NEW SOLUTION METHODS FOR JOINT CHANCE-CONSTRAINED STOCHASTIC PROGRAMS WITH RANDOM LEFT-HAND SIDE

A Dissertation

by

MATTHEW WILEY TANNER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2009

Major Subject: Industrial Engineering

NEW SOLUTION METHODS FOR JOINT CHANCE-CONSTRAINED STOCHASTIC PROGRAMS WITH RANDOM LEFT-HAND SIDE

A Dissertation

by

MATTHEW WILEY TANNER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Lewis Ntaimo
Committee Members,	Guy Curry
	Sergiy Butenko
	Faming Liang
Head of Department,	Brett Peters

May 2009

Major Subject: Industrial Engineering

ABSTRACT

New Solution Methods for Joint Chance-Constrained Stochastic Programs with Random Left-Hand Side. (May 2009)

Matthew Wiley Tanner, B.S.E., Princeton University

Chair of Advisory Committee: Dr. Lewis Ntaimo

We consider joint chance-constrained programs with random lefthand sides. The motivation of this project is that this class of problem has many important applications, but there are few existing solution methods. For the most part, we deal with the subclass of problems for which the underlying parameter distributions are discrete. This assumption allows the original problem to be formulated as a deterministic equivalent mixed-integer program.

We first approach the problem as a mixed-integer program and derive a class of optimality cuts based on irreducibly infeasible subsets of the constraints of the scenarios of the problem. The IIS cuts can be computed efficiently by means of a linear program. We give a method for improving the upper bound of the problem when no IIS cut can be identified. We also give an implementation of an algorithm incorporating these ideas and finish with some computational results.

We present a tabu search metaheuristic for finding good feasible solutions to the mixed-integer formulation of the problem. Our heuristic works by defining a *sufficient* set of scenarios with the characteristic that all other scenarios do not have to be considered when generating upper bounds. We then use tabu search on the one-opt neighborhood of the problem. We give computational results that show our metaheuristic outperforming the state-of-the-art industrial solvers.

We then show how to reformulate the problem so that the chance-constraints are monotonic functions. We then derive a convergent global branch-and-bound algo-

rithm using the principles of monotonic optimization. We give a finitely convergent modification of the algorithm. Finally, we give a discussion on why this algorithm is computationally ineffective.

The last section of this dissertation details an application of joint chance-constrained stochastic programs to a vaccination allocation problem. We show why it is necessary to formulate the problem with random parameters and also why chance-constraints are a good framework for defining an optimal policy. We give an example of the problem formulated as a chance constraint and a short numerical example to illustrate the concepts.

To my parents, Steven and Lisa

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Lewis Ntaimo for all the support and advice that he has given me over the last few years. I would like to thank the Industrial and Systems Engineering Department here at Texas A&M for all the support they have given me while I was studying for my doctorate. I would especially like to thank Judy for taking care of us graduate students all these years. Finally, I would like to thank my friends for making graduate school a fun time.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

A. Problem Definition

We consider joint chance-constrained stochastic programming problems allowing for the left-hand side of the constraints to be random. Instead of requiring feasibility almost surely, a chance-constraint within a stochastic program must be satisfied at least with probability α. A joint chance-constraint introduces dependency into this concept, requiring that a subset of constraints in the formulation are satisfied at least with probability α. Chance-constraints are used to model systems for which a certain quality-of-service is required, or for when a problem has extreme cases for which satisfying the chance-constraints for all possible parameter values is either too expensive or impossible.

The goal of mathematical programming is to identify an optimal solution for a problem. Traditionally it is assumed that all problem data are known precisely, which implies that an optimal solution for the problem is truly the best (Bazaara et al., 1990). Unfortunately, in most cases, problem data cannot be known exactly and instead the data can take a range of values or perhaps can be defined by a probability distribution. Sensitivity analysis can be used to identify the range of problem parameter values for which an optimal solution to a problem remains optimal. If this range encompasses the entire range of possible parameter values, then the optimality of the solution can be guaranteed. However, if the parameter values vary widely, then there may be possible parameter values for which the optimal solution is suboptimal or even infeasible (Bazaara et al., 1990).

This dissertation follows the style of *IIE Transactions*

Stochastic programming is an extension of mathematical programming in which the assumption that all data are known is relaxed; instead, a subset of the parameter values of the problem are characterized by probability distributions (Ruszczyński and Shapiro, 2003). The goal of a mathematical programming problem is to identify an optimal solution, where optimality is defined in terms of a cost function to be minimized or maximized. The most popular measure of optimality in stochastic programming is in terms of the expected value of the objective function, but other risk functions can also be used (Szegö, 2002). Except for a few cases, solving stochastic programs with continuously distributed parameters is extremely difficult, so in most cases parameters are given discrete distributions or the continuous distributions are discretized through sampling. A realization ω of the vector of random variables $\tilde{\omega}$ of the problem is known as a scenario and is defined on the sample space Ω. The decision variables of the mathematical programs are given by the vector $x \in \mathcal{R}^n$. The types of information that can be gained by optimizing a general, random function $z(x, \omega)$ in the stochastic programming framework can be described as follows:

DEFINITION A.1. The stochastic programming solution (SPP) is the minimum expected value of the function $z(x, \omega)$ in terms of the random variables $\tilde{\omega}$. In other words,

$$SPP = \min_x E_{\tilde{\omega}}\big[z(x, \omega)\big]. \tag{1.1}$$

Besides the optimal objective value and optimal decision variables, other useful statistics can be computed to show the effect of parameter uncertainty on the policies suggested by the model. Assuming that the random parameters have discrete distributions, the problem has a finite number of scenarios. The "wait-and-see" solution assumes that the decision maker can see the realization of the random variables before any decisions need to be made.

DEFINITION A.2. The "wait-and-see" solution (WS) given in equation (1.2) is the expected value of the solutions found by assuming that decisions are made after the random parameters are realized.

$$WS = E_{\tilde{\omega}}\big[\min_{x} z(x, \omega)\big] \tag{1.2}$$

DEFINITION A.3. The value of perfect information (VPI) is given by the equation

$$VPI = SPP - WS. \tag{1.3}$$

VPI measures how much improvement can be gained if the true values of the parameters are known. This information can be used to decide how much effort should be expended trying to improve estimates of the parameters.

Another important statistic found by analyzing the stochastic model is the value of including uncertainty in the formulation. To compute this value, a deterministic linear program is set up using the expected values of the random parameters as deterministic parameters. This deterministic linear program is solved to find optimal decision variable values.

DEFINITION A.4. The expected result of using the expected value solution (EEV) is the expected objective value of the solution found with the mean point estimates of the random parameters. EEV is given by

$$EEV = E_{\tilde{\omega}}\big[z(\bar{x}(\bar{\omega}), \omega)\big]. \tag{1.4}$$

Since the goal is to lower the probability of a disease spreading widely, the benefit of solving these problems as a stochastic program is in the added robustness of the

optimal value. In this case, the EEV shows the probability of failure for the vaccination policy found by using point estimates of the parameter values and so gives an estimate of the gain in solution robustness due to including parameter uncertainty in the problem formulation.

It is assumed that the random variables of a problem have known distributions, and that there is a set of decision variables that must be decided before the values of the random parameters are realized. Two popular stochastic programming models are chance-constrained programs and stochastic programs with recourse. The basic formulation for a joint chance-constrained program is given by equations (1.5a)-(1.5c).

$$\begin{aligned} \text{Min} \quad & f(x) && \text{(1.5a)} \\ \text{s.t.} \quad & \mathbb{P}\big\{\omega \in \Omega : g_i(x,\omega) \leq 0,\ i = 1 \dots m\big\} \geq \alpha, && \text{(1.5b)} \\ & x \in \mathcal{X}. && \text{(1.5c)} \end{aligned}$$

In formulation (1.5a) - (1.5c), $x \in \mathbb{R}^n$ is the decision variable vector, $f(x)$ is the objective function, $g_i(x,\omega),\ i = 1 \dots m$ are real valued functions that make up the random constraints within the joint chance-constraint (1.5b), and $\mathcal{X}$ is the feasible space of the decision variables. In this formulation, individual outcomes 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 with probability less than $\alpha \in [0,1]$.

Besides the possible nonconvexity of the objective function and feasible space, there are several reasons specific to chance constraints that formulation (1.5) is difficult to solve. The first is that in the case of general probability distributions for the random data, evaluating whether a point satisfies the joint chance-constraints (1.5c)

involves the solution of a multi-dimensional integral. So even finding a feasible point for the problem may not be possible. The second main difficult is that even under strong assumptions on the probability distributions, the feasible region of problem (1.5) may be nonconvex.

Various assumptions about the distribution functions of the random parameters must be made in order to formulate computationally tractable problems with chance constraints. Depending on where in the problem the randomness is situated, it can be shown that the problem is convex for various probability distributions of the random data and various constraint function types. In this case, convex programming methods can be used to find optimal solutions. In the case of discrete probability distributions, most solution methods use integer programming or other discrete programming techniques.

B. Computation Details and Test Instances

To test the effectiveness of the methods given in this thesis, 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 Chapter VI. The problem formulation and the random parameter distributions that we assumed can be found in Appendix A. Problem size information can be found in Table I. 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 B, while the Problem size information is given in Table II. For each of these two sets, we generated 5 problems of each size using random sampling.

In both Table I and Table II, the first column gives the names of the test instance.

Table I. Problem Sizes for Vaccination Test Instances

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

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 with our algorithms, For all of our computational tests, we used a Dell Optiplex GX620 computer with a PentiumD 3.0 GHz processor and 4.0 GB RAM. We implemented our algorithms using the CPLEX 9.0 callable library within the C++ environment.

Table II. Problem Sizes for Production Planning 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

C. Thesis Outline

This dissertation focuses on mathematical programming solution techniques for chance-constrained stochastic programs with with random constraint functions. Not all of the results presented in this dissertation will be valid for the general formulation (1.5). We will make assumptions at the beginning of each chapter. One of the main assumptions that we will make in most cases is that the random parameters of the problem have discrete distributions. Studying discrete parameter distributions is particularly important because it allows the use of sampling to generate computationally tractable problems for cases in which assumptions on the probability distributions that make the problem convex are too stringent. The discrete distributions allow the problem to be reformulated as a deterministic equivalent mixed-integer programming problem.

The integer programming reformulations of problem (1.5) tend to have very weak

linear programming relaxations and so traditional branch-and-cut methods are often not effective at solving them. Also, mixed-integer programming problems can not be solved effectively for general objective and constraint functions. We will first present novel integer programming techniques that can be used to solve the IP formulation of problem (1.5) more effectively in the case where the objective and constraint functions are linear. We will then give a heuristic method for the problem that is valid for more general problem with discretely distributed parameters. We will also present a branch-and-bound algorithm that branches on the continuous decision variables of the problem using a monotonic reformulation of the chance constraints. This algorithm can be applied to problems with continuously distributed parameters. This dissertation includes computational results of these methods on several test problem sets that are much larger than have previously been solved.

Chapter II gives extensive background on chance-constrained stochastic programs. It begins with a description of a few different applications for chance-constrained programming from the literature. It then continues with a short introduction to the various types of chance-constrained programs that have been studied. We give a review of the various solution techniques for the different classes of problems. The chapter finishes with a more extensive review of results on chance-constrained stochastic programs with discretely distributed random parameters, including a discussion of where this dissertation fits into the literature.

Chapter III presents a new class of optimality cuts called IIS cuts that we have derived for solving joint chance-constrained programs in a branch-and-cut framework in the case where the constraints are linear. These cuts are derived using irreducibly infeasible subsets of scenarios which can be identified using linear programming. We prove that the derived cuts do not cut off all optimal solutions and present a separating algorithm. We also give a routine for quickly improving the best feasible solution

found for the problem in the case when no IIS cut can be found. The methods are only valid for problems with discretely distributed random parameters. We finish the chapter with a description of our implementation of the algorithm and computational results that show the effectiveness of the methods.

In Chapter IV, we reformulate the problem as finding subsets of scenarios such that the sum of the probabilities of those scenarios is greater than the reliability parameter α. This reformulation implies a finite feasible solution space Φ. Given a solution $C \in \Phi$, we define a neighborhood $\mathcal{N}(C)$. We then describe a method for quickly searching $\mathcal{N}(C)$ for improving solutions, and give a random tabu search metaheuristic for searching our solution space. This method is valid for any problem with discretely distributed random parameters. We finish the chapter with some computational results from our heuristic.

In Chapter V, we give a novel branch-and-bound strategy for solving a class of joint chance-constrained problems. Our method requires the functions that make up the chance-constraint to be either all increasing or all decreasing, an assumption that is satisfied by all chance-constrained programs with linear constraints. We also require an oracle to evaluate the feasibility of the chance constraints, easily done in the case of discrete distributions. The branching is done on the decision variables of the problem. We give some possible branching rules, prove convergence of the algorithm in the general case of continuous distribution functions for the random parameters, and finite ϵ-convergence given some extra assumptions. The chapter concludes with a discussion of why the algorithm is computationally infeasible.

Chapter VI is a detailed description of the application that inspired this dissertation. The chapter describes how chance-constrained programming can be applied to the problem of optimal vaccine allocation. Stochastic programming is a particularly apt framework for vaccine allocation because the parameters of disease spread models

that underly the problem of distributing vaccines are particularly hard to estimate. The chapter gives a few different formulations of the vaccine allocation problem as a chance-constrained program and also describes the class of disease models for which stochastic programming can be used to define a vaccination program.

Chapter VII finishes with some conclusions and future work on this subject. We also include some appendices that give the problem formulations and parameter data that we used in our computational studies.

CHAPTER II

BACKGROUND

This chapter gives an introduction to the history and current status of research into chance-constrained stochastic programs. There are a large number of potential applications for chance constrained programming including maintaining proper aquifer levels (Curry et al., 1973; Morgan et al., 1993), maintaining a continuous distillation process (Henrion and Möller, 2003), optimizing a portfolio (Pagnoncelli et al., 2008), air quality management with a required reliability level (Watanabe and Ellis, 1993; An and Eheart, 2007), and optimal scheduling (Tayur et al., 1995). Chapter VI introduces an application of chance-constraints to the problem of finding optimal vaccination policies. Obviously, this is not a complete list of all applications of chance-constrained programming but it does give an idea of the wide range that have been studied.

There are several important types of chance-constraints which drive the various solution techniques that have been developed to tackle them. Table III includes sections on each of these different types and lists the references for each of them, the rest of the section will explain the papers in detail. The first column lists the categories: problems with continuous distributions, discrete distributions, algorithms that find approximate results, application papers, papers on convex functions, and robust optimization papers applied to chance constrained programming. The second column gives the type of random distribution for the papers: fixed technology matrix, random technology matrix, and either. The final column lists the papers for each of the combinations of categories. The rest of the chapter summarizes and highlights the important results in chance-constrained programming.

Table III.: Chance-Constrained Programming Papers

Category	Randomness	Papers
Continuous Distributions	Fixed LHS	Charnes and Cooper (1959)
		Miller and Wagner (1965)
		Prékopa (1971)
		Henrion and Strugarek (2006)
		Cheon et al. (2006)
Continuous Distributions	Random LHS	Kataoka (1963)
		Jagannathan (1974)
		Prékopa (1974)
		Watanabe and Ellis (1993)
		Lagoa et al. (2005)
		Henrion and Strugarek (2006)
Discrete Distributions	Fixed LHS	Prékopa (1990)
		Sen (1992)
		Dentcheva et al. (2000)
		Dentcheva et al. (2002)
		Beraldi and Ruszczyński (2002b)
		Beraldi and Ruszczyński (2002a)
		Cheon et al. (2006)
		Saxena (2007)
		Luedtke et al. (2007)
Discrete Distributions	Random LHS	Morgan et al. (1993)
		Tayur et al. (1995)

TABLE III continued

Category	Randomness	Papers
		Ruszczyński (2002)
		Pang and Leyffer (2004)
		Tanner and Ntaimo (2008)
Approximations	Any	Pintér (1989)
		Iwamura and Liu (1996)
		Aringhieri (2004)
		Nemirovski and Shapiro (2004)
		Calafiore and Campi (2005)
		Nemirovski and Shapiro (2006)
		Calafiore and Campi (2006)
		Haneveld and Vlerk (2006)
		An and Eheart (2007)
		Luedtke and Ahmed (2007)
		Pagnoncelli et al. (2008)
		Tanner and Beier (2008)
Applications	Any	Curry et al. (1973)
		Morgan et al. (1993)
		Watanabe and Ellis (1993)
		Henrion and Möller (2003)
		Tanner et al. (2008)
Convex Functions	Any	Nemirovski and Shapiro (2004)
		Dentcheva et al. (2004)
		Erdogan and Iyengar (2005)

TABLE III continued

Category	Randomness	Papers
Robust Optimization	Any	Calafiore and Campi (2006)
		Chen et al. (2007)
		Parpas et al. (2007)

A. Convexity Results

Most early results in the field of chance constrained programming deals with deriving conditions for which the structure of the constraint functions and the probability distributions of the random parameters cause the feasible space of the problem to be convex. In these cases, standard convex programming techniques can be used to determine optimal solutions. Due to the difficulty of evaluating the feasibility of the chance constraints in the case of general probability distributions for the parameters, pretty much all exact solution methods for chance constrained programs with continuously distributed parameters are limited to cases where the problem is convex. All of these results require the assumption that the chance constraints of the problem are linear functions.

The earliest formulation of chance constraints within a stochastic programming framework was given by Charnes and Cooper (1959). They presented a model with single chance constraints (e.g. $m = 1$) and fixed left-hand side. Using these assumptions, they showed that the problem can be reformulated as a deterministic nonlinear programming problem equivalent by taking the inverse of the distribution

of the random righthand size. With known distributions, this transformation is linear and results in an efficiently solvable problem. For the case of single chance constraints and random left-hand sides, Kataoka (1963) showed that the problem is convex when the left-hand side is independently normally distributed and $\alpha \geq 0.5$.

Problems with joint chance-constraints ($m > 1$) were introduced by Miller and Wagner (1965). They focused on problems with fixed technology matrices. They showed that when the random righthand side parameter distributions are independent, the logarithmic transforms of the products of the CDFs are convex and hence computationally tractable for a large class of probability distributions. In the case when the random righthand sides are dependent, Prékopa (1971) showed that a convex deterministic equivalent problem can be formulated when the righthand sides have log-concave distributions. This class includes such distributions as multi-variate normal and multi-variate beta.

Problems with random left-hand side are significantly more difficult to solve than are problems with randomness just in the righthand side vector. Early results on the convexity of this case are given by Jagannathan (1974) who showed that if the random coefficients of the technology matrix are independent and normally distributed, then the problem can be reformulated as a parametric convex program. Prékopa (1974) showed that problem is convex if all the covariance and cross-covariance matrices of the columns or rows of the normally distributed parameters of the left-hand side are proportional to each other. Later, Watanabe and Ellis (1993) reformulated the problem as a deterministic nonlinear programming problem for the more general case that the rows are allowed to be dependent. They gave an algorithm to find upper bounds on the solution heuristically. Finally, Henrion and Strugarek (2006) gave conditions for which the problem is convex as long as the problem rows are independent and the random coefficients are normally distributed. For the restriction to single chance

constraints Lagoa et al. (2005) showed that the problem is convex as long as both the left-hand and righthand sides have symmetric log-concave distributions.

In cases for which the random parameters do not have normal distributions in the left-hand side, log-concave distributions in the righthand side, or the constraint functions are nonlinear there have not been any conditions found for which problem (1.5) is convex. Therefore, there has been a lot of interest in nonlinear and integer programming techniques for solving more general instances of the problem.

B. Discrete Distributions, Fixed Left-Hand Side

An important branch of research in chance-constrained programs is on problems with discretely distributed parameters. The main advantage of the discrete distribution assumption is that it allows problems to be reformulated as deterministic equivalent integer programs. Many problems have parameters with distributions that do not fit the convexity assumptions summarized previously or else distributions that can only be estimated empirically. So the only hope of solving the problem comes from discretizing the distribution through sampling and then solving the deterministic equivalent integer programming problem. The results in this subsection focus on problems with linear functions in the chance constraints.

A wide variety of stochastic linear chance problems with discretely distributed parameters and fixed left-hand sides use the enumeration of p-efficient points to aid in solution. Defining $F(\cdot)$ as the cumulative distribution function of the random parameters and defining $z' \leq z$ if $z'_i \leq z_i, \quad \forall i = 1 \dots n$. A p-efficient point z is one such that $F(z) \geq p$ but there is no possible $z' \leq z$ such that $z' \neq z$ and $F(z') > p$. The main use of p-efficient points is that they can be enumerated efficiently and guarantee that the optimal solution to the problem is within the set of points greater than the

p-efficient points of the problem.

Prékopa (1990) gave the earliest example of using p-efficient points to solve a linear chance-constrained problems with fixed left-hand side. He did not address the problem of identifying p-efficient points but given the full enumeration of them $\mathcal{E}$, he introduced the reformulation that rewrites the chance-constraint as a disjunctive program.

$$\begin{aligned}
\min \quad & c^\top x && \text{(2.1a)}\\
\text{s.t.} \quad & Ax \leq b && \text{(2.1b)}\\
& Tx - y = 0 && \text{(2.1c)}\\
& y \in \bigcup_{z_l \in \mathcal{E}} \{H_l := y | y \geq z_l\} && \text{(2.1d)}\\
& x \geq 0 && \text{(2.1e)}
\end{aligned}$$

Sen (1992) derived valid inequalities using disjunctive programming for formulation (2.1). For most problems the number of p-efficient points is too large for complete enumeration, therefore Dentcheva et al. (2000) addressed the problem of identifying useful p-efficient points. They also give a method for bounding the optimal objective value of the chance-constrained program under the assumption of r-concave discrete distributions for the parameters.

More recent work using p-efficient points has focused on extensions to the problem. Beraldi and Ruszczyński (2002a) gave a branch-and-bound algorithm for problem (2.1) in the case where the decision variables are integer valued. Dentcheva et al. (2002) also analyzed problems with integer decision variables, giving valid upper and lower bounds on the optimal objective values that can be computed using nonlinear programming formulations. Dentcheva et al. (2004) extends the formulation to the

case of general convex constraint functions. Their method also depends on solving nonlinear programming reformulations. The probabilistic set covering problem is analyzed and solved using p-efficient points in both Beraldi and Ruszczyński (2002b) and Saxena (2007).

More recent results have branched out from the focus on identifying p-efficient points and solving formulation (2.1). Cheon et al. (2006) showed that the feasible region of a chance-constraint with fixed left-hand side is a reverse normal set. They then use methods from monotonic optimization to develop a branch-reduce-cut algorithm with the branching on the continuous variables y as formulated in problem (2.1). More details on this approach are given in Chapter V which uses similar ideas to approach the case of random left-hand side. Finally, Luedtke et al. (2007) reformulates problem (1.5) as a mixed-integer program. The special structure of this problem allow for strong valid inequalities based on mixing cuts to be derived for the problem. They also derive strengthened formulations for the problem. Computational results show that this approach is particularly promising.

C. Discrete Distributions, Random Left-Hand Side

Only a few papers have been published studying exact solution methods for the important case of problem (1.5) in which the left-hand side of the problem is allowed to be random and the parameter distributions do not make the problem convex. For this case, most results are for problems with $f(x)$, $g_i(x)$ for all $i = 1 \dots m$ as linear functions and $\mathcal{X}$ (see formulation 1.5) defined by linear functions and possibly integer requirements on the decision variables as well as discrete distributions for the parameters. These assumptions allow the problem to be formulated as a mixed-integer linear program and research effort has focused on mixed-integer programming

solution techniques. The mixed-integer formulation is given below.

$$\begin{aligned}
\min \quad & c^\top x && \text{(2.2a)}\\
\text{s.t.} \quad & T(\omega)x - M_\omega e z_\omega \leq r(\omega) \quad \forall \omega \in \Omega && \text{(2.2b)}\\
& \sum_{\omega \in \Omega} p_\omega z_\omega \leq 1 - \alpha && \text{(2.2c)}\\
& Ax \leq b && \text{(2.2d)}\\
& x \geq 0,\ z \in \mathbb{B}^{|\Omega|} && \text{(2.2e)}
\end{aligned}$$

Where $c \in \mathcal{R}^n$ is the cost matrix, The technology matrix $T(\omega) \in \mathcal{R}^{n \times m}$ and the righthand side $r(\omega) \in \mathcal{R}^m$ are the random constraint matrix and righthand side, $M_\omega \in \mathcal{R}$ is an appropriate 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. We will refer to a scenario ω is satisfied if the binary decision variable $z_\omega = 0$. The scenario ω is unsatisfied if $z_\omega = 1$. A scenario is considered binding if the scenario is satisfied and the slack variables associated with the constraints of that scenario are equal to 0. The chance constraint is forced to be satisfied by the knapsack inequality (2.2c).

An early formulation of problem (1.5) as a mixed-integer program was given by Morgan et al. (1993). Applying chance-constraints to an aquifer remediation problem, they formulate the problem as a mixed-integer program and assume that each scenario has the same probability. In their approach to the problem, they first solved a master problem formulation with a reliability level $\alpha = 1$ in level $k = 0$. This can be solved without adding any of the extra variables z_ω and can be solved efficiently as long as the set $\mathcal{X}$ is convex. In subsequent levels, their algorithm searches the feasible space of problem (2.2) by branching on all possible nodes created by dropping a

single binding scenario. In any given level k of the search tree, each node gives a feasible solution to problem (2.2) for reliability level $\alpha = 1 - \frac{k}{|\Omega|}$ while the minimum solution on a given level k is the optimal solution for that reliability level. Thus the algorithm finds optimal solutions to problem (2.2) for all possible reliability levels. The authors recognized that the search tree tends to become extremely large for even small instances and so concluded with several heuristics based upon this idea in order to identify decent solutions at each reliability level with less computational effort.

Tayur et al. (1995) gave a method based on algebraic geometry for solving chance constrained programs with pure integer decision variables. The method searches the solution space of the integer variables by solving a reduced integer program without the chance-constraints and then searching the feasible space of this reduced integer program for points that are feasible for the chance constraints. The main contribution of their approach is that the method can find the optimal solution for any chance-constrained program with pure integer decision variables and an oracle to evaluate the feasibility of a candidate solution. Computational results show that the method can solve small problems.

A more traditional mixed-integer programming approach to formulation (2.2) is given in Ruszczyński (2002). A property of the scenarios of a chance constrained program is that if the parameter values of $T(\omega_1)$ are greater than the parameter values of $T(\omega_2)$ for all the elements of the matrices, then scenario ω_1 dominates scenario ω_2. Formulation (2.2) can then be strengthened with the added precedence constraints $z_{\omega_2} \leq z_{\omega_1}$. Ruszzyński then derives valid inequalities for the polyhedron defined by the precedence constraints and the knapsack inequality. The paper also gives a branch-and-bound method for solving the algorithm the reduces the number of scenarios that need to be considered by dropping scenarios that are particularly easy or hard to satisfy. New MIP results for problem (2.2) are given in Chapter III.

Most recently, Pang and Leyffer (2004) gave a finite branch-and-bound algorithm for minimizing Value-at-Risk (VaR) which is equivalent to a single chance-constraint. Their algorithm starts with a reformulation of the problem with linear complementarity constraints for which they derive linear programming upper and lower bounds that can be used in a branch-and-bound framework. A novel branch-and-bound approach for solving formulation (1.5) with joint chance-constraints is given in Chapter V.

D. Approximations and Sampling

Given the difficulty of solving the mixed-integer formulation (2.2) to optimality when there are a large number of scenarios, another branch of research has been into finding nearly optimal solutions. Some of these results focus on sampling methods that allow statistically bounds to be put on the true optimal solution. Another type of approximation paper focuses on developing convex feasible regions that are guaranteed to be contained in the feasible region of the chance constraint, thus allowing for an efficiently computable upper bound. A third type of approximation is in heuristics for quickly finding the best upper bound possible. A common weakness of all these approaches is that it is quite difficult to determine tight upper and lower bounds on the optimal solutions and so most of the results only terminate with a solution that is guaranteed to be feasible for the chance constraint with a high probability.

An early paper by Pintér (1989) gave an explicit convex approximation of the chance constraint called the Bernstein approximation. The feasible region of a Bernstein approximation constraint was derived to be contained in the feasible region of the chance constraint. Thus the Bernstein approximation gives a computationally tractable method for finding an upper bound on the optimization of formulation (1.5)

for problems with only limited information available about the parameter distributions such as the mean, variance, range, or upper bound values. For problems with known parameter distributions, another common conservative approximation is given by Conditional Value-at-Risk (CVaR) (Uryasev, 2000). Using CVaR constraints in place of chance constraints again gives an upper bound on the true optimal objective value and guarantees that a feasible solution is found. With either of these methods, there is no guarantee on the quality of the solutions found.

In stochastic programming sampling is a popular method for finding approximate solutions. For chance-constrained programs Calafiore and Campi (2005, 2006) sampled scenarios from the parameter distributions. They then solved a deterministic convex program with all the sampled scenarios required to be satisfied. They were able to derive a lower bound on the sample size that guarantees the probability that the optimal solution to this convex program is a feasible solution to the original chance-constrained program. Nemirovski and Shapiro (2006) were able to tighten the bounds on the required sample size and Nemirovski and Shapiro (2004) were able to extend their results to general convex constraint functions rather than just linear functions. The importance of these bounds is that the sample size N is polynomial in terms of the log of the required probability that the chance-constraint is satisfied. The weakness is that for some problems, the approximation is extremely conservative.

Luedtke and Ahmed (2007) gave a sampling method with stronger bounds on the optimal solution. They find an upper bound by sampling scenarios and then solving the mixed-integer formulation of a small chance-constrained program with a higher reliability requirement than the original problem. The paper gives results on how large the sample needs to be in order to guarantee with high probability that they have found a feasible solution. These results depending on the assumption that the problem has fixed technology matrix. They also derived a method for finding a lower

bound by solving a sampled problem with a lower reliability level than required in the original problem. They are able to prove the convergence of the lower bound for all cases.

An and Eheart (2007) studied chance-constrained programming applied to air-quality management. They looked at problems with normally distributed parameters but allowing for general dependence between the different parameters. They derived convex bounds for the optimal value of the program by looking at the extreme cases of row dependence: complete codependence, zero codependence, and complete negative codependence. Depending on the values computed, the results are used to identify problem instances for which assumptions that make the overall program convex are used, or cases for which a more detailed nonconvex program must be analyzed.

Another class of approximations for chance constraints are derived by adapting the concepts of robust optimization. Chen et al. (2007) used uncertainty sets to define a convex feasible space of deviations around parameter values. They were able to prove that the new convex space was contained within the convex space of the chance constraints, thus providing an approximation. Parpas et al. (2007) used similar ideas to optimize a chance constrained program for which only the moments of the parameter distributions are known.

Only a few traditional combinatorial heuristics have been developed for finding good feasible solutions to problem (1.5). Up to now all have them have dealt with the special case of the problem with pure integer decision variables. Iwamura and Liu (1996) gave a genetic algorithm that used a Monte Carlo simulation to check the feasibility of any candidate solution. Aringhieri (2004) developed a tabu search heuristic for the same problem. Again simulation is used to evaluate the feasibility of any candidate solution. Chapter IV gives a tabu search heuristic for problems with continuous decision variables and discretely distributed left-hand side parameters.

CHAPTER III

IIS CUTS

A. Preliminaries

This chapter presents a branch-and-cut approach to chance-constrained programs with linear constraints and discretely distributed parameters. The formulation of a chance constrained program with random left-hand side and linear constraints is given by equations (3.1a) - (3.1c).

$$\text{Min} \quad c^\top x \tag{3.1a}$$

$$\text{s.t.} \quad \mathbb{P}\big\{\omega \in \Omega : T(\omega)x \leq r(\omega)\big\} \geq \alpha, \tag{3.1b}$$

$$Ax \leq b. \tag{3.1c}$$

Specifically, the IIS cuts described here are derived for the MIP reformulation of the problem given below by equations (3.2a)-(3.2e).

$$\min \quad c^\top x \tag{3.2a}$$

$$\text{s.t.} \quad T(\omega)x - M_\omega e z_\omega \leq r(\omega) \quad \forall \omega \in \Omega \tag{3.2b}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \leq 1 - \alpha \tag{3.2c}$$

$$Ax \leq b \tag{3.2d}$$

$$x \geq 0, \ z \in \mathbb{B}^{|\Omega|} \tag{3.2e}$$

The primary weakness of this MIP reformulation of joint chance-constrained stochastic programs is that the "Big-M" constraints of the problem mean that the

linear programming relaxation of the problem is often extremely weak. The result of this is that branch-and-bound algorithms tend to be ineffective for many instances of the problem. A common way to strengthen the linear programming relaxation of an MIP is the addition of cutting planes. In this section we give some background on the problem and present the MIP reformulation of the problem that can be solved directly. We then review a class of cutting planes derived by Codato and Fischetti (2006) called combinatorial Benders cuts, which are derived similarly to IIS cuts. We describe the differences between the combinatorial Benders cuts and our IIS cuts.

When the problem parameters have discrete distributions, the original problem (3.1) can be considered as finding the optimal solution where the sum of the probabilities of scenarios that are satisfied is at least α. The IIS cuts are defined by proving that a set of scenarios cannot all be satisfied in an optimal solution to the problem. We also show that if no such sets of scenarios can be found, then an improved upper bound for the problem can quickly be found. We make the following assumptions throughout the rest of the chapter:

(A1) $|\Omega| < \infty$.

(A2) Bounds on the decision variables x are included in the constraint set $Ax \leq b$.

(A3) The polyhedron $P_1 = \{x \in \mathbb{R}^n \mid Ax \leq b\} \neq \emptyset$.

Assumption (A1) requires the random parameters to be discretely distributed thus allowing the MIP reformulation of the problem. 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 chapter. Assumption (A3) keeps the problem from being trivially infeasible and is also not very restrictive.

Based on our computational experience we have seen that often a relatively few number of 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 (3.2). The cutting planes are based on *irreducibly infeasible subsystems* (IISs). An IIS is defined as follows:

DEFINITION A.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 linear programs with the goal of figuring out the optimal strategy for changing problem parameters to make the system feasible. Several methods for identifying these sets using LP methods have been developed (Chinnneck, 1997; Gleeson and Ryan, 1990; Loon, 1981). In more recent years, IISs have been used to generate valid inequalities for the maximum feasible subsystem problem (Amaldi et al., 2003; Pfetsch, 2008). In integer programming, IISs have been used to derive combinatorial Benders (CB) cuts for a class of MIP problems (Codato and Fischetti, 2006).

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 with the master problem having pure integer decision variables and the subproblem having pure continuous decision variables. While the CB cuts were designed specifically to solve problems for which the objective function depends entirely on the integer decision variables, they can also be used for problems in which the objective function depends entirely on the continuous decision variables such as chance-constrained programs. However, the cut generated in Codato and Fischetti (2006) tends to be weak for such problems.

To derive CB cuts, a MIP with "big-M" constraints is decomposed into a master program (3.3) and a subproblem (3.4).

$$\begin{aligned} \min \quad & p^\top w && \text{(3.3a)} \\ \text{s.t.} \quad & Dw \leq d && \text{(3.3b)} \\ & w \in \{0,1\}, && \text{(3.3c)} \end{aligned}$$

where w is a vector of binary variables, p is the cost vector of the binary variables, D is the constraint matrix for constraints that depend only on the binary variables, and d is the righthand side for those constraints.

$$\begin{aligned} \min \quad & c^\top x && \text{(3.4a)} \\ \text{s.t.} \quad & Ax \leq b && \text{(3.4b)} \\ & Tx \leq r - Mew && \text{(3.4c)} \\ & x \in \mathcal{X}, && \text{(3.4d)} \end{aligned}$$

where all parameters and variables are the same as formulation (2.2). Furthermore, either p or c must be a vector of all zeroes for CB cuts to be valid. After finding an integer feasible solution w^* to problem (3.3), an IIS S of problem (3.4) must be identified. A CB cut is then given by the equation

$$\sum_{i \in S: w_i^* = 0} w_i + \sum_{i \in S: w_i^* = 1} (1 - w_i) \geq 1 \tag{3.5}$$

The following fundamental result gives a method for determining IISs that can be used to derive CB cuts.

THEOREM A.2. *IISs of* (3.7) *are in one-to-one correspondence with the supports*

of the vertices of the polyhedron

$$\begin{aligned}\Pi := \Big\{ & y_1 \in \mathbb{R}^{m_1}, y_2(\omega) \in \mathbb{R}^{m_2}, \forall \omega \in \Omega \setminus U, y_3 \in \mathbb{R} \mid \\ & y_1^\top A + \sum_{\omega \in \Omega \setminus U} y_2^\top T(\omega) + y_3^\top c = 0 \\ & y_1^\top b + \sum_{\omega \in \Omega \setminus U} y_2^\top r(\omega) + y_3^\top V \leq -1 \\ & y_1, y_2, y_3 \geq 0. \Big\}\end{aligned} \tag{3.6}$$

Proof. See Gleeson and Ryan (1990) □

Note that the *support* of a vector is the set of indices of its nonzero components. Theorem A.2 is a direct application of the theorem in Gleeson and Ryan (1990) to (3.7) to give us a polyhedron with the property that every extreme point of the polyhedron corresponds with an IIS of (3.7). This means we can simply use LP to identify IISs.

CB cuts can be generated at every feasible integer solution encountered in the pure integer master program. Without decomposing the problem, it is possible to derive CB cuts whenever an integer feasible solution to the problem is encountered. This method was shown to be computationally ineffective in (Codato and Fischetti, 2006) because cuts can only be generated deep in the branch-and-bound tree. 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 not often useful.

This chapter focuses on theoretical results that can be used for general solution techniques for problem (3.2). The main significance of these results is that they are

valid for joint chance-constrained problems with discretely distributed random technology matrices and righthand side vectors. We introduce a new class of optimality cuts, called irreducibly infeasible subsystem (IIS) cuts, for strengthening the LP relaxations of (3.2). We also present a method for quickly 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, and to test instances from a production planning problem.

The rest of the chapter is organized as follows. In Section B we derive IIS cuts and an upper bound improvement strategy to be used in the IIS Branch-and-Cut algorithm. We present and discuss an implementation of the IIS Branch-and-Cut algorithm in Section C and give computational results in Section D. Finally, we finish with a summary and point out some future research topics in Section E.

B. IIS Cuts

For chance-constrained programs, we need cuts that are effective for problems where the objective function depends on the continuous variables. While similar to the CB cuts described in the previous section, IIS cuts are derived so that cuts can be generated at every solution to the linear relaxation of formulation (3.2). This means that the cuts are more likely to tighten the formulation where it is needed and so be much more effective at aiding solution.

Let us begin by defining some notation we will use throughout the rest of the chapter. At an arbitrary node of a branch-and-bound 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 $V - \epsilon$ denote the current incumbent objective value minus a sufficiently small value. We will generate IIS cuts from IISs of the polyhedron defined by forcing every scenario in $\Omega \setminus \mathcal{U}$ to be satisfied, restricted by an optimality cut generated from the upper bound. We will refer to a scenario ω being forced into or out of the problem if the binary decision variable z_ω for that scenario is forced to equal 0 or 1, respectively. Since $\mathbb{P}\{\Omega \setminus \mathcal{U}\} \geq \alpha$, 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:

$$\begin{aligned}
\text{Min} \quad & c^\top x && \text{(3.7a)}\\
\text{s.t.} \quad & Ax \leq b && \text{(3.7b)}\\
& T(\omega)x \leq r(\omega), \quad \forall \omega \in \Omega \setminus \mathcal{U} && \text{(3.7c)}\\
& c^\top x \leq V - \epsilon && \text{(3.7d)}\\
& x \geq 0. && \text{(3.7e)}
\end{aligned}$$

The advantage to formulation (3.7) is that it can be set up at fractional solutions of problem (3.2). This means that we can generate optimality cuts early in the branch-and-bound tree when they are most effective rather than deep in the tree at integer solutions. Also, since the fractional points that we are trying to separate with our cuts are found via the linear relaxation of the problem, the part of the solution space that we are cutting off is more useful than the region cut off by CB cuts.

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 (3.2). The following fundamental results show how such sets $\mathcal{D}$ can be determined as well as the separation

problem that we are trying to solve.

THEOREM B.1. *Given an IIS S of formulation* (3.7)*, let the subset of scenarios* $\mathcal{D} = \{\omega \in \Omega \mid T(\omega)x \leq r(\omega) \cap S \neq \emptyset\}$*. The set* $\mathcal{D} \neq \emptyset$ *defines the IIS cut*

$$\sum_{\omega \in \mathcal{D}} z_\omega \geq 1. \tag{3.8}$$

Equation (3.8) *is valid in the sense that it does not cut off all optimal solutions to problem* (3.2)*.*

COROLLARY B.2. *Given a fractional solution* $(\bar{x}, \{\bar{z}_\omega\}_{\omega \in \Omega})$ *to the LP relaxation of problem* (3.2)*. The separation problem for the CB cut is to find an IIS S of problem* (2.2) *and generating a subset* $\mathcal{D} \subseteq \Omega$ *as in* (3.8) *such that*

$$\sum_{\omega \in \mathcal{D}} \bar{z}_\omega < 1. \tag{3.9}$$

The separation problem defined by Corollary B.2 is NP-hard (Amaldi et al., 2003). Therefore, heuristics have to be used to find valid inequalities quickly. Pfetsch (2008) suggests finding IISs by solving an LP constrained by Π with the objective function coefficients given by the non-integer solution to the LP relaxation. To find IIS cuts, a good possible objective function is:

$$\text{Min} \sum_{\omega \in \Omega \setminus U} \bar{z}_\omega y_2(\omega). \tag{3.10}$$

However, using this objective function we may not find an IIS cut that separates the current non-integer solution. Nevertheless, the generated IIS cuts are valid for the entire branch-and-bound tree and may cut off some non-integer solution at some later node in the branch-and-bound tree. Notice that only the values of the binary variables z_ω are used in (3.10). The reason for this is that cuts tend to be stronger

when $|\mathcal{D}|$ is small. Furthermore, the cardinality $|\mathcal{D}|$ only depends on the constraints defined by the z_ω variables and therefore, the other variables should not affect the objective function of the cut generating LP.

Since every extreme point of Π defines an IIS, it is possible to generate rounds of cuts using linear programming. 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 (Pfetsch, 2008). A more effective method would be 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.

1. Improving the Upper Bound

One situation that may arise when generating IIS cuts is that $\Pi = \emptyset$ and thus the cut generating LP is infeasible. This implies that every scenario in $\Omega \setminus \mathcal{U}$ can be satisfied with the current upper bound V. In the original decomposition approach to the problem, an upper bound to the optimal solution was found by solving the subproblem to optimality. With a chance constrained program, it is possible to improve upon the upper bound found in this way. We would like to either improve V by dropping more scenarios from $\Omega \setminus \mathcal{U}$, or be able to show that no improvement is possible and fathom the current node.

PROPOSITION B.3. *Let* Π *be as defined in* (3.6). *If* $\Pi = \emptyset$ *and the original problem* (3.1) *is bounded, then formulation* (3.7) *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* (3.2) *with* $c^\top x \leq V$. *Furthermore, a possibly*

improved integer feasible solution can be found by dropping a set of scenarios $\mathcal{F} \subseteq \Omega$ *from problem* (3.7) *for any set* $\mathcal{F}$ *such that* $\mathbb{P}(\Omega \setminus (\mathcal{U} \cup \mathcal{F})) \geq \alpha$.

Proof. Since $\Pi = \emptyset$, it implies there are no IISs for formulation (3.7) and hence it is feasible. Since it cannot be unbounded as it is a restriction of problem (3.1), it has an optimal solution $\bar{x}$. The solution $(\bar{x}, \{\bar{z}_\omega\}_{\omega \in \Omega})$ satisfies $\{Ax \leq b, T(\omega)x - Mez_\omega \leq r(\omega), \forall \omega \in \Omega, x \geq 0\}$. Since $\mathbb{P}(\mathcal{U}) \leq 1 - \alpha$ or else the node would have been fathomed by *infeasibility*, constraint (3.2c) must also be satisfied, and thus $(\bar{x}, \{\bar{z}_\omega\}_{\omega \in \Omega})$ is an integer feasible point. Also, $c^\top \bar{x} \leq V$, otherwise constraint (3.7c) would have been violated. Finally, dropping the set of scenarios $\mathcal{F}$ from formulation (3.7) provides a relaxation whose optimal solution will be no worse. □

To improve the upper bound on the problem, it is necessary to carefully choose the set $\mathcal{F}$ of scenarios to remove from formulation (3.7). If the slack variables associated with the constraints of a scenario are all basic, then removing those constraints will not improve the optimal objective value when they are removed. This means that the only scenarios whose removal will affect the objective value are those in which some constraint is binding.

One way to identify the set $\mathcal{F}$ of scenarios to be removed is to rank the slack variables for each scenario $\omega \in \Omega \setminus \mathcal{U}$ and greedily add the ones with the minimum values as long as $\mathbb{P}(\Omega \setminus (\mathcal{U} \cup \mathcal{F})) \geq \alpha$. A more time consuming method is to take the scenario with the minimum slack variable and remove it from the problem. Then re-solve the problem and repeat until no more scenarios can be removed. It would also be possible to use a more complicated heuristic such as local or tabu search to better identify sets of scenarios to remove. Note that an IIS cut can always be found after the upper bound improvement step has been run by generating a cut using the new upper bound in formulation (3.7). The following proposition gives an instance

when it is possible to show that no improvement to the upper bound is possible which allows for early fathoming of the node.

PROPOSITION B.4. *Given an optimal solution $\hat{x}$ to problem* (3.7) *with optimal objective value $c^\top \hat{x}$. Define $\mathcal{B} := \{\omega \in \Omega | \text{at least one slack variable associated with the constraints } T(\omega)x \leq r(\omega) \text{ is nonbasic}\}$. If $\mathcal{B} \cap (\Omega \setminus \mathcal{U}) = \emptyset$ then no upper bound improvement is possible and the node can be fathomed.*

Proof. Since $\mathcal{B} = \emptyset$, for any set of scenarios $\mathcal{F} \subseteq \Omega \setminus \mathcal{U}$ the slack variables associated with these constraints are non-basic. Therefore, removing these constraints will not effect the optimal solution to the problem and no set of scenarios $\mathcal{F}$ exists that will improve the solution. □

C. A Branch-and-Cut Algorithm

This section illustrates the use of the IIS ideas within a branch-and-cut framework. The point of this algorithm is to explicitly show how the IIS cuts and the upper bound improvement fit into an exact method to solve formulation (3.2). First, define k as the node index and K as the total number of nodes in the 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, the current best upper bound on the optimal solution is given by V.

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 $V = \infty$

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

Step 2: Solve LP Solve the linear relaxation of formulation (2.2) including the proper constraints for the scenarios that have been set in branch-and-bound. This will either find an optimal solution $(\bar{x}, \{\bar{z}_\omega\}_{\omega\in\Omega})$ or else that the problem is infeasible.

Step 3: Fathoming Rules If the problem is infeasible or $c^\top \bar{x}^k \geq V$ fathom the node and return to step 1.

Else, if constraint (2.2c) is satisfied, set $V = c^\top \bar{x}^k$, fathom the node and return to step 1.

Else, continue to step 4.

Step 4: Cut Generation If cuts are to be generated, find extreme points of (3.6) that give IISs. Generate and add the IIS cuts implied by these sets and go to step 2.

If (3.6) is empty, improve the upper bound as allowed by Proposition B.3 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.

REMARK C.1. The finite convergence of the above algorithm is guaranteed by the branching on the binary variables of formulation (3.2). By Theorem B.1 and Proposition B.4, not all alternative optimal solutions are eliminated by the IIS cuts, hence the algorithms 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.

D. Computational Results

We now present some computational results showing the effectiveness of the IIS branch-and-cut algorithm in solving formulation (3.2). We ran our tests on two test sets, the first is an application developed in Chapter VI involving the optimal allocation of vaccines under parameter uncertainty. The second is a chance-constrained multistage production planning problem adapted from the standard models in the literature (Nemhauser and Wolsey, 1999). The implementation was completed in C++ 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 callable library. 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 directly on the MIP formulation of the problem to provide a benchmark. The second is computations with the IIS branch-and-cut algorithm without adding the IIS cuts. This implementation is pure branch-and-bound 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.

1. Optimal Vaccine Allocation

Detailed background on this application is available in Chapter VI. We have provided details of the formulation of the chance-constrained problem as well as the probability distributions assumed for the random parameters of the original disease model in Appendix A for the interested reader.

Table IV gives the results of the computational tests on the vaccination allocation

Table IV. IIS Branch-and-Cut Computations on Vaccination Problems

	CPLEX Results				B&B No IIS Cuts		IIS Cuts Added			
Instances	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	10815	65.27	211	330	33.34
vac750e	64.77	0%	8432	163.74	64.77	20387	64.77	155	294	27.27
vac1000a	65.11	0%	22505	469.16	65.11	85687	65.11	207	387	47.69
vac1000b	65.02	0%	74615	1527.72	65.02	58815	65.02	493	766	104.16
vac1000c	64.57	0%	32481	642.87	64.57	83623	64.57	387	645	72.08
vac1000d	65.50	0%	25604	678.98	65.50	27691	65.50	299	458	63.07
vac1000e	64.31	0%	23140	570.12	64.31	>110000	64.31	431	768	97.63
vac2000a	65.05	10.89%	>71181	>7200	65.16	>54004	64.98	1643	2660	1001.31
vac2000b	65.50	2.10%	>85385	>7200	65.50	>57729	65.48	1901	2829	1045.24
vac2000c	66.07	10.27%	>71001	>7200	65.55	>55733	65.50	1895	3122	1109.76
vac2000d	64.95	3.18%	>92311	>7200	65.58	>51927	64.95	1491	2432	837.00
vac2000e	65.06	5.24%	>109881	>7200	66.05	>45497	65.06	1737	2660	1119.82

test instances. The first column of Table IV 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 branch-and-bound tree, and the fifth column gives the time to prove optimality. The next two columns give the results of our implementation of branch-and-bound without any added cuts or upper bound improvement. The first of these columns gives the best solution found, while the second of these columns gives the average number of nodes searched in the branch-and-bound tree. For either CPLEX or the branch-and-bound 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.

The IIS branch-and-cut algorithm is able to greatly reduce both the number of nodes of the branch-and-bound 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 branch-and-bound. A relatively few number of cuts allow for the optimal solution to be found with much less computational effort. For example, notice that for the vac1000 test instances, after two hours the branch-and-bound algorithm can only prove optimality for four of the test instances, and these four require an average of about 63950 nodes in the branch-and-bound 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 branch-and-bound tree and a reduction of 90% in computation time. Even accounting accounting for the number of cuts which each require solving an LP about the size of 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 branch-and-bound algorithm without IIS cuts actually identified the optimal solution for each of the five 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 two hours. The branch-and-bound algorithm by itself does even worse and now searches an average of over 50000 nodes but still cannot identify an optimal solution in two 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.

2. Production Planning Application

We also tested the IIS algorithm on a second set of test instances generated from a standard production planning application from the literature (Nemhauser and Wolsey, 1999). Details of the problem and formulation are given in Appendix B including the probability distributions of the random parameters.

Table V gives the results of CPLEX, branch-and-bound, and the IIS branch-and-cut algorithm on this set of test instances. The tables are organized the same as Table IV with a final column added to give the percentage objective value improvement found by the IIS branch-and-cut algorithm over CPLEX. This value is computed using the equation percentage improvement $= |\frac{\text{IIS Ubound - CPLEX Ubound}}{\text{CPLEX Ubound}}|$. As evidenced by the inability of CPLEX to solve any of these problems except for the smallest, these test problems are significantly more difficult to solve to optimality than the vaccination application problems. For these problems, we stopped the cut generation

Table V. IIS Branch-and-Cut Computations on Production Planning Problems

	CPLEX Results				B&B No IIS Cuts		IIS Cuts Added				Improv
Instances	Objval	Gap	Nodes	Time	Objval	Nodes	Objval	Nodes	Cuts	Time	CPLEX
Prod100a	-93008.4	0%	28282	927.48	-93008.4	81207	-93008.4	48095	1000	2073.93	0%
Prod100b	-92737.9	0%	42022	1011.07	-92737.9	79439	-92737.9	81199	1000	3235.13	0%
Prod100c	-89277.4	0%	8917	346.79	-89277.4	67277	-89277.4	62355	1000	2382.24	0%
Prod100d	-92382.8	0%	25594	812.57	-92382.8	66881	-92382.8	37487	1000	1910.96	0%
Prod100e	-90293.4	0%	19025	733.90	-90293.4	85093	-90293.4	43927	1000	2041.24	0%
Prod250a	-88675.8	15.04%	>33063	>7200	-81489.1	>257981	-88463.1	>41968	1000	>7200	-0.24%
Prod250b	-86388.4	14.26%	>29526	>7200	-80666.5	>417253	-86836.7	>34947	1000	>7200	0.52%
Prod250c	-86986.0	15.20%	>35330	>7200	-81215.2	>200000	-87775.7	>91123	1000	>7200	0.91%
Prod250d	-89248.6	15.95%	>29415	>7200	-81492.6	>300000	-89508.6	>25348	1000	>7200	0.29%
Prod250e	-86506.7	16.65%	>26846	>7200	-81492.6	>353550	-87011.5	>42435	1000	>7200	0.58%
Prod500a	-84932.3	31.03%	>8987	>7200	-79109.6	>68970	-83620.3	>18912	1000	>7200	-1.55%
Prod500b	-85810.2	31.23%	>7500	>7200	-76886.2	>78324	-86021.0	>3424	1000	>7200	1.17%
Prod500c	-85426.8	30.61%	>7839	>7200	-78498.4	>60012	-86121.4	>6555	1000	>7200	0.81%
Prod500d	-84501.7	31.03%	>7877	>7200	-74826.6	>102613	-85003.4	>24994	500	>7200	0.59%
Prod500e	-85441.9	29.96%	>7055	>7200	-74637.2	>73964	-85195.8	>25631	500	>7200	-0.29%
Prod750a	-84245.7	37.01%	>3105	>7200	-75831.4	>45000	-84931.5	>6419	500	>7200	0.81%
Prod750b	-85222.3	37.61%	>2581	>7200	-78290.1	>43953	-86021.0	>25736	500	>7200	0.94%
Prod750c	-84371.5	37.86%	>2789	>7200	-75153.5	>40705	-84546.4	>12594	500	>7200	0.21%
Prod750d	-82774.6	37.23%	>2600	>7200	-72805.7	>41772	-82849.6	>16723	500	>7200	0.09%
Prod750e	-84275.9	37.73%	>3047	>7200	-76743.8	>45000	-85845.8	>20236	500	>7200	1.86%
Prod1000a	-84411.0	41.67%	>1100	>7200	-75793.3	>41436	-83129.1	>11531	500	>7200	-1.52%
Prod1000b	-83005.5	43.21%	>1640	>7200	-73462.6	>24535	-84145.0	>6160	500	>7200	1.37%
Prod1000c	-81771.0	44.67%	>1542	>7200	-72828.9	>38428	-83416.5	>8852	500	>7200	2.01%
Prod1000d	-82814.3	43.20%	>1559	>7200	-75918.7	>29629	-84921.5	>18587	500	>7200	2.54%
Prod1000e	-80816.7	45.27%	>1500	>7200	-75862.3	>26000	-83735.5	>2351	500	>7200	3.61%
Prod2000a	-70779.3	70.71%	>0	>7200	-70837.9	>16740	-82991.0	>1206	500	>7200	17.25%
Prod2000b	-71890.6	67.26%	>0	>7200	-71962.9	>17676	-82445.9	>2546	500	>7200	14.68%
Prod2000c	-72090.4	85.26%	>0	>7200	-72179.0	>14458	-82145.1	>856	500	>7200	13.95%
Prod2000d	-73985.9	62.91%	>0	>7200	-74041.9	>8250	-82849.0	>1833	500	>7200	11.98%
Prod2000e	-71739.0	67.97%	>0	>7200	-71793.1	>17154	-82990.7	>4540	500	>7200	15.68%

loop of the algorithm after a certain number of cuts had been generated in order to reduce the extra computation time due to the cuts. Too many extra cuts slowed down the solution time for linear relaxation of this problem and hence made the algorithm ineffective.

All the three algorithms were able to solve the smallest test instances prod100 to optimality in the time allotted, however optimality cannot be proven on any of the larger test problems. The results on Prod100 with our bare implementation of branch-and-bound shows that use of the IIS branch-and-cut algorithm again result in a significant decrease in the number of nodes searched by the branch-and-bound algorithm in order to converge to an optimal solution. CPLEX is able to solve the smallest test problems to optimality more quickly than the IIS methods. For the test

problems with more than 100 scenarios, the IIS branch-and-cut algorithm improves upon the best solution found by CPLEX in 21 out of the 25 instances. For the prod2000 test problems, the improvement is an average of over 14%.

E. Conclusion

In this chapter 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 fractional 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 quickly improving the upper bound during a branch-and-bound method when there is a node for which no IIS cut can be found. These IIS cuts are similar to the combinatorial Benders cuts of Codato and Fischetti (2006), but they specifically derived for a problem with the objective function depending on the continuous decision variables rather than on the integer decision variables. This addresses some of the weakness of the CB cuts for our class of problems. The chapter also gives some computational results from our algorithm on two sets of test instances from two different applications. The computational results are very 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 set than does CPLEX or our implementation of branch-and-bound with no cuts added. The production planning problems are more difficult to solve to optimality, however the IIS branch-and-cut algorithm is able to find significantly improved solutions than either of the other

two algorithms. Possible extensions to this work include implementing the cuts in a branch-and-bound 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 for a stronger formulation of the problem that gives stronger convex relaxations.

CHAPTER IV

TABU SEARCH METAHEURISTIC

A. Introduction

Despite the positive computational results shown in Chapter III for solving problem (2.2) to optimality using the IIS branch-and-cut algorithm, problems with huge numbers of scenarios are still intractable. A common approach to intractable combinatorial problems is through metaheuristics based on local search. The main function of a combinatorial metaheuristic is to find a good feasible solution from among a finite set of possible solutions for the case when exact solution of the problem is not computational feasible. The weakness of heuristic solutions is that there is no guarantee on the quality of the solution found. For many instances of problem (1.5), the problem is defined with an extremely large number of scenarios and it is important to at least find some feasible solution.

The heuristic presented in this chapter is designed to exploit the scenario structure of the problem. Under the finite number of scenarios assumption, a point is feasible for constraint (1.5b) if at least $\lfloor |\Omega| * \alpha \rfloor$ scenarios are satisfied at that point. This suggests a combinatorial structure to the problem by looking at scenarios that are satisfied or not satisfied. The basic idea behind our heuristic is to switch scenarios in and out of the solution by requiring them to be satisfied or not.

The main contributions of this chapter 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 chapter is organized as follows: In Section B 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 C gives a metaheuristic based on tabu search using our reformulation and our sampling methods. Section D gives some randomly generated problem test sets and computational results using our heuristic. We finish with some conclusions and future work.

B. Local Search

This section presents a reformulation of problem (1.5) 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.

1. Defining the Neighborhood

DEFINITION B.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 (2.2a) - (2.2d), 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 B.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 (2.2) 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 (4.1a) - (4.1d) 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,

$$
\begin{aligned}
f(C) = \quad & \min \quad c^\top x && \text{(4.1a)} \\
& \text{s.t.} \quad Ax \leq b && \text{(4.1b)} \\
& \qquad T(\omega)x \leq r(\omega) \; \forall \omega \in C && \text{(4.1c)} \\
& \qquad x \geq 0 && \text{(4.1d)}
\end{aligned}
$$

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 B.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 B.4. Restricting the search space Φ to its minimal elements will not eliminate all optimal solutions of the problem.

REMARK B.5. In the case where every scenario has the same probability, remark B.4 implies that the search can be restricted to solutions where exactly $\lceil \alpha|\Omega| \rceil$ scenarios are forced to be satisfied.

DEFINITION B.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 $\omega \in \Omega \backslash C$ 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 (4.1). If formulation (4.1) is infeasible, then $f(C) = \infty$. Define $I(C)$ as the sum of the infeasibilities of the constraints of formulation (4.1), with $I(C) = 0$ when formulation (4.1) 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 (Balas and Zemel, 1980). 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 B.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.

$$
\begin{aligned}
f(C) = \quad & \min \quad c^\top x && \text{(4.3a)} \\
& \text{s.t.} \quad Ax \leq b && \text{(4.3b)} \\
& \qquad T(\omega)x \leq r(\omega) \ \forall \omega \in D(C) && \text{(4.3c)} \\
& \qquad x \geq 0 && \text{(4.3d)}
\end{aligned}
$$

Clearly, given a valid sufficient set $D(C)$, the optimal objective value for problem (4.3) is equal to the optimal objective value of (4.1) 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 B.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 (4.3).

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.

C. 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 (1.5) with discretely distributed

random parameters. Our algorithm uses a random tabu search (Glover and Laguna, 1997; Hoos and Stutzle, 2005) 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.

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 (2.2) 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.

$$
\begin{aligned}
l_\omega = \min \quad & c^\top x && \text{(4.4a)}\\
\text{s.t.} \quad & Ax \leq b && \text{(4.4b)}\\
& T(\omega)x \leq r(\omega) && \text{(4.4c)}\\
& x \geq 0 && \text{(4.4d)}
\end{aligned}
$$

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.

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 (4.1). 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. 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 (4.1).

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 (4.3) 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.

D. Computational Results

1. 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.

Table VI. Sampling Tabu Computations on Vaccination Test Instances

	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

Tables VI and VII give the basic results of CPLEX and our heuristic. For both

Table VII. Sampling Tabu Computations on Production Planning Instances

	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

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 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 VI, 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 VII 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.

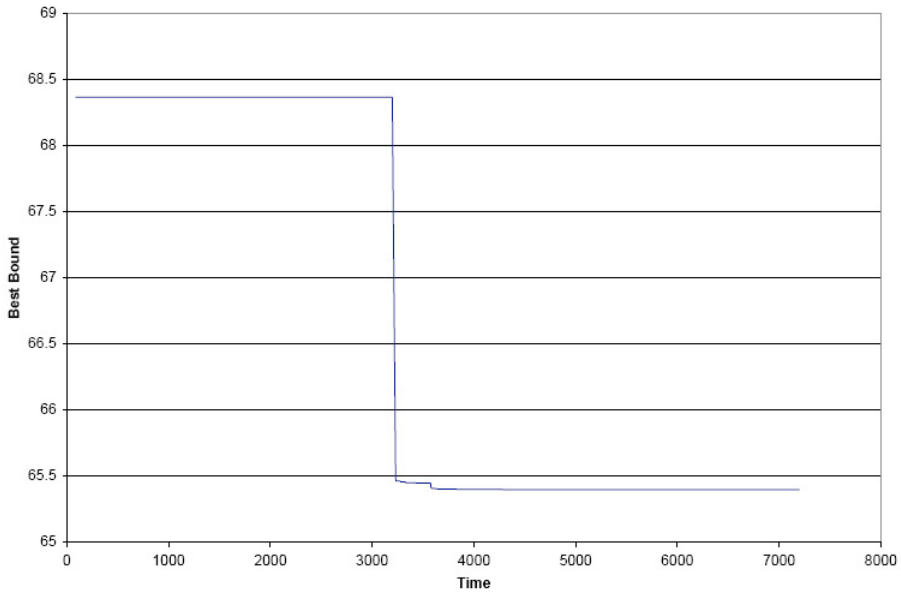

Fig. 1. Plot of Best Feasible Solution vs. Time for Vac20000

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

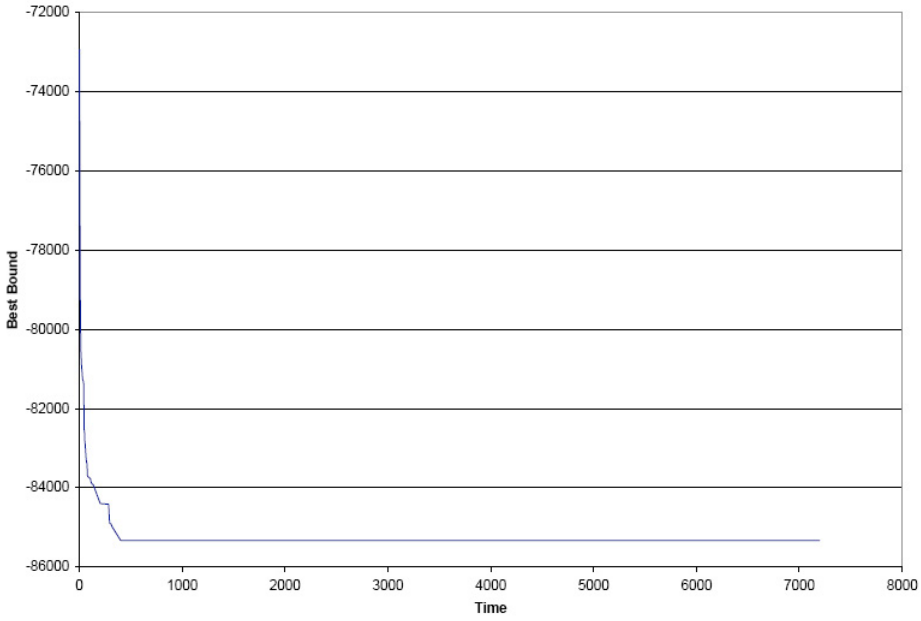

Fig. 2. Plot of Best Feasible Solution vs. Time for Prod2000

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. 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 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.

E. 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.

CHAPTER V

A MONOTONIC OPTIMIZATION ALGORITHM

A. Introduction

In both the IIS branch-and-cut algorithm of Chapter III and the tabu search heuristic presented in Chapter IV, the computational difficulty of solving chance-constrained programs increases dramatically with the number of scenarios. Up until now we have tried to deal with this with preprocessing techniques and the identification of critical subsets of scenarios. However, the number of scenarios that the methods can deal with is still fundamentally limited. In this chapter, we derive an algorithm that deals with the chance constraint implicitly in the hope that this will expand the size of problems that can be solved.

In this chapter we show how to reformulate the problem so that the chance constraint is a monotonic function of the decision variables. This allows us to prove some bounds on the optimal objective function within any hyper-rectangular partition of the decision variables by solving small linear programs and evaluating the feasibility of the chance constraint at single points. The method is a branch-and-bound algorithm on the continuous feasible space of the decision variables. The computationally efficient evaluation and bounding methods allow us to search a much larger tree than is possible in traditional branch-and-bound on the MIP formulation of the problem. Furthermore, the algorithm can be used on a more general class of chance-constrained problems given by formulation (5.1). Note that there are significantly less assumptions in this formulation, however a few extra ones will be added in the next section.

$$\text{Min} \quad f(x) \tag{5.1a}$$

$$\text{s.t.} \quad \mathbb{P}\big\{\omega \in \Omega : g_i(x,\omega) \leq 0, \quad i = 1 \dots m\big\} \geq \alpha \tag{5.1b}$$

$$x \in \mathcal{X}. \tag{5.1c}$$

In formulation (5.1a) - (5.1c), $x \in \mathbb{R}^n$ is the decision variable vector, $f(x)$ is the continuous objective function, $g_i(x,\omega),\ i = 1 \dots m$ are measurable, real valued, lower semi-continuous functions that make up the random constraints within the joint chance-constraint (5.1b), and $\mathcal{X}$ is the compact feasible space of the decision variables. In this formulation, individual outcomes 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 with probability less than $\alpha \in [0,1]$.

Monotonic optimization (Tuy, 2000) concerns the optimization of a monotonic function over a constraint set characterized by monotonic functions. The main idea is that the objective function of any point x_1 dominates the objective value of any point x_2 in the cone of all points less than or equal to x_1. This is used to partition the feasible space of the constraint sets in a branch-and-bound algorithm and determine the global optimum for this class of nonlinear, nonconvex optimization problems. In Cheon et al. (2006), a chance-constrained program with fixed left-hand side is reformulated as a monotonic optimization program. Then, monotonic optimization concepts are used to develop a finitely convergent optimization algorithm for the problem. The algorithm was shown to be effective computationally on a test set.

This chapter presents a generalization of monotonic optimization problems to chance-constrained programs with random left-hand side. We are able to give some

conditions under which the more general problem can be reformulated as a monotonic optimization problem. We give an algorithm that exploits this structure and prove convergence. We finish with a discussion of some computational results which have proven to be disappointing. We end the chapter with a discussion of some of the reasons why the algorithm does not seem to work.

B. Background

In this section we present the assumptions necessary for our algorithm as well as some properties of joint chance-constrained programs that we exploit in our solution method. Second, we describe the classes of chance constrained problems for which our assumptions hold and give an example of reformulating a linear chance-constrained program with fixed left-hand side so that our assumptions hold. We then review monotonic optimization and show how it relates to our problem. Finally, we present results that allow us to fathom regions of the feasible space and identify locally optimal solutions using ideas from monotone optimization (Cheon et al., 2006; Tuy, 2000).

DEFINITION B.1. For any $x, y \in \mathcal{R}^n$, $x \succeq y$ if $x_i \geq y_i \ \forall i = 1 \dots n$

DEFINITION B.2. A function f is *increasing* if for any $x, y \in \mathcal{R}^n$ such that $x \succeq y$, $f(x) \geq f(y)$.

DEFINITION B.3. A function f is *decreasing* if for any $x, y \in \mathcal{R}^n$ such that $x \succeq y$, $f(x) \leq f(y)$.

Assumptions

(A1) The random constraint functions $g_i(x, \omega)$ are increasing for $i = 1 \dots m$ or decreasing for $i = 1 \dots m$ for all $\omega \in \Omega$.

(A2) We have an oracle to evaluate $\mathbb{P}\{\omega \in \Omega : g_i(x,\omega) \leq 0, \quad i = 1 \dots m\} \geq \alpha$ for any point $x \in \mathcal{R}^n$.

(A3) The feasible region of the chance constraint $\Pi = \{x : \mathbb{P}\{g_i(x,\omega) \leq 0\} \geq \alpha, \quad i = 1 \dots m\}$ is contained within the hypercube $\{x : a \preceq x \preceq b\}$ for some $a, b \in \mathcal{R}^n$.

Assumption (A1) is the main assumption needed for our algorithm. It allows us to use the ideas of monotone optimization to solve chance-constrained programs with random left-hand sides. While this assumption is limiting, it holds for an important class of formulations and applications such as problems for which the decisions x are investments in capacity subject to service requirements where the investments increase the probability that the required service is met. An analogous situation is when the decisions are investments in capacity subject to resource constraints and an increase in capacity always increases resource use. These are classes of problems with a wide number of applications. The assumption also holds for any problem with polynomial constraints with either all positive or all negative coefficients.

Several types of problems can be reformulated so that Assumption (A1) holds. First, any problem with fixed left-hand side can be reformulated by adding extra decision variables y as used in (Cheon et al., 2006). Any problem with linear g_i can also be reformulated by using complements of the decision variables in the chance constraint and extra constraints added to the deterministic constraint set. This is proved in Proposition B.4. Finally, any combinations of the above cases can be reformulated so that Assumption (A1) holds.

Assumption (A2) is needed so that our results hold for general probability distributions of the random parameters the make up the chance-constraints. This assumption is limiting in the sense that such an oracle does not exist for many problems with

continuously distributed parameters. Calculating the multi-dimensional integral that is needed to evaluate the feasibility of a constraint of form (**??**) may be extremely difficult. However, for any problem for which the parameters have discrete distribution, possibly determined through sampling, evaluating the required probability is not difficult. Assumption (A3) is needed in order to prove the convergence of our algorithm. It is not particularly limiting.

PROPOSITION B.4. *Under Assumption (A3), any joint chance-constrained problem for which g_i is linear for all $i = 1 \dots m$ and $|\Omega| < \infty$ can be reformulated so that Assumption (A1) holds.*

Proof. Since the feasible space of the problem is bounded, we can assume without loss of generality that $0 \leq x_j \leq m_j$ for all $j = 1 \dots n$ and some upper bound m_j. The linear chance constrained problem can be formulated as:

$$\text{Min} \quad c^\top x \tag{5.2a}$$

$$\text{s.t.} \quad \mathbb{P}\big\{\omega \in \Omega \sum_{j=1}^{n} t_{ij}(\omega) x_j \leq r_i(\omega)\big\} \geq \alpha, \quad i = 1 \dots m \tag{5.2b}$$

$$x \in \mathcal{X}. \tag{5.2c}$$

$$0 \leq x_j \leq m_j \ \forall j = 1 \dots n \tag{5.2d}$$

For any $t_{ij}(\omega) : t_{ij}(\omega) < 0$, replace x_j with $m_j - \bar{x_j}$ where $\bar{x_j}$ is the complementary decision variable to x_j. Define the set $\mathcal{D}_\omega$ as the set of indices for which $t_{ij}(\omega) \geq 0$. This leads to the following linear chance-constrained formulation with all positive

technology matrix which is sufficient for Assumption (A1) to hold.

$$\text{Min} \quad c^\top x \tag{5.3a}$$

$$\text{s.t.} \quad \mathbb{P}\big\{\omega \in \Omega : \sum_{j\in\mathcal{D}} t_{ij}(\omega)x_j + \sum_{j\notin\mathcal{D}} -t_{ij}(\omega)\bar{x_j} \leq r_i(\omega) + \sum_{j\notin\mathcal{D}} -t_{ij}(\omega)m_j\big\} \geq \alpha, \quad i = 1\dots m \tag{5.3b}$$

$$x \in \mathcal{X}. \tag{5.3c}$$

$$0 \leq x_j \leq m_j \ \forall j = 1\dots n \tag{5.3d}$$

$$x_j + \bar{x_j} = m_j \ \forall j = 1\dots n \tag{5.3e}$$

□

Note that the reformulation given in Proposition B.4 only requires the addition of n extra decision variables and n extra constraints. This is a useful result for the computational effectiveness of our algorithm on a wide range of problems. Since our feasible region is bounded, by using the complements of our decision variables we can assume without loss of generality that $g_i(x,\omega)$ are increasing for $i = 1\dots m$. Now we define some terminology that will be used throughout the development of the algorithm.

DEFINITION B.5. A set $\mathcal{H} \subseteq \mathcal{R}^n_+$ is *normal* if for any $x, y \in \mathcal{R}^n_+$ such that $y \preceq x$ and $x \in H$, $y \in H$.

Using the definition of reverse normal sets as well as Assumption (A1) it is clear that the feasible region of the chance constraint (5.1b) is a reverse normal set if $g_i(x,\omega)$ are increasing for $i = 1\dots m$. Thus problem (5.1) can be reformulated as the following, where $\mathcal{G} := \{x : \mathbb{P}\big\{\omega \in \Omega g_i(x,\omega) \leq 0, \ i = 1\dots m\big\} \geq \alpha\}$ is normal set:

$$\text{Min} \quad f(x) \tag{5.4a}$$

$$\text{s.t.} \quad x \in \mathcal{X} \bigcap \mathcal{G}. \tag{5.4b}$$

PROPOSITION B.6. *The set $\mathcal{G}$ is closed.*

Proof. See Proposition 14 in (Ruszczyński and Shapiro, 2003) □

In our proposed algorithm, we partition the feasible space of the decision variables into hyper-rectangles $\{x : x_l \preceq x \preceq x_u\}$ where $a \preceq x_l \preceq x_u \preceq b$. Define $C(x_l, x_u) := \min_x\{f(x) : x \in \mathcal{X} \bigcap [x_l, x_u]\}$. The following proposition gives the important properties of formulation (5.4) that allows us to fathom partitions of the feasible space. The proof follows from the properties of normal sets given in Tuy (2000).

PROPOSITION B.7.

(i) $C(x_l, x_u)$ gives a valid lower bound on the optimal objective value of formulation (5.1) *over the hyper-rectangle $[x_l, x_u]$.*

(ii) If $argmin_x\{C(x_l, x_u)\} \in \mathcal{G}$, then $C(x_l, x_u)$ gives a valid upper bound on the optimal objective value of formulation (5.1).

(iii) If $x_l \notin \mathcal{G}$, then there is no feasible $x \in [x_l, x_u]$.

Since $C(x_l, x_u)$ can be computed by solving a mathematical program that avoids the difficulty of the chance constraint, Proposition B.7 suggests a branch-and-bound procedure on partitions of the continuous feasible space that will converge to an optimal solution to the problem. The benefit of such an algorithm is that since the chance-constraint is always dealt with implicitly, there is no need for the large number of binary variables as was needed in previous exact solution methods. The weakness

of this type of algorithm is that the information given by the chance-constraint is dropped when finding bounds at the nodes of the branch-and-bound tree. This could prevent the bounding procedure from sufficiently pruning the tree thus making the algorithm inefficient.

C. A Branch-and-Bound Algorithm

In this section, we formally present our *Monotone Chance Constraints Algorithm (MCC-Algorithm)*. The algorithm works by making refining partitions of the continuous feasible space of the decision variables and fathoming according to the rules laid out in the previous section. The goal of this algorithm is to exploit the ability to search a huge number of nodes quickly even for problems with large numbers of scenarios. The hope is that this will allow for tight bounds on the optimal objective value for a wide range of problems that are much bigger than more traditional MIP methods can handle. In the second part of this section, we prove convergence of the algorithm in the general case and give a modification to give finite convergence in the case of a finite number of scenarios. We end the section with a discussion of our implementation of the algorithm.

1. An Algorithm

In the following algorithm, $\mathcal{N}$ is the set of open nodes, v is the lower bound, V is the upper bound, and $\bar{x}^k$ is the current incumbent solution. For a given node η, x_l^η and x_u^η are the upper and lower bounds, also $x_{l,i}^\eta$ and $x_{u,i}^\eta$ are the ith components of the respective vectors and e_i is a vector of zeroes with a one in position i. Also define $x^\eta := \text{argmin}_x\{C(x_l, x_u)\}$.

Monotone Chance Constraints Algorithm (MCC-Algorithm)

Step 0: Initialize Set $\mathcal{N} = \{[a, b]\}$, $v = C(a, b)$, $V = \infty$, and $k = 0$.

Step 1: Choose Node Choose a node $\eta = [x_l, x_u] \in \mathcal{N}$. Call the oracle to find if $x^\eta := \text{argmin}_x\{C(x_l, x_u)\} \in \mathcal{G}$ and/or if $x_u \in \mathcal{G}$.

Step 2: Fathoming

(i) If $x^\eta \in \mathcal{G}$, set $V^k = C(x_l^\eta, x_u^\eta)$, set $\bar{x}^k = \text{argmin}_x\{C(x_l^\eta, x_u^\eta)\}$. Fathom η and set $k = k + 1$. Return to Step 1.

(ii) Else if $x_u^\eta \notin \mathcal{G}$, fathom η. Return to Step 1.

(iii) Else, continue to Step 3.

Step 3: Branch Choose $\text{argmax}_i\{x_{u,i}^\eta - x_{l,i}^\eta\}$. Create two new nodes $\eta_1 = [x_l^\eta, x_u^\eta - (\frac{e_i}{2}x_u^\eta - x_l^\eta)]$ and $\eta_2 = [x_l^\eta + (\frac{e_i}{2}x_u^\eta - x_l^\eta), x_u^\eta]$.

Step 4: Updating Compute $C(x_l^{\eta_1}, x_u^{\eta_1})$ and $C(x_l^{\eta_2}, x_u^{\eta_2})$. Set $v = \min_{\eta\in\mathcal{N}}\{C(x_l^\eta, x_u^\eta)\}$. Fathom all $\eta \in \mathcal{N}$ such that $C(x_l^\eta, x_u^\eta) \geq V$. Return to Step 1.

2. Convergence

Define k as the iteration index of the branch-and-bound algorithm with partition $\eta_k = [x_{l,k}, x_{u,k}]$. Define v_k and V_k as the lower and upper bounds on η_k. The following definitions come from Horst et al. (2000).

DEFINITION C.1. A bounding operation is called *consistent* if at every step any unfathomed partition element can be further refined, and if any infinite sequence $\{k_q\}$ of successively refined partition elements satisfies

$$\lim_{q\to\infty}(V_{k_q} - v_{k_q}) = 0$$

DEFINITION C.2. A selection operation is called *complete* if for every hyper-

rectangle $[x_l, x_u]$ we have

$$\min_x \{f(x) : x \in \mathcal{X} \bigcap \mathcal{G} \bigcap [x_l, x_u]\} \geq V := \lim_{k \to \infty} V_k$$

DEFINITION C.3. A branching strategy is *exhaustive* if

$$\lim_{q \to \infty} |x_{l,k_q} - x_{u,k_q}|_1 = 0$$

for all infinite sequences $\{k_q\}$ of successively refined partition elements.

LEMMA C.4. *The following is true of the MCC-Algorithm*

(i) The branching strategy is exhaustive.

(ii) The bounding operation is consistent

(iii) The selection operation is complete

Proof.

(i) For any node η, we partition on the longest edge in Step 3 of the MCC-Algorithm. This guarantees that the branching strategy is exhaustive.

(ii) The exhaustiveness of the branching strategy implies that the bounding operation is consistent.

(iii) The completeness of the selection operation is guaranteed by the bounds and fathoming given in Proposition B.7 as well as the bound improving strategy of searching the node with the minimum lower bound each iteration.

□

Using these definitions and Lemma C.4, we can now show convergence using standard convergence proofs for global optimization branch-and-bound algorithms. The next theorem shows that if the algorithm terminates in a finite number of steps, then the solution that it finds is the optimal solution. This theorem does not imply

that the algorithm necessarily terminates and it is left to the theorems following it to show that the algorithm converges to the optimal solution in the case where it requires an infinite number of steps.

THEOREM C.5. *If the MCC-Algorithm terminates, then it terminates with a global optimal solution to problem* (5.1) *or resolves that problem* (5.1) *is infeasible.*

Proof. Suppose that the MCC-Algorithm finds a feasible point $\hat{x}$ after finite steps and has terminated. Since the feasible region is non-empty, problem (5.1) has an optimal solution x^*. If $f(x^*) < f(\bar{x})$, then by the exhaustiveness of the branching rule there exists $x^* \in [x_l, x_u]$ such that $[x_l, x_u]$ is some leaf node in the branch-and-bound tree. Therefore since x^* is feasible for the chance constraint (5.1b), x_u is also feasible for constraint (5.1b) so the node was not fathomed by infeasibility. Also since $C(x_l, x_u) \leq f(x^*) < f(\bar{x})$, the node was not fathomed for being above the upper bound. Therefore, the algorithm would not have terminated without further refining the node $[x_l, x_u]$ which is contradiction to our supposition that the algorithm terminated so $f(x^*) = f(\bar{x})$.

Suppose that the algorithm terminates without identifying any feasible solutions. If x^* is a feasible point for problem (5.1), then by the exhaustiveness of the branching rule there exists $x^* \in [x_l, x_u]$ such that $[x_l, x_u]$ is some leaf node in the branch-and-bound tree. Since x^* is a feasible point, x_u is a feasible point for constraint (5.1b) so the node would have been further refined. This contradicts our statement that the algorithm terminated. Therefore, no feasible solution must exist. □

The following two theorems from Horst et al. (2000) establish the convergence of the MCC-Algorithm for the case when the algorithm does not terminate.

THEOREM C.6 ((Horst et al., 2000) Theorem IV.2)**.** *In the case where the branch-and-bound continues for an infinite number of steps, suppose that the bounding op-*

eration is consistent and the selection operation is complete. The the procedure is convergent and

$$\lim_{k\to\infty} f(\bar{x}^k) = f(x^*)$$

where $f(x^*)$ *is an optimal solution to problem* (5.1)

Proof. □

COROLLARY C.7 ((Horst et al., 2000) Corollary IV.2)**.** *If* $f : \mathcal{R}^n \to \mathcal{R}$ *is continuous,* $\mathcal{X} \bigcap \mathcal{G}$ *is closed, and* $Q(x^0) := \{x \in \mathcal{X} \bigcap \mathcal{G} : f(x) \leq f(x^0)$ *is bounded. In an infinite branch-and-bound procedure with a consistent bounding operation and a complete selection operation, every accumulation point of* $\{\bar{x}^k\}$ *solves problem* (5.1)*.*

Proof. □

While this proof only guarantees that the algorithm will converge to the optimal solution after a possibly infinite number of steps, it is still useful. The main goal of this algorithm is to obtain tight bounds on the optimal objective value. Given that the size of the problem that can be solved exactly is so limited, a tight bound on the optimal objective for larger instances would be extremely useful.

However, computationally we would like to guarantee the convergence of this algorithm to the optimal solution in a finite number of steps. To do this we take advantage of the following property of the monotonic chance constraint (5.1b) for the case when the random parameters are discretely distributed. Under the discrete assumption, the realizations of the random variables $\omega \in \Omega$ are known as scenarios. Define a scenario as satisfied if the constraints associated with that scenario are satisfied. Since $|\Omega| < \infty$, the chance constraint can be reformulated as the requirement that at least $\lceil \alpha * |\Omega| \rceil$ of the scenarios be satisfied. Define $\mathcal{S}_x$ and $\mathcal{U}_x$ as the set of scenarios satisfied and unsatisfied respectively at point x.

PROPOSITION C.8. *Assuming $|\Omega| < \infty$, for any node $\eta = [x_l, x_u]$ of the MCC-Algorithm, the following are true*

(i) $\mathcal{S}_{x_l} \subseteq \mathcal{S}_{x_u}$

(ii) $\mathcal{U}_{x_u} \subseteq \mathcal{U}_{x_l}$

With Proposition C.8 an optimal solution for any node of the branch-and-bound tree can be found by solving the following mixed-binary program. Where $\mathcal{Z} = \mathcal{S}_{x_u} \setminus \mathcal{S}_{x_l}$, M is a large number, p_ω is the probability of scenario ω, and the chance constraint is enforced with a knapsack constraint (5.5c).

$$\begin{aligned}
\text{Min} \quad & f(x) && \text{(5.5a)}\\
\text{s.t.} \quad & g_i(x,\omega) - M z_\omega \leq 0 \;\; i = 1 \dots m \;\; \forall \omega \in \mathcal{Z} && \text{(5.5b)}\\
& \sum_{\omega \in \mathcal{Z}} p_\omega z_\omega \geq 1 - \alpha && \text{(5.5c)}\\
& x \in \mathcal{X} \bigcap [x_l, x_u], \;\; z_\omega \in \{0,1\} \; \omega \in \mathcal{Z} && \text{(5.5d)}
\end{aligned}$$

To make the MCC-algorithm finite, any node of the branch-and-bound tree can be pruned by solving problem (5.5). This would be used to stop the branching once a given level of fineness in the partition has been reached. Depending on the application and the properties of the original problem, this may be too computationally intensive to be useful. However, in many cases small mixed-binary problems ($|\mathcal{S}_{x_u} \setminus \mathcal{S}_{x_l}| < 50$) can be solved quickly be commercial solvers such as CPLEX. So the finiteness modification may be computational feasible for guaranteeing finite convergence of the algorithm or at least finding strong upper bounds.

3. Initial Implementation and Results

Our implementation of the algorithm centered around developing effective data structures to deal with the large number of branch-and-bound nodes that will need to be searched. Each iteration, the algorithm searches the node with the lowest objective value. This means that the nodes have to be arranges in order from lowest objective to highest. After selection the node with the lowest objective function, it is partitioned into two other nodes and the lower bound on both of those nodes is computed. The LP solved at each node is small so this is not computationally intensive. The algorithm then follows the fathoming and bounding rules and iterates until an optimal solution is identified and optimality is proven.

Computationally, the algorithm did not succeed. For vac100, the algorithm searches several hundred thousand nodes in an hour but still has a lower bound of 0.7 when the optimal solution is 65.2. No upper bound is found in this time either. Any larger size instances would be even worse because the relaxation at each node is the same for all test instances. This gets at the fundamental weakness of the algorithm in that there is too much information being lost by dealing implicitly with the chance constraint. Essentially, even though the basic design of the algorithm to be able to search a large number of nodes quickly worked, branch-and-bound was not able to fathom nodes quickly enough for this to make a difference. In the final section we will give some insights into why the MCC-Algorithm failed.

D. Conclusions

The MCC-algorithm is significant in that it gives a reformulation of and an exact solution method for joint chance-constrained programs. The benefit of this approach is that the difficulty of the chance-constraint can largely be avoided by dealing with

it implicitly. The algorithm has potential for being scalable in terms of the number of scenarios. However, the results from the implementation were disappointing and we will give some explanations for why this might be the case. We will give some of our insights into why this form of branching may be significantly less effective than branching on the binary decision variables of the MIP formulation. Finally we will give a few ideas for possibly modifying the MCC-Algorithm so that it is computationally feasible.

The primary goal of the MCC-Algorithm was to have a method for solving joint chance-constrained problems that is relatively insensitive to the number of scenarios of the problem. This would avoid the scaling issue that we encountered in our earlier methods. The implementation of the algorithm was in fact insensitive to the number of scenarios in the sense that huge numbers of nodes could be searched quickly because the chance constraint was dealt with implicitly. This is the main success of the algorithm and the main reason that the idea is still attractive.

The main problem with the MCC-Algorithm is that it is a trade-off between the advantages gained by dropping the chance-constraints and dealing with them implicitly and the loss of information about the feasible space due to dropping the chance-constraints. From looking at the best lower bound found by the algorithm, there was a large amount of symmetry in the problem. Basically, without the chance-constraint, there was not enough information left to differentiate between different solutions. Thus the lower bound never improved. Some more evidence for why this is the case is the fact that after every partition, the previous optimal solution to the node LP is always optimal for one of the new nodes. This means that unless the node can be fathomed because the upper righthand corner does not allow feasibility of the chance constraint, the lower bound found by the algorithm will not improve. We found that this is the primary failing of the algorithm. It can search millions of

nodes, but the lower bound improves extremely slowly.

The paper that inspired this research (Cheon et al., 2006) was able to solve some test instances. It is interesting to contrast the two algorithms to figure out why one worked and the other did not. Looking at the two algorithms, the main difference is that the MCC-Algorithm is partitioning the continuous space of the original decision variables while the earlier algorithm is partitioning the space defined by new variables (one for each constraint of the joint chance-constraint). The test instances solved in the earlier paper were limited to problems with 7 constraints within the joint chance-constraint. There are a number of applications for which this would be enough, but there are also many in which the size of the joint chance-constraint would be much bigger.

CHAPTER VI

OPTIMAL VACCINE ALLOCATION UNDER UNCERTAINTY

A. Introduction

Vaccination is one of the primary strategies used by public health authorities to control human infectious diseases. Mathematical models have long played a major role in identifying and evaluating strategies to allocate resources in order to guarantee maximum effectiveness of vaccination in controlling infectious disease outbreaks. Three primary modeling approaches have been used in this effort – deterministic analytical models, stochastic analytical models, and computer simulations. The determination of optimal vaccination strategies may be sensitive to changes in model parameter values, however, so there is a need for new methods that can take parameter uncertainty into account in order to find more robust vaccination policies. We present here a description of one such method, stochastic programming, and illustrate how this method can improve our ability to find optimal vaccination strategies.

The goal of most deterministic and stochastic epidemiological models addressing vaccination strategies is to derive appropriate strategies analytically. Deterministic models focused on identifying reasonable vaccination strategies for the control of infectious diseases date back to at least the 1960s (early papers include, for example, Brogger, 1967; Hethcote and Waltman, 1973; Revelle et al., 1967; Waaler et al., 1962). In general, deterministic vaccination models fall into two major groups. The majority of these models are used to evaluate predetermined vaccination strategies to see which of the proposed strategies may be most effective. Analysis of most of these models generally involves exploration of the steady state behavior of the model system and determination of an epidemic threshold. The effectiveness of different proposed vacci-

nation strategies in reducing the susceptible population below the epidemic threshold for the minimum cost is then evaluated. In some of the recent more complex models, computer simulation is used to assess the effectiveness of different strategies. Models of this type have been developed for a number of infectious diseases, including tuberculosis (Brogger, 1967; Waaler et al., 1962), measles (Agur et al., 1993; Babad et al., 1995; Hethcote, 1988; Shulgin et al., 1998), rubella (Anderson and May, 1983; Dietz, 1981; Hethcote, 1983; Knox, 1980), pertussis (Hethcote, 2002, 1997, 1999; Hethcote et al., 2004), and respiratory illnesses (Pourbohloul et al., 2005).

The second group of deterministic vaccination models do not start with predetermined strategies; rather, they center on the use of optimization methods in combination with deterministic epidemic models to identify the optimal vaccination strategy. Optimization methods have been used both in a theoretical framework (Hethcote and Waltman, 1973) and to guide the development of vaccination policies for specific diseases, including tuberculosis (Revelle et al., 1967), influenza (Longini et al., 1978), and smallpox (Frauenthal, 1981).

A number of stochastic models have also been developed to determine optimal vaccination strategies. For example, Ball et al. (1997) develop an SIR epidemic model with both local mixing at the household level and global mixing at the community level. They introduce the notation, R_*, to represent the threshold parameter for a community of households. They analyze the case of a perfect vaccine and show that under this condition, a strategy that allocates vaccines to those households with the

---

Epidemiological models are often formulated as a series of compartments corresponding to different disease states, e.g. susceptible, exposed, infectious, recovered, etc. The models are then referred to by the series of capital letters that corresponds to the compartments within the basic model structure. For example, an SIR model considers individuals to be either susceptible (S), infectious (I), or recovered (R) and to progress through the stages in that order; an SIS model would consist of the stages susceptible - infectious - susceptible and would represent a disease for which there was no immunity.

largest number of unvaccinated individuals is best for reducing R_* to a level that will control an epidemic. Becker and Starczak (1997) study vaccination policies in a stochastic SIR model divided into a community of households. They derive a closed form equation for the post-vaccination reproductive number, R_*, then formulate and numerically solve a linear program to find the minimum vaccination coverage under the constraint $R_* \leq 1$. This constraint ensures that the disease will tend to die out. (Becker and Starczak (1997) use the notation R_{HV} rather than R_*, but the concepts are equivalent.) Drawing upon the earlier work of Ball et al. (1997), Ball and Lyne (2002) consider the case of an all-or-nothing vaccine where a person is either totally immune following vaccination or the vaccine does not work at all. They show that if the sequence $\{n\mu_n\}$ is convex, where μ_n is the mean size of a local outbreak within a household of size n, then the optimal solution to the linear programming problem formulation of Becker and Starczak (1997) can be characterized explicitly. Ball et al. (2004) use the model described by Ball and Lyne (2002) to address the question of optimal allocation of vaccines. They show that an explicit characterization of the optimal vaccination strategy is only possible in certain special cases, such as proportionate mixing. Müller (1997) uses an SIRS epidemic model to derive optimal vaccination strategies in an age structured population and compares the conditions needed for optimal vaccination coverage of individuals as opposed to entire populations. Hill and Longini (2003) use a general framework that could apply to several epidemic situations (e.g., diseases with permanent immunity (SIR models), incorporation of latent periods (SEIR models), or no immunity (SIS models) with and without vital dynamics). They develop a method to derive optimal vaccination strategies for populations divided into m heterogenous subgroups and fully examine the use of the model in populations with two subgroups and proportionate mixing.

Very few of these analytical models include discussion of the effect of parameter

uncertainty on the vaccination policies identified and/or evaluated but this uncertainty can have major consequences. For example, Longini et al. (1978) show that the optimal allocation of vaccines derived from their influenza model is highly sensitive to both the epidemiological characteristics of the virus and to the choice of the objective function used in the optimization process. Similar conclusions about the sensitivity of model outcomes to epidemiological and structural uncertainty are reached by Bansal et al. (2006), who use a contact network model to compare morbidity-based strategies that target high prevalence populations and mortality-based strategies that target high risk populations, Dushoff et al. (2007) who use a very simple model to explore the consequences of different vaccine allocation strategies, and Clancy and Green (2007) who use a Bayesian-decision theoretic approach and a general stochastic SIR model with a homogenous population under parameter uncertainty.

Computer simulation models within a fully stochastic framework have also been used to assess the effectiveness of various potential strategies to control infectious disease spread. Most of these papers focus on pure control strategies, such as antivirals, vaccines, quarantine, and travel restrictions, that are implemented over the entire population. The effect of these strategies used individually and in different combinations are analyzed through simulation (see, for example, Ferguson et al., 2006; Germann et al., 2006; Morris et al., 2001). As an example of a simulation model focused specifically on the identification of an optimal vaccination strategy, Patel et al. (2005) use a genetic algorithm within the framework of a simulation of pandemic influenza. Their algorithm is a heuristic; in other words, it is designed to find feasible solutions to the problem but there is no guarantee for how close those solutions are to the true optimal solution. It is important to note that at the present time heuristic approaches are all that are available for this class of problems. Also, due to the large amount of computer time per simulation run, none of the simulation

papers discussed here consider the effects of parameter uncertainty.

Both analytical models and simulation models have weaknesses that must be considered in light of the goals of a modeling project. A major criticism of analytical deterministic and stochastic vaccination models that allow closed form representations of R_* is that many assumptions are needed to have this property. These assumptions generally result in a model that is only a rough approximation of the actual spread of a disease through a population. Despite this weakness, analytical models can still be useful because they can give a clearer picture of the crucial parameters in a model (Ball et al., 2004). For the task of identifying appropriate vaccination strategies, analytical models provide a good way to find mixed strategies that can provide insight into the groups that need to be particularly targeted by health authorities. Simulation models, which generally incorporate more realistic assumptions about population structure and disease transmission processes, are usually limited to pure or simple strategies because of the time required to run simulations given their complexity and the necessity of running them repeatedly because of their inherent randomness. Another important use of optimal strategies derived from analytical disease models is as a benchmark for strategies found via a heuristic on simulation models. The cost and effectiveness of the heuristic strategies can be checked against the optimal strategies of the analytical models to provide information on the quality of the heuristic strategies.

The complexity of human interactions means that parameter estimation for epidemiological models is notoriously difficult. Thus, vaccination policies found for any kind of model should be considered very carefully, especially if the uncertainty of the parameters is not taken into account. Policies derived from models with deterministic parameters may not be robust in the sense that even an optimal strategy might be highly suboptimal or even infeasible if parameters are changed slightly. Stochastic programming is a popular method for incorporating uncertainty in mathematical op-

timization problems by finding optimal decisions given that some parameter values are not deterministically known (Birge and Louveaux, 1997).

Using stochastic programming to include parameter uncertainty when finding optimal vaccination strategies can give several clear benefits. The stochastic programming framework allows for more robust vaccination strategies that are not as reliant on point estimates of parameter values. Stochastic programming can also help identify parameters to which optimal decisions are particularly sensitive, and so can provide guidance for allocation of resources for estimating parameters of the model.

The formulations presented in this chapter include chance constraints to require $R_* \leq 1$ with at least a minimum probability, a random objective to minimize the probability that an epidemic will occur under resource constraints, and a cost-benefit formulation making the required probability for $R_* \leq 1$ a decision variable. The problem is formulated in a general framework that is valid for a wide class of epidemic models. We illustrate the stochastic programming formulation using the heterogeneous household model of Becker and Starczak (1997) and we provide a numerical example to show why including parameter uncertainty is important when devising an optimal vaccination strategy.

The rest of this chapter is organized as follows: Section B gives a short introduction to stochastic programming and presents a general problem framework for finding optimal vaccination strategies under parameter uncertainty including a discussion of applying this technique to a variety of disease spread models. Section C describes the model of Becker and Starczak (1997) and some of the basic results and extensions. The section continues with an example reformulation of their linear program as a stochastic program with probabilistic constraints. Section D gives a numerical illustration of this technique focusing on the value of information and the effects of not including uncertainty. Section E finishes with some conclusions and future work.

B. Stochastic Programming

1. Stochastic Programming Formulations

For this chapter, we consider three possible formulations of the problem as a chance-constrained program. We consider a disease to be controlled if $R_* \leq 1$ and not controlled otherwise. Setting a reliability parameter α, the first formulation minimizes the cost of vaccination under a chance constraint that requires $R_* \leq 1$ with at least probability α. The second is the case where the vaccination supply is limited and the probability that the vaccination is insufficient to control the disease spread is minimized. The final formulation that we consider is a cost/benefit analysis with α as a variable instead of a parameter. These formulations are general and can be applied to a number of disease spread models; hence, the specific structures of the decision variables and constraint sets are all problem dependent.

In the context of finding vaccination policies, decision variables $x \in \mathcal{R}^n$ define possible vaccination policies implied by the model. These decision variables are constrained by an arbitrary set X, which defines allowable vaccination policies. The post-vaccination reproductive number is a function both of the vaccination policy x and the realization of the random variables ω and is given by $R_* = h(x, \omega)$. The cost of a vaccination policy is a function of x and is given by $c(x)$.

Equations (6.1a)-(6.1c) give a general formulation of the problem as a chance constrained stochastic program. The objective function (6.1a) is to minimize the cost of the vaccination policy. The constraint (6.1b) is the probabilistic constraint and requires that $R_* \leq 1$ with probability greater than or equal to the reliability parameter, α. The constraint (6.1c) defines the feasible space of allowable vaccination policies.

$$\min \quad c(x) \tag{6.1a}$$

$$\text{s.t.} \quad \mathbb{P}\big(h(x,\omega) \leq 1\big) \geq \alpha \tag{6.1b}$$

$$x \in X \tag{6.1c}$$

Higher values of α mean higher costs for the optimal prevention strategy since the disease must be prevented for a larger number of scenarios. The parameter is often chosen through computational experimentation, trading off the much higher cost of policies under extreme values of α with the costs of allowing too many infeasible policies.

A possible criticism of using chance constraints to formulate this problem is that the goal of policy makers is to prevent a major epidemic from ever occurring. Explicitly finding a vaccination strategy that allows for an acceptable failure rate clearly goes against this ideal. However, the problem of finding the vaccination strategy that requires $R_* \leq 1$ for all scenarios consists of defining the worst possible parameter values for the disease and solving the deterministic program for those values. The weakness of this approach is that strategies that are feasible for the worst values are often much more expensive than a strategy that is feasible for the vast majority of cases. Therefore in terms of vaccination strategies, the strategy to prevent every possible epidemic might just mean that everyone has to be vaccinated. This is not as useful as knowing how many fewer vaccine doses are needed to control the spread of a disease with high probability. Also, since estimating upper bounds is just as inexact as estimating average values, there is no guarantee that the upper bound estimate will actually mean that $R_* \leq 1$ in all cases. We feel that chance constraints allow for a more natural way to plan for bad scenarios and better reflect disaster planning in the

real world. If failure rate of an optimal vaccination strategy found with formulation (6.1) is too low, then the value of α can always be increased.

Another situation for which an optimal vaccination policy might be required is when the vaccination budget is limited. In this case, it is not always possible to vaccinate in a way that makes $R_* \leq 1$, but it is still necessary to distribute the vaccines effectively. A measure of the effectiveness of a vaccine distribution with limited supplies that is analogous to the chance constraints is to optimally distribute the vaccines while minimizing the probability that $R_* \geq 1$. Adding a budget parameter B, equations (6.2a) - (6.2c) give a problem formulation to do this.

$$\begin{aligned} \min \quad & \mathbb{P}\big(h(x,\omega) \geq 1\big) && \text{(6.2a)} \\ \text{s.t.} \quad & c(x) \leq B && \text{(6.2b)} \\ & x \in X && \text{(6.2c)} \end{aligned}$$

A third possible problem formulation is to explicitly consider the costs and benefits of lowering the probability that the reproductive number is less than one. Now, instead of a problem parameter, the reliability α is a decision variable of the problem. The costs of a less reliable vaccination strategy are modeled with the cost function $p(\alpha)$ leading to the following mathematical program.

$$\begin{aligned} \min \quad & c(x) + p(\alpha) && \text{(6.3a)} \\ \text{s.t.} \quad & \mathbb{P}\big(h(x,\omega) \leq 1\big) - \alpha \geq 0 && \text{(6.3b)} \\ & x \in X,\ 0 \leq \alpha \leq 1 && \text{(6.3c)} \end{aligned}$$

2. Application to Various Disease Spread Models

This section will give a short introduction to some of the types of disease spread models for which stochastic optimization can be used to find optimal vaccination strategies. We discuss two main types of models, those for which an explicit function for $h(x, \omega)$ can be derived and those for which $h(x, \omega)$ can only be calculated implicitly through simulation. We will include a few remarks on solution methods for these different types of problems. The results here refer to problem (6.1), but they can easily be extended for the other two formulations.

Under general random parameter distributions, even finding a feasible solution to problem (6.1) may be impossible since computing the probability in constraint (6.1b) may be too computationally intensive (Prékopa, 2003). In order to avoid this, it is necessary to assume that the distributions are discrete. This assumption is not too limiting as a discrete distribution can always be created from a continuous one by sampling.

For problems with discretely distributed random data, Morgan et al. (1993) gives a formulation of the problem as a mixed-binary program that is more amenable to solution. There are a finite number of scenarios ω, each with a probability p_ω. Every scenario has a corresponding binary variable $z_\omega \in \mathcal{B}$ which takes the value of 0 if the disease is controlled in that scenario by the optimal vaccination policy. The variable takes a value of 1 if the disease is not controlled in that scenario. In the following formulations M is a sufficiently large number to guarantee that constraint (6.4b) is satisfied under scenario ω whenever $z_\omega = 1$. A knapsack constraint (6.4c) ensures that the probabilistic constraint is satisfied by forcing the sum of the probabilities of scenarios where epidemics occur to be less than $1 - \alpha$.

$$\begin{aligned}
\min \quad & c(x) && \text{(6.4a)}\\
\text{s.t.} \quad & h(x,\omega) - Mz_\omega \leq 1 \ \forall \omega \in \Omega && \text{(6.4b)}\\
& \sum_{\omega \in \Omega} p_\omega z_\omega \leq 1 - \alpha && \text{(6.4c)}\\
& x \in X,\ z \in \mathcal{B}^{|\Omega|} && \text{(6.4d)}
\end{aligned}$$

In the general case, the functions $h(x,\omega)$, $c(x)$, and the constraints that define the feasible set X are nonlinear, nonconvex functions in terms of the decision variables x. However, solving such a nonlinear mathematical programming problem may not possible. It is possible to use nonlinear programming techniques or heuristics to find local minima or feasible solutions that one hopes will be good enough (Horst et al., 2000), but these do not give any guarantee of solution quality. The formulation is more computationally tractable in the case where X, $c(x)$, and $h(x,\omega)$ are given by convex functions in terms of x. Such problems can then be solved using commercial optimization programs. Our example formulation gives an instance where X, $c(x)$, and $h(x,\omega)$ are given by linear functions.

The other class of epidemic models for which the stochastic programming framework can possibly be used is that for which R_* can only be calculated by means of simulation. In this case, a problem of type (6.1) only exists implicitly and it is necessary to use heuristic methods in the simulation optimization framework in order to search for a feasible solution Tekin and Sabuncuoglo (2004). Simulation optimization methods are usually not particularly good at finding optimal solutions, but even a feasible solution to problem (6.1) can be valuable for defining a robust vaccination policy.

REMARK B.1. Many of the diseases for which vaccination is an option vary sea-

sonally in their transmission, a feature that is usually modeled by incorporating a sinusoidal or similarly varying transmission parameter. Most vaccine optimization models do not take seasonality into account, however. One simple way to apply the stochastic programming framework to find an optimal vaccination policy in such a situation is to estimate the parameter distributions for the high season of the disease when the transmission parameter is maximal. In this case, $\mathbb{P}(R_* \leq 1) \geq \alpha$ for the worst season, which means that the reliability requirement is guaranteed to be satisfied for the entire year. Of course using a strategy that guarantees reliability over an entire year based on transmission rates during the worst part of the year can be expensive, especially if resources are limited. In this case it might be more expeditious to use a cost/benefit formulation that is tied to the cyclicity of the transmission parameter.

C. Example Model

We next present an example formulation of a stochastic program in the case where the constraints that define $c(x)$, $h(x, \omega)$, and X are all linear. In Section 1, the SIR epidemic model and a formulation of the optimal vaccination problem as a linear program as given in Becker and Starczak (1997) are described. Note that although the linear programming formulation given here is the same as their formulation, we have changed the notation for clarity. In Section 2, our extension of this linear program to a stochastic program is explained.

1. Linear Programming Formulation

As a first step in formulating their linear program for finding optimal vaccination strategies, Becker and Starczak (1997) compute R_0 and R_* (the post-vaccination re-

production number) for their model. They assume in the model that the disease spreads quickly within individual households and spreads more slowly between them through close contacts between infected and susceptible members of different households. To ensure that the problem constraints are linear, they also assume proportionate mixing between households. This allows them to find a closed form equation for the post-vaccination reproduction number.

To formulate the program, it is necessary to define groups within the population that have different susceptibilities and infectivities. It is also necessary to define the different types of families that make up the overall population. The decision variables x_{fv} of the program represent the proportion of households of type f that are vaccinated under policy v. The rest of the model parameters and their descriptions are given in Table VIII.

The full formulation of the linear program is given in equations (6.5a)-(6.5d). The objective function (6.5a) minimizes the vaccine coverage. The first constraint (6.5b) balances all the decision variables for each family type, ensuring that the proportions assigned sum to one. The second constraint (6.5c) requires that that reproductive number of the disease be brought below one. This constraint is a linear function of a_{fv}, which is itself a function of the parameters given by equation (6.6). The parameter a_{fv} is derived in (Becker and Starczak, 1997) and the value $\sum_{f\in F}\sum_{v\in V} a_{fv}x_{fv}$ gives the post-vaccination reproduction number of the model.

Table VIII. Vaccination Stochastic Programming Model Parameters

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}	computed parameter for impact of immunization decisions
μ_F	the average size of a household
Parameters to compute $a_{nv}(\omega)$	
m	the average contact rate of infected people
u_t	the relative infectivity of people of type t
s_t	the relative susceptibility of people of type t
b	the transmission proportion within a household
ϵ	the vaccine efficacy
Decision Variables	
x_{fv}	the proportion of families of type f vaccinated under policy v

$$\min : \quad \sum_{f\in F}\sum_{v\in V}\sum_{t\in T} v_t h_f x_{fv} \tag{6.5a}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{fv} = 1 \quad \forall f \in F \tag{6.5b}$$

$$\sum_{f\in F}\sum_{v\in V} a_{fv} x_{fv} \leq 1 \tag{6.5c}$$

$$0 \leq x_{fv} \leq 1 \quad \forall f \in F, v \in V \tag{6.5d}$$

$$a_{fv}(\omega) = \frac{mh_f}{\mu_F}\Big(\sum_{t\in T} u_t s_t\big[(1-b)(f_t - v_t\epsilon) + bv_t\epsilon(1-\epsilon)\big] + b\sum_{t\in T}\sum_{r\in T} u_r s_t (f_t - v_t\epsilon)(f_r - v_r\epsilon)\Big) \tag{6.6}$$

In the case where vaccination supplies are limited and planners wish to minimize R_*, the formulation can be modified in the following manner. The left-hand side of constraint (6.5c) is moved to the objective and is minimized, and a constraint limiting the total number of vaccine doses that can be allotted to D is created from the objective function (6.5a). This formulation is given by equations (6.7a) - (6.7d).

$$\min \quad \sum_{f\in F}\sum_{v\in V} a_{fv}x_{fv} \tag{6.7a}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{fv} = 1 \quad \forall f \in F \tag{6.7b}$$

$$\sum_{f\in F}\sum_{v\in V}\sum_{t\in T} v_t h_f x_{fv} \leq D \tag{6.7c}$$

$$0 \leq x_{fv} \leq 1 \quad \forall f \in F, v \in V \tag{6.7d}$$

Becker and Starczak (1997) and Ball et al. (2004) show that this linear program does not allow an easy characterization of the optimal strategy, meaning that the optimal strategy may not be easy to implement. However, they claim that it is still useful from a policy standpoint because it gives insight into groups that should be targeted in any vaccination plan. Constraints can be added to the model if more implementable plans are desired. For example, to limit policies to those where either an entire household is vaccinated or no members are, all decision variables corresponding to partially vaccinating a household are set to zero.

2. Stochastic Programming Formulation

We extend the linear programming model of Becker and Starczak (1997) to the stochastic setting by considering the following parameters as random variables: the vaccine efficacy ϵ, the average contact rate of infected people m, the relative infectivities and susceptibilities of people of different types u_t and s_t, and the transmission proportion within a household b. The rest of the parameters of the model can be estimated more easily than these from census data and similar sources so they are assumed to be deterministic in our model. Depending on the goals of the modeler, a different number of random parameters could be included in the stochastic model. Methods for estimating the distributions of the random parameters can be found in Becker (1995).

We reformulate problem (6.5) as a stochastic program with probabilistic constraints considering the previously mentioned parameters as random variables. This formulation is a special case of the general structure that was defined in Section 2. The stochastic formulation is given by equations (6.8a) - (6.8d).

$$\min \quad \sum_{f\in F}\sum_{v\in V}\sum_{t\in T} v_t h_f x_{fv} \tag{6.8a}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{fv} = 1 \quad \forall f \in F \tag{6.8b}$$

$$\mathbb{P}\Big(\sum_{f\in F}\sum_{v\in V} a_{fv}(\omega) x_{fv} \leq 1\Big) \geq \alpha \tag{6.8c}$$

$$0 \leq x_{fv} \leq 1 \quad \forall f \in F, v \in V \tag{6.8d}$$

where α_{fv} now becomes a function of the random parameters that we have defined

$$a_{fv}(\omega) = \frac{m(\omega)h_f}{\mu_F}\Big(\sum_{t\in T} u_t(\omega)s_t(\omega)\big[(1-b(\omega))(f_t - v_t\epsilon(\omega)) + b(\omega)v_t\epsilon(1-\epsilon)\big] + b\sum_{t\in T}\sum_{r\in T} u_r(\omega)s_t(\omega)(f_t - v_t\epsilon(\omega))(f_r - v_r\epsilon(\omega))\Big) \quad (6.9)$$

Formulation (6.8) can be reformulated as a mixed-binary program as was described in Chapter I. The remainder of this paper will be concerned with this mixed-binary formulation (given in equations (6.10a) - (6.10e)), which can be solved without any modification. A mixed-binary formulation can be similarly derived for the case with a limited vaccination budget.

$$\min \quad \sum_{f\in F}\sum_{v\in V}\sum_{t\in T} v_t h_f x_{fv} \quad (6.10a)$$

$$\text{s.t.} \quad \sum_{v\in V} x_{fv} = 1 \quad \forall f \in F \quad (6.10b)$$

$$\sum_{f\in F}\sum_{v\in V} a_{fv}(\omega)x_{fv} - Mz_\omega \leq 1 \ \forall \omega \in \Omega \quad (6.10c)$$

$$\sum_{\omega\in\Omega} p_\omega z_\omega \leq 1-\alpha \quad (6.10d)$$

$$0 \leq x_{fv} \leq 1 \ \ \forall f \in F, v \in V, \ z_\omega \in \{0,1\} \ \forall \omega \in \Omega \quad (6.10e)$$

D. Numerical Example

We generated a small set of random test instances in order to illustrate the effect of random parameter values on the optimal policies found by mathematical programming. The goal of this section is to use example instances of the problem to show why stochastic programming with probabilistic constraints is useful for finding optimal vaccination policies. In particular, this section will show the importance of using random distributions for the model parameters rather than point estimates in terms

of the cost and effectiveness of the vaccination policy given in the solution.

To set up our random test instances, we generated values and distributions for the model parameters, which included family group parameters that define the makeup of the population and parameters that control the spread of the disease. For this example we chose parameter distributions that seemed plausible based on information in the epidemiological literature; to properly estimate them is beyond the scope of this project.

The family group parameters included three different types of people: children, adults, and the elderly. We defined 30 possible family groupings comprised of different numbers of these types, the details of which can be found in Appendix Table 3. The disease parameters were more difficult to estimate. According to Longini et al. (2004) a plausible value of R_0 for influenza is estimated to be around 2.0. We defined our parameters so that the distribution of the R_0 values would be mostly in the interval $[1.5, 2.5]$. For the efficacy of the vaccine we assumed a truncated normal distribution with a mean of 0.85 and a standard deviation of 0.1. See Appendix Table 4 for the assumed distributions of the remaining parameters.

Our test instances were created by independently sampling the parameters from their defined distributions. We formulated the problem with probabilistic constraints as given in formulation (6.10). We limited the number of scenarios of the instance to 500 so that the instance could be solved quickly by a commercial solver. We found solutions to the problems by using the mixed-integer programming solver CPLEX 9.0. The objective values of the formulations were weighted so that the numerical value was equal to the percentage of people who would need to be vaccinated in the optimal strategy.

Figure 3 shows how increasing α affects the percentage of people who need to be vaccinated in the optimal strategy. The striking detail in this plot is that the

percentage of people who need to be vaccinated increases relatively linearly when the probability that $R_* \leq 1$ is between 20% and 95%. However, when $\alpha > 0.95$, the required number of vaccine doses increases at a much faster rate. The plot gives evidence for why stochastic programming with probabilistic constraints is a good framework for finding optimal vaccination policies. Since resources for the prevention of disease are limited, it is important to be able to identify a level for which the probability that $R_* > 1$ is low, but the vaccination coverage is not too extreme. This plot shows that this constraint can be satisfied with high probability using relatively few doses of a vaccine, but that increasing the probability that $R_* \leq 1$ beyond that requires a huge increase in vaccine supplies. A plot such as this can be used to set the parameter α. Since the number of doses needed to prevent extra epidemics starts to increase quickly above $\alpha = 0.95$, this value is a reasonable choice for that parameter.

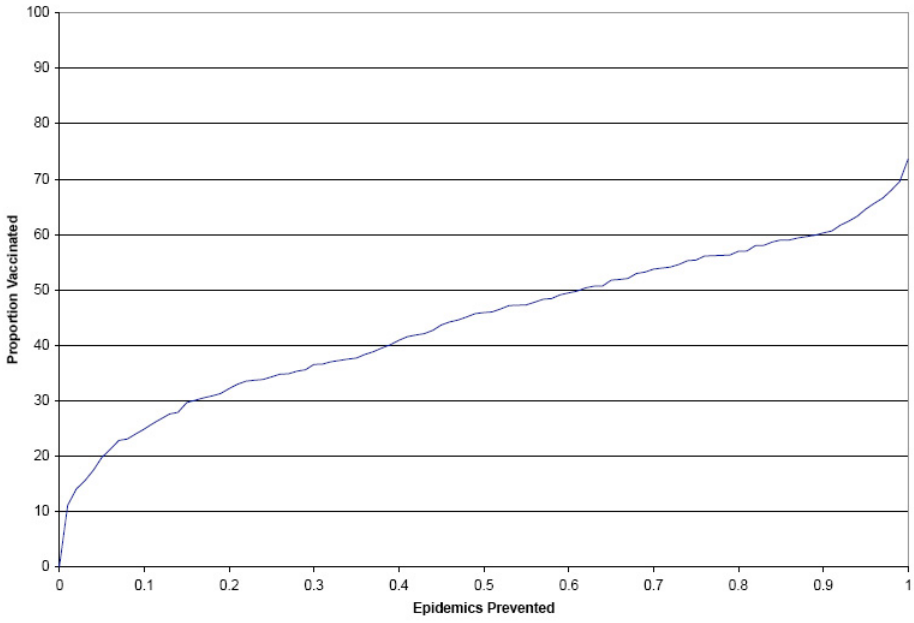

Fig. 3. Plot of Vaccine Proportion vs Epidemic Prevention Rate

We next solved all the test problems with $\alpha = 0.95$, computed the expected

value of the expected solution, the wait-and-see solution, and the value of perfect information. The results of these computations are given in Table IX. In this table SPP designates the stochastic problem solution, EEV designates the expected value of the expected value solution, WS designates the wait-and-see solution, and VPI designates the value of perfect information respectively.

Table IX. Value of Information on Small Vaccination Example

Test Instance	SPP	EEV	WS	VPI
vac500a	64.53	0.52	40.64	23.89
vac500b	65.49	0.57	40.38	25.11
vac500c	66.41	0.56	41.42	25.00
vac500d	66.63	0.56	40.90	25.73
vac500e	65.16	0.56	41.37	23.79
Average	65.54	0.55	40.94	24.70

The SSP column of Table IX shows that $R_* \leq 1$ with probability 95% can be achieved by vaccinating on average 65.54% of the population. This number gives the absolute minimum percentage of the population that would have to be vaccinated in order to guarantee that the chance constraint is satisfied, however this information alone is not particularly valuable in terms of defining a vaccination strategy to prevent a real disease. The optimal decision variable values give the exact proportion of each family type $f \in F$ that need to be vaccinated according vaccination policy $v \in V$. This means that the optimal vaccination policy found by solving formulation (6.1) is not likely to be implementable from a public policy standpoint because it is

unrealistically specific about which people need to be vaccinated. Nevertheless, the optimal objective value is an absolute lower bound on the cost of any effective strategy so it does give us a starting point for budgeting for a real vaccination program. Also, the optimal decision variable values can be used to identify particular groups of people that are crucial to vaccinate and hence should be specifically targeted in the actual vaccination plan.

We computed the expected value of the expected value solution (EEV) for the test instance. To compute the expected value decision, formulation (6.5) was set up and solved using the expected values of the parameter distributions to find optimal decision variable x^*. Then for each scenario ω, we tested whether using the vaccination policy given by x^* satisfies the constraint $h(x, \omega) \leq 1$. As is shown in the EEV column of the table, in this case, an $R_* > 1$ occurs an average of 55.0% of the time. This is clearly unacceptable for a vaccination policy. This result indicates that the effectiveness of a vaccination policy is highly susceptible to how the parameters are distributed. Using the traditional expected values of parameters gives a solution that is not robust enough to be useful.

We also computed the value of perfect information $VPI = WS - SPP$ for our example. To compute this value it was necessary to compute the cost of the optimal objective value using formulation (6.5) for each of the 500 scenarios and then take the average of those values. The average optimal objective percentage of the population to vaccinate when the stochastic data are known is 40.94%. Hence the value of perfect information is 24.70%. This means that if parameters were able to be estimated perfectly, $R_* \leq 1$ with an average of over a third fewer vaccine doses. From a policy standpoint, this helps decide how much effort and resources should be spent finding more exact estimates of the model parameters.

The purpose of setting up and solving a small example like this is to show that

traditional linear programming may be insufficient for finding good vaccination strategies even under the assumptions for which it can be used. In particular, the expected value solution shows that using point estimates of the epidemic parameters is not robust enough to be used to plan for actual epidemics. On the other hand, the stochastic programming solution gives a vaccine allocation program that does not require the vast majority of people to be vaccinated, but is robust enough to prevent most epidemics from occurring. Also, the value of perfect information shows that because parameter uncertainty has a substantial effect on policies that the program returns it may be worthwhile to expend significant effort to better estimate parameters.

E. Conclusions

This paper introduces stochastic programming with chance constraints as a framework for including parameter uncertainty when finding optimal vaccination policies. We give general stochastic programming formulations of the problem that can be used for a wide class of epidemic models. As an example, the linear programming formulation of Becker and Starczak (1997) is extended to this stochastic programming framework. We then use a numerical example to show the large effect that including parameter uncertainty has on the optimal vaccination strategies. We believe that since accurate parameter estimation can be extremely difficult for epidemic models, ignoring parameter uncertainty is not a good assumption to make when creating a vaccination policy. Extensions of this work include creating realistic estimates of the different parameter values and testing the robustness of the optimal vaccination schemes through simulation, as well as extending the stochastic programming framework to other epidemic models.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

This dissertation introduces new algorithmic methods for joint chance-constrained programs with random left-hand side. Our solution techniques include both exact MIP approaches, exact global optimization techniques, and some heuristic methods. We present implementations of all these methods and report on computational results from several sets of test instances. We also describe how to solve the problem of optimally allocating vaccines under parameter uncertainty as a chance-constrained program.

Joint chance-constrained programs are an important branch of stochastic programming in which a subset of the problem constraints are allowed to be violated with a user-defined probability. The applications for these problems include finance, production planning, aquifer management, and many others. When the left-hand side matrix of the problem is allowed to be random, few results exist for finding the optimal solution to the problem. The primary goal of this dissertation is to expand on the set of tools available to mathematical programmers for solving this important class of problems. We also want to point future research in important directions that may lead to fruitful advancements in algorithms for the problems.

One contribution of this work is that we showed how to find optimal vaccination policies under parameter uncertainty using chance constrained programming. We started with the linear programming formulation of the problem given by Becker and Starczak (1997) and relaxed the assumption that the data are known. We showed how this specific example could be reformulated as a chance constrained program and presented a small numerical example showing that the chance constrained formulation solution is much more robust than the traditional solution.

More importantly, we extended the chance constrained programming formulation to the general problem of finding an optimal vaccination policy independent of the underlying disease spread model. We showed how the problem can be formulated as a traditional chance constrained problem, a problem maximizing the probability that an epidemic is prevented, and a cost-benefit formulation. We also discussed various solution techniques for different types of formulations dependent on the underlying disease spread model. Our hope is that this formulation will lead to a new way of dealing with the fundamental problem of parameter uncertainty in the disease modeling and vaccination communities.

The results presented in this thesis are certainly not the final word on using stochastic programming to find optimal vaccination strategies under parameter uncertainty. There are several important extensions to these results. The first is to develop practical vaccination policies using improved estimates of the random parameters and solving the chance-constrained programs that arise. The other major extension is using stochastic programming techniques for finding optimal vaccination strategies in the case where the underlying disease spread model is a simulation. There have been a few results applying simulation-optimization to this problem when the parameter distributions are known. However, since parameters are inherently uncertain in disease spread, it is necessary to look at robust and stochastic approaches to the simulation-optimization of the problem.

Another contribution of this dissertation is our derivation of IIS cutting planes for strengthening the relaxation of the MIP reformulation of the problem. The problem is that the MIP reformulation contains "big-M" constraints which cause it to have particularly weak relaxations. We used irreducibly infeasible subsets of constraints of an upper bound generating formulation of the problem to derive subsets of constraints that cannot all be satisfied in the optimal solution to the problem. We then use these

constraints to derive optimality cuts that can be added to the MIP formulation. In the case in which no cut can be found, we should how the upper bound of the problem can be quickly improved, thus helping with the computational effectiveness of the algorithm. We combine all these results into a branch-and-cut algorithm that will solve joint chance-constrained problems to optimality.

After deriving the IIS branch-and-cut algorithm, we implemented it to see how effective it is at solving problems in our test sets. The implementation is much less sophisticated than commercial branch-and-bound code with node choice simply being most fractional and no other cuts added besides the IIS cuts. In computational tests, we showed that the IIS cut-and-branch algorithm is effective at solving the MIP formulation of joint chance-constraints. It was superior to commercial solvers despite its lack of sophistication. On one set of test problems, the IIS branch-and-cut algorithm was able to solve significantly larger problems than could the commercial solver. On the other, more difficult set, it managed to improve upon the best solutions found.

An important extension to this algorithm would be to implement the IIS cuts within a good commercial solver to see how much it improves the workings of the algorithm in that case. Much work needs to be done improving the generation procedure of the cuts especially in terms of generating rounds of cuts, or deciding when to stop or start generating cuts. It would also be useful to thoroughly test other types of MIP cuts on the problem to see what cuts are most useful when used together. There is also a need for more analysis of the polyhedral structure of the problem. Ruszczyński (2002) provided a first cut at it, but with the success of (Luedtke and Ahmed, 2007) similar results for chance-constrained programs with random left-hand sides are needed.

This dissertation also includes a description of a tabu search heuristic that we

developed to find good feasible solutions to joint chance constrained problems. The main problem encountered by the IIS branch-and-cut algorithm and other exact algorithms for chance constrained programs is that they have scaling issues as the number of scenarios increases. This makes sense because there is a binary variable for every scenario. The goal of our heuristic is to develop a method to find good feasible solutions for cases in which there are too many scenarios to allow exact solution. Such a heuristic can give a decent bound on the optimal solution and also is "better than nothing".

The first need for our heuristic was to reformulate the problem so that it has a finite solution space. This is important because there is a much wider range heuristic results for problems with finite solution spaces as apposed to problems with infinite solution spaces. We reformulated the problem as finding a minimal element of the set of all sets of scenarios C such that the probability of C is sufficient. We then defined the neighborhood of a solution as any other solution that can be found by putting a scenario into the existing solution and then removing elements until it is minimal.

We had to make some modifications to existing tabu search methods in order to ensure that our algorithm could iterate sufficiently quickly. We defined a sufficient set of scenarios that imply the rest of the scenarios of the problem. Computationally we showed that this sufficient set is often much smaller than the original set of the problem. This allowed us to solve a relatively small linear program at each iteration. We chose outgoing scenarios from the set of scenarios with a constraint with slack value 0 and we chose incoming scenarios from the set of scenarios with minimum infeasibility. Finally, we derived a new construction heuristic to give our algorithm a good starting point.

Computational results show our heuristic to be effective in finding good feasible solutions to the problems in our test sets. For one set of test instances, the heuristic

was able to find the optimal solution for every problem that it was known. It was able to find significant improvements on the other problems. On the other set of test problems, the heuristic was able to find significant improvements to the commercial solver. We envision this heuristic being used in tandem with exact methods to determine tight bounds on the optimal solutions to these problems.

The ideas that we used to make our heuristic effective could also be useful for exact solution methods. A major issue with the branch-and-cut in the IIS cuts is the computational expense of solving large linear programs at each node. Sufficient sets could be used to shrink the size of these nodes and so allow for larger trees to be searched. Combined with other computational improvements and cutting planes, this could lead to big increases in the size of problems that can be solved.

The final contribution of this dissertation is showing how the chance constraint can be reformulated so that it is monotonic and then using this fact to develop a monotonic optimization algorithm for the problem. We derived the monotonic branch-and-bound algorithm and proved that it converges to the optimal solution. The computational experience with our implementation was negative. The algorithm is successful in being able to search a large number of nodes quickly, however the pruning is not effective enough to make the tree small enough to solve practical problems. Whether or not the reformulation of the problem as a monotone optimization problem can be made useful remains an open question.

The main extension needed for the monotonic branch-and-bound is figuring out improvements that allow it to be effective. The algorithm as it is currently devised loses too much information by dropping the chance-constraints. It is necessary to figure out a branching strategy or a cutting plane method that could be used in combination with the monotonic structure of the problem that would take into account the information present in the chance-constraint without too much computational

expense.

REFERENCES

Agur, Z., Danon, Y. L., Anderson, R. M., Cojocaru, L., and May, R. M. (1993). Measles immunization strategies for an epidemiologically heterogeneous population - the Israeli case-study. *Proceedings of the Royal Society of London Series B-Biological Sciences*, **252**(1334):81–84.

Amaldi, E., Pfetsch, M., and Trotter, L. (2003). On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs. *Mathematical Programming*, **95**(3):533–554.

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

Anderson, R. M. and May, R. M. (1983). Vaccination against rubella and measles: quantitative investigation of different policies. *Journal of Hygiene-Cambridge*, **90**:259–325.

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

Babad, H. R., Nokes, D. J., Gay, N. J., Miller, E., Morgan-Capner, P., and Anderson, R. M. (1995). Predicting the impact of measles vaccination in England and Wales: model validation and analysis of policy options. *Epidemiology and Infection*, **114**:319–344.

Balas, E. and Zemel, E. (1980). An algorithm for large zero-one knapsack problems. *Operations Research*, **28**:1130–1155.

Ball, F., Britton, T., and Lyne, O. (2004). Stochastic multitype epidemics in a

community of households: estimation and form of optimal vaccination schemes. *Mathematical Biosciences*, **191**(1):19–40.

Ball, F., Mollison, D., and Scalia-Tomba, G. (1997). Epidemics with two levels of mixing. *Annals of Applied Probability*, **7**:46–89.

Ball, F. G. and Lyne, O. D. (2002). Optimal vaccination policies for stochastic epidemics among a population of households. *Mathematical Biosciences*, **177-178**:333–354.

Bansal, S., Pourbohloul, B., and Meyers, L. A. (2006). A comparative analysis of influenza vaccination programs. *PLoS Medicine*, **3**:e387.

Bazaara, M. S., Jarvis, J. J., and Sherali, H. D. (1990). *Linear Programming and Network Flows*. John Wiley & Sons, Inc., New York, NY.

Becker, N. (1995). Estimation of parameters relevant for vaccination strategies. *Bulletin de l'Institut International de Statistique*, **56**(2):1279–1289.

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

Beraldi, P. and Ruszczyński, A. (2002a). A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software*, **17**:359–382.

Beraldi, P. and Ruszczyński, A. (2002b). The probabilistic set covering problem. *Operations Research*, **50**:956–967.

Birge, J. R. and Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer Series in Operations Research. Springer-Verlag, New York.

Brogger, S. (1967). Systems analysis in tuberculosis control: a model. *American Review of Respiratory Diseases*, **95**:419–434.

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

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

Charnes, A. and Cooper, W. W. (1959). Chance-constrained programming. *Management Science*, **6**:73–89.

Chen, X., Sim, M., and Sun, P. (2007). A robust optimization perspective on stochastic programming. *Operations Research*, **55**:1058–1071.

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

Chinnneck, J. W. (1997). Finding a useful subset of constraints for analysis in an infeasible linear program. *INFORMS Journal on Computing*, **9**:164–174.

Clancy, D. and Green, N. (2007). Optimal intervention for an epidemic model under parameter uncertainty. *Mathematical Biosciences*, **205**:297–314.

Codato, G. and Fischetti, M. (2006). Combinatorial benders cuts for mixed-integer linear programming. *Operations Research*, **54**:756–766.

Curry, G., Helm, J., and Clark, R. (1973). Chance-constrained model of system of reservoirs. *Journal of the Hydraulics Division*, **12**:2353–2366.

Dentcheva, D., Lai, B., and Ruszczyński, A. (2004). Dual methods for probabilistic optimization problems. *Mathematical Methods of Operations Research*, **60**:331–346.

Dentcheva, D., Prekopa, A., and Ruszczyński, A. (2000). Concavity and efficient points of discrete distributions in probabilistic programming. *Mathematical Programming*, **89**(1):55–77.

Dentcheva, D., Prékopa, A., and Ruszczyński, A. (2002). Bounds for probabilistic integer programming problems. *Discrete Applied Mathematics*, **124**:55–65.

Dietz, K. (1981). The evaluation of rubella vaccination strategies. In Hiorns, R. W. and Cooke, D., editors, *The Mathematical Theory of the Dynamics of Biological Populations II*, pages 81–97. Academic Press, London.

Dushoff, J., Plotkin, J. B., Viboud, C., Simonsen, L., Miller, M., Loeb, M., and Earn, D. J. D. (2007). Vaccinating to protect a vulnerable subpopulation. *PLoS Medicine*, **4**:e174.

Erdogan, E. and Iyengar, G. (2005). On two-stage convex chance constrained programs. *http://www.stoprog.org/*.

Ferguson, N. M., Cummings, D. A. T., Fraser, C., Cajka, J. C., Cooley, P. C., and Burke, D. S. (2006). Strategies for mitigating an influenza pandemic. *Nature*, **442**:448–452.

Frauenthal, J. C. (1981). *When should routine vaccination be discontinued?* The UMAP Expository Monograph Series. Birkhäuser, Boston.

Germann, T. C., Kadau, K., Longini, Ira M., J., and Macken, C. A. (2006). Mitigation strategies for pandemic influenza in the United States. *Proceedings of the National Academy of Science, USA*, **103**(15):5935–5940.

Gleeson, J. and Ryan, J. (1990). Identifying minimally infeasible subsystems of inequalities. *ORSA Journal on Computing*, **2**(1):61–63.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston.

Haneveld, W. and Vlerk, M. v. d. (2006). Integrate chance constraints: Reduced forms and an algorithm. *Computational Management Science*, **3**:245–269.

Henrion, R. and Möller, A. (2003). Optimization of a continuous distillation process under random inflow rate. *Computers and Mathematics with Applications*, **45**:247–262.

Henrion, R. and Strugarek, C. (2006). Convexity of chance constraints with independent random variables. *http://www.stoprog.org/*.

Hethcote, H. W. (1983). Measles and rubella in the united states. *American Journal of Epidemiology*, **117**:2–13.

Hethcote, H. W. (1988). Optimal ages of vaccination for measles. *Mathematical Biosciences*, **89**:29–52.

Hethcote, H. W. (1997). An age-structured model for pertussis transmission. *Mathematical Biosciences*, **145**:89–136.

Hethcote, H. W. (1999). Simulations of pertussis epidemiology in the United States. *Mathematical Biosciences*, **158**:47–73.

Hethcote, H. W. (2002). New vaccination strategies for pertussis. In Castillo-Chavez, C., Blower, S., van den Driessche, P., Kirschner, D., and Yakubu, A.-A., editors, *Mathematical Approaches for Emerging and Reemerging Infectious Diseases: An Introduction*, The IMA Volumes in Mathematics and its Applications, pages 97–118. Springer, New York.

Hethcote, H. W., Horby, P., and McIntyre, P. (2004). Using computer simulations to compare pertussis vaccination strategies in Australia. *Vaccine*, **22**:2181–2191.

Hethcote, H. W. and Waltman, P. (1973). Optimal vaccination schedules in a deterministic epidemic model. *Mathematical Biosciences*, **18**:365–381.

Hill, A. N. and Longini, Jr., I. M. (2003). The critical vaccination fraction for heterogeneous epidemic models. *Mathematical Biosciences*, **181**:85–106.

Hoos, H. and Stutzle, T. (2005). *Stochastic Local Search: Foundations and Applications*. Elsevier, San Francisco.

Horst, R., Pardalos, P. M., and Thoai, N. V. (2000). *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordtrecht, The Netherlands.

Iwamura, K. and Liu, B. (1996). A genetic algorithm for chance constrained programming. *Journal of Information and Optimization Sciences*, **17**(2):409–422.

Jagannathan, R. (1974). Chance-constrained programming with joint constraints. *Operations Research*, **22**:358–372.

Kataoka, S. (1963). A stochastic programming model. *Econometrica*, **31**:181–196.

Knox, E. G. (1980). Strategy for rubella vaccination. *International Journal of Epidemiology*, **9**:13–23.

Lagoa, C., Li, X., and Sznaier, M. (2005). Probabilistically constrained linear programs and risk-adjusted controller design. *SIAM Journal on Optimization*, **15**:938–951.

Longini, Jr., I. M., Ackerman, E., and Elveback, L. R. (1978). An optimization model for influenza A epidemics. *Mathematical Biosciences*, **38**:141–157.

Longini, Jr., I. M., Halloran, M. E., Nizam, A., and Yang, Y. (2004). Containing pandemic influenza with antiviral agents. *American Journal of Epidemiology*, **159**(7):623–633.

Loon, J. V. (1981). Irreducibly inconsistent systems of linear inequalities. *European Journal of Operational Research*, **8**:283–288.

Luedtke, J. and Ahmed, S. (2007). A sample approximation approach for optimization with probabilistic constraints. *Submitted to Mathematical Programming*.

Luedtke, J., Ahmed, S., and Nemhauser, G. (2007). An integer programming approach for linear programs with probabilistic constraints. *Submitted to Mathematical Programming*.

Miller, B. and Wagner, H. (1965). Chance constrained programming with joint constraints. *Operations Research*, **13**:930–945.

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

Morris, R. S., Wilesmith, J. W., Stern, M. W., Sanson, R. L., and Stevenson, M. A. (2001). Predictive spatial modelling of alternative control strategies for the foot-and-mouth disease epidemic in Great Britain, 2001. *Veterinary Record*, **149**(5):137–144.

Müller, J. (1997). Optimal vaccination strategies–for whom? *Mathematical Biosciences*, **139**(2):133–154.

Nemhauser, G. and Wolsey, L. (1999). *Integer and Combinatorial Optimization*. John Wiley & Sons, New York.

Nemirovski, A. and Shapiro, A. (2004). Scenario approximations of chance constraints. *http://www.optimization-online.org/*.

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

Pagnoncelli, B., Ahmed, S., and Shapiro, A. (2008). Computational study of a chance constrained portfolio selection problem. *http://www.optimization-online.org/*.

Pang, J. and Leyffer, S. (2004). On the global minimization of the value-at-risk. *Optimization Methods and Software*, **19**:611–631.

Parpas, P., Rustem, B., and Pistikopoulos, E. (2007). Global optimization of robust chance constrained programs. *to appear in Journal of Global Optimization*.

Patel, R., Longini, Jr, I. M., and Halloran, E. M. (2005). Finding optimal vaccination strategies for pandemic influenza using genetic algorithms. *Journal of Theoretical Biology*, **234**(2):201–212.

Pfetsch, M. E. (2008). Branch-and-cut for the maximum feasible subset problem. *SIAM Journal on Optimization*, **19**:21–38.

Pintér (1989). Deterministic approximations of probability inequalities. *Methods and Models of Operations Research*, **33**:219–239.

Pourbohloul, B., Meyers, L. A., Skowronski, D. M., Krajden, M., Patrick, D. M., and Brunham, R. C. (2005). Modeling control strategies of respiratory pathogens. *Emerging Infectious Diseases*, **11**(8):1249–1256.

Prékopa, A. (1971). Logarithmic concave measures with application to stochastic programming. *Acta Scientiarum Mathematicarum*, **32**:301–316.

Prékopa, A. (1974). Programming under probabilistic constraints with a random technology matrix. *Mathematische Operationsforshung und Statistik*, **5**:109–116.

Prékopa, A. (1990). Dual method for the solution of a one-stage stochastic programming problem with random rhs obeying a discrete probability distribution. *Methods and Models of Operations Research*, **34**:441–461.

Prékopa, A. (2003). Probabilistic programming. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming*, Handbooks in Operations Research and Management Science, pages 267–345. Elsevier, Amsterdam, The Netherlands.

Revelle, C. S., Lynn, W. R., and Feldmann, F. (1967). Mathematical models for the economic allocation of tuberculosis control activities in developing nations. *American Review of Respiratory Diseases*, **96**:893–909.

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

Ruszczyński, A. and Shapiro, A., editors (2003). *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, N. Holland, The Netherlands.

Saxena, A. (2007). A short note on the probabilistic set covering problem. *Stochastic Programming E-Print Series*.

Sen, S. (1992). Relaxations for probabilistically constrained programs with discrete random variables. *Operations Research Letters*, **11**:81–86.

Shulgin, B., Stone, L., and Agur, Z. (1998). Pulse vaccination strategy in the sir epidemic model. *Bulletin of Mathematical Biology*, **60**:1–26.

Szegö, G. (2002). Measures of risk. *Journal of Banking and Finance*, **26**:1253–1272.

Tanner, M. and Beier, E. (2008). A general heuristic method for joint chance-constrained stochastic programs with discretely distributed parameters. *submitted to Computers and Operations Research*.

Tanner, M. and Ntaimo, L. (2008). Iis branch-and-cut for joint chance-constrained programs with random technology matrices. *submitted to European Journal of Operational Research*.

Tanner, M., Sattenspiel, L., and Ntaimo, L. (2008). Finding optimal vaccination

strategies under parameter uncertainty using stochastic programming. *Mathematical Biosciences*, page doi:10.1016/j.mbs.2008.07.006.

Tayur, S. R., Thomas, R. R., and Natraj, N. (1995). An algebraic geometry algorithm for scheduling in presence of setups and correlated demands. *Mathematical Programming*, **69**:369–401.

Tekin, E. and Sabuncuoglo, I. (2004). Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions*, **36**:1067–1081.

Tuy, H. (2000). Monotonic optimization: Problems and solution approaches. *Siam Journal on Optimization*, **11**:464–494.

Uryasev, S. P., editor (2000). *Probabilistic Constrained Optimization: Methodology and Application*. Nonconvex Optimization and Its Applications. Springer, Dordtrecht, The Netherlands.

Waaler, H., Geser, A., and Anderson, S. (1962). The use of mathematical models in the study of the epidemiology of tuberculosis. *American Journal of Public Health*, **52**:1002–1013.

Watanabe, T. and Ellis, H. (1993). A joint chance-constrained programming model with row dependence. *European Journal of Operational Research*, **77**:325–343.

APPENDIX A

OPTIMAL VACCINATION POLICY FORMULATION AND DATA

We use the deterministic linear program of Becker and Starczak (1997) 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.

Table X. 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 X gives the problem sizes for the set of optimal vaccination test instances. We created 5 random replications of each problem size in order to ensure the robustness of the computational results. 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. While the last column gives the number of binary variables of the problem. For these test problems $m_1 = 31$ and $m_2 = 1$. These instances tend to be difficult to solve because the MIP formulation is

extremely dense. The parameters and details of the stochastic program are given in Table XI.

Table XI. 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

$$\min : \sum_{f\in F}\sum_{v\in V}\sum_{t\in T} v_t h_f x_{fv} \tag{A.1a}$$

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

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

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

Equations (A.1a) - (A.1d) give the formulation of the stochastic programs. The objective function minimizes the vaccine coverage. The first constraint (A.1b) balances all the decision variables for each family type, ensuring that the proportions assigned sum to one. The second, probabilistic constraint (A.1c) 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 (6.6).

$a_{fv}(\omega)$ 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 (1997) and is given below. It includes the assumption that between household contacts occur proportionately to the size of the household. Table XII and Table XIII give the exact household makeups and probability distributions that we assumed.

$$a_{fv}(\omega) = \frac{m(\omega)h_f}{\mu_F}\Big(\sum_{t\in T} u_t(\omega)s_t(\omega)\big[(1-b(\omega))(f_t - v_t\epsilon(\omega)) + b(\omega)v_t\epsilon(1-\epsilon)\big] \tag{A.2}$$

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

Table XII. 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 XIII. List of Vaccination 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, $p = 0.5$

APPENDIX B

PRODUCTION PLANNING FORMULATION AND DATA

The model is a standard multistage production planning problem with the goal of maximizing profit. In this particular model, a company is producing and selling a set of products over time. The company has limited production capacity and must decide how much of each product to make, sell, or store in each time period. Also, there is limited capacity for inventory storage and the company is constrained to sell a minimum amount of each product in each time period. Furthermore, the company is constrained by a maximum amount of each product that can be sold. The randomness in this problem appears in the amount of resources that is required for the company to produce each product during each time period, and in the maximum and minimum amount of each product that must be sold.

Table XIV gives the details on the sizes of the production planning test instances. The table is set up in the same way as Table 1. Again, we created a set of five test instances for each size problem. In this case, $m_1 = 31$ and $m_2 = 55$. The joint chance-constraint makes these problems difficult to solve as the MIP formulation becomes extremely large as the number of scenarios increases.

The first constraint is a mass balance constraint (B.1b). The second constraint is the joint chance-constraint (B.1c) 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 XV, while the distributions of the random parameters are given in Table XVI.

Table XIV. Problem Sizes for Production Planning Test Instances

Instance	Rows	Cont. Vars.	Bin. 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

$$\max \quad \sum_{kt} -c_{kt}m_{kt} + p_{kt}s_{kt} \tag{B.1a}$$

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

$$\mathcal{P}\left(\begin{array}{l} \sum_k n_{kt}(\omega) m_{kt} \leq r_{kt} \;\; \forall t \in T \\ s_{kt} \leq \max_{kt}(\omega) \;\; \forall t \in T, \forall k \in K \\ s_{kt} \geq \min_{kt}(\omega) \;\; \forall t \in T, \forall k \in K \end{array} \right) \geq \alpha \tag{B.1c}$$

Table XV. 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 XVI. List of Production 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]

VITA

Matthew Wiley Tanner is from Columbia, Missouri. He received his B.S.E. in operations research and financial engineering from Princeton University in 2004. This dissertation is the culmination of 5 years of study in the Industrial and Systems Engineering Department at Texas A&M. He graduated in May 2009. Matthew's mailing address is 241 Zachry, 3131 TAMU, College Station, TX 77843-3131. His email address is mtanner@tamu.edu.

The typist for this thesis was the author.