n^{K+2} = (\mathcal{L}^k, \mathcal{U}^k \cup z_\omega)$. Add these nodes to $\mathcal{N}$, set $K = K + 2$, and return to Step 1.

The finite convergence of the IIS branch-and-cut algorithm is guaranteed by the branching on the binary variables of formulation (2). By Theorem 3.2, not all alternative optimal solutions are eliminated by the IIS cuts. Hence, the algorithm is assured of converging to an optimal solution. Note that the IIS cuts and upper bound improvement strategy cannot guarantee an optimal solution without branching, however as computational results show, they are able to significantly reduce the size of the search tree necessary to find an optimal solution and prove optimality.

5. Computational results

We now present some computational results showing the effectiveness of the IIS branch-and-cut algorithm in solving formulation (2). We ran our tests on instances from an application developed in Tanner et al. [32] involving the optimal allocation of vaccines under parameter uncertainty. We created five random replications of each problem size to guard against pathological cases and better evaluate the robustness of the computational results. The instances were created by sampling uniformly from the probability distributions given in the Appendix. The constant M used in inequalities (2c) was chosen for each instance through experimentation to be the smallest value to ensure feasibility without causing computational instability.

We implemented the IIS branch-and-cut algorithm in C++ using the CPLEX 9.1 callable library on a Dell Optiplex GX620 with a 3.00 GHz dual processor and 4.0 GB of RAM. The solution of any LPs in the algorithm was done using the CPLEX 9.1 LP solver. For these computational results, all solution times are given in seconds and a time limit of 7200 seconds (2 hours) of CPU time was imposed. Throughout our computational results, we compare three sets of tests. The first is computations using the CPLEX MIP solver under default settings directly on the MIP formulation (2) to provide a benchmark. The second is computations with the IIS branch-and-cut algorithm without adding the IIS cuts. This implementation is pure BAB using a depth-first search strategy and was done to provide a benchmark to assess the effectiveness of the IIS cuts. Finally, the third computations were performed with the IIS branch-and-cut algorithm with the IIS cuts added at each node of the BAB tree.

5.1. Optimal vaccine allocation

Optimally allocating scarce vaccines is important in order to prevent the occurrence of disease epidemics [4,16,23]. A common method for deciding on an optimal allocation strategy is to solve a mathematical program minimizing the cost of the vaccinations subject to the requirement that the post-vaccination reproductive number $R^* \leqslant 1$ [5]. The post-vaccination reproductive number is defined as the average number of new cases of a disease caused by a single infected individual after the population has been vaccinated. Constraining this value to be below one causes a disease to tend to die out.

Chance constraints are a natural framework for including parameter uncertainty in finding optimal vaccine allocations. If an epidemic occurs, it is a disaster and so the penalties are binary. On the other hand, it is impossible to stop all epidemics from occurring. Sometimes a strain will arise that will spread through the population no matter how many vaccines are given out. Chance constraints gives a method for determining the optimal strategy that will prevent the vast majority of epidemics from starting. We created a test set of random instances extending the deterministic LP devised by Becker and Starczak [5] to find optimal vaccination policies for a population divided into a community of households. The background on this application is available in Tanner et al. [31]. We have provided details of the formulation of the chance-constrained problem and the probability distributions as-

sumed for the random parameters of the original disease model in Appendix A for the interested reader.

Table 1 gives the problem sizes for the set of optimal vaccination test instances. The first column gives the name of the test instance with the number of the test instance corresponding to the number of scenarios. The second column gives the number of rows of the problem. The third column gives the number of continuous variables in the problem. Finally, the last column gives the number of binary variables of the problem. For these test problems the number of rows in matrices A and $T(\omega)$ are $m_1 = 31$ and $m_2 = 1$, respectively. These instances are clearly difficult to solve because the MIP formulations are very large and are extremely dense.

Table 2 gives the results of the computational tests on the vaccination allocation test instances. The first column of Table 2 gives the name of the test instance. The next four columns give the CPLEX results: the second column gives the best solution found by CPLEX; the third column gives the optimality gap returned by CPLEX; the fourth column gives the number of nodes searched in the BAB tree; and the fifth column gives the time to prove optimality. The next two columns give the results of our implementation of BAB without any added cuts or upper bound improvement. The first of these columns gives the best solution value found, while the second of these columns gives the number of nodes searched in the BAB tree. For either CPLEX or the BAB implementation, if the table shows that the number of nodes searched is greater than some number, it means that the algorithm was unable to prove optimality within the 2-hours time limit. The final four columns give the results of the IIS branch-and-cut algorithm on these test instances. The first of these columns gives the best objective value found, the second column gives the number of nodes searched, the third column gives the number of cuts added to the formulation, and the fourth column gives the solution time in seconds.

The IIS branch-and-cut algorithm is able to reduce both the number of nodes of the BAB tree that have to be searched in order to find the optimal solution and the time that is required to prove optimality. The advantages of the IIS methods hold over both CPLEX and our implementation of BAB. Relatively fewer cuts allow for the optimal solution to be found for several instances. For example, notice that for the vac1000 test instances, after 2 hours the BAB algorithm can only prove optimality for four of the test instances, and these four require an average of about 63,950 nodes in the BAB tree. CPLEX requires an average of over 35,000 nodes and about 770 seconds to prove optimality. With IIS branch-and-cut algorithm, we are able to find an optimal solution by searching an average of about 360 nodes in an average time of less than 80 seconds. This is a reduction of 99% in the nodes of the BAB tree and a reduction of 90% in computation time. Even accounting for the number of cuts which each require solving an LP at a node, the total number of LPs solved by the IIS algorithm is less than 1000.

It is also interesting to note that the BAB algorithm without IIS cuts actually identified the optimal solution for each of the five

Table 1
Problem sizes for vaccination test instances.

Instance	Rows	Cont. vars.	Bin. vars.
vac100	131	302	100
vac250	281	302	250
vac500	531	302	500
vac750	781	302	750
vac1000	1031	302	1000
vac2000	2031	302	2000

Table 2
Computational results on vaccination problems.

Instances	CPLEX results				B&B No IIS cuts		IIS cuts added			
	Objval	Gap (%)	Nodes	Time	Objval	Nodes	Objval	Nodes	Cuts	Time
vac100a	65.28	0	18	0.61	65.28	23	65.28	7	7	0.28
vac100b	62.39	0	30	0.77	62.39	69	62.39	9	9	0.37
vac100c	65.15	0	11	0.67	65.15	19	65.15	5	4	0.36
vac100d	69.81	0	13	0.56	69.81	23	69.81	3	3	0.25
vac100e	65.99	0	8	0.56	65.99	13	65.99	1	1	0.27
vac250a	63.69	0	59	4.41	63.69	371	63.69	29	14	1.25
vac250b	62.34	0	549	7.16	62.34	855	62.34	41	58	2.17
vac250c	65.52	0	486	5.64	65.52	433	65.52	29	46	1.87
vac250d	62.92	0	211	4.59	62.92	667	62.92	31	39	1.75
vac250e	66.59	0	208	4.11	66.59	175	66.59	7	6	0.91
vac500a	64.53	0	4074	30.82	64.53	3075	64.53	111	210	12.66
vac500b	65.49	0	3249	30.82	65.49	3727	65.49	111	229	13.20
vac500c	66.41	0	520	30.82	66.41	893	66.41	49	107	6.42
vac500d	66.63	0	805	30.82	66.63	1863	66.63	57	121	7.72
vac500e	65.16	0	784	30.82	65.16	931	65.16	47	88	6.92
vac750a	65.17	0	2833	75.39	65.17	6207	65.17	211	335	31.69
vac750b	66.10	0	3690	87.99	66.10	6353	66.10	175	275	27.04
vac750c	64.85	0	1912	54.36	64.85	7967	64.85	115	223	21.68
vac750d	65.27	0	7135	143.61	65.27	10,815	65.27	211	330	33.34
vac750e	64.77	0	8432	163.74	64.77	20,387	64.77	155	294	27.27
vac1000a	65.11	0	22,505	469.16	65.11	85,687	65.11	207	387	47.69
vac1000b	65.02	0	74,615	1527.72	65.02	58,815	65.02	493	766	104.16
vac1000c	64.57	0	32,481	642.87	64.57	83,623	64.57	387	645	72.08
vac1000d	65.50	0	25,604	678.98	65.50	27,691	65.50	299	458	63.07
vac1000e	64.31	0	23,140	570.12	64.31	>110,000	64.31	431	768	97.63
vac2000a	65.05	10.89	>71,181	>7200	65.16	>54,004	64.98	1643	2660	1001.31
vac2000b	65.50	2.10	>85,385	>7200	65.50	>57,729	65.48	1901	2829	1045.24
vac2000c	66.07	10.27	>71,001	>7200	65.55	>55,733	65.50	1895	3122	1109.76
vac2000d	64.95	3.18	>92,311	>7200	65.58	>51,927	64.95	1491	2432	837.00
vac2000e	65.06	5.24	>109,881	>7200	66.05	>45,497	65.06	1737	2660	1119.82

vac1000 test problems. However, without the IIS cuts added, the linear relaxations of these problems were too weak for the algorithm to prove the optimality of the solutions. This gives empirical evidence that the IIS cuts offer a significant increase in the strength of the relaxations of the problem. The advantage of the IIS branch-and-cut algorithm is even more significant for the larger test instances. For the vac2000 problems, CPLEX is unable to identify optimal solutions for any of the test instances after 2 hours. The BAB algorithm by itself does even worse and now searches an average of over 50,000 nodes but still cannot identify an optimal solution in 2 hours. The IIS branch-and-cut algorithm finds the optimal solution in an average time of about half an hour for each of the five replications. A final observation on the results is that the IIS branch-and-cut algorithm results seem to have significantly less variation in computation time than CPLEX. For the vac1000 problems, the CPLEX computation times vary by over 1000 seconds. With the IIS cuts, the variation in times for the vac1000 test instances is only 40 seconds, the variation in time for the vac2000 problems is 300 seconds.

6. Conclusion

In this paper we have derived a class of optimality cuts for jointly chance-constrained stochastic programs with random technology matrices. We have defined an upper bound generating formulation of the problem that allows cuts to be generated at every non-integer point of the linear relaxation of the problem. The cuts are derived in order to identify sets of scenarios that cannot all be satisfied in the optimal solution to the problem. We also have given a method for improving the upper bound during BAB when there is a node for which no IIS cut can be found. The paper also gives computational results on several test instances from optimal vaccine allocation. The computational results are promising as they show that the method can be used on its own to solve quite large instances. Also, the results show that the IIS branch-and-cut algorithm requires many fewer nodes in the search tree and much less computational effort in order to prove optimality for the vaccination allocation test instances than does CPLEX or our implementation of BAB with no cuts added. Possible extensions to this work include implementing the cuts in a BAB framework including other cuts that have been derived for general MIPs. It would also be important to study branching rules and other implementation issues that may make for a more effective algorithm to solve this class of problems. Another need is deriving stronger formulations of the joint chance-constrained problem that give stronger convex relaxations.

Acknowledgments

The authors are grateful to the anonymous referees' useful comments which greatly helped improve the presentation of this paper.

Appendix A

We use the deterministic linear program of Becker and Starczak [5] as the basis for a stochastic formulation for finding optimal vaccination strategies. The authors model a community divided up into households, each of which contains a heterogeneous population. We consider the random elements of the model to be the vaccine efficacy, the average contact rate of an infective, and the relative infectivities and susceptibilities. The parameters and details of the stochastic program are given in Table 3.

$$\text{Min}: \quad \sum_{f\in F}\sum_{v\in V}\sum_{t\in T} v_t h_f x_{fv} \tag{7a}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{fv} = 1 \quad \forall f \in F \tag{7b}$$

$$\mathbb{P}\left(\sum_{f\in F}\sum_{v\in V} a_{fv}(\omega)x_{fv} \leqslant 1\right) \geqslant \alpha \tag{7c}$$

$$0 \leqslant x_{fv} \leqslant 1 \quad \forall f \in F,\ v \in V. \tag{7d}$$

Table 3
Parameters for vaccination problem.

Sets	
F	Set of family types
T	Set of types of people
V	Set of vaccine policies
Ω	The set of scenarios
Indices	
f	Index for a family type in F
v	Index for a vaccination policy in V
t	Index for a person type in T
f_t	Index for the number of people of type t in a family of type f
v_t	Index for the number of people of type t vaccinated in v
ω	Index for a particular scenario in Ω
Parameters	
h_f	The proportion of households in the population that are of type f
$a_{nv}(\omega)$	Computed random parameter for impact of immunization decisions
μ_F	The average size of a household
Parameters to compute $a_{ijkl}(\omega)$	
$m(\omega)$	The average contact rate of infected people
$u_t(\omega)$	The relative infectivity of people of type t
$s_t(\omega)$	The relative susceptibility of people of type t
$b(\omega)$	The transmission proportion within a household
$\epsilon(\omega)$	The vaccine efficacy
Decision variables	
x_{fv}	The proportion of families of type f vaccinated under policy v

Table 4
List of family types and frequency.

Household size	Children	Adults	Elderly	Frequency
1	0	1	0	0.05
1	0	0	1	0.05
2	0	2	0	0.10
2	0	0	2	0.05
2	1	1	0	0.08
2	0	1	1	0.02
3	1	2	0	0.10
3	0	2	1	0.05
3	0	0	3	0.05
3	1	0	2	0.05
3	0	3	0	0.05
4	2	2	0	0.03
4	3	1	0	0.03
4	0	2	2	0.03
4	0	4	0	0.03
4	0	0	4	0.03
5	3	2	0	0.03
5	2	2	1	0.03
5	0	5	0	0.02
5	0	0	5	0.02
6	4	2	0	0.01
6	0	6	0	0.01
6	0	0	6	0.01
6	3	2	1	0.01
7	2	2	2	0.01
7	5	2	0	0.01
7	0	7	0	0.01
7	0	0	7	0.01
7	4	2	1	0.01
7	3	2	2	0.01

Table 5
List of parameters and distributions.

Parameter name	Symbol	Distribution
Vaccine efficacy	$\epsilon(\omega)$	Truncated normal (0.85,0.32) in interval [0,1]
Inter-household contact rate	$m(\omega)$	Truncated normal (1,0.5) in interval $[0,\infty]$
Intra-household spread rate	$b(\omega)$	Truncated normal (0.6,0.32) in interval [0,1]
Relative infectivity, person type t	$\mu_t(\omega)$	Low value 0.7, $p = 0.5$, high value 1.3, $p = 0.5$
Relative susceptibility, person type t	$\mu_t(\omega)$	Low value 0.7, $p = 0.5$, high value 1.3, $p = 0.5$

The objective function minimizes the vaccine coverage. The first constraint (7b) balances all the decision variables for each family type, ensuring that the proportions assigned sum to one. The second, probabilistic constraint (7c) requires that the reproductive number of the disease be brought below one at least α proportion of the time. $a_{fv}(\omega)$ is a function of the random variable realization given by (8).

$a_{fv}(\omega)$ is computed using the random infectivity, susceptibility, contact rate, and vaccine efficacy parameters of the original model. The equation to compute $a_{fv}(\omega)$ comes from Becker and Starczak [5] and is given below. It includes the assumption that between household contacts occur proportionately to the size of the household. Tables 4 and 5 give the exact household makeups and probability distributions that we assumed.

$$a_{fv}(\omega) = \frac{m(\omega)h_f}{\mu_F}\left(\sum_{t\in T} u_t(\omega)s_t(\omega)[(1-b(\omega))(f_t - v_t\epsilon(\omega)) + b(\omega)v_t\epsilon(1-\epsilon)] + b\sum_{t\in T}\sum_{r\in T} u_r(\omega)s_t(\omega)(f_t - v_t\epsilon(\omega))(f_r - v_r\epsilon(\omega))\right). \quad (8)$$

References

[1] E. Amaldi, M.E. Pfetsch, L.E. Trotter, On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs, Mathematical Programming 95 (3) (2003) 533–554.
[2] H. An, J.W. Eheart, A screening technique for joint chance-constrained programming for air quality management, Operations Research 55 (4) (2007) 792–798.
[3] R. Aringhieri, Solving chance-constrained programs combining tabu search and simulation, in: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms, Lecture Notes in Computer Science, vol. 3059, Springer-Verlag, Heidelberg, 2004, pp. 30–41.
[4] F. Ball, P. Neal, A general model for stochastic sir epidemics with two levels of mixing, Mathematical Biosciences 180 (2002) 73–102.
[5] N.G. Becker, D.N. Starczak, Optimal vaccination strategies for a community of households, Mathematical Biosciences 139 (2) (1997) 117–132.
[6] J.F. Benders, Partitioning procedures for solving mixed-variable programming problems, Numerische Mathematik 4 (1962) 238–252.
[7] J. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer-Verlag, New York, 1997.
[8] M.C. Campi, S. Garatti, Chance-constrained optimization via randomization: Feasibility and optimality, Optimization online, 2009. <http://www.optimization-online.org/DB_FILE/2008/09/2092.pdf>.
[9] M.S. Cheon, S. Ahmed, F. Al-Khayyal, A branch-reduce-cut algorithm for the global optimization of probabilistically constrained linear programs, Mathematical Programming 108 (2006) 617–634.
[10] John W. Chinnneck, Finding a useful subset of constraints for analysis in an infeasible linear program, INFORMS Journal on Computing 9 (1997) 164–174.
[11] G. Codato, M. Fischetti, Combinatorial benders cuts for mixed-integer linear programming, Operations Research 54 (2006) 756–766.
[12] D. Dentcheva, B. Lai, A. Ruszczyński, Efficient point methods for probabilistic optimization problems, Mathematical Methods of Operations Research 60 (2004) 331–346.
[13] D. Dentcheva, A. Prekopa, A. Ruszczyński, Concavity and efficient points of discrete distributions in probabilistic programming, Mathematical Programming 89 (1) (2000) 55–77.
[14] J. Gleeson, J. Ryan, Identifying minimally infeasible subsystems of inequalities, ORSA Journal on Computing 2 (1) (1990) 61–63.
[15] R. Henrion, A. Möller, Optimization of a continuous distillation process under random inflow rate, Computers and Mathematics with Applications 45 (2003) 247–262.
[16] Ira M. Longini Jr., M. Elizabeth Halloran, Azhar Nizam, Yang Yang, Containing pandemic influenza with antiviral agents, American Journal of Epidemiology 159 (7) (2004) 623–633.
[17] J.N.M. Van Loon, Irreducibly inconsistent systems of linear inequalities, European Journal of Operational Research 8 (1981) 283–288.
[18] J. Luedtke, S. Ahmed, A sample approximation approach for optimization with probabilistic constraints, SIAM Journal of Optimization 19 (2) (2008) 674–699.
[19] J. Luedtke, S. Ahmed, G. Nemhauser, An integer programming approach for linear programs with probabilistic constraints, Mathematical Programming 112 (2010) 247–272.
[20] D. Morgan, J.W. Eheart, A. Valocchi, Aquifer remediation design under uncertainty using a new chance constrained programming technique, Water Resources Research 29 (1993) 551–561.
[21] A. Nemirovski, A. Shapiro, Scenario approximations of chance constraints, published online at Optimization Online, 2004.
[22] S. Pagnoncelli, B. Ahmed, A. Shapiro, The sample average approximation method for chance constrained programming: Theory and applications, Journal of Optimization Theory and Applications 142 (2) (2009) 399–416.
[23] Rajan Patel, Ira M. Longini Jr., M. Elizabeth Halloran, Finding optimal vaccination strategies for pandemic influenza using genetic algorithms, Journal of Theoretical Biology 234 (2) (2005) 201–212.
[24] Marc E. Pfetsch, Branch-and-cut for the maximum feasible subset problem, SIAM Journal on Optimization 19 (2008) 21–38.
[25] A. Prékopa, A class of stochastic programming decision problems, Matematische Operationsforschung und Statistick (1972) 349–354.
[26] A. Prékopa, Programming under probabilistic constraints with a random technology matrix, Mathematische Operationsforshung und Statistik 5 (1974) 109–116.
[27] A. Prékopa, Probabilistic programming, in: Andrzej Ruszczyński, Alexander Shapiro (Eds.), Stochastic Programming, Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, The Netherlands, 2003, pp. 267–345.
[28] A. Ruszczyński, Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra, Mathematical Programming 93 (2) (2002) 195–215.
[29] S. Sen, Relaxations for probabilistically constrained programs with discrete random variables, Operations Research Letters 11 (1992) 81–86.
[30] A. Shapiro, Monte carlo sampling methods, in: A. Ruszczyński, A. Shapiro (Eds.), Stochastic Programming Handbooks in Operations Research and Management Science, vol. 10, North-Holland Publishing Company, Amsterdam, The Netherlands, 2003, pp. 353–425.
[31] M.W. Tanner, L. Sattenspiel, L. Ntaimo, Finding optimal vaccination strategies under uncertainty using stochastic programming, Mathematical Biosciences 215 (2008) 144–151.
[32] S.R. Tayur, R.R. Thomas, N.R. Natraj, An algebraic geometry algorithm for scheduling in presence of setups and correlated demands, Mathematical Programming 69 (1995) 369–401.