A General Heuristic Method for Joint Chance-Constrained Stochastic Programs with Discretely Distributed Parameters

Matthew W. Tanner* and Eric B. Beier
Department of Industrial and Systems Engineering
Texas A & M University, mtanner@tamu.edu and ebeier@tamu.edu

Abstract

We present a general metaheuristic for joint chance-constrained stochastic programs with discretely distributed parameters. We give a reformulation of the problem that allows us to define a finite solution space. We then formulate a novel neighborhood for the problem and give methods for efficiently searching this neighborhood for solutions that are likely to be improving. We formulate a reduced linear program that is all that needs to be solved each iteration of the algorithm, reducing the computational effort required for the search. Finally, we give a random tabu search heuristic to search our solution space. The algorithm differs from other work on joint chance-constrained problems in that it is valid for problems with randomness both in the constraint matrix and in the righthand side. We finish with computational experiments on two sets of test instances that show that our algorithm is highly effective at finding good feasible solutions to these problems. Keywords{stochastic programming, chance constraints, probabilistic constraints, tabu search}

1 Introduction

In many optimization models, assuming deterministic data when actual data is stochastic or hard to estimate can lead to undesirable outcomes. Decisions made with a deterministic model can, after the realization of randomness, be highly suboptimal or even infeasible.

Stochastic programming is one framework for taking the stochastic nature of the data into account when formulating and solving an optimization problem.

In stochastic programming formulations, decisions are divided into those that need to be made "here and now" and those that can be made after the values of the random variables become known. This framework can encompass both a two-stage setting where the random data is realized all at once, or the multistage setting where the random data is realized at the beginning of each stage.

Adding chance constraints to a linear program is a popular method for introducing stochasticity into a linear program. In the basic model that we will consider, a set of decisions is made and then the probability that those decisions are feasible is computed using a probability distribution of the random elements of the problem. The requirement that the constraints need to be satisfied almost surely is relaxed, and instead, the goal is to find the minimum cost decisions such that a subset of the problem constraints are allowed to be violated an "acceptable" amount of time. There are no decisions to be made after the uncertain information has become known. In this paper, we focus on problems where the distribution of the random variables is discrete.

The general problem formulation that we consider is shown in equations (1a) - (1d), where $x \in \mathcal{R}^n$ is the decision variable vector, $c \in \mathcal{R}^n$ is the cost parameter vector, $A \in \mathcal{R}^{n \times m_1}$ is the deterministic constraint matrix, and $b \in \mathcal{R}^{m_1}$ is the deterministic right-hand side. In this formulation, uncertainty appears as the multi-dimensional random variable $\tilde{\omega}$, that gives rise to the random parameter matrix $T(\tilde{\omega}) \in \mathcal{R}^{n \times m_2}$ and the random right-hand side vector $r(\tilde{\omega}) \in \mathcal{R}^{m_2}$. Individual outcomes of the random variable are represented as realizations $\omega \in \Omega$ of the probability space . A reliability parameter $\alpha \in [0, 1]$ is included in constraint (1c)to require the random constraints of this problem be satisfied with at least probability α. Since the random variables are discrete, the chance constraint (1c) can be thought of as $|\Omega|$ sets of m_2 constraints, one set for each realization of the random variable $\tilde{\omega}$ with the property that up to $100 * (1 - \alpha)\%$ of the constraints can be violated.

$$\min \quad c^\top x \tag{1a}$$

$$\text{s.t.} \quad Ax \leq b \tag{1b}$$

$$\mathbb{P}\big(T(\tilde{\omega})x \leq r(\tilde{\omega})\big) \geq \alpha \tag{1c}$$

$$x \geq 0 \tag{1d}$$

In general, the above problem is nonconvex and quite difficult to solve. Applications of formulation (1) include optimization of machine capacities on a shop floor [23], multi-commodity supply chains [21], optimal vaccination strategies under uncertainty [22], and optimal investment strategies with a certain level of risk [10]. Up to this point most research into probabilistic constraints has focused on identifying distributions of Ω for which the problem becomes convex [5, 19] or on the special case of the above problem where all the randomness is in the righthand side [8, 9, 10, 15].

One branch of research into these problems has focused on convex programming approaches to identifying feasible solutions to problem (1). These methods focus on finding convex restrictions to the feasible space of the chance-constraints that can then be optimized efficiently by standard methods [1, 17, 18, 20]. Another method for finding feasible solutions to the problem with high probability is through sampling [6, 7, 14]. One weakness with all these methods is that while they can be used to find feasible solutions to formulation (1), the solutions found are often conservative. There is still a need for heuristic and exact methods for finding good quality feasible solutions.

To the best of our knowledge, there have only been a few heuristics developed to find good solutions to formulation (1). Iwamura and Liu [13] present a genetic algorithm approach to the above problem, while Aringhieri [2] gives a tabu search heuristic in the case where the decision variables are limited to pure integer. In both papers, the algorithms search the candidate solutions and test their objective value and feasibility through simulation. Simulation is needed because direct computation of the objective value is extremely difficult. However, this problem can be addressed by assuming discrete distributions of the random variables.

Morgan et al. [16] show that when $|\Omega| < \infty$ formulation (1) can be reformulated as a deterministic mixed-binary linear program with a knapsack constraint and a binary variable for each scenario. A given binary variable takes the value of zero if the corresponding constraint is satisfied, and one otherwise. The mixed-binary formulation is given in equations (2a) - (2e), where $M \in \mathcal{R}$ is a large number, $z_\omega \in \mathcal{B}^{|\Omega|}$ is a vector of binary decision variables, p_ω is the probability of a scenario $\omega \in \Omega$, and e is an appropriately sized vector of ones.

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

$$\text{s.t.} \quad T(\omega)x - Mez_\omega \leq r(\omega) \quad \forall \omega \in \Omega \tag{2b}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \leq 1 - \alpha \tag{2c}$$

$$Ax \leq b \tag{2d}$$

$$x \geq 0,\ z \in \mathbb{B}^{|\Omega|} \tag{2e}$$

Each scenario $\omega \in \Omega$ corresponds to a set of rows of constraints (2b). The constraint allows $T(\omega)x > r(\omega)$ as long as $z_\omega = 1$. The "big M" ensures that $z_\omega = 1$ implies satisfaction of the constraint. Constraint (2c) is a knapsack constraint that limits the proportion of z_ω that can be set to 1, this ensures satisfaction of the original chance-constraint. The above formulation can be solved using traditional integer programming techniques and has $|\Omega|$ binary variables, n linear variables, and $m_1 + |\Omega| m_2$ constraints. However, the "Big M" formulation tends to have weak LP relaxations which limit the effectiveness of traditional branch-and-bound algorithms. Ruszczyński [21] gives an exact algorithm for this formulation based on induced cover inequalities for the knapsack constraint.

The main contributions of this paper are threefold. First, we give a reformulation of the problem that suggests methods to exploit the scenario structure. Second, we present methods to identify subsets of scenarios that are most important in identifying good solutions, which lead to a new tabu search metaheuristic that can quickly find good, feasible solutions for the problem using our reformulation. Finally, we give some computational results on several test problem sets to demonstrate the effectiveness of the algorithm.

The rest of this paper is organized as follows: In Section 2 we give the reformulation and discuss some properties that allow us to restrict the solution search space. We also define a neighborhood and describe methods for quickly searching that neighborhood for improving solutions. Section 3 gives a metaheuristic based on tabu search using our reformulation and our sampling methods. Section 4 gives some randomly generated problem test sets and computational results using our heuristic. We finish with some conclusions and future work.

2 Local Search

This section presents a reformulation of problem (1) that allows us to define a finite solution set based on sets of scenarios to be satisfied. We then define a local neighborhood that can be searched for improving solutions. Naive exploration of this neighborhood is computationally infeasible and so the section concludes with a discussion of how to efficiently search the neighborhood.

2.1 Defining the Neighborhood

DEFINITION 2.1. A scenario ω is said to be *satisfied* by solution x if $T(\omega)x \leq r(\omega)$. Otherwise, the scenario ω is unsatisfied.

In the mixed integer formulation of problem (2a) - (2d), each scenario is represented by a binary variable that defines whether or not the constraints associated with that scenario are satisfied by the solution. Fixing a variable $z_\omega = 0$ or $z_\omega = 1$ means that the scenario ω corresponding to that variable must be satisfied or unsatisfied respectively.

DEFINITION 2.2. A *candidate solution* is a subset of scenarios $C \subseteq \Omega$ such that $\sum_{\omega \in C} p_\omega \geq \alpha$. Define the set of feasible solutions Φ as the set of all candidate solutions.

For any such candidate set $C \subseteq \Omega$, a feasible solution to the problem can be found by solving the linear relaxation of (2) with the variables $z_\omega \; \forall \omega \in C$ fixed to zero and all other binary variables fixed to one. This is true because in such a solution $\sum_\omega p_\omega z_\omega \leq 1 - \alpha$ by the definition of the set C. A reformulation of the problem based on searching the scenario space is given below. Solving the linear program (3a) - (3d) defined by a set $C \in \Phi$ gives an upper bound on the objective value to the original problem.

Find $C \in \Phi$ such that $f(C)$ is minimized
where,

$$
\begin{aligned}
f(C) = \quad & \min \quad c^\top x && \text{(3a)} \\
& \text{s.t.} \quad Ax \leq b && \text{(3b)} \\
& \qquad T(\omega)x \leq r(\omega) \; \forall \omega \in C && \text{(3c)} \\
& \qquad x \geq 0 && \text{(3d)}
\end{aligned}
$$

Our algorithm searches the feasible space of candidate sets looking for improving solutions. From here on, we refer to scenarios in the current candidate set C as being forced into the problem, and scenarios outside of that set C as being forced out of the problem.

DEFINITION 2.3. A candidate solution $C \in \Phi$ is a *minimal element* of Φ if $\forall \omega_j \in C$, $\sum_{\omega \in C \setminus \omega_j} p_\omega < \alpha$.

REMARK 2.4. Restricting the search space Φ to its minimal elements will not eliminate all optimal solutions of the problem.

REMARK 2.5. In the case where every scenario has the same probability, remark 2.4 implies that the search can be restricted to solutions where exactly $\lceil \alpha |\Omega| \rceil$ scenarios are forced to be satisfied.

DEFINITION 2.6. Define the neighborhood $\mathcal{N}(C)$ of any element $C \in \Phi$ as all sets C' that can be constructed by adding a scenario $\omega \in \Omega \setminus C$ and then removing scenarios from C until C' is a minimal element of Φ.

The objective value of a candidate solution C, $f(C)$ can be found by solving problem (3). If formulation (3) is infeasible, then $f(C) = \infty$. Define $I(C)$ as the sum of the infeasibilities of the constraints of formulation (3), with $I(C) = 0$ when formulation (3) is feasible. Define $g(C')$ as the evaluation function for a potential solution $C' \in \mathcal{N}(C)$ with the goal of first finding feasible solutions and then improving the objective function.

$$g(C') = \begin{cases} I(C'), & \text{if } I(C) > 0; \\ f(C'), & \text{otherwise.} \end{cases} \tag{4}$$

2.2 Efficiently Searching the Neighborhood

The naive way to search the neighborhood of C is to evaluate $g(C')$ $\forall C' \in \mathcal{N}(C)$. This process is impractical because it requires solving $O(|\Omega|^2)$ relatively large linear programs. Computational results from an early implementation took up to 10 minutes to exhaustively search the neighborhood of a single candidate solution. This subsection gives heuristic methods that allow us to solve a few greatly reduced linear programs each iteration of our search algorithm to compute $f(C)$, and to efficiently search the neighborhood of C without having to solve any linear programs.

For knapsack problems, it has been observed computationally that a relatively small "core" of variables can be identified such that solving a reduced problem with just these variables gives the same optimal solution as the entire problem [3]. We have observed a similar structure in discretely distributed chance-constrained problems. Many scenarios are often redundant in the sense that their constraints are much easier to satisfy than the constraints of a "core" set of scenarios. Thus, such scenarios can be implicitly included in the candidate solution C, shrinking the size of the linear program necessary to compute $f(C)$.

DEFINITION 2.7. A set of scenarios $D(C) \subseteq \Omega$ is a *sufficient set* for a candidate solution C if all scenarios $\omega \in C$ are satisfied by the optimal solution to the following LP.

$$\begin{aligned} f(C) = \quad & \min \quad c^\top x && \text{(5a)} \\ & \text{s.t.} \quad Ax \leq b && \text{(5b)} \\ & \qquad T(\omega)x \leq r(\omega)\ \forall \omega \in D(C) && \text{(5c)} \\ & \qquad x \geq 0 && \text{(5d)} \end{aligned}$$

Clearly, given a valid sufficient set $D(C)$, the optimal objective value for problem (5) is equal to the optimal objective value of (3) and still gives a valid upper bound on the optimal objective value of the original problem. In section 3, we show how to construct an initial sufficient set $D(C)$ and also give a method for quickly updating $D(C)$ as the algorithm progresses.

DEFINITION 2.8. Define a scenario ω as *tight* if at least one of the slack variables associated with the constraints of scenario ω has value 0.

The other computational intensive step in searching the neighborhood of a candidate solution C is evaluating $f(C')\ \forall C' \in \mathcal{N}(C)$. In the definition of $\mathcal{N}(C)$, for any $C' \in \mathcal{N}(C)$ there is exactly one scenario $\omega(C')$ included in C that is outside C'. When $f(C) < \infty$, the only $C' \in \mathcal{N}(C)$ that can result in an improved objective function are those for which at least one constraint of scenario $\omega(C')$ is tight. In the case that $f(C) = \infty$, we only consider $C' \in \mathcal{N}(C)$ for which the constraints of scenario $\omega(C')$ are most infeasible. To choose which $C' \in \mathcal{N}(C)$ are most likely to result in an improved solution, it is necessary to identify scenarios not included in C that are the best to include. One measure of the

quality of a scenario to include is the maximum infeasibility of the constraints associated with that scenario under the optimal solution to formulation (5).

Using these criteria, the best $C' \in \mathcal{N}(C)$ are ranked first by choosing which scenario $\omega(C')$ to remove from C. The leaving scenario is chosen as either a tight scenario if $f(C) < \infty$, or the maximally infeasible scenario if $f(C) = \infty$. The entering scenarios are chosen by ranking them by the least maximum infeasible constraint. These search methods do not guarantee that the best possible move is chosen, however it is more important to be able to search the neighborhood quickly as long as there is a reasonable chance that improving solutions can be identified.

3 Tabu Search for Probabilistically Constrained Programs

In the previous section, we defined a finite set of candidate solutions and a neighborhood for each solution. We also gave some methods for quickly searching the neighborhood of a candidate solution C to identify search steps that are likely to result in an improvement to the objective value. In this section, we define a new, general heuristic for solving problems of the form (1) with discretely distributed random parameters. Our algorithm uses a random tabu search [11, 12] to identify candidate solutions with the goal of converging to optimal or nearly optimal solutions. The algorithm includes preprocessing steps that identify scenarios that can be completely dropped from consideration, a construction heuristic to identify a good initial feasible solution, and a tabu search method to improve the initial solution.

Define V as the objective value of the current incumbent solution found by the algorithm. Define $\bar{x}$ as the decision variable values associated with the current candidate solution $\bar{C}$. For each scenario $\omega \in \bar{C}$, define $\bar{s}_\omega$ as the minimum slack variable value of the constraints of that scenario. Define l_ω as a lower bound for the cost of including a given scenario in any solution.

3.1 Preprocessing

Before the main step of the algorithm, our method gathers information about the respective costs of the different scenarios of the problem that can be used to improve the

speed of the algorithm and the quality of the solution that it finds. By solving a linear relaxation of problem (2) for each scenario where only the deterministic constraints and the constraint set of that one scenario are required to be satisfied, we obtain a lower bound for any possible solution in the search space that includes that scenario. A formulation the problem to compute l_ω is given below.

$$
\begin{aligned}
l_\omega = \min \quad & c^\top x && \text{(6a)}\\
\text{s.t.} \quad & Ax \leq b && \text{(6b)}\\
& T(\omega)x \leq r(\omega) && \text{(6c)}\\
& x \geq 0 && \text{(6d)}
\end{aligned}
$$

If $V \leq l_\omega$, then for all candidate solutions $C : \omega \in C$, $f(C) \geq V$. Thus the scenario ω can be dropped from further consideration by the algorithm. Computing l_ω is also useful because it allows for a rough sorting of the scenarios by "cost" of satisfying them.

3.2 Construction

The goal of this construction heuristic is to quickly define a good candidate solution to be sent to the main tabu search loop. The heuristic works by first selecting a candidate solution C by greedily choosing scenarios in terms of the lowest values l_ω as found in the preprocessing step. Then, the the objective value of solution C is computed. Scenarios are put into C if they are satisfied by the optimal decision variable values associated with solution C. Scenarios are removed from C in order to keep the solution minimal. These steps are repeated until no more feasible solutions can be added.

Construction Heuristic

Step 0: Initialization Set $V = \infty$. Sort the scenario lower bounds l_ω. Construct an initial candidate solution $\bar{C}$ by adding scenarios with the lowest l_ω until $\sum_{\omega \in C} p_\omega \geq \alpha$.

Step 1: Update Compute $f(\bar{C})$ by solving (3). Assign the optimal decision variable values to $\bar{x}$. Assign the minimum slack variable value for each scenario ω to $\bar{s}_\omega$. If $f(\bar{C}) \leq V$, set $V = f(\bar{C})$.

Step 2: Add Scenarios For every $\omega \in \Omega \setminus \bar{C}$, if scenario ω is satisfied by $\bar{x}$ set $\bar{C} = \bar{C} \cup \omega$. If no such scenarios can be added, return $\bar{C}$ as the initial candidate solution for tabu search.

Step 3: Remove Scenarios Remove scenarios from $\bar{C}$ until $\bar{C}$ is a minimal element of Φ. Remove scenarios in order of lowest minimum slack value $\bar{s}_\omega$. Return to Step 1.

3.3 A Tabu Search Algorithm

This section presents the tabu search and a routine for updating the sufficient set $D(C)$. In each step of the tabu search algorithm, a scenario that is in the candidate solution C is exchanged with a scenario that is not a current member of our candidate solution C. We use one tabu list to prevent scenarios that were recently put into C from being removed, and another tabu list to prevent scenarios that were recently taken out of C from being added. The sizes of the respective tabu lists are user defined parameters set by computational experiments. The algorithm also includes an element of randomness as the leaving scenario is chosen at random from among the set of tight scenarios of C that are not on the tabu list. We choose the scenario to remove using uniform probabilities on the set of tight scenarios.

In the following algorithm, the tabu lists are implemented with lengths defined by the modeler. As the algorithm iterates, the first scenario on each tabu list is removed from the list and the scenarios that were added or removed to C are added to the end of the list respectively.

Tabu Search

Step 0: Initialization Use the construction heuristic to generate an initial feasible solution with objective value V. Set $D(C)$ to be the set of all tight scenarios in the final solution of formulation (3).

Step 1: Calculate $\mathbf{f(C)}$ Update $D(C)$ using the sufficient set subroutine. Assign the optimal value returned by the subroutine to $f(C)$ and assign the decision variable values to $\bar{x}$. If $f(C) < V$, set $V = f(C)$.

Step 2: Choose Leaving Scenario Depending on $f(C)$,

- If $f(C) < \infty$, pick a nontabu scenario $\omega \in D(C)$ at random from among the set of tight scenarios. If no such scenario exists, pick the nontabu scenario ω with the minimum s_ω.
- If $f(C) = \infty$, pick the nontabu scenario ω with the maximum infeasibility.

Step 3: Choose Entering Scenarios While $\sum_{\omega \in C} p_\omega < \alpha$, add scenarios to C in order of minimum infeasibility under decisions $\bar{x}$.

Step 4: Stopping Stop the algorithm if the maximum time is reached, otherwise return to Step 1.

The sufficient set updating subroutine takes a candidate sufficient set as input. Define k_ω as the number of iterations that a scenario ω has been included in $D(C)$ since that scenario was tight. Define K as the maximum value of k_ω before scenario ω is removed from $D(C)$.

Sufficient Set Updating

Step 1: Solve LP Solve formulation (5) using set $D(C)$. Assign the optimal decision variables to $\bar{x}$.

Step 2: Add Scenarios For all scenarios $\omega \in C$ such that $\omega \notin D(C)$, if scenario ω is satisfied by $\bar{x}$, then add scenario ω to $D(C)$.

Step 3: Remove Scenarios For all scenarios $\omega \in D(C)$, if scenario $k_\omega > K$ remove scenario ω from $D(C)$.

Step 4: Stopping If no scenarios have been added to $D(C)$ in Step 2 then stop. Otherwise, return to Step 1.

4 Computational Results

4.1 Test Instance Generation

To test the effectiveness of our heuristic, we generated two sets of random test instances. The first set of test instances is a chance-constrained program applied to finding an optimal vaccination policy. The details of the application can be found in [22]. The

problem formulation and the random parameter distributions that we assumed can be found in Appendix 1. Problem size information can be found in Table 1. The second set of test instances that we generated is a chance-constrained formulation of a production planning problem. The exact formulation and probability distributions can be found in Appendix 2, while the Problem size information is given in Table 2. For each of these two sets, we generated 5 problems of each size using random sampling.

Instance	Rows	Cont. Vars.	Binary Vars.
vac500	531	302	500
vac750	781	302	750
vac1000	1031	302	1000
vac2000	2031	302	2000
vac3500	3531	302	3500
vac5000	5031	302	5000
vac10000	10031	302	10000

Table 1: Problem Sizes for Vaccination Test Instances

Instance	Rows	Cont. Vars.	Binary Vars.
Prod100	5531	75	100
Prod250	13781	75	250
Prod500	27531	75	500
Prod750	41281	75	750
Prod1000	55031	75	1000
Prod2000	110031	75	2000

Table 2: Problem Sizes for Production Planning Test Instances

In both Table 1 and Table 2, the first column gives the names of the test instance. The number in these names refers to the number of scenarios. The second column gives the number of rows of the MIP formulation of the problem. The third column gives the number of continuous variables that the problem has, while the fourth column gives the number of binary variables that the problem has. The vaccination test instances have much few rows than the production planning test instances, but they are difficult because the instances tend to be extremely dense. The production planning test instances are

difficult because the MIP formulation becomes extremely large as the number of scenarios increases.

For comparison, we ran all test instances using CPLEX 9.0 directly on formulation (2). For all of our tests, we used a Dell Optiplex GX620 computer with a PentiumD 3.0 GHz processor and 4.0 GB RAM. We implemented our heuristic using the CPLEX 9.0 callable library within the C++ environment.

4.2 Algorithm Results

This subsection gives the results of our algorithm on our two sets of test instances. We used CPLEX on the MIP formulation of each test instance as a control case to compare the effectiveness of our heuristic. Both heuristic and CPLEX tests were run for 2 hours because that is around the time that CPLEX's branch-and-bound tends to run out of memory. This subsection gives tables of the respective results of the two methods, a discussion of how we chose good parameter values for our tabu search, and plots showing the convergence of the heuristic upper bounds that were found.

	CPLEX			Tabu Search	
Instance	Obj. Val.	Time	Opt. Gap	Best Bound	Improvement
vac500	65.64	30.82	0%	65.64	0
vac750	65.23	105.02	0%	65.23	0
vac1000	64.90	777.77	0%	64.90	0
vac2000	65.33	>7200	6.34%	65.19	0.13
vac3500	65.87	>7200	24.20%	65.34	0.53
vac5000	66.07	>7200	27.39%	65.11	0.96
vac10000	67.86	>7200	33.86%	65.27	2.59
vac20000	80.42	>7200	46.37%	65.35	14.60

Table 3: Results on Vaccination Test Instances

Tables 3 and 4 give the basic results of CPLEX and our heuristic. For both sets of test instances, the results are averaged over 5 test cases of each size problem. In each table, the first column gives the name of the test instance. The second column gives the average best objective value found by CPLEX. The third column gives the average amount of time CPLEX took to find an optimal solution. The fourth column gives the optimality gap of CPLEX using the relation $\frac{\text{upper bound - lower bound}}{\text{upper bound}}$. The fifth column gives

	CPLEX			Tabu Search	
Instance	Obj. Val.	Time	Opt. Gap	Obj. Val.	Improvement
Prod100	-91539.98	766.36	0%	-91481.02	-58.96
Prod250	-87561.1	>7200	15.42%	-88194.18	633.08
Prod500	-85222.58	>7200	30.77%	-86311.62	1089.04
Prod750	-84178.00	>7200	37.49%	-85443.88	1265.88
Prod1000	-82563.7	>7200	43.60%	-85044.06	2480.36
Prod2000	-72097.04	>7200	70.82%	-84451.22	12354.18

Table 4: Results on Production Planning Instances

the average best objective value found by our heuristic, while the last column gives the average absolute improvement of our heuristic over the best bound found by CPLEX.

The tables clearly show our heuristic consistently finds better solutions for both instances of joint chance-constrained stochastic programs. For the vaccination test cases in Table 3, CPLEX can prove optimality for all problems up to 1000 scenarios. Our heuristic finds the same optimal solutions for each of these instances. For larger problems CPLEX is unable to find optimal solutions and as the problem size increases, the optimality gap of CPLEX quickly increases. Our heuristic is able to find improved solutions in every test case that CPLEX in unable to solve in 2 hours. In the very largest test instance, CPLEX stops with a gap of almost 50% while our heuristic is able to find a much better solution.

Table 4 shows a similar dynamic in the comparison of CPLEX and our heuristic for the production planning problems. In this case, the problems are more difficult and CPLEX is only able to prove optimality for the smallest test instances. Also, the optimal gaps reported by CPLEX are much higher for these tests, culminating in an average gap of 71% for the largest test problems. The heuristic was only able to find the optimal solution in 4 out 5 of the problems with 100 scenarios which is why the average improvement for those is negative. For the larger problems, the heuristic is again able to find much better solutions in the two hours. Our heuristic finds an improved solution for each of the larger problems in this set of test instances. For the largest problems, the heuristic is able to find solutions with an average improvement of over 10,000.

The two major user defined parameters of our tabu search heuristic that can be set by the modeler are the sizes of the two tabu lists. For the vaccination test instances, the

algorithm gives the best results when the tabu list that prevents scenarios from being put back into the candidate solution C had size 5, while the tabu list that keeps scenarios from being taken out of C had size 1. The interesting thing is that these parameter values worked well regardless of problem size. The results were relatively insensitive to the parameter values, although if the size of the first tabu list was made a lot bigger or a lot smaller, then the algorithm returned much worse solutions. The very small size of the tabu list for keeping scenarios from being taken out of C is not actually surprising because there are often only a small number of scenarios that are candidates for removal and picking from among those candidates at random has much the same effect as a tabu list.

For the production planning test instances, choosing the correct sizes for the tabu lists was more challenging. In these cases, the best of tabu list length was more dependent on the size of the test problem. As a basic rule of thumb, we started with each tabu list of size 2 for the smallest test problems, increasing the sizes by 1 for each increase in the size of the problem. For the larger test instances, we used size 5 for both tabu lists. The heuristic has similar performance for a range around these sizes, providing empirical evidence that the performance of our heuristic is relatively insensitive to the exact sizes of the tabu lists.

Figures 1 and 2 show the best feasible solutions found by our heuristic as a function of time. Figure 1 shows the results for vac20000. For this test instance, our construction heuristic finds a good initial solution that is already much better than the best bound found by CPLEX after only 85 seconds. The heuristic then spends a long time searching without finding any improving solutions for over an hour. Starting at around an hour, the heuristic found another improvement and then quickly finds a series of slightly improving solutions. The algorithm was not able to find any improvements over the final 3000 seconds of runtime.

Figure 2 shows the results for prod2000. The construction heuristic finds about the same quality solution as CPLEX is able to find in the two hour test but it only requires 4.5 seconds. The tabu search is then able to find a series of improving solutions over 400 seconds that result in a final solution that is much better than CPLEX was able to find. The heuristic does not find any improvements over the final 6000 seconds of runtime.

First, these charts show that our construction heuristic is quite effective at finding good initial solutions. This means that we have an excellent starting point for tabu search.

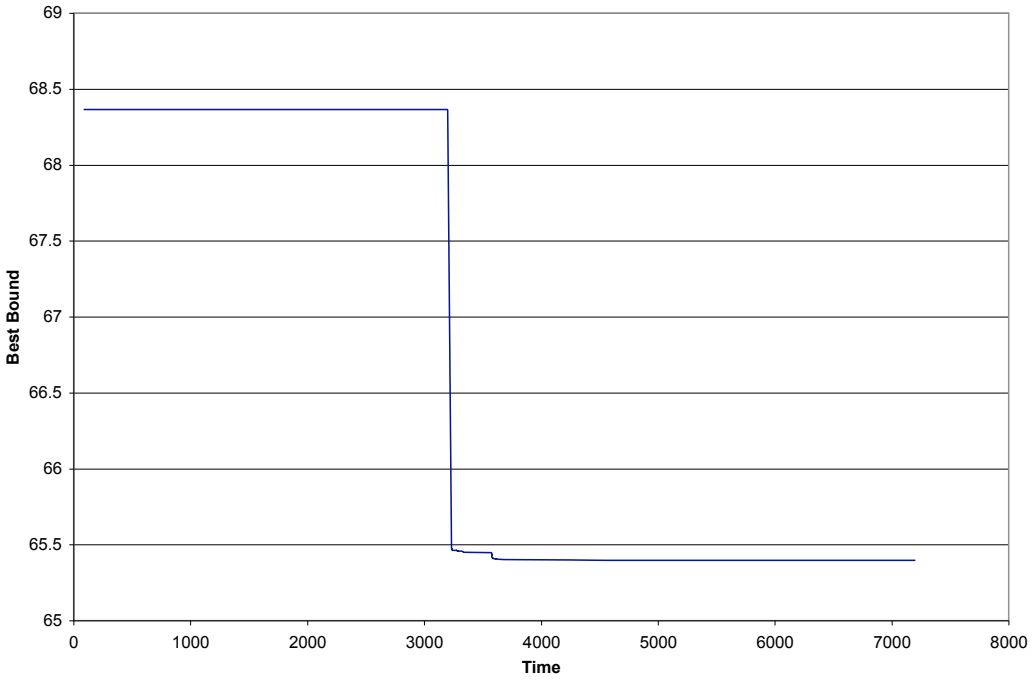

Figure 1: Plot of Best Feasible Solution vs. Time for Vac20000

Second, especially with production test instance, tabu search was able to find improving solutions quickly. This shows the effectiveness of our neighborhood search techniques and leads to the possibility of combining branch-and-bound with our heuristic in order to make a more effective exact algorithm.

The results of our heuristic on these two sets of test instances provide evidence that it can be used to effectively find good feasible solutions for joint chance-constrained stochastic programs with random constraint matrices. This is a significant result because few computational results exist for this class of problems because of their intractability. Our heuristic is able to find good feasible solutions much quicker than CPLEX. Also, it is more scalable than CPLEX because our neighborhood search methods are effective at limiting the extra computation effort required during each search step of the algorithm

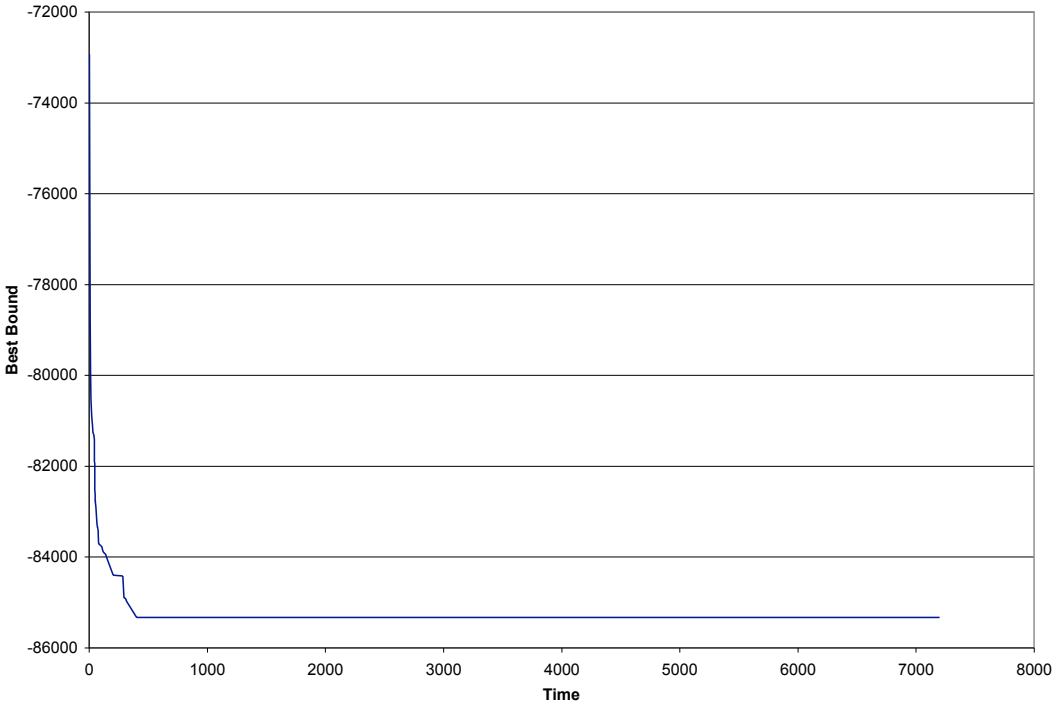

Figure 2: Plot of Best Feasible Solution vs. Time for Prod2000

due to increasing problem size. It is also important to note that our heuristic does not require much computer memory, so it can be used effectively on much larger problem instances than can branch-and-bound on the MIP formulation.

5 Conclusions and Future Work

We have presented a general metaheuristic for finding good, feasible solutions to joint chance-constrained stochastic programs for the case in which the random parameters have discrete distributions. Our algorithm is a random tabu search over a novel neighborhood that we formulated for this class of problems. We gave some methods for efficiently searching our neighborhood for likely improving solutions. We presented an effective

construction heuristic as well as our tabu search main loop. Computational results showed that our heuristic is highly effective at finding good feasible solutions. We were able to beat the best bound found by CPLEX for all test cases for which CPLEX could not find the optimal solution. The computational results also show that our heuristic is able to find improving solutions more quickly than the 2 hours allotted suggesting that our heuristic could be used in tandem with a general MIP algorithm to make an effective exact algorithm for this class of problems.

Acknowledgements

We would like to sincerely thank Dr. Lewis Ntaimo of Texas A & M University for helpful comments and suggestions.

Funding for Eric B. Beier: This research was performed while on appointment as a U.S. Department of Homeland Security (DHS) Fellow under the DHS Scholarship and Fellowship Program, a program administered by the Oak Ridge Institute for Science and Education (ORISE) for DHS through an interagency agreement with the U.S Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORISE.

References

[1] H. An and J.W. Eheart. A screening technique for joint chance-constrained programming for air quality management. *Operations Research*, 55(4):792–798, 2007.

[2] R. Aringhieri. A tabu search algorithm for solving chance-constrained programs. *Journal of the ACM*, 5:1–14, 2004.

[3] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.

[4] Niels G. Becker and Dianna N. Starczak. Optimal vaccination strategies for a community of households. *Mathematical Biosciences*, 139(2):117–132, 1997.

[5] J. R. Birge and F. V. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.

[6] G. Calafiore and M. C. Campi. Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming*, 102:25–46, 2005.

[7] G. Calafiore and M. C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51:742–753, 2006.

[8] MS Cheon, S Ahmed, and F Al-Khayyal. A branch-reduce-cut algorithm for the global optimization of probabilistically constrained linear programs. *Mathematical Programming*, 108:617–634, 2006.

[9] Darinka Dentcheva, A. Prékopa, and Andrzej Ruszczyński. On convex probabilistic programming with discrete distributions. *Nonlinear Analysis*, 47:1997–2009, 2001.

[10] Darinka Dentcheva, Bogumila Lai, and Andrzej Ruszczyński. Efficient point methods for probabilistic optimization problems. *Mathematical Methods of Operations Research*, Vol. 60:331–346, 2004.

[11] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston/Dutrecht/London, 1997.

[12] H. Hoos and T. Stutzle. *Stochastic Local Search: Foundations and Applications*. Elsevier, San Francisco, 2005.

[13] K. Iwamura and B. Liu. A genetic algorithm for chance constrained programming. *Journal of Information and Optimization Sciences*, 17:409–422, 1996.

[14] J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *Submitted*, 2007.

[15] J. Luedtke, S. Ahmed, and G. Nemhauser. An integer programming approach for linear programs with probabilistic constraints. *Submitted*, 2007.

[16] D. Morgan, J.W. Eheart, and A. Valocchi. Aquifer remediation design under uncertainty using a new chance constrained programming technique. *Water Resources Research*, 29:551–561, 1993.

[17] A. Nemirovski and A. Shapiro. Convex approximation of chance constrained programs. *SIAM Journal on Optimization*, 17:969–996, 2006.

[18] J. Pinter. Deterministic approximations of probability inequalities. *Operations Research*, 33:219–239, 1989.

[19] A. Prékopa. Programming under probabilistic constraints with a random technology matrix. *Optimization*, 5:109–116, 1974.

[20] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000.

[21] Andrzej Ruszczyński. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, 93(2):195–215, 2002.

[22] M.W. Tanner, L. Sattenspiel, and L. Ntaimo. Finding optimal vaccination strategies under parameter uncertainty using stochastic programming. *submitted*.

[23] Sridhar R. Tayur, Rekha R. Thomas, and N.R. Natraj. An algebraic geometry algorithm for scheduling in presence of setups and correlated demands. *Mathematical Programming*, 69:369–401, 1995.

Appendix 1

We use the deterministic linear program of Becker and Starczak [4] 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 5.

Table 5: 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

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

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

$$\mathbb{P}\Big(\sum_{f\in F}\sum_{v\in V} a_{fv}(\omega)x_{fv} \leq 1\Big) \geq \alpha \tag{7c}$$

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

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 that 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 [4] and is given below. It includes the assumption that between household contacts occur proportionately to the size of the household. Table 6 and Table 7 give the exact household makeups and probability distributions that we assumed.

$$a_{fv}(\omega) = \frac{m(\omega)h_f}{\mu_F}\Big(\sum_{t\in T} u_t(\omega)s_t(\omega)\big[(1-b(\omega))(f_t - v_t\epsilon(\omega)) + b(\omega)v_t\epsilon(1-\epsilon)\big] \tag{8}$$

$$+b\sum_{t\in T}\sum_{r\in T} u_r(\omega)s_t(\omega)(f_t - v_t\epsilon(\omega))(f_r - v_r\epsilon(\omega))\Big)$$

Appendix 2

We start with a general deterministic production planning model for five products over a five period time horizon. The objective is to maximize profit minus the production costs. The first constraint is a mass balance constraint (9b). The second constraint is the joint chance-constraint (9c) made up of constraints that set the amount of raw materials available in each time to produce all products and upper and lower bounds on the production levels. The parameters and decision variables of the model are given in Table 8, while the distributions of the random parameters are given in Table 9.

$$\max \quad \sum_{kt} -c_{kt} m_{kt} + p_{kt} s_{kt} \tag{9a}$$

$$\text{s.t.} \quad m_{kt} + I_{kt-1} - I_{kt} - s_{kt} = 0 \;\; \forall t \in T, \forall k \in K \tag{9b}$$

$$\mathcal{P}\left(\begin{array}{l} \sum_k n_{kt}(\omega) m_{kt} \leq r_{kt} \;\; \forall t \in T \\ s_{kt} \leq \max_{kt}(\omega) \;\; \forall t \in T, \forall k \in K \\ s_{kt} \geq \min_{kt}(\omega) \;\; \forall t \in T, \forall k \in K \end{array} \right) \geq \alpha \tag{9c}$$

Footnotes

Affiliation of Authors:
Matthew W. Tanner*, Department of Industrial and Systems Engineering Texas A & M University, mtanner@tamu.edu
Eric B. Beier, Department of Industrial and Systems Engineering Texas A & M University, ebeier@tamu.edu

*denotes corresponding author

Table 6: 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 7: 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, $p = 0.5$

Table 8: Parameters for Production Planning Problems

Sets	
K	set of product times
T	time
Indices	
k	index for a product type K
t	index for a time period in T
Deterministic Parameters	
c_{kt}	cost of production
p_{kt}	selling price
r_{kt}	maximum production capacity
Random Parameters	
$n_{it}(\omega)$	resource requirement to make products
$\min_{it}(\omega)$	minimum production requirement
$\max_{it}(\omega)$	maximum production level
Decision Variables	
m_{kt}	production quantities
I_{kt}	inventory levels
s_{kt}	sales quantities

Table 9: List of Parameters and Distributions

Parameter Name	Symbol	Distribution
resource requirement	$n_{it}(\omega)$	truncated Normal(3, 4) in interval [1,10]
minimum production	$\min_{it}(\omega)$	truncated Normal(200, 50) in interval $[50, 400]$
maximum production	$\max_{it}(\omega)$	truncated Normal(800, 50) in interval [600,1000]