An Integer Programming and Decomposition Approach to General Chance-Constrained Mathematical Programs

James Luedtke*

November 20, 2009

Abstract. We present a new approach for exactly solving general chance constrained mathematical programs having discrete distributions. Such problems have been notoriously difficult to solve due to nonconvexity of the feasible region, and currently available methods are only able to find provably good solutions in certain very special cases. Our approach uses both decomposition, to enable processing subproblems corresponding to one possible outcome at a time, and integer programming techniques, to combine the results of these subproblems to yield strong valid inequalities. Computational results on a chance-constrained two-stage problem arising in call center staffing indicate the approach works significantly better than both an existing mixed-integer programming formulation *and* a simple decomposition approach that does not use strong valid inequalities. Thus, the strength of this approach results from the successful merger of stochastic programming decomposition techniques with integer programming techniques for finding strong valid inequalities.

*Department of Industrial and Systems Engineering, University of Wisconsin, Madison, Wisconsin 53706 (jrluedt1@wisc.edu).

1 Introduction

We introduce a new approach for exactly solving general chance-constrained mathematical programs (CCMPs). A chance constraint states that the chosen decision vector should, with high probability, lie within a region that depends on a set of random variables. A generic CCMP can be stated as

$$\min\{f(x) \mid \mathbb{P}\{x \in P(\omega)\} \geq 1-\epsilon,\ x \in X\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables to be chosen to minimize $f(x)$, ω is a random vector and $P(\omega) \subseteq \mathbb{R}^n$ is a region parameterized by ω. The interpretation is that the region $P(\omega)$ is defined such that the event $x \notin P(\omega)$ is an undesirable outcome. The parameter $\epsilon \in (0,1)$ is a risk tolerance, typically small, that limits the likelihood of such an outcome. A problem with uncertain *linear* constraints is the special case of this problem in which $P(\omega) = \{x \mid T(\omega)x \geq b(\omega)\}$ and a two-stage problem with the possibility to take recourse after observing the random outcome has $P(\omega) = \{x \mid \exists y \text{ with } T(\omega)x + W(\omega)y \geq b(\omega)\}$. (In §5.1 we describe an example application.)

Our approach works for CCMP's with *discrete* (and finite support) distribution. Specifically, we assume that[1] $\mathbb{P}\{\omega = \omega^k\} = 1/N$ for $k = 1, \ldots, N$. We shall refer to the possible outcomes as *scenarios*. While this is certainly a restriction, recent results on using sample average approximations on problems with more general distributions [19] demonstrate that such finite support approximations, when obtained from a Monte Carlo sample of the original distribution, can be used to find good feasible solutions to the original problem *and* statistical bounds on solution quality. We will also assume that the sets $P_k := P(\omega^k)$ are polyhedra described by

$$P_k = \{x \in \mathbb{R}^n_+ \mid \exists y \in \mathbb{R}^d_+ \text{ with } T^k x + W^k y \geq b^k\}, \tag{2}$$

where $b^k \in \mathbb{R}^m$ and T^k and W^k are appropriately sized matrices. Note the special case with $d = 0$ yields a mathematical program with chance-constrained linear constraints having random coefficients: $\mathbb{P}\{T(\omega)x \geq b(\omega)\} \geq 1-\epsilon$. However, our approach can be extended to the more general case in which $P(\omega)$ is convex, provided we have oracles for separation and optimization over $P(\omega)$.

CCMPs have a long history dating back to Charnes, Cooper and Symonds [11]. The general version considered here, which enforces a *system* of constraints to be enforced with high probability, was introduced by Prékopa [24] (where chance constraints are referred to as probabilistic constraints). However, solution of the general problem (1) has remained computationally challenging for two reasons: the feasible region is generally not convex, and evaluating solution feasibility requires multi-dimensional integration. As discussed in the previous paragraph, the latter difficulty can be addressed by a sample-average approximation approach. However, this approach still requires a computationally efficient method for solving the resulting approximation problem which still has the form (1), except that the probability distribution is simplified to one with finite support.

Methods for obtaining provably good solutions for CCMPs have been successful in only a couple very special cases. If the chance constraint consists of a single row and all random coefficients are normally distributed [10, 9], then a deterministic (nonlinear and convex) reformulation is possible. If the randomness appears only in the right-hand side of the chance constraints (i.e. $P(\omega) = \{x \mid Tx \geq b(\omega)\}$) and the random vectors $b(\omega)$ have continuous and log-concave distributions, the resulting feasible region is convex and so nonlinear programming techniques can be used [24]. If the randomness appears only in the right-hand side and the distribution of $b(\omega)$ is discrete, then approaches based on certain "efficient points" of the random vector [5, 12] or on strong integer programming formulations [21, 26] have been proposed.

[1] The extension to more general discrete distributions of the form $\mathbb{P}\{\omega = \omega^k\} = p_k$, where $p_k \geq 0$ and $\sum_k p_k = 1$, is straightforward and is omitted to simplify exposition.

Very few methods are available for finding provably good solutions for CCMPs with the general structure we consider here, e.g. for problems having linear constraints with random coefficients or two stage problems as in (2). In [25], an approach based on an integer programming formulation (which we give in §2), strengthened with precedence constraints is presented. In more recent work, [28] presents a specialized branch-and-cut algorithm based on identification of irreducible infeasible sets of certain linear inequality systems. While these are important contributions, the size of instances that are demonstrated to be solvable with these approaches is very limited, in particular, because these approaches do not enable decomposition. In another recent important stream of research, a number of *conservative* approximations [4, 6, 8, 22, 23, 13, 14] have been studied that solve tractable (convex) approximations to yield feasible solutions to general CCMPs. However, these approaches do not say anything about the cost of the resulting solutions relative to the optimal, and tend to yield highly conservative solutions.

Our approach is important because it is an *exact* approach for solving problems with general chance constraints, and as we show in §5, has the potential to solve problems with high-dimensional random parameters and a large number of scenarios. The approach builds on the ideas of [20, 21] that were very successful for solving chance-constrained problems with *random right-hand side only* by developing a method to apply the same types of valid inequalities used there to the much more general case considered here. The other important aspect of our approach is that it enables decomposition of the problem into single scenario subproblems. This is important for solving CCMPs with discrete distributions because the problem size grows as the size of the support increases. The ability of this approach to solve large instances of this problem, even for the particular structure of the test problem described in §5.1, is significant because, until now, a major impediment to using a chance-constrained model has been the difficulty in solving such problems in all but a few very special cases. The approach we present here, when combined with the sample average approximation results of [19], has the potential to remove this barrier.

Decomposition has long been used for solving traditional two-stage stochastic programming problems, where the objective is to minimize the sum of costs of the first stage decisions and the expected costs of second-stage recourse decisions (see, e.g. [7, 29, 17]). For CCMPs, the only existing paper we are aware of that considers a decomposition approach is [27] which applies a decomposition approach to a chance-constrained formulation of an application insuring vital arcs in a critical path network. The decomposition idea is similar to what we present here, but the mechanism for generating cuts is significantly different: they use a convex hull reformulation (based on Relaxation-Linearization techniques) which involves "big-M" constants, likely leading to weak inequalities. In contrast, we combine the valid inequalities we obtain from different subproblems in a way that avoids the need for "big-M" constants and hence yields strong valid inequalities for the overall problem. As we will see in the computational results in §5, the use of strong valid inequalities makes a very significant difference beyond the benefits obtained from decomposition.

The remainder of this extended abstract is organized as follows. We start with an overview of the approach in §2. In §3 we describe how we generate strong valid inequalities, and in §4 we describe the actual decomposition branch-and-cut algorithm. Finally, we present preliminary computational results of the approach in §5.

2 Overview of the approach

To fix notation, and motivate the approach, we first describe a standard integer programming formulation of problem (1). We also make a couple assumptions that will assure this formulation is well-defined, and that also simplify exposition of the main results in the rest of the paper. We

assume without loss of generality that the sets P_k are non-empty for all $k \in \mathcal{N}$, since we could otherwise discard such a scenario and consider a problem with risk tolerance $\epsilon' = \epsilon - 1/N$. We also assume that the sets $P_k \cap X$ are compact for all $k \in \mathcal{N}$. Finally, for notational convenience we define the scenario index set $\mathcal{N} = \{1, \ldots, N\}$.

The standard mixed-integer programming formulation (e.g. [25]) uses a binary variable z_k for each scenario k, where $z_k = 0$ implies the constraints of scenario k should be satisfied:

$$\min\ f(x) \tag{3a}$$

$$\text{s.t. } T^k x + W^k y^k + z_k M_k \geq b^k, \quad k \in \mathcal{N}, \tag{3b}$$

$$\sum_{k=1}^{N} z_k \leq p \tag{3c}$$

$$x \in X, \quad z \in \{0,1\}^N, y^k \in \mathbb{R}^d_+,\ k \in \mathcal{N}. \tag{3d}$$

Here $p := \lfloor (1-\epsilon)N \rfloor$ and $M_k \in \mathbb{R}^m_+$ are sufficiently large to ensure that when $z_k = 1$, constraints (3b) are not active. On the other hand, when $z_k = 0$, constraints (3b) enforce $x \in P_k$. Thus, (3c), which is a rewritten and strengthened version of the constraint

$$\frac{1}{N}\sum_{k=1}^{N}(1 - z_k) \geq 1 - \epsilon,$$

successfully models the constraint $\mathbb{P}\{x \in P(\omega)\} \geq 1 - \epsilon$. Our approach is motivated by the desire to avoid the use of big-M constants as in (3b), which are likely to lead to weak lower bounds when solving a continuous relaxation of (3), and also to use decomposition to avoid explicit introduction of the constraints (3b) and recourse variables y^k which may make (3) very large-scale if N is large.

Our decomposition algorithm is based on a master problem that includes the original variables x, and the binary variables z. The constraints (3b) are enforced implicitly with cutting planes, similar in spirit to a Benders' decomposition approach. The key difference, however, is that given the mixed-integer nature of our master problem, we seek to add cutting planes that are strong. Specifically, we are interested in strong valid inequalities for the projection of the feasible region of (3) into the space of x and z variables. Specifically, we define this projection as

$$F = \left\{x \in X, z \in \{0,1\}^N \mid \exists y \in \mathbb{R}^{d\times N}_+ \text{ s.t. (3b)} - \text{(3c) hold}\right\}. \tag{4}$$

Note that $(x, z) \in F$ if and only if $x \in X$, $z \in \{0,1\}^N$ satisfies (3c), and $x \in P_k$ for any k with $z_k = 0$. Given this definition of F, we can then succinctly state a reformulation of the original chance-constrained problem (1) as:

$$\min\{f(x) \mid (x, z) \in F\}. \tag{5}$$

Our algorithm will solve this reformulation.

In §3 we describe how we obtain strong valid inequalities for F, for a *given* set of coefficients on the x variables. Then, in §4 we describe the decomposition approach which naturally suggests a choice for the coefficients on the x variables that leads to a convergent branch-and-cut algorithm. In our current implementation, we use only this approach for choosing these coefficients, but we believe that, depending on the problem structure, alternative approaches may be useful for yielding additional strong valid inequalities.

3 Generating strong valid inequalities

We now describe our procedure for generating strong valid inequalities of the form

$$\alpha x + \pi z \geq \beta \tag{6}$$

for the set F defined in (4), where $\alpha \in \mathbb{R}^n$, $\pi \in \mathbb{R}^N$, and $\beta \in \mathbb{R}$. We assume here that the coefficients α are given so our task is find π and β that make (6) valid for F. In addition, given a possibly fractional solution $(\hat{x}, \hat{z})$ our *separation* task is to find, if possible, π and β such that $(\hat{x}, \hat{z})$ violate the resulting inequality.

The approach is very similar to that used in [20, 21], which applies only to chance-constrained problems with random right-hand side. However, by exploiting the fact that we have assumed α to be fixed, we are able to reduce our significantly more general problem to the structure studied in [20, 21] and ultimately apply the same types of valid inequalities.

The first step in our procedure is to solve the following auxiliary "single scenario" problems:

$$h_k(\alpha) := \min\{\alpha x \mid x \in P_k \cap \bar{X}\}, \quad k \in \mathcal{N}. \tag{7}$$

Here $\bar{X} \subseteq \mathbb{R}^n$ is a relaxation of the set X, i.e. $\bar{X} \supseteq X$, chosen such that $P_k \cap \bar{X}$ is non-empty and compact, guaranteeing that the above optimal values exist. The choice of $\bar{X}$ represents a trade-off in time to compute the values $h_k(\alpha)$ and strength of the resulting valid inequalities. Choosing $\bar{X} = \mathbb{R}^n$ leads to a problem for calculating $h_k(\alpha)$ that has the fewest number of constraints (and presumably the shortest computation time), but choosing $\bar{X} = X$ will yield the strongest inequalities. In particular, if X is described as a polyhedron with additional integer restrictions on some of the variables, problem (7) would become a mixed-integer program and hence could be computationally demanding to solve, although doing so may yield significantly better valid inequalities.

Observe that calculation of the $h_k(\alpha)$ values decomposes by scenario and can be easily implemented in parallel. Having obtained the values $h_k(\alpha)$ for $k \in \mathcal{N}$, we then sort them to obtain a permutation σ of $\mathcal{N}$ such that:

$$h_{\sigma_1}(\alpha) \geq h_{\sigma_2}(\alpha) \geq \cdots \geq h_{\sigma_N}(\alpha).$$

Although the permutation depends on α, we will suppress this dependence to simplify notation. Our first lemma uses these values to establish a set of "base" inequalities that are valid for F, which we ultimately combine to obtain stronger valid inequalities.

Lemma 1. *The following inequalities are valid for F:*

$$\alpha x + (h_{\sigma_i}(\alpha) - h_{\sigma_{p+1}}(\alpha)) z_{\sigma_i} \geq h_{\sigma_i}(\alpha), \quad i = 1, \ldots, p. \tag{8}$$

Proof. Observe that for each $k \in \mathcal{N}$, by construction, the inequality $\alpha x \geq h_k(\alpha)$ is valid for P_k, and hence must be satisfied for any $(x, z) \in F$ such that $z_k = 0$. Thus, (8) holds for any $(x, z) \in F$ with $z_{\sigma_i} = 0$. On the other hand, inequality (3c) ($\sum_k z_k \leq p$) implies that for any $(x, z) \in F$ there must be an index $i \in \{1, \ldots, p+1\}$ with $z_{\sigma_i} = 0$, and hence $\alpha x \geq h_{\sigma_{p+1}}(\alpha)$ is a valid inequality for F. This implies that (8) is also satisfied for any $(x, z) \in F$ with $z_{\sigma_i} = 1$. □

Now, as was done in [20, 21], we can apply the *star* inequalities of [3], or equivalently in this case, the mixing inequalities of [15] to "mix" the inequalities (8) to obtain additional strong valid inequalities.

Theorem 2. *Let $T = \{t_1, t_2, \ldots, t_l\} \subseteq \{\sigma_1, \ldots, \sigma_p\}$ be such that $h_{t_i}(\alpha) \geq h_{t_{i+1}}(\alpha)$ for $i = 1, \ldots, l$, where $h_{t_{l+1}}(\alpha) := h_{\sigma_{p+1}}(\alpha)$. Then the inequality*

$$\alpha x + \sum_{i=1}^{l} (h_{t_i}(\alpha) - h_{t_{i+1}}(\alpha)) z_{t_i} \geq h_{t_1}(\alpha) \tag{9}$$

is valid for F.

These inequalities are strong in the sense that if we consider the set Y defined by

$$Y = \left\{ (y, z) \in \mathbb{R} \times \{0,1\}^p \mid y + (h_{\sigma_i}(\alpha) - h_{\sigma_{p+1}}(\alpha)) z_{\sigma_i} \geq h_{\sigma_i}(\alpha), \quad i = 1, \ldots, p \right\}$$

then the inequalities (9), with αx replaced by y, define the convex hull of Y [3]. Furthermore, the inequalities of Theorem 2 are facet-defining for the convex hull of Y (again with $y = \alpha x$) if and only if $h_{t_1}(\alpha) = h_{\sigma_1}(\alpha)$, which suggests that when searching for a valid inequality of the form (9), one should always include $\sigma_1 \in T$. In particular, the valid inequalities

$$\alpha x + (h_{\sigma_1}(\alpha) - h_{\sigma_i}(\alpha)) z_{\sigma_1} + (h_{\sigma_i}(\alpha) - h_{\sigma_{p+1}}(\alpha)) z_{\sigma_i} \geq h_{\sigma_1}(\alpha), \quad i = 1, \ldots, p. \tag{10}$$

dominate the inequalities (8) which can be obtained by aggregating (10) with the valid inequalities $z_{\sigma_1} \leq 1$ with a weight of $(h_{\sigma_1}(\alpha) - h_{\sigma_i}(\alpha))$ on the latter.

Theorem 2 presents an exponential family of valid inequalities, but given a point $(\hat{x}, \hat{z})$ separation of these inequalities can be accomplished very efficiently. In [3] an algorithm based on finding a longest path in an acyclic graph is presented that has complexity $O(p^2)$, and [15] gives an $O(p \log p)$ algorithm. We use the algorithm of [15].

4 Decomposition algorithm

We are now ready to describe the branch-and-cut decomposition algorithm. The algorithm will work with a master relaxation defined as follows:

$$\begin{aligned}
\mathrm{RP}^*(N_0, N_1, C) \;:=\; & \min\ f(x) && \text{(11a)}\\
& \text{s.t.}\ \sum_{k=1}^{N} z_k \leq p, \quad (x, z) \in C, \quad x \in X, \quad z \in [0,1]^N && \text{(11b)}\\
& z_k = 0, k \in N_0, z_k = 1,\ k \in N_1. && \text{(11c)}
\end{aligned}$$

Here N_0 is the set of binary variables currently fixed to 0, N_1 is the set of binary variables currently fixed to 1, and C is the relaxation defined by all the globally valid inequalities added so far when the relaxation is solved. At the root node in the search tree, we will have $N_0 = N_1 = \emptyset$ and $C = \mathbb{R}^n$.

In Algorithm 1, we describe a simple version of the proposed approach. The algorithm is a basic branch-and-bound algorithm, with branching being done on the binary variables z_k, with the only important difference being the way each node is processed (Step 2 in the algorithm). This step consists of a loop in which the current node relaxation (11) is solved repeatedly until no cuts have been added to the description of C or the lower bound exceeds the incumbent objective value U. Whenever an integer feasible solution $\hat{z}$ is found, and optionally otherwise, the cut separation routine SepCuts is called. The SepCuts routine *must* be called when $\hat{z}$ is integer feasible to check whether the solution $(\hat{x}, \hat{z})$ is truly feasible to the set F. The routine is optionally called otherwise to possibly improve the lower bound.

Algorithm 1: Simple version of branch-and-cut decomposition algorithm.

```
1  t ← 0, N_0(0) ← ∅, N_1(0) ← ∅, C ← ℝ^{n×N}, OPEN ← {0}, U ← +∞;
2  while OPEN ≠ ∅ do
3  |  Step 1: Choose l ∈ OPEN and let OPEN ← OPEN \ {l};
4  |  Step 2: Process node l;
5  |    repeat
6  |    |  Solve (11);
7  |    |  if (11) is infeasible then
8  |    |  |  CUTFOUND ← FALSE;
9  |    |  else
10 |    |  |  Let (x̂, ẑ) be an optimal solution to (11), and lb ← RP*(N_0(l), N_1(l), C);
11 |    |  |  if ẑ ∈ {0,1}^N then
12 |    |  |  |  CUTFOUND ← SepCuts(x̂, ẑ, C);
13 |    |  |  |  if CUTFOUND = FALSE then U ← lb;
14 |    |  |  else
15 |    |  |  |  CUTFOUND ← FALSE;
16 |    |  |  |  Optional: CUTFOUND ← SepCuts(x̂, ẑ, C);
17 |    |  |  end
18 |    |  end
19 |    until CUTFOUND ≠ TRUE or lb ≥ U ;
20 |  Step 3: Branch if necessary;
21 |    if lb < U then
22 |    |  Choose k ∈ 𝒩 such that ẑ_k ∈ (0,1);
23 |    |  N_0(t+1) ← N_0(l) ∪ {k}, N_1(t+1) ← N_1(l);
24 |    |  N_0(t+2) ← N_0(l), N_1(t+2) ← N_1(l) ∪ {k};
25 |    |  t ← t + 2;
26 |  end
27 end
```

The SepCuts routine, described in Algorithm 2, attempts to find strong violated inequalities using the approach described in §3. The key here is the method for selecting the coefficients α that are taken as given in §3. The idea is to consider all scenarios k such that $\hat{z}_k = 0$, so that the associated constraints $x \in P_k$ are supposed to be satisfied, and for such scenarios test whether indeed this holds. If $\hat{x} \in P_k$, then the condition that $\hat{z}_k = 0$ should imply $\hat{x} \in P_k$ is not violated. However, if $\hat{x} \notin P_k$, this contradicts the value of $\hat{z}_k$, and hence we seek to find an inequality that cuts off this infeasible solution. We therefore find an inequality, say $\alpha x \geq \beta$, that is facet-defining for P_k, and that separates $\hat{x}$ from P_k. We then use the coefficients α to generate one or more strong valid inequalities as derived in §3. While stated as two separate steps, the test of $\hat{x} \in P_k$ (line 3) and subsequent finding of a facet-defining inequality of P_k that cuts off $\hat{x}$ if not would typically be done together. For example, if we have an inequality description of P_k (possibly in a lifted space such as in (2)) then this can be accomplished by solving an appropriate linear program. If P_k has special structure (such as the constraint set of a shortest path problem) it may be accomplished with a specialized (e.g. combinatorial) algorithm.

Observe that, in line 2 of Algorithm 2, we actually test whether $\hat{x} \in P_k$ for any k such that $\hat{z}_k < 1$. To obtain a convergent algorithm, it would be sufficient to check only those k such that

Algorithm 2: Cut separation routine SepCuts($\hat{x}, \hat{z}, C$).

Data: $\hat{x}, \hat{z}, C$

Result: If one or more valid inequalities for F are found that are violated by $(\hat{x}, \hat{z})$, adds these to description of C and returns TRUE, else returns FALSE.

```
1  CUTFOUND ← FALSE;
2  for k ∈ 𝒩 such that ẑ_k < 1 do
3  |  if x̂ ∉ P_k then
4  |  |  Separate x̂ from P_k: Find an inequality αx ≥ β that is facet-defining for P_k such that
   |  |  αx̂ < β;
5  |  |  Using the coefficients α, find a violated inequality for F of the form (9) that is
   |  |  violated by (x̂, ẑ) and add this to the description of C;
6  |  |  CUTFOUND ← TRUE;
7  |  |  Optionally break;
8  |  end
9  end
10 return CUTFOUND;
```

$\hat{z}_k = 0$; we also optionally check k such that $\hat{z}_k \in (0, 1)$ in order to possibly generate additional strong valid inequalities. We now establish that Algorithm 1 solves (5).

Theorem 3. *Algorithm 1 terminates finitely, and at termination if $U = +\infty$, problem* (5) *is infeasible, otherwise U is the optimal value of* (5).

Proof (Sketch). The details of the proof are left out of this extended abstract. However, the first main point is that the algorithm terminates finitely because it is based on branching on a finite number of binary variables, and the processing of each node terminates finitely because the valid inequalities are derived from a finite number of facet-defining inequalities (for the polyhedral sets P_k). The second point is that the algorithm never cuts off an optimal solution because the branching never excludes part of the feasible region and only valid inequalities for the set F are added. The final point is that no solutions that are *not* in the feasible region F are accepted for updating the incumbent objective value U (in line 13 of the algorithm) because the SepCuts routine is always called for integer feasible solutions $\hat{z}$ and it can be shown that it is guaranteed to find a violated inequality if $(\hat{x}, \hat{z}) \notin F$. □

Aside from solving the master relaxation (11) the main work of Algorithm 1 happens within the SepCuts routine. An advantage of this approach is that most of this work is done for one scenario at a time (i.e. it is decomposed) and can be implemented to be done in parallel. In particular, checking whether $\hat{x} \in P_k$ (and finding a violated facet-defining if not) for any k such that $\hat{z}_k < 1$ can be done in parallel. The subsequent work of generating a strong valid inequality is likely dominated by calculation of the values $h_k(\alpha)$ for each k as in (7), which can also be done in parallel.

We have stated our approach in relatively simple form in Algorithm 1. However, as this approach is essentially a variant of branch-and-cut for solving a (particularly structured) integer programming problem, we can also use all the computational enhancements commonly used in such algorithms. In particular, using heuristics to find good feasible solutions early and using some sort of pseudocost branching [18], strong branching [2], or reliability branching [1] approach for choosing which variable to branch on would be important. In our implementation (described in §5.2) we have embedded the key cut generation step of our algorithm within the CPLEX commercial integer programming solver which has such enhancements already implemented.

In our definition of the master relaxation (11), we have enforced the constraints $x \in X$. If, X is a polyhedron and $f(x)$ is linear, (11) is a linear program. However, if X is not a polyhedron, suitable modifications to the algorithm could be made to ensure that the relaxations solved remain linear programming problems. For example, if X is defined by a polyhedron Q with integrality constraints on some of the variables, then we could instead define the master relaxation to enforce $x \in Q$, and then perform branching both on the integer-constrained x variables and on the z_k variables. Such a modification is also easy to implement within existing integer programming solvers. Note also that, in this case, Q would be a natural choice for the relaxation $\bar{X}$ of X used in §3 when obtaining the $h_k(\alpha)$ values as in (7).

5 Preliminary computational results

5.1 Call center staffing problem

We tested our approach on randomly generated instances of a call center staffing problem recently studied in [16]. In this problem the staffing levels of different types of available servers (x_i for $i = 1, \ldots, n$) must be set before knowing what the actual arrival rates of the customers will be. The routing of arriving customers to servers, however, can be done as the arrivals are observed. In [16], a static and fluid approximation of the second-stage dynamic routing problem is used in which servers are simply (fractionally) allocated to customers, leading to the following formulation:

$$\min\big\{cx \;\big|\; \mathbb{P}\{x \in P(\Lambda)\} \geq 1 - \epsilon,\ x \in \mathbb{R}^n_+\big\}$$

where $c \in \mathbb{R}^n_+$ represent the staffing costs, Λ is a m-dimensional random vector of arrival rates, and

$$P(\lambda) = \big\{x \in \mathbb{R}^n_+ \mid \exists y \in \mathbb{R}^{n\times m}_+ \text{ s.t. } \sum_{j=1}^{m} y_{ij} \leq x_i,\ i = 1, \ldots, n,\ \sum_{i=1}^{n} \mu_{ij} y_{ij} \geq \lambda_j,\ j = 1, \ldots, m\ \big\}. \quad (12)$$

Here μ_{ij} is the service rate of server type i when serving customer type j ($\mu_{ij} = 0$ if server type i cannot serve customer type j). This formulation aims to choose minimum cost staffing levels such that the probability of meeting quality of service targets is high.

When generating the test instances, we first generated the service rates, and the mean and covariance of the arrival rate vector, then generated the cost vector in such a way that "more useful" server types were generally more expensive, in order to make the solutions nontrivial. Finally, to generate specific instances with finite support, we sampled N joint-normally distributed arrival rate vectors independently using the fixed mean and covariance matrix for various sample sizes N. In all our test instances we use $\epsilon = 0.1$ as the risk tolerance.

We see that this problem has the two-stage structure given in (2), and hence available methods for finding exact solutions (or even any solution with a bound on optimality error) are very limited. However, we do want to point out that the form of $P(\lambda)$ still possesses some special structure in that the second-stage constraints have no random coefficients (i.e. in the form of (2) the matrices T^k and W^k do not vary with k). In addition, the constraints $x \in X$ are very simple for this problem; we simply have $X = \mathbb{R}^n_+$. Thus, while this test problem is certainly beyond the capabilities of existing approaches, it is not yet a test of the algorithm in the most general settings.

Technically, this problem does not satisfy our assumptions given in §2 because the sets $\mathbb{R}^n_+ \cap P(\lambda)$ are not bounded. However, our approach really only requires that the optimal solutions to (7) always exist for any coefficient vector α of a facet-defining inequality for $P(\lambda)$. As valid inequalities for $P(\lambda)$ necessarily have non-negative coefficients, this clearly holds.

5.2 Implementation details

We implemented our approach within the commercial integer programming solver CPLEX 11.2. The main component of the approach, separation of valid inequalities of the form (9), was implemented within a cut callback that CPLEX calls whenever it has finished solving a node (whether the solution is integer feasible or not) and also after it has found a heuristic solution. In the feasibility checking phase of the SepCuts routine (line 2) we searched for k with $\hat{z}_k < 1$ and $x \notin P_k$ in increasing order of $\hat{z}_k$ (so, in particular we always first check the scenarios k with $\hat{z}_k = 0$). For the *first* such k we find (and only the first) we add *all* the violated valid inequalities of the form (10) as well as the single most violated inequality of the form (9). Our motivation for adding the inequalities (10) is that they are sparse and this is a simple way to add additional valid inequalities in one round; we found that doing this yielded somewhat faster convergence.

5.3 Results

We compared our algorithm against the Big-M formulation (3) (with the M values chosen as small as possible) and also against a basic decomposition algorithm that does not use the strong valid inequalities of §3. We compare against this simple decomposition approach to understand whether the success of our algorithm is due solely to the decomposition, or whether the strong inequalities are also important. The difference between the basic decomposition algorithm and the strengthened version is in the type of cuts that are added in the SepCuts routine. Specifically, in the case of an uncertainty set P_k of the form (12), if we find a scenario k with $\hat{z}_k = 0$, and a valid inequality $\alpha x \geq \beta$ for the set P_k that is violated by $\hat{x}$, the basic decomposition algorithm simply adds the inequality

$$\alpha x \geq \beta z_k.$$

It is not hard to see that when the sets P_k have the form (12), this inequality is valid for F because $x \geq 0$ and any valid inequality for P_k has $\alpha \geq 0$. Furthermore, this inequality successfully cuts off the infeasible solution $(\hat{x}, \hat{z})$.

n	m	N	Big-M	Basic Decomp	Strong Decomp
10	20	500	23.0%	1752[a]	2.9
		1000	27.3%	5.5%	17.3
		2000	-	10.1%	143.4
20	30	1000	28.9%[b]	7.3%	11.8
		2000	-	16.7%	27.5
		3000	-	24.3%	73.9
40	50	1000	-	16.2%	65.3
		2000	-	24.1%	190.9
		3000	-	28.7%	256.3

[a] Average based on nine instances that solved in time limit.
[b] Gap for one instance, remaining nine instances failed.

Table 1: Results on call center staffing instances; solution time (sec) or final optimality gap (%).

Table 1 presents the results of these three approaches for varying problem size in terms of number of agent types (n), number of customer types (m) - which is also the dimension of the random vector Λ, and number of scenarios N. These tests were done with a time limit of one hour. Unless stated otherwise, each entry is an average over ten randomly generated samples from

the same base underlying instance (i.e. the instance is fixed, but ten different samples of the N scenarios are taken). The big-M formulation (3) only successfully solves the LP relaxation and finds a feasible solution for the two smallest instance sizes. The entries '-' in the other cases mean that either no solution was found in the time limit, or that the LP relaxation did not solve in the time limit. For the largest instances, CPLEX failed with an out-of-memory error before the time limit was reached. Using the basic decomposition approach makes a significant improvement over the big-M formulation in that feasible solutions are now found for all instances. However, only the smallest of the instances (and only 9 of 10 of them) could be solved to optimality within the time limit, and the larger instances had very large optimality gaps after the limit. Combining decomposition with strong valid inequalities ("Strong Decomp" in the table) we are able to solve all the instances to optimality in an average of less than five minutes.

To understand these results a little better, we present in Table 2 the root gaps (relative to the optimal values) after all cuts have been added for the two decomposition approaches. We also present the average number of nodes processed in each approach (to the time limit for the basic approach, and to optimality for our approach). It is clear that the strong valid inequalities lead to very strong relaxations for this particular problem, and hence almost no nodes need to be explored. In comparison, for the smallest instance size, in which the basic decomposition approach can solve most of the instances, the average number of nodes required is over 20,000. (The smaller number of nodes for the larger instances merely reflects that fewer could be processed in the time limit.)

			Root gap (%)		Nodes	
n	m	N	Basic	Strong	Basic	Strong
10	20	500	20.3%	0.00%	22969	0
		1000	20.1%	0.01%	15034	0
		2000	19.5%	0.01%	4641	6.7
20	30	1000	20.2%	0.00%	6271	0
		2000	19.9%	0.01%	557	0
		3000	20.4%	0.00%	399	0.1
40	50	1000	20.1%	0.00%	878	0.3
		2000	20.7%	0.00%	101	0.1
		3000	20.7%	0.00%	13	1

Table 2: Average root gaps and nodes for decomposition approaches.

6 Discussion

We have presented a promising approach for solving general CCMPs, although additional computational tests are needed on problems having more general structures than the test problem we considered. The approach uses both decomposition, to enable processing subproblems corresponding to one scenario at a time, and integer programming techniques, to yield strong valid inequalities. From a stochastic programming perspective, it is not surprising that decomposition is necessary to yield an efficient algorithm, as this is well-known for traditional two-stage stochastic programs. From an integer programming perspective, it is not surprising that using strong valid inequalities has an enormous impact. The approach presented here represents a successful merger of these approaches to solve CCMPs.

Acknowledgments. The author thanks Shabbir Ahmed for the suggestion to compare the presented approach with a basic decomposition algorithm.

References

[1] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Oper. Res. Lett.*, 33:42–54, 2004.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding cuts in the TSP. Technical Report 95-05, DIMACS, 1995.

[3] A. Atamtürk, G. L. Nemhauser, and M. W. P. Savelsbergh. The mixed vertex packing problem. *Math. Program.*, 89:35–53, 2000.

[4] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Program.*, 88:411–424, 2000.

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

[6] D. Bertsimas and M. Sim. The price of robustness. *Oper. Res.*, 52:35–53, 2004.

[7] J. Birge and F. Louveaux. *Introduction to stochastic programming.* Springer, New York, 1997.

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

[9] G. Calafiore and L. El Ghaoui. On distributionally robust chance-constrained linear programs. *J. Optim. Theory Appl.*, 130:1–22, 2006.

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

[11] A. Charnes, W. W. Cooper, and G. H. Symonds. Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil. *Manage. Sci.*, 4:235–263, 1958.

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

[13] E. Erdoğan and G. Iyengar. Ambiguous chance constrained problems and robust optimization. *Math. Program.*, 107:37–61, 2006.

[14] E. Erdoğan and G. Iyengar. On two-stage convex chance constrained problems. *Math. Meth. Oper. Res.*, 65:115–140, 2007.

[15] O. Günlük and Y. Pochet. Mixing mixed-integer inequalities. *Math. Program.*, 90:429–457, 2001.

[16] I. Gurvich, J. Luedtke, and T. Tezcan. Call center staffing with uncertain arrival rates: a chance-constrained optimization approach. Technical report, 2009. Submitted to *Management Science*.

[17] J. L. Higle and S. Sen. Stochastic decomposition: an algorithm for two-stage stochastic linear programs. *Math. Oper. Res.*, 16:650–669, 1991.

[18] J. Linderoth and M. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS J. Comput.*, 11:173–187, 1999.

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

[20] J. Luedtke, S. Ahmed, and G. Nemhauser. An integer programming approach for linear programs with probabilistic constraints. In M. Fischetti and D. Williamson, editors, *IPCO 2007*, Lecture Notes in Comput. Sci., pages 410–423, Berlin, 2007. Springer-Verlag.

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

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

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

[24] A. Prékopa. On probabilistic constrained programmming. In H. W. Kuhn, editor, *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138, Princeton, N.J., 1970. Princeton University Press.

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

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

[27] S. Shen, J. Smith, and S. Ahmed. Expectation and chance-constrained models and algorithms for insuring critical paths. Submitted for publication, 2009.

[28] M. Tanner and L. Ntaimo. IIS branch-and-cut for joint chance-constrained programs with random technology matrices. Under second review, 2008.

[29] R. Van Slyke and R. J. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math*, 17:638–663, 1969.