SUBGRADIENT-BASED DECOMPOSITION METHODS FOR STOCHASTIC MIXED-INTEGER PROGRAMS WITH SPECIAL STRUCTURES

A Dissertation

by

ERIC BENJAMIN BEIER

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

December 2011

Major Subject: Industrial Engineering

SUBGRADIENT-BASED DECOMPOSITION METHODS FOR STOCHASTIC MIXED-INTEGER PROGRAMS WITH SPECIAL STRUCTURES

A Dissertation

by

ERIC BENJAMIN BEIER

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

December 2011

Major Subject: Industrial Engineering

ABSTRACT

Subgradient-based Decomposition Methods
for Stochastic Mixed-integer Programs with Special Structures. (December 2011)
Eric Benjamin Beier, B.S., Lamar University;
M.E., Texas A&M University
Chair of Advisory Committee: Dr. Lewis Ntaimo

The focus of this dissertation is solution strategies for stochastic mixed-integer programs with special structures. Motivation for the methods comes from the relatively sparse number of algorithms for solving stochastic mixed-integer programs. Two stage models with finite support are assumed throughout. The first contribution introduces the nodal decision framework under private information restrictions. Each node in the framework has control of an optimization model which may include stochastic parameters, and the nodes must coordinate toward a single objective in which a single optimal or close-to-optimal solution is desired. However, because of competitive issues, confidentiality requirements, incompatible database issues, or other complicating factors, no global view of the system is possible.

An iterative methodology called the nodal decomposition-coordination algorithm (NDC) is formally developed in which each entity in the cooperation forms its own nodal deterministic or stochastic program. Lagrangian relaxation and subgradient optimization techniques are used to facilitate negotiation between the nodal decisions in the system without any one entity gaining access to the private information from other nodes. A computational study on NDC using supply chain inventory coordination problem instances demonstrates that the new methodology can obtain good solution values without violating private information restrictions. The results also show that the stochastic solutions outperform the corresponding expected value solutions.

The next contribution presents a new algorithm called scenario Fenchel decomposition (SFD) for solving two-stage stochastic mixed 0-1 integer programs with special structure based on scenario decomposition of the problem and Fenchel cutting planes. The algorithm combines progressive hedging to restore nonanticipativity of the first-stage solution, and generates Fenchel cutting planes for the LP relaxations of the subproblems to recover integer solutions.

A computational study SFD using instances with multiple knapsack constraint structure is given. Multiple knapsack constrained problems are chosen due to the advantages they provide when generating Fenchel cutting planes. The computational results are promising, and show that SFD is able to find optimal solutions for some problem instances in a short amount of time, and that overall, SFD outperforms the brute force method of solving the DEP.

To my wife, Hope Thomas Beier.

ACKNOWLEDGMENTS

Much appreciation is due my advisor, Dr. Lewis Ntaimo for his support and advice as I pursued my doctorate. I would also like to thank Dr. Jorge Leon for his part in the Nodal Decomposition-Coordination algorithm. Thank you to the Department of Industrial and Systems Engineering at Texas A&M University for providing financial support and experience in the teaching side of academia. Special thanks go to Judy for always being there to answer my questions and hear my grievances. Finally, thanks to my friends, without whom I could have not made it this far.

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

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Stochastic programming (SP) is a branch of mathematical programming that seeks optimal solutions to mathematical optimization problems containing uncertain data. The field of stochastic programming includes two stage stochastic problems, multistage stochastic problems, and problems with chance constraints. This dissertation considers two stage stochastic problems and that is the focus here. Two stage stochastic programs are SPs with two distinct sets of decision variables: a first-stage decision vector representing the "here and now" decisions and a vector of second-stage decision variables, usually called *recourse* decisions, which do not have to be decided upon until after the uncertainty in the problem data has been realized. The general form of a two stage stochastic program is given below:

$$\begin{aligned} \text{SP1}: \text{Min}\ & c^\top x + \mathbb{E}[f(\tilde{\omega}, x)] \\ \text{s.t.}\ & Ax \geq b \\ & x \in X^{n_1}, \end{aligned} \tag{1.1}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, and for an outcome ω of $\tilde{\omega}$

$$\begin{aligned} f(\omega, x) = \text{Min}\ & q(\omega)^\top y(\omega) \\ \text{s.t.}\ & W(\omega)y(\omega) \geq h(\omega) - T(\omega)x \\ & y(\omega) \in Y^{n_2}. \end{aligned} \tag{1.2}$$

In the first stage of SP1 (1.1), x denotes the vector of first-stage decisions, $c \in \mathbb{R}^{n_1}$ denotes the first-stage objective cost vector, $A \in \mathbb{R}^{m_1 \times n_1}$ denotes the first-stage constraint matrix,

This dissertation follows the style of *IIE Transactions*.

$b \in \mathbb{R}^{m_1}$ denotes the first-stage right-hand side vector, and X^{n_1} denotes special restrictions on the first-stage decisions (such as integral restrictions and upper and lower bounds.) In the second stage of SP1(1.2), $f(\omega, x)$ is called the *recourse function*, $y(\omega)$ denotes the recourse decision vector for scenario ω, $q(\omega) \in \mathbb{R}^{n_2}$ is the second-stage objective cost vector, $W(\omega) \in \mathbb{R}^{m_2 \times n_2}$ is the recourse matrix, $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$ is the technology matrix, $h(\omega)$ denotes the second-stage right hand side, and Y^{n_2} denotes special restrictions on the recourse decisions. In this work, the following assumptions are made on SP1:

(A1) The random variable $\tilde{\omega}$ follows a discrete distribution with finite support Ω.

(A2) The first-stage feasible set $\{Ax \geq b_i,\ x \in X^{n_1}\}$ is nonempty.

(A3) The second-stage feasible set $\{Wy(\omega) \geq h(\omega) - T(\omega)x,\ y(\omega) \in Y^{n_2}\}$ is nonempty and bounded for all feasible first-stage x.

Assumption A1 ensures that the formulation is tractable, A2 guarantees the existence of a feasible solution, and A3 is called the "relatively complete recourse" assumption, which guarantees feasibility of the recourse function for all feasible x. Satisfying A3 can always be accomplished through careful modelling of the problem of interest, such as the imposition of suitably penalized artificial variables to ensure constraint feasibility. Given Assumption A1, let $K = |\Omega|$ and assign an ordering $k = 1, \ldots, K$ to each realization $\omega \in \Omega$, and let p^k denote the probability of outcome k. Then (1.1) can be restated:

$$\begin{aligned} \text{SP2 : Min } & c^\top x + \sum_{k=1}^{K} p^k f(k, x) \\ \text{s.t. } & Ax \geq b \\ & x \in X^{n_1} \end{aligned} \tag{1.3}$$

where, for realization $k = 1, \ldots, K$, $f(k, x)$ is

$$\begin{aligned} f(k,x) = \text{Min } & q^{k\top} y^k \\ \text{s.t. } & W^k y^k \geq h^k - T^k x \\ & y^k \in Y^{n_2}. \end{aligned} \tag{1.4}$$

Formulation (1.4) represents the general form of a two stage stochastic program with finite support, where realization k dictates the data for the matrices from the multivariate random variable. When the sets X^{n_1} and Y^{n_2} require x and y^k to be continuous variables, SP2 is referred to as a stochastic linear program (SLP). If either set requires all or part of the decision variables to be integer, SP2 is referred to as a stochastic mixed-integer program (SMIP). In order to generate tractable problem formulations, the number of realizations can be limited by restricting how many of these matrices are described by random variables. When the recourse matrix $W^k = W, k = 1, \dots, K$, the SP is said to have *fixed recourse* and otherwise it has *random recourse*. Similarly, SP2 with $T^k = T,\ k = 1, \dots, K$ are said to have *fixed technology.*

Another classification of SP2 considers feasibility of the recourse function. When $f(x,k)$ is feasible for any $x \in \mathbb{R}$ SP2 is said to have *complete recourse*, and the weaker assumption, *relatively complete recourse* was already introduced as assumption A3. A special case of complete recourse problems is called *simple recourse*. In simple recourse models $n_2 = 2n_1$ and the recourse matrix $W^k = [I, -I]\ \forall k$ (where I denotes the identity matrix). Two stage SPs with simple recourse have special properties and are among the simplest SPs to solve.

SP2 can be reformulated as the following deterministic equivalent problem (also referred to as the extensive form):

$$\text{DEP: Min } c^\top x + \sum_{k=1}^{K} p^k q^{k\top} y^k \tag{1.5a}$$

$$\text{s.t. } Ax \geq b \tag{1.5b}$$

$$T^k x + W^k y^k \geq h^k,\ k = 1, 2, \ldots, K \tag{1.5c}$$

$$x \in X,\ y^k \in Y, \qquad k = 1, 2, \ldots, K. \tag{1.5d}$$

DEP can be solved directly using a suitable off the shelf optimizer, providing the optimal solution to (1.3). However, for even moderately dimensioned problems, solving 1.5 directly can be a daunting task, as the number of variables and constraints in (1.5c) and (1.5d) grows exponentially in terms of K. For this reason, decomposition approaches are usually required in order to find solutions in a reasonable amount of time.

There are two main approaches for decomposing SP2. The first, called *stagewise decomposition*, involves relaxing the constraints defining the recourse function and approximating the value of the recourse function by linear approximations. The second approach, called *scenario-wise decomposition* uses variable splitting on x allowing K scenario subproblems to be formulated. One benefit of scenario-wise strategies is that they can generally be applied to multi-stage stochastic programs without much alteration.

In the following chapter, this work is motivated by discussing previous work concerning solving SP2. As the theme of this dissertation is focuses on subgradient techniques for solving SMIPs, a brief review of Lagrangian relaxation and subgradient optimization is given. Next, some important theory regarding SLPs is reviewed including a description of the two prevailing decomposition and solution strategies for solving SLP. Finally, some of the major algorithms for solving SMIPs are reviewed.

Chapter III introduces the nodal decomposition-coordination (NDC) algorithm for SMIPs. NDC is motivated by a nodal decision structure where each node represents an entity with its own optimization model. The nodes in the structure form a group with a common objective, and each node owns its own optimization model whose parameters and decision variables are known only to itself except for a small subset of the variables whose values must be coordinated with some subset of the other nodes. NDC is developed in order to coordinate these decision variables's values and the algorithm is tested on a set of instances describing a supply chain inventory coordination problem.

Chapter III develops a new algorithm called scenario Fenchel decomposition (SFD) for solving SMIPs based on the progressive hedging algorithm (Rockafellar and Wets, 1991). The algorithm iteratively finds the optimal solution to the LP relaxation of the SMIP using PHA, and then uses Fenchel cutting planes to separate the noninteger point from the convex hull of the (integer) scenario subproblems. The algorithm is tested on a set of randomly generated multidimensional knapsack problems. Finally, Chapter V summarizes the contributions of this dissertation and describes the avenues for ongoing and future research.

CHAPTER II

LITERATURE REVIEW

This dissertation focuses subgradient based decomposition methods for stochastic integer programs. This chapter reviews theory necessary for later chapters and summarizes current state-of-the art approaches for solving stochastic programs. To begin, the concepts of Lagrangian duality and subgradient optimization from linear programming are discussed followed by a review of optimization methods for stochastic programs.

A. Lagrangian Duality and Subgradient Optimization

Lagrangian relaxation techniques are motivated by mathematical programs whose feasible region includes a set of constraints which, if relaxed, would result in problem which is much easier to solve than the original problem. Lagrangian relaxation is a technique commonly used to relax these *complicating constraints* in order to have an easier problem to optimize over. Consider the following linear program (LP):

$$P_{LP} = \min cx \tag{2.1a}$$

$$\text{s.t. } Ax = b \tag{2.1b}$$

$$Dx \leq d \tag{2.1c}$$

$$x \geq 0. \tag{2.1d}$$

Assume that constraint (2.1b) is a complicating constraint. Lagrangian relaxation is a well-known method for relaxing complicating constraints from a LP or mixed-integer program (MIP) into the objective. This is accomplished by relaxing the constraint into the objective and penalizing deviations from the constraint. Performing Lagrangian relaxation on (2.1b) yields the following form:

$$\begin{aligned} P_{LR}(\lambda) := \min_{x}\ & cx + \lambda(Ax - b) \\ \text{s.t. } & Dx \leq d \\ & x \geq 0. \end{aligned} \tag{2.2}$$

where the vector λ is referred to as the Lagrangian multipliers. For a given value of λ, $P_{LR}(\lambda)$ (2.2) is known as the *Lagrangian relaxation problem*. The Lagrangian dual is formed by finding maximal Lagrangian multipliers for $P_{LR}(\lambda)$ (2.2). The Lagrangian dual has the form

$$P_{LD} := \max_{\lambda \text{ free}} P_{LR}(\lambda). \tag{2.3}$$

Due to the presence of a maximization over the the Lagrangian dual problem is sometimes referred to as the max-min dual problem.

The Lagrangian relaxation and Lagrangian dual problems are well studied in the literature. They have many properties that are useful in optimization techniques. (The proofs for these theorems are well documented in the literature and textbooks. See for example Bazaraa et al. (1993).) The first, known as the *weak duality* theorem for $P_{LR}(\lambda)$ readily allows for computing a lower bound to the optimal solution of P.

THEOREM II.1. *Given feasible solutions x to P and λ to $P_{LR}(\lambda)$, the following relationship holds:*

$$P_{LR}(\lambda) \leq P_{LP} \tag{2.4}$$

Given the weak duality theorem, the question arises of whether it is possible to guarantee that a solution to P_{LD} is optimal for P_{LP}. This is known as the *strong duality* theorem, and it is well-known to hold for linear programs of the form P_{LP}.

THEOREM II.2. *Assume the feasible region of P_{LP} is nonempty and bounded and P_{LP} has a finite optimal and let the optimal values to P_{LD} and P_{LP} be P^*_{LD} and P^*_{LP}, respec-*

tively. Then

$$P^*_{LD} = P^*_{LP}. \tag{2.5}$$

Theorem II.2 guarantees that for LPs, an optimum to P_{LP} can be found by solving P_{LD}. The challenge in solving P_{LD} is in resolving the outer maximization with the inner minimization. A well-known method for solving the Lagrangian dual is to employ the use of subgradient optimization. In order to apply subgradient optimization to the Lagrangian dual problem, the following concavity theorem is necessary:

THEOREM II.3. *Assume the feasible region of P_{LP} is nonempty and bounded and P_{LP} has a finite optimal. Then $P_{LR}(\lambda)$ is a piecewise linear concave function of λ..*

For smooth functions, maximization of a concave function can be done by any of several Newton's method-based algorithms. However, the piecewise linearity of the function means that $P_{LR}(\lambda)$ is not everywhere-differentiable over its domain. Fortunately, given concavity, an adaptation of Newton's method known as subgradient optimization can be used to solve P_{LD}.

DEFINITION II.4. Given concave function $g(\lambda)$, ξ is called a subgradient of g at $\bar{\lambda}$ if

$$g(\lambda) \leq g(\bar{\lambda}) + \xi(\lambda - \bar{\lambda})$$

Showing that the Lagrangian-relaxed constraint $Ax - b$ in satisfies the requirement for ξ is a straightforward application of the theorem. The main idea of a subgradient approach is to use a subgradient in lieu of a gradient in a Newton's method based approach. A basic subgradient optimization algorithm for P_{LD} is formally stated in Figure 1.

Choosing a step size in Step 1 of the subgradient algorithm can be done in several ways. Theoretically, any divergent series that satisfies $\sum_t \mu^t \to \infty$, $\mu^t \to 0$ as $t \to \infty$ is sufficient, but the convergence rate of such a choice is usually too slow to be used in practice. Instead, the step size $\mu^t = \rho^t \frac{UB - P_{LR}(\lambda^t)}{\|Ax^t - b\|^2}$, (where UB is an an upper bound on

Subgradient Algorithm

Step 0 Select an initial point λ^0. Let iteration counter $t = 0$.

Step 1 Let x^t denote the inner minimization solution vector from solving $P_{LR}(\lambda^t)$. If $Ax^t - b = 0$ stop. λ^t and x^t are optimal for P_{LD}. Otherwise, choose step size $\mu^t > 0$.

Step 2 Let $\lambda^{t+1} = \lambda^t + \mu^t(Ax^t - b)$ and return to Step 1.

Fig. 1. A Subgradient Optimization Algorithm

the optimal solution) is typically chosen in practice, as the rate of convergence is faster, but optimality is not guaranteed. This method is particularly popular when applying Lagrangian relaxation to integer programming problems as optimality cannot be guaranteed for general integer programs because Theorem II.2 does not hold, introducing a duality gap between the optimal solution of the Lagrangian dual and the original problem. For details on the choice of a step size in subgradient optimization methods, see Bazaraa et al. (1993) or Nemhauser and Wolsey (1999).

A practical problem is often experienced by researchers employing subgradient optimization. If the step size in subsequent iterations is too large, the subgradient step calculation can return solutions that move between a small number of candidates with no improving solution. This phenomenon is known as solution oscillation, and can be overcome by the introduction of a regularization term into the objective. The resulting augmented Lagrangian problem is formally stated as

$$\begin{aligned} P_{ALR}(\lambda) := \min_x \; & cx + \lambda(Ax - b) + \frac{\rho}{2}|Ax - b|^2 \\ \text{s.t. } & Dx \leq d \\ & x \geq 0 \\ & \lambda \text{ free,} \end{aligned} \tag{2.6}$$

where ρ is a user-defined scalar. The quadratic regularization term in the objective seeks to keep the subgradient ascent direction from moving too far from the current x^k, thus reducing the liklihood of solution oscillations in an iterative approach. For details on augmented Lagrangian approaches see, for example Ruszczyński (1986), Rockafellar (1976) or Ruszczyński (1995)

B. Stochastic Linear Programming

Next, consider instances of (1.3) in which both the first-stage variables (x) and second-stage variables (y) are continuous. These so called stochastic linear programs (SLP) are among the simplest SPs to solve because when $y \in \mathbb{R}$, the expected recourse function $\mathbb{E}[f(\tilde{\omega}, x_N)]$ is convex Wets (1974).

The first formulation of a stochastic linear program is generally credited to George Dantzig (Dantzig, 1955), where he introduced the general form of a stochastic linear program and pointed out the special structure of the formulation. It wasn't until much later that SLP theory was developed that allowed for efficient decomposition strategies for SLPs. The information contained in this chapter only touches on the basics of stochastic programming. For a thorough treatment, see Birge and Louveaux (1997), Ruszczyński and Shapiro (2003) or Shapiro et al. (2009).

In general, decomposition methods for two-stage SLPs fall into two categories. The first, discussed in Section 1 is called stage-wise decomposition which adapts theory from

Benders decomposition (Benders, 1962) in order to decompose the problem into a single master problem representing the first-stage decisions and one subproblem for each scenario which contains the second-stage decisions. The second is called stagewise decomposition, which uses variable splitting on the first-stage variables to decompose the problem into one subproblem for each scenario.

1. Stage-wise Decomposition

One of the most well-known decomposition approaches for SLP is Benders decomposition (Benders, 1962). Notice that the dual of 1.5 exhibits a block-angular structure (Dantzig and Wolfe, 1960), which is amenable to Dantzig-Wolfe decomposition (DWD) (Dantzig and Wolfe, 1961). Benders decomposition is a generalized programming method that takes advantage of dual block angular structures and is often considered to be a dual algorithm to DWD. Benders decomposition was first applied to stochastic programs with the development of the L-shaped algorithm Slyke and Wets (1969). L-shaped methods get their name because of the shape that the first-stage (x) decision vectors make with each set of scenario constraints 1.5c-1.5d in the DEP formulation 1.5.

The L-shaped method is an outer linearization technique which relaxes the second-stage constraints 1.5c-1.5d from DEP and replaces them with linear approximations derived from scenario subproblems. With this in mind, the L-shaped master problem is defined as

$$\text{MP : Min } c^\top x + \theta \tag{2.7a}$$

$$\text{s.t. } \quad Ax \geq b \tag{2.7b}$$

$$\alpha_t x + \theta \geq \beta_t \; t = 1, \ldots, \tau \tag{2.7c}$$

$$x \in X^{n_1} \tag{2.7d}$$

where the optimality cuts 2.7c are derived from dual vectors of the subproblems:

$$\begin{aligned} \text{SP}(k,\hat{x}) = &\text{Min } q^{k\top} y^k \\ &\text{s.t.} \quad W^k y^k \geq h^k - T^k \hat{x} \\ &\qquad\; y^k \in Y^{n_2}. \end{aligned} \tag{2.8}$$

The L-shaped algorithm is formally stated in Figure 2.

L-Shaped Algorithm

Step 0 Initialize. $\tau \leftarrow 0$, $ub \leftarrow \infty$, $lb \leftarrow -\infty$. Solve MP to get solution $\hat{x}$.

Step 1 Solve Subproblems. Solve SP$(k, \hat{x})$ $\forall k$. Compute π^k, the dual vector. Compute $\beta_\tau = \sum_{k=1}^{K} \pi^k h^k$ and $\alpha_\tau = \sum_{k=1}^{K} \pi^k T^k$

Compute $ub = \min\{ub, c^\top x + \sum_{k=1}^{K} p^k f(k,x)\}$

If ub updated, record x_τ as incumbent.

Step 2 Add MP cut and solve. Add cut $\alpha_\tau x + \theta \geq \beta_\tau$ to MP. Solve MP, returning solutions $\hat{x}$ and $\hat{\theta}$ Compute $lb = max\{lb, c^\top \hat{x} + \theta\}$

Step 3 Check termination conditions. If $ub - lb < \epsilon$, stop. Report incumbent as ϵ-optimal. Else, $\tau \leftarrow \tau + 1$. Go to Step 1.

Fig. 2. The L-Shaped Algorithm

Several modifications to the L-shaped algorithm exist. The form given is the classic single cut L-shaped algorithm algorithm. Instead of adding a single cut to MP at each iteration in Step 2, multiple cuts can be added to the MP at each iteration. At each iteration, the multicut L-shaped algorithm of Birge and Louveaux (1988) adds a cut to the master program for each subproblem $k = 1, \dots, K$. Theoretically, the benefits of the multicut version over the single cut version are a reduction in the number of L-shaped iterations, but in practice, the larger size of the MP makes choosing the best option problem dependent.

Additionally, if the relatively complete recourse assumption (Assumption (A3)) does not hold, feasibility cuts can be implemented to force the master problem to find feasible first-stage solutions. Consider the situation where a subproblem k is infeasible with the given $\hat{x}$ solution in Step 1. In this case, Step 2 can be modified to compute a feasibility cut based on the dual extreme ray, which cuts off that $\hat{x}$ from the feasible region of first-stage decisions. A complete treatment of the L-shaped algorithm can be found in Slyke and Wets (1969).

An important extension to stage-wise decomposition approaches is the regularized L-shaped algorithm Ruszczyński (1986). This method augments MP (2.7) with a quadratic term similar to that seen in the augmented Lagrangian relaxation problem. Direct implementations of the L-shaped algorithm can result in a sequence of first-stage x points which converge slowly toward the optimum due to the nature of simplex-based approaches for solving the MP. The regularized L-shaped algorithm seeks to eliminate this inefficiency by penalizing new solutions that step too far from the current incumbent.

One final extension of the L-shaped algorithm involves finding solutions when K becomes too large for direct application of the L-shaped algorithm. In this case a sampling method such as the stochastic decomposition algorithm Higle and Sen (1991) can be used. The algorithm is an internal sampling method that adds another scenario realization in each iteration to approximate the value of the expected recourse function. Exterior sampling methods, where a new sample of the random variables is drawn at each iteration are referred to as sample average approximation (SAA) methods. For a review of SAA algorithms, see Shapiro (2003).

2. Scenario-wise Decomposition

As discussed, stage-wise decomposition strategies seek to exploit the temporal form of a two stage SMIP. The L-shaped method relaxes the second-stage constraints (1.5c) and (1.5d) and iteratively performs an outer linearization technique, generating the feasibility cuts (2.7c). In scenario-wise decomposition approaches, scenario subproblems are formed by copying the first-stage x decision vector K times using variable splitting, and a Lagrangian relaxation approach is typically used to guarantee a feasible solution at termination.

Consider the form of DEP (1.5). The first-stage decision vector x can be thought of as "complicating" variables in the sense that if those variables did not exist, the problem would be directly decomposable into a subproblem for each scenario $k = 1, \ldots, K$ consisting of only the second-stage decisions y. Scenario-wise decomposition approaches seek to find optimal solutions by using variable splitting on the x variables, giving each scenario k its own x^k variable. This strategy can be thought of as moving the first-stage decisions into the second stage where the probability measure acts on the objective coefficients in the same way as it does the y^k variables. Following this approach yields the following form:

$$
\begin{aligned}
\text{VSDEP: Min } & \sum_{k=1}^{K} p^k c^{k\top} x^k + q^{k\top} y^k && && \text{(2.9a)}\\
\text{s.t. } & A x^k \geq b && k = 1, 2, \ldots, K && \text{(2.9b)}\\
& T^k x + W^k y^k \geq h^k, && k = 1, 2, \ldots, K && \text{(2.9c)}\\
& x \in X,\ y^k \in Y, && k = 1, 2, \ldots, K. && \text{(2.9d)}
\end{aligned}
$$

VSDEP decomposes naturally into a subproblem for each k, but solving the resulting subproblems returns an implementable solution in the general case since the x^k solutions returned cannot be translated into a single optimal x solution to DEP. Requiring equality in the x^k solutions is a concept known as *nonanticipativity*. Nonanticipativity is the restriction that the first-stage (known as here-and-now) decisions must be decided before the realization of the random variable is known in the second stage. In the case of scenario-wise decomposition strategies, nonanticipativity must be enforced explicitly with the addition of nonanticipativity constraints. Several forms of nonanticipativity exist. Three forms reported in Caroe (1998) are

$$\begin{aligned} & x^k - x^{k+1} = 0, \qquad k = 1, \dots, K-1 \\ & x^K - x^1 = 0 \end{aligned} \tag{2.10a}$$

$$x^k - \sum_{s=1}^{K} p^s x^s = 0, \quad k = 1, \dots, K \tag{2.10b}$$

$$\begin{aligned} x^k - x^s = 0, \qquad & k = 1, \dots, K; \\ & s \in \{1, 2, \dots, K\}. \end{aligned} \tag{2.10c}$$

The form given in (2.10a) represents a cyclical form, equation (2.10b) gives a form of nonanticpativity in expectation, and (2.10c) chooses a single representative scenario and enforces that value across all scenarios. In this dissertation, the “in expectation” form given in (2.10b) will be used. Adding nonanticipativity constraints of the form (2.10b) to VSDEP (2.9) yields the following deterministic equivalent program.

$$\text{SWDEP: Min} \sum_{k=1}^{K} p^k c^{k\top} x^k + q^{k\top} y^k \tag{2.11a}$$

$$\text{s.t. } Ax^k \geq b \qquad k = 1, 2, \ldots, K \tag{2.11b}$$

$$T^k x + W^k y^k \geq h^k, \quad k = 1, 2, \ldots, K \tag{2.11c}$$

$$x^k - \sum_{s=1}^{K} p^s x^s = 0, \quad k = 1, 2, \ldots, K \tag{2.11d}$$

$$x \in X,\ y^k \in Y, \qquad k = 1, 2, \ldots, K. \tag{2.11e}$$

An optimal solution to DEP2 is the solution vector $(\bar{x}, y^k)$ where $\bar{x} = \sum_{k=1}^{K} p^k x^k$. Solving SWDEP directly is harder than solving DEP since K constraints and K variables have been added to the problem. However, by using Lagrangian relaxation, SWDEP can be reformulated into a form whose feasible region is decomposable into K subproblems. Performing the variable substitution $\bar{x} = \sum_{k=1}^{K} p^k x^k$, relaxing the nonanticipativity constraints (2.11d) via Lagrangian relaxation and introducing the Lagrangian penalty vector $\lambda = \{\lambda^1, \lambda^2, \ldots, \lambda^K\}$ yields the following Lagrangian relaxation formulation:

$$\text{DEP}_{LR}(\lambda) : \text{Min} \sum_{k=1}^{K} p^k c^{k\top} x^k + q^{k\top} y^k + \lambda^k (x^k - \bar{x}) \tag{2.12a}$$

$$\text{s.t. } Ax^k \geq b, \qquad k = 1, 2, \ldots, K \tag{2.12b}$$

$$T^k x + W^k y^k \geq h^k, \qquad k = 1, 2, \ldots, K \tag{2.12c}$$

$$x \in X,\ y^k \in Y, \qquad k = 1, 2, \ldots, K. \tag{2.12d}$$

As with the deterministic case, the Lagrangian dual function becomes

$$\text{DEP}_{LD} = \underset{\lambda \text{ free}}{\text{Max}}\ \text{DEP}_{LR}(\lambda). \tag{2.13}$$

For SLPs, subgradient optimization can now be applied to find an optimal solution to SP2 by solving DEP$_{LD}$.

The progressive hedging algorithm (PHA) presents an augmented Lagrangian approach for solving SP2 (Rockafellar and Wets, 1991). Adding a quadratic regularization term yields the following augmented form:

$$\begin{aligned}
\text{DEP}_{ALR}(\lambda) : \text{Min } & \sum_{k=1}^{K} p^k c^{k\top} x^k + q^{k\top} y^k + \lambda^k (x^k - \bar{x}) + \frac{\rho}{2} |x^k - \bar{x}|^2 \\
\text{s.t. } & Ax^k \geq b, && k = 1, 2, \ldots, K \\
& T^k x + W^k y^k \geq h^k, && k = 1, 2, \ldots, K \\
& x \in X, \; y^k \in Y, && k = 1, 2, \ldots, K.
\end{aligned} \tag{2.14}$$

Notice that DEP$_{ALR}$ now has a feasible region that is decomposable into K subproblems. Assuming an iterative procedure (with counter t) for updating λ^k as discussed in the subgradient optimization review define the PHA subproblem as

$$\text{SP}^k_{PHA}(\lambda) : \text{Min } p^k [c^{k\top} x^k + q^{k\top} y^k + \lambda^k (x^k - \bar{x}_t) + \frac{\rho^k_t}{2} |x^k - \bar{x}_t|^2] \tag{2.15a}$$

$$\text{s.t. } Ax^k \geq b, \qquad k = 1, 2, \ldots, K \tag{2.15b}$$

$$T^k x + W^k y^k \geq h^k, \qquad k = 1, 2, \ldots, K \tag{2.15c}$$

$$x \in X, \; y^k \in Y, \qquad k = 1, 2, \ldots, K, \tag{2.15d}$$

where $\bar{x}_t$ is calculated by taking the weighted sum of the previous iteration's solution vector and ρ^k_t is chosen to keep x^k from moving too far from $\bar{x}_t$ from one iteration to the next. Given this formulation, PHA is formally stated in Figure 3.

PHA Algorithm

Step 1: Choose an initial $\bar{x}_0$, step sizes ρ_0^k and multipliers λ_0^k and $\epsilon > 0$. Set PHA iteration counter $t = 0$.

Step 2: Solve $\mathrm{P}_{LP}^k(\lambda^k)$ $\forall k$ (2.15). Let the solutions be $(\hat{x}_t^k, \hat{y}_t^k, \hat{\lambda}_t^k)$. Update $\bar{x}_{t+1} = \sum\limits_{k=1}^{K} p^k \hat{x}_t^k$.

Step 3: Update $\lambda_{t+1} = \lambda_t + \rho^k(\hat{x}_t^k - \bar{x}_{t+1})$. If $\sum_{k=1}^{K} p^k ||\hat{x}_t^k - \bar{x}_t|| < \epsilon$, $(\hat{x}_t^k, \hat{y}_t^k, \hat{\lambda}_t^k)$ is optimal for $P_{LP}^k(\lambda^k)$; Report optimal solution variables $\hat{x}_t^k$ and $\hat{y}_t^k$ and corresponding objective $\sum\limits_{k=1}^{K} P_{LP}^k(\lambda^k)$. Otherwise increment t. Go to Step 2.

Fig. 3. The Progressive Hedging Algorithm

C. Stochastic Mixed-Integer Programming

To this point, our treatment of SP has assumed that all variables are continuous. Now consider the case when the sets X^{n_1} and Y^{n_2} of SP2 (1.3) restrict some or all of the variables to be integer, in which case SP2 becomes a SMIP. SMIPs are among the most difficult optimization models to solve because the expected recourse function $E[f(\tilde{\omega}, x)]$ of a SMIP with integer restrictions included in the set Y^{n_2} is in general discontinuous and noncovex (Blair and Jeroslow, 1982) (Birge and Louveaux, 1997) (Schultz, 1993). This section summarizes some of the important algorithms for SMIP. An extensive review article can be found in Haneveld and van der Vlerk (1999).

Because solving a SMIP is such a difficult prospect, algorithmic development tends to exploit specific types of variables in each stage in order to be able to find solutions within a reasonable amount of time. Thus algorithms for SMIP can be classified by the types of variables that they work with. Table I simplifies this classification the major algorithms discussed here. The first two columns describe the types of variables allowed in the algorithm: C denotes continuous variables, G denotes pure integer, MG denotes mixed-integer, B denotes pure binary and MP denotes mixed-binary. The third is the name of the algorithm and author.

Table I. Classification of SMIP algorithms

First Stage	**Second Stage**	**Algorithm**
Any	C	L-shaped algorithm (Slyke and Wets, 1969)
C	G	Grobner basis enumeration (Shultz et al., 1998)
	G	Branch-and-bound on tender variables (Ahmed et al., 2004)
B	C	Branch-and-fix coordination (Alonso-Ayuso et al., 2003)
	B	Dynamic programming (Lageweg et al., 1985)
	MB	Disjunctive decomposition (D2) (Sen and Higle, 2005)
	MB	Branch-and-bound with lift and project cuts (Caroe and Tind, 1997)
	MG	Integer L-shaped (Laporte and Louveaux, 1993)
MG	G	F-dual L-shaped algorithm (Caroe and Tind, 1998)
	MG	Dual decomposition(Caroe and Schultz, 1999)
	MG	Stochastic branch-and-bound (Norkin et al., 1998)

Examination of Table I shows that the types of variables in the SMIP dictate the solution strategy used. The simplest case of SMIP is whenever the second stage has all continuous variables, in which case the expected recourse function $E[f(\tilde{\omega}, x)]$ retains its structure and the L-shaped algorithm can be used. However, integer variables in the first stage make convergence of the L-shaped algorithm slow in practice.

While the L-shaped algorithm is not able to solve general SMIP models, many researchers adopt the stagewise decomposition framework when developing new algorithms for solving SMIPs. The integer L-shaped method (Laporte and Louveaux, 1993) merges branch-and-cut with the L-shaped algorithm in order to guarantee convergence of SMIPs with binary first-stage variables and mixed-binary variables in the second stage. The algorithm begins by relaxing integrality restrictions and initializing a branch-and-bound tree. The algorithm proceeds by solving the LP relaxation via the L-shaped method and then branching on variables which violate the integrality restrictions. An illustrative example is given, but no computational results are reported.

The D^2 algorithm (Sen and Higle, 2005) has proven very capable of solving SMIPs with binary first-stage and mixed-binary second-stage. The algorithm solves the first-stage problem as a mixed-binary problem and the linear relaxations of the second-stage subproblems. The algorithm then uses lift-and-project cuts on the scenario subproblems in order to eventually arrive at an optimal solution. Computational experiments using the D^2 algorithm can be found in Ntaimo and Sen (2008).

The $\mathcal{F}$-dual L shaped algorithm of Caroe and Tind (1998) adapts the L-shaped algorithm to solve SMIP problems with mixed-integer first-stage variables and pure integer second-stage variables. Instead of using LP-duality ideas to generate optimality cuts, the authors use IP duality theory. The tradeoff is that the master problem becomes nonlinear, and the authors suggest some modifications that make the master problem easier to solve at the expense of optimality.

The next algorithm reviewed is a branch-and-bound algorithm that exploits lift-and-project cuts (Caroe and Tind, 1997). The algorithm iteratively solves the linear relaxation of SMIP and generates lift-and-project cuts Balas et al. (1993) to recover integer solutions. When SMIP has fixed recourse, the authors offer a means by which the cuts generated for one scenario can be translated to another scenario. Theory is developed for SP2 with continuous first-stage variables, then extended to the binary first-stage case by employing branch-and-bound. No computational results are reported.

A dynamic programming approach can be used to solve some instances of SMIP with pure binary first and second-stage variables (Lageweg et al., 1985). The approach requires that the second-stage problem has a very special structure that can be solved using a dynamic programming approach. Since the set of problems solvable by the algorithm is, the method has not seen wide-spread application, but applications to stochastic scheduling, bin packing and multi-dimensional knapsack problems were demonstrated.

The stochastic branch-and-bound algorithm (Norkin et al., 1998) is able to accommodate mixed-integer variables in both the first and second stage of a SMIP. This enumerative method iteratively partitions the feasible region, forms estimates of the objective based on those subsets, and then removes subsets that contain no feasible points. The estimates of the objective are statistical estimates and optimality cannot be finitely guaranteed, but statistical estimates of the optimality gap are computed at each iteration. Convergence of the algorithm is illustrated by tests on randomly generated instances from a financial planning application.

Another enumeration based approach uses the Gröbner basis (Shultz et al., 1998). The method divides the first-stage feasible region into a countable number of sections by using properties obtained from the expected recourse function. The authors then make the argument that the optimal solution must be contained in this countable set, and an enumeration algorithm is given to find the optimum from amongst the set.

Ahmed et al. (2004) introduce a branch-and-bound approach on the *tender variables*. The tender variables are defined as right hand side ($h^k - T^k x$) of the second-stage problem. The authors assume fixed h and T matrices for their method. The algorithm reformulates SMIP into a global optimization problem in terms of the tender variables. The reformulation results in a semicontinuous function, and the algorithm proceeds by breaking the function into continuous pieces and searching them for the optimum. Computational results compare the performance of the proposed algorithm with other methods from the literature.

The final two algorithms for SMIP use a scenario-wise decomposition of SP2. The branch-and-fix coordination algorithm (BFC) of Alonso-Ayuso et al. (2003) can be used to solve SMIPs with binary first-stage variables and continuous second-stage variables. The method branches on the first-stage x solutions and uses Dantzig cuts to fix some of the variables which have not been fixed by the branch-and-bound tree. Computational experiments are reported for some large scale models.

A well-known algorithm for solving SMIPs using a scenario-wise decomposition framework is the dual decomposition algorithm (Caroe and Schultz, 1999). The method is applicable to problems with mixed-integer variables in both the first and second stages. The problem is decomposed through the use of nonanticipativity constraints and Lagrangian relaxation. The Lagrangian dual of the LP-relaxation of the problem is then solved, and branch-and-bound is used in order to recover integer solutions. A computational study demonstrates the effectiveness of the algorithm.

CHAPTER III

NODAL DECOMPOSITION-COORDINATION FOR SMIP WITH PRIVATE INFORMATION

A. Introduction

Traditional optimization problems assume that there exists some omnipotent governing body that has access to all the operational information of the system being examined. This coordinating decision maker can then take all of this information and find a solution that is globally optimal for the system. In reality, many systems are composed of independent bodies that are unable or unwilling to reveal their private information regarding their operations to such a decision maker. Thus finding a globally optimal solution when private information restrictions are in place is an important topic in many areas of operations research, including supply chain coordination and homeland security. One application where private information is often ignored is in supply chain coordination of inventory and production schedules. An issue in supply chain management is inventory control, which deals with decisions related to how much inventory to keep at each facility and when to reorder so as to minimize overall inventory ordering and holding costs, service levels, or other relevant objectives.

Problems that arise when attempting global coordination can be very complex even when dealing with a single facility. The difficulty is augmented when expanding to supply chains, and further complicated when considering competitors in the same supply chain, and unwillingness (or practical difficulties) to share critical information. For instance, demand rates and transport times are random variables, while inventory holding costs, pro-

The notation regarding Lagrangian relaxation in this chapter differs from that used in Chapter II.

duction capacity, operational costs, budgets and lead times are typically private information between competitors. Also, the amount of information that must flow from all firms in the supply chain to the central decision maker can be very large and frequent. It is evident from the literature that private data restrictions in coordinated systems are a common concern. However, there still exists a need for methodologies to solve such problems without relaxing this private information requirement and information uncertainty.

This paper introduces nodal decomposition-coordination (NDC) for optimization models with the goal of facilitating global coordination under private information restrictions and stochastic demand. The approach presented is an iterative methodology, whereby each entity in the cooperation forms its own nodal problem, which can be a mixed-integer program (MIP) or stochastic mixed-integer program (SMIP). Lagrangian relaxation and subgradient optimization techniques are used to facilitate negotiation between the nodal decisions without any one entity gaining access to the private information of other entities. In this coordinated system, optimal or close-to-optimal solutions for the system are desired.

This work makes the following contributions to the literature on optimization with private information restrictions: a) stochastic programming model with private information restrictions; b) NDC solution methodology for SMIP with private information restrictions; c) application of the NDC method to supply chain inventory coordination; and d) computational results demonstrating that close-to-optimal solutions can be obtained using the NDC method without compromising private information restrictions. To the best our knowledge, this the first time private information restrictions have been applied to SMIP.

The rest of this paper is organized as follows: In the next section, relevant research from the literature is reviewed. In Section C a model formulation is given for a general problem. In Section D the foundation is developed for the NDC method for the general problem and in Section E the NDC algorithm is presented. Section F introduces a supply chain inventory control model as a special case of the general model and the solution method is applied. Section G discusses solving the models, presents results and draws conclusions.

B. Related Work

Some algorithms suggesting how to solve optimization problems when global sharing of information is not allowed exist in the literature. Fox et al. (2000) explains some of the emerging strategies in supply chain coordination, which employ agents outside of the chain to coordinate the needs of organizations in the supply chain. The view taken in the paper is one which considers each agent in the supply chain to be its own intelligent software system. The authors suggest several traits that the next generation of agile planning systems should address in order to meet the challenges required. They make the case that such supply chain software must be distributed, dynamic, intelligent, integrated, responsive, reactive, cooperative, interactive, anytime, complete, reconfigurable, general, adaptable, and backwards compatible.

Shehory and Kraus (1998) takes a look at the problem of assigning a group of agents to a group of tasks. They use a greedy heuristic algorithm for the set covering and set partitioning problems to aid in grouping the agents into coalitions to solve the set of tasks to be completed. They demonstrate the effectiveness of their algorithm on a simulation of assigning a group of agents to a set of precedence constrained tasks.

Cruijssen et al. (2007) give a transportation example and explain how some of the information sharing concerns affect that industry. In their paper, they discuss the horizontal cooperation practices in Flanders, Belgium. The study is survey based, and includes responses from 1537 logistic service providers in the Flemish region which asked them to agree or disagree with 16 propositions concerning the implementation and effectiveness of horizontal cooperation. Among the conclusions drawn is that even when customer information (which would ordinarily be closely guarded private information) is required to be shared among competing companies, the potential reduction in costs and increase in profits by cooperating with competing firms is attractive to most firms.

Jeong and Leon (2002a) derive a methodology by which to globally solve MIPs displaying a structure that does not allow for some constraints to be globally viewed. The authors term their methodology ‘cooperative interaction via coupling agents’ (CICA). In such a problem structure, different entities in a system where a global optimal value is desired are competing for some resource. The method involves using an artificial entity called a ‘coupling agent’ to handle conflict resolution among entities vying for the use of a shared resource. The work in Jeong and Leon (2002a) has been expanded upon in several other papers. In Jeong and Leon (2003) the authors apply their CICA method to coordination of a facility among business competing for use of the resource. In Jeong and Leon (2005) the method is applied to solve the minimum makespan problem when a machine is required in different steps in an assembly line. In Jeong and Leon (2002b) the method is applied to a system where only two resources are available to produce a final product, but multiple firms have a stake in the production line.

An interesting problem where private information is often ignored is in supply chain coordination of inventory and production schedules. A scalable methodology is presented in Chu and Leon (2009). The method involves using Lagrangian relaxation to take care of constraints that would be private to certain firms in the supply chain and appropriate Lagrangian penalty functions are developed to force the method toward a globally optimal solution. However, the solution strategy presented does not take into consideration the stochastic nature of the parameters in the model. In Chu and Leon (2008) this procedure is applied to a deterministic supply chain model. This work develops a new decomposition-coordination method for the stochastic setting and in essence extends the work of Chu and Leon (2009). Further, the application describing supply chain coordination of inventory and production introduced in Chu and Leon (2009) is used to test the new methodology.

Schneeweiss and Zimmer (2004) describe a made to order production line in which coordination between entities in the system is maintained through the use of external agents. These external agents shield private information of each node from the other node. Giannoccaro and Pontrandolfo (2004) use a contract scheme with revenue sharing to coordinate the efforts of the members in the supply chain while maintaining private information. However, such revenue sharing contract schemes have limitations in some industries, as pointed out in Cachon and Lariviere (2005).

Other applications for which finding a globally optimal solution when private information restrictions are in place exist. An example is in matters of homeland security. Chen et al. (2004) provides a review of relevant areas where information is kept private in a cooperative arrangement. Application areas mentioned include information sharing across jurisdictional bounds, terrorist information collection, modeling and analysis, bioterrorism applications, and border security. The paper proposes the use of current trends in data analysis and information sharing, including system interoperability, data mining, automated event monitoring and visualization.

Phillips Jr. et al. (2002) explains structures where militaries, governments, and civilian companies have to work together in crisis situations, and makes the case that military and government agencies cannot always divulge information to the private sector. The focus of the paper is on coalitions which are formed very quickly in the wake of a large scale crisis such as a natural disaster or terrorist attack. Mendonca (2007) provides a critique of coordination methods used to solve problems in the chaotic and multi-tasked situation that arose in response to the 2001 World Trade Center attack and focuses on developing a set of requirements for computer systems to allow interoperability in future emergency situations. The paper concludes that for a system to be effective in responding to an extreme disaster situation it needs to focus on the areas of categorization, search, assembly, constraint satisfaction, communication, and inference.

There is a limited bank of information in the literature regarding the addition of stochastic data in formulations of supply chain models. In Santoso et al. (2005), the authors apply sampling strategies based on the sample average approximation methodology (Shapiro, 2003) to find high quality solutions to two supply chain coordination formulations. Alonso-Ayuso et al. (2003) formulates a two stage binary supply chain problem with the objective of maximizing expected profit. The model presents the production topology, plant sizing, product selection, product allocation and vendor selection decisions as 0-1 variables in the first-stage, and a solution strategy based on the Branch-and-Fix Coordination strategy is proposed. Escudero et al. (1999) develops a model for a manufacturing, assembly and distribution supply chain with uncertain demand and prices and uses a dual approach splitting variable scheme to solve the formulation. Petrovic et al. (1999) use the idea of fuzzy sets to handle stochastic parameters in a serial supply chain structure. The next section introduces a general problem formulation for problems fitting the nodal decision structure.

C. Problem Formulation

Consider models with a special nodal decision structure in which each node is responsible for optimally assigning its resources toward the node's operations. In this framework, the nodal structure faces some degree of uncertainty in the last node of the cooperation. Such a model forms whenever a group of independent entities work together as a corporation or alliance to reach the common goal of obtaining an optimal *global solution* to a problem under uncertainty. Finding a global solution requires that independent nodal decisions coordinate with other nodes. In this setting, this coordination is represented by a set of resources or decisions called *complicating* or *linking variables* whose allocation must be coordinated with other nodes in the system. A pair of nodes who share linking variables will be called *neighbors* and the neighbors of node i will be denoted $\mathcal{N}(i)$. In the system, no node is allowed to gain knowledge of the operations (represented in the constraint and objective function parameters) of any of its neighbors or any other nodes in the system.

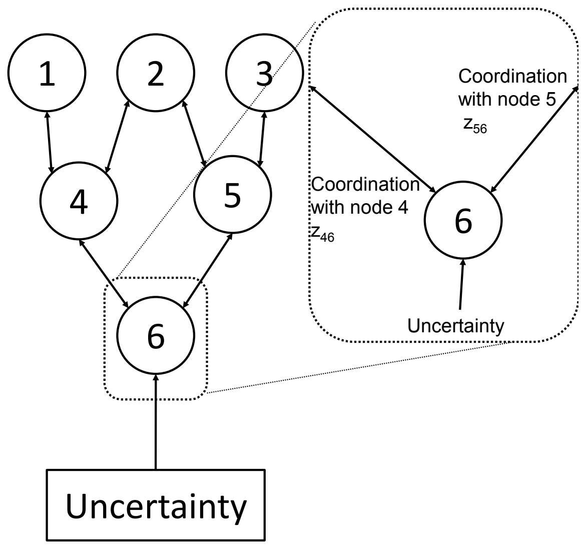

Fig. 4. Graphical representation of nodal decision structure

Figure 4 depicts an example nodal structure with 6 nodes. The “Uncertainty” box represents outside influences (exogenous uncertainty) on the system. For example, in the case of a manufacturing setting demand for final product may not be directly controllable by the system, and thus would fall into this category. The exploded view of node 6 at the right emphasizes that there exist linking variables (represented by the double arrow) between the neighboring node pairs 3 and 6 and 5 and 6. The single arrow at the bottom of the node represents that some parameters from outside the system influence the operations of the node.

The nodal decision problem, which takes the form of a two-stage SLP with special private information constraints can now be formulated. Assume the structure describes a corporation with N nodes, numbered 1 to N. For the problems which fit into this modelling framework the decision problems for nodes 1 through $N-1$ take the form of a deterministic MIP and the decision problem for node N takes the form of a SMIP. The vector z_i will denote the linking variables at node i. For example, in Figure 4, z_6 is a vector consisting of the variables z_{46} and z_{56}. The nodal stochastic program with private information (NSP-PI) restrictions can be given as follows:

$$\text{NSP-PI} : \text{Min} \sum_{i=1}^{N} \left(c_i^\top x_i + d_i^\top z_i \right) + \mathbb{E}[f(\tilde{\omega}, x_N)] \tag{3.1a}$$

$$\text{s.t.} \quad A_i x_i + G_i z_i \geq b_i \qquad i = 1, \ldots, N \tag{3.1b}$$

$$x_i \in X_i, \ z_i \in Z_i \qquad i = 1, \ldots, N \tag{3.1c}$$

$$c_i, d_i, A_i, G_i, b_i, x_i \in \mathcal{P}(i) \qquad i = 1, \ldots, N, \tag{3.1d}$$

where $\mathbb{E}$ is the mathematical expectation operator, $\tilde{\omega}$ is a multivariate random variable, and for each outcome ω of $\tilde{\omega}$, the *recourse* function $f(\omega, x_N)$ for node N is defined as

$$f(\omega, x_N) = \text{Min } q_N(\omega)^\top y_N(\omega) \tag{3.2a}$$

$$\text{s.t.} \quad W_N y_N(\omega) \geq h_N(\omega) - T_N(\omega) x_N \tag{3.2b}$$

$$y_N(\omega) \in Y \tag{3.2c}$$

$$q_N(\omega), W_N, T_N(\omega), V_N(\omega), h_N(\omega), y_N \in \mathcal{P}(N). \tag{3.2d}$$

In NSP-PI (3.1), $x_i \in \mathbb{R}^{n_i^x}$ and $z_i \in \mathbb{R}^{n_i^z}$ are the first-stage decision vectors for node i, while $c_i \in \mathbb{R}^{n_i^x}$ and $d_i \in \mathbb{R}^{n_i^z}$ are the first-stage cost vectors. $A_i \in \mathbb{R}^{m_i^1 \times n_i^x}$ and $G_i \in \mathbb{R}^{m_i^1 \times n_i^z}$ are the first-stage constraints and $b_i \in \mathbb{R}^{m_i^1}$ is the first-stage righthand side. The sets X_i and Z_i impose possible integer (binary) restrictions on all or some components of x_i

and z_i, respectively, in constraints (3.1c). In constraint (3.1d) the set $\mathcal{P}(i)$ imposes private information restrictions on the first-stage problem data for node i.

In the second-stage (recourse) problem (3.2), $y_N(\omega) \in \mathbb{R}^{n_N^y}$ is the recourse decision vector for node N and $q_N(\omega) \in \mathbb{R}^{n_N^y}$ is the second-stage cost vector. $W_N \in \mathbb{R}^{m_N^2 \times n_N^y}$ is the recourse constraint matrix, $T_N(\omega) \in \mathbb{R}^{m_N^2 \times n_N^x}$ is the technology constraint matrix, $V_N(\omega) \in \mathbb{R}^{m_N^2 \times n_N^z}$ is the node linking constraint matrix, and $h_N(\omega) \in \mathbb{R}^{m_N^2}$ is the second-stage righthand side. The set Y_N imposes possible integer (binary) restrictions on all or some components of y_N in constraints (3.1c). Like constraint (3.1c), constraint (3.1d) imposes private information restrictions on the second-stage problem data for node N.

In the above modeling framework the key issue is optimizing the first-stage decisions (x_i, z_i) which have to be made without anticipation of future realizations of $\{q_N(\omega), T_N(\omega), V_N(\omega), h_N(\omega)\}$. After having decided for (x_i, z_i) and observed $\{q_N(\omega), T_N(\omega), V_N(\omega), h_N(\omega)\}$, the remaining decisions $y_N(\omega)$ are made in an optimal way. The constraints (3.1d) and (3.2d) are termed *private information constraints* and represent the restriction that the operations taking place in each node of the system remain private to that node. The imposition of this constraint causes the model shown to be *impossible* to formulate in this explicit form, since no single node nor any overseeing entity is allowed to gather the information required to formulate the model directly! For this reason, a nodal decomposition-coordination approach that preserves the private information constraints will be necessary in order to generate solutions. The following assumptions are made on NSP-PI:

(A1) The random variable $\tilde{\omega}$ follows a discrete distribution with finite support Ω.

(A2) For each node i, the first-stage feasible set $\{A_i x_i + G_i z_i \geq b_i,\ x_i \in X_i,\ z_i \in Z_i,\ c_i, d_i, A_i, G_i, b_i, x_i \in \mathcal{P}(i)\}$ is nonempty.

(A3) For node N, the second-stage feasible set $\{W_N y_N(\omega) \geq h_N(\omega) - T_N(\omega) x_N - V_N(\omega) z_N,\ y_N(\omega) \in Y,\ q_N(\omega), W_N, T_N(\omega), V_N(\omega), h_N(\omega), y_N(\omega) \in \mathcal{P}(N)\}$ is nonempty and bounded for all feasible first-stage x_N.

Assumption (A1) is required to make the problem tractable while assumption (A2) is required to guarantee the existence of an optimal solution. Assumption (A3) is the so-called *relatively complete recourse* assumption in stochastic programming. This assumption can easily be satisfied by modeling the problem with this assumption in mind, or by adding appropriate artificial variables and corresponding induced constraints to the second-stage problem to satisfy relatively complete recourse.

NSP-PI without the private information restrictions falls under two-stage SMIP with recourse, which is still a vibrant area of study (Ruszczyński and Shapiro, 2003, e.g.). In that case if the integer restrictions on the $y_N(\omega)$'s are relaxed, then the expected recourse function $\mathbb{E}[f(\tilde{\omega}, x_N)]$ is a well-behaved piecewise linear and convex function of x_i. Thus Benders decomposition (Benders, 1962) is applicable (Wollmer, 1980). Otherwise, when the second-stage variables involve integrality restrictions, $\mathbb{E}[f(\tilde{\omega}, x_N)]$ is lower semicontinuous with respect to (x_N) (Blair and Jeroslow, 1982), and is generally nonconvex (Schultz, 1993). Thus SMIP problems are generally difficult to solve. In this case, in addition to the computational difficulty that stems from SMIP, NSP-PI involves private information restrictions. Thus the problem being addressed here is a much more difficult in general.

D. A Decomposition-Coordination Method

We now propose an iterative nodal decomposition-coordination method for NSP-PI that allows for negotiation between the nodes in the supply chain. As mentioned previously, formulating NSP-PI explicitly for all nodes is not possible due to the private information constraints (3.1d) and (3.2d). Observe that the only decision variables linking two nodes i and j in NSP-PI are z_{ij}, and that this variable affects only nodes i and j. These linking variables can be considered as being 'complicating' variables in the sense that if each node's decision was allowed to disagree with its neighbors, the problem would decompose directly into N subproblems. Similar to Chu and Leon (2009), introduce the following auxiliary variables to facilitate negotiation:

u_{ij}: Negotiation variable in node i's subproblem denoting node i's proposed value to node j for variable z_{ij} (u_i denotes the vector of u_{ij} variables.)

v_{ij}: Negotiation variable in node i's subproblem denoting node j's proposed value to node i for variable z_{ij} (v_i denotes the vector of v_{ij} variables.)

For a given node i, substitute the variables u_{ij} in (3.1) for z_{ij} to represent the node's complicating decision vector with its neighbor j, and v_{ij} for z_{ij} to denote the corresponding decision vector for neighbor j. In order to solve the original problem, u_{ij} and v_{ij} must be equal for each pair of neighboring nodes i and j. To achieve this, one can implement one of several equivalent negotiation constraints. Two are given below:

$$u_{ij} = \bar{z}_{ij} \tag{3.3a}$$

$$u_{ij} - v_{ij} = 0. \tag{3.3b}$$

The first form (3.3a) forces equality of the negotiation variable in each subproblem to some third value. The constraint works because u_{ij} is found in the subproblem of each pair of neighboring nodes. In practice, the value $\bar{z}_{ij}$ can be set by taking the average $\frac{1}{2}(u_{ij}+v_{ij})$, which is the approach implemented here. The second constraint set is a constraint that directly forces each pair of coordination variables to agree.

Using these new variables and equation, a variable splitting technique can be applied to (3.1) to get N decoupled subproblems. The resulting subproblem for node $i \neq N$ becomes:

$$\text{NSP-PI2}(i): \text{Min } c_i^\top x_i + d_i^\top u_i \tag{3.4a}$$

$$\text{s.t.} \quad A_i x_i + G_i u_i \geq b_i \tag{3.4b}$$

$$u_{ij} = \bar{z}_{ij} \qquad j \in \mathcal{N}(i) \tag{3.4c}$$

$$x_i \in X_i,\ u_i \in Z_i,\ v_i \in Z_i \tag{3.4d}$$

$$c_i, d_i, A_i, G_i, b_i, x_i \in \mathcal{P}(i) \tag{3.4e}$$

and, for node N:

$$\text{NSP-PI2}(N): \text{Min } c_N^\top x_N + d_N^\top u_N + E[f(\tilde{\omega}, x_N)] \tag{3.5a}$$

$$\text{s.t.} \quad A_N x_N + G_N u_N \geq b_N \tag{3.5b}$$

$$u_{ij} = \bar{z}_{ij} \qquad j \in \mathcal{N}(i) \tag{3.5c}$$

$$x_N \in X_N,\ u_N \in Z_N,\ v_N \in Z_N \tag{3.5d}$$

$$c_N, d_N, A_N, G_N, b_N, x_N \in \mathcal{P}(N) \tag{3.5e}$$

where, for a particular realization $\omega \in \tilde{\omega}$:

$$f(\omega, x_N) = \text{Min } q_N(\omega)^\top y_N(\omega) \tag{3.6a}$$

$$\text{s.t.} \quad W_N y_N(\omega) \geq h_N(\omega) - T_N(\omega) x_N \tag{3.6b}$$

$$y_N(\omega) \in Y \tag{3.6c}$$

$$q_N(\omega), W_N, T_N(\omega), V_N(\omega), y_N(\omega) \in \mathcal{P}(N). \tag{3.6d}$$

Lagrangian relaxation can now be applied to relax constraints (3.4c) into the objective. By choosing appropriate penalty parameters, negotiation between nodes i and j can be achieved. The solution procedure will require solving two subproblems at each node: a *coordinated subproblem* which takes into account the propositions made by node (i)'s neighbors by penalizing deviations from $\bar{z}_{ij}$ in node (i)'s decisions, and a *non-coordinated subproblem* that requires that the (i)'s neighbors receive exactly the *compromise* target value $\bar{z}_{ij}$. In each iteration all nodes will first solve their non-coordinated problems. Then, each node will solve their coordinated problem and pass their requests for order and delivery quantities to their neighbors for the next iteration. The procedure will be further explained after these two subproblems are defined.

For each node i define the *non-coordinated subproblem* (NCP) as the optimization problem specific to the node that is required to satisfy exactly the compromise value. For consistency and convenience whenever a variable is treated as data, determined based on the node (i)'s neighbors, it will be designated as such with the ˆ notation. Then substituting for $\bar{z}_{ij}$ in NSP-PI2(i) 3.4 for node i results in the following non-coordinated subproblem for node $1 \neq N$:

$$\text{NCP}(i): \text{Min } c_i^\top x_i + d_i^\top \hat{z}_i \tag{3.7a}$$

$$\text{s.t.} \quad A_i x_i + G_i \hat{z}_i \geq b_i \tag{3.7b}$$

$$x_i \in X_i \tag{3.7c}$$

$$c_i, d_i, A_i, G_i, x_i \in \mathcal{P}(i) \tag{3.7d}$$

and for node N:

$$\text{NCP}(N): \text{Min } c_N^\top x_N + d_N^\top \hat{z}_N + E[f(\tilde{\omega}, x_N)] \tag{3.8a}$$

$$\text{s.t. } A_N x_N + G_N \hat{z}_N \geq b_N \tag{3.8b}$$

$$x_N \in X_N \tag{3.8c}$$

$$c_N, d_N, A_N, G_N, x_N \in \mathcal{P}(i) \tag{3.8d}$$

An issue with feasibility can arise when solving NCP(i). Infeasibility can occur in cases where the compromise value resulting from node (i)'s neighbor's requests generates value for the linking variables which causes NCP(i) to become infeasible. One method to maintain feasibility is to add an artificial variable to 3.7b and penalize it in the objective. This method is adopted later in the computational study section of this paper.

Next the *coordinated subproblem* (CP) needs to be defined. Contrary to NCP, the coordinated problem for node i allows for deviations from the compromise quantities negotiated with its neighboring nodes ($j \in \mathcal{N}(i)$) in the cooperation. An augmented Lagrangian objective function is adopted for CP. Let μ_{ij} denote the Lagrangian multipliers associated with the negotiation constraints (3.3a) in CP for node (i)'s neighbors For each $j \in \mathcal{N}(i)$, let the Lagrangian relaxed constraints be defined as

$$U_{ij} = u_{ij} - \hat{z}_{ij}. \tag{3.9}$$

Note that in the equation $\hat{z}_{ij}$ is known (data) based on calculations from node (i)'s neighbors $j \in \mathcal{N}(i)$. Then CP for node $i \neq N$ can be stated as follows:

$$\text{CP}(i): \text{Min } c_i^\top x_i + d_i^\top u_i + \sum_{j \in \mathcal{N}(i)} \left(\mu_{ij} |U_{ij}| + \frac{\xi_{ij}}{2} U_{ij}^2 \right) \tag{3.10a}$$

$$\text{s.t.} \quad A_i x_i + G_i u_i \geq b_i \tag{3.10b}$$

$$x_i \in X_i, \ u_i \in Z_i \tag{3.10c}$$

$$c_i, d_i, A_i, G_i, x_i \in \mathcal{P}(i) \tag{3.10d}$$

and for node N:

$$\text{CP}(N): \text{Min } c_N^\top x_N + d_N^\top u_N + \sum_{j \in \mathcal{N}(N)} \left(\mu_{Nj} |U_{Nj}| + \frac{\xi_{Nj}}{2} U_{Nj}^2 \right) + E[f(\tilde{\omega}, x_N)] \tag{3.11a}$$

$$\text{s.t.} \quad A_N x_N + G_N u_N \geq b_N \tag{3.11b}$$

$$x_N \in X_N, \ u_N \in Z_N \tag{3.11c}$$

$$c_N, d_N, A_N, G_N, x_N \in \mathcal{P}(N) \tag{3.11d}$$

The term $|.|$ in the objectives of (3.10) and (3.11) denotes the absolute value, and ξ_{ij} is a user defined scalar that keeps node i's decisions from straying too far from the current compromise solution $\hat{z}_{ij}$.

Now consider the calculation of the μ_{ij} penalties using a *subgradient optimization* procedure. To update the penalties using subgradient optimization, a lower and upper bound on the global optimal value is required. However, due to the private data restrictions (constraints (3.10d)) it is *not* possible to know each node's objective function value! A novel way to calculate the bounds without violating the private data requirements will now be presented.

PROPOSITION III.1. *Let UB denote an upper bound on the global optimal value, and* *let NCP$^*_i$ denote the optimal value to NCP* (3.7) *for node $i \in N$. Then UB $= \sum_{i=1}^{N}$NCP$^*_i$*

Proof. NCP$^*_i$ is an upper bound for each node i since a subset of the decision variables are fixed at some feasible values. The result follows by summing the NCP$^*_i$ values for all the nodes. □

PROPOSITION III.2. *Let LB denote a lower bound on the global optimal value, and let CP$^*_i$ denote the optimal value to CP(i)* (3.10) *ignoring the augmented penalty terms ($\frac{\xi_{ij}}{2}U^2_{ij}$) for node i. Then LB $= \sum_{i=1}^{N}$CP$^*_i$*

Proof. Since each nodal CP(i) is a relaxation of the original problem, by dropping the penalty terms from the objective and summing over all the nodes the result follows.

□

While the bounds given in Propositions III.2 and III.1 would be appropriate for use as the lower and upper bounds, doing so *violates private information* constraints (3.7d and 3.10d) since this would require nodes to give their optimal values to all other nodes. To circumvent this problem, penalties will be passed between nodes during the negotiation process. Intuitively, these penalties should be based on the difference between what is best for the node and what would be best for the node's neighbors. Thus, for each node i define π_i to be the *compensation* that node i would be willing to pay to have their optimal solution accepted over the its neighbor's proposed solutions. Then the bounds on the global optimal value can be calculated by having all nodes i and j exchange the penalty values π_i without having to relate the objective values for either of its subproblems.

COROLLARY III.3. *For each node $i = 1, \ldots, N$, let LB and UB be as defined in Propositions III.2 and III.1. Then*

$$UB - LB = \sum_{i=1}^{N} \pi_i. \tag{3.12}$$

Proof. For each node, let $\pi_i = \text{NCP}_i^* - \text{CP}_i^*$. where CP_i^* and NCP_i^* are as defined in Proposition III.2 and III.1, respectively. Then

$$\begin{aligned} \text{UB} - \text{LB} &= \sum\nolimits_{i=1}^{n} \text{NCP}_i^* - \sum\nolimits_{i=1}^{n} \text{CP}_i^* \\ &= \sum\nolimits_{i=1}^{n} \left(\text{NCP}_i^* - \text{CP}_i^*\right) \\ &= \sum\nolimits_{i=1}^{n} \pi_i \end{aligned}$$

□

Corollary III.3 allows the calculation of $UB - LB$ for use in the subgradient procedure using only the penalty values $\pi_i, \forall i \in N$ without gaining access to each node's objective (cost) value information.

One other piece of information is required in order to appropriately update the Lagrangian penalties: a measure of the infeasibility of the current solution in terms of the relaxed constraints. In practice, this is typically accomplished by squaring the norm of the violated constraints. In this case, this is accomplished by requiring each node to pass a single value calculated as follows:

$$s_i = \sum_{j \in \mathcal{N}(i)} (\hat{u}_{ij} - \hat{z}_{ij})^2 \tag{3.13}$$

where $\hat{u}_{ij}$ is the current solution.

Next it is proven that the objective functions of all CP_i's are suitable for employing subgradient optimization. Recall that in the objective function of CP_i the relaxed linking constraints (3.3a) appear with the μ penalty terms as defined in equation (3.9). For the cost function in (3.1), the Lagrangian relaxation function for node $i \neq N$, denoted $\phi_i(\mu)$, is given by

$$\phi_i(\mu) = \text{Min } c_i^\top x_i + d_i^\top z_i + \sum_{j \in \mathcal{N}(i)} \mu_{ij} \left|(u_{ij} - \hat{z}_{ij})\right| \tag{3.14}$$

and for node N:

$$\phi_N(\mu) = \text{Min } c_N^\top x_N + d_N^\top z_N + \sum_{j \in \mathcal{N}(N)} \mu_{Nj} \left|(u_{Nj} - \hat{v}_{Nj})\right| + E[f(\tilde{\omega}, x_N)] \tag{3.15}$$

The presence of the the penalty terms $\mu_{ij} \left|(u_{ij} - \hat{v}_{ij})\right|$ in CP$_i$ is clearly nonlinear due to the presence of the absolute value function $|.|$. To linearize it, a standard variable substitution technique can be employed. For each $|u - \hat{v}|$, replace $|u - \hat{v}|$ by $u^+ - u^-$ in the objective, and add the constraint $u - \hat{u} = u^+ - u^-,\ u^+, u^- \geq 0$ to the formulation. Next observe that since the objective of CP(i) consists of linear functions, the subproblem in each node $i = 1, \ldots, N$, $\phi_i(\mu)$ is *concave*. Therefore, subgradient optimization is appropriate for this setting. However, it is important to note that there may be a duality gap between the subgradient optimal solution and the optimal solution to NSP-PI due to the integer restrictions on the decision variables. The Lagrangian dual objective for node i is given by

$$\phi_i = \underset{\mu}{\text{Max}} \quad \phi_i(\mu). \tag{3.16}$$

Next, the appropriate *subgradients* to use in the subgradient procedure must be defined. This will be based on the coordinated problems CP$_j$ (3.10). Recall that for a concave function $f : \mathbb{R}^m \mapsto \mathbb{R}$ a subgradient $s(\bar{x}) \in \mathbb{R}^m$ of f at $\bar{x} \in \mathbb{R}^m$ must satisfy $f(x) - f(\bar{x}) \leq s(\bar{x})(x - \bar{x})$.

PROPOSITION III.4. *Consider the dual objective function ϕ_i (3.16) for node i. Then*

$$s_i(\bar{u}_i) = \sum_{j \in \mathcal{N}(i)} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right| \tag{3.17}$$

is a subgradient of $\phi_i(\mu)$ at $\bar{\mu}_{ij}$ for node i, provided $\bar{u}_i$ is feasible to CP$_i$.

Proof. Given $\bar{\mu}_{ij}$, let the optimal solution to $\phi_i(\mu_i)$ (3.14) be $(\bar{x}_i, \bar{u}_i)$ for $i \neq N$. For any

μ_i, let the optimal solution be (x_i, u_i). Then $\phi_i(\mu_i)$ (3.14) becomes:

$$\phi_i(\bar{\mu}_i) = c_i^\top \bar{x}_i + d_i^\top \bar{u}_i + \sum_{j \in \mathcal{N}(i)} \bar{\mu}_{ij} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right|.$$

And for any μ_{ij},

$$\phi_i(\mu_i) = c_i^\top x_i + d_i^\top u_i + \sum_{j \in \mathcal{N}(i)} \mu_{ij} \left|(u_{ij} - \hat{v}_{ij})\right|.$$

Then

$$\begin{aligned}\phi_i(\mu_i) - \phi_i(\bar{\mu}_i) &= c_i^\top x_i + d_i^\top u_i + \sum_{j \in \mathcal{N}(i)} \mu_{ij} \left|(u_{ij} - \hat{v}_{ij})\right| - \\ &\quad \left(c_i^\top \bar{x}_i + d_i^\top \bar{u}_i + \sum_{j \in \mathcal{N}(i)} \bar{\mu}_{ij} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right|\right) \\ &\le c_i^\top \bar{x}_i + d_i^\top \bar{u}_i + \sum_{j \in \mathcal{N}(i)} \mu_{ij} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right| - \\ &\quad \left(c_i^\top \bar{x}_i + d_i^\top \bar{u}_i + \sum_{j \in \mathcal{N}(i)} \bar{\mu}_{ij} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right|\right) \\ &= \sum_{j \in \mathcal{N}(i)} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right|(\mu_{ij} - \bar{\mu}_{ij}),\end{aligned}$$

where the inequality comes from replacing the minimal values from $\phi(\mu_i)$ with $(\bar{x}_i, \bar{u}_i)$. Thus it holds,

$$\phi_i(\mu_i) - \phi_i(\bar{\mu}_i) \le \sum_{j \in \mathcal{N}(i)} \left|(\bar{u}_{ij} - \hat{v}_{ij})\right|(\mu_{ij} - \bar{\mu}_{ij})$$

The proof for node $i = N$ is similar; the expectation drops out after the inequality.

$\square$

Now that it has been proven that subgradient optimization can be applied to this problem, a solution procedure can be formally outlined. The regularization term is added and the Lagrangian penalties are updated using subgradient optimization. In the iterative process, let k be the iteration index for parameters that depend on the iteration. Then, for each constraint of type (3.3a) in the objective with associated penalty μ^k, the penalty between supplier node $i \in \alpha(j)$ and j is updated using the following scheme:

$$\mu_{ij}^{k+1} = \mu_{ij}^k + \delta_{ij}^k \big|(u_{ij} - \hat{z}_{ij})\big|,$$

where

$$\delta_{ij}^k = \frac{\lambda^k (\sum_{i=1}^N \pi_i)}{\sum_{i=1}^N s_i}$$

and λ^k is a user defined scalar initialized to some value $0 < \lambda^0 < 2$.

Several heuristics for updating λ^k have been proposed in the literature. A common scheme for updating λ is to halve λ^k whenever the best lower bound found has failed to increase in some fixed number of iterations. However, preliminary computational experiments suggested that this approache converged too slowly to be practical. For this reason, another approach was adopted for this paper.

Consider the heuristic found in Caprara et al. (1999), which allows for the value of λ to increase or decrease based on the rate of convergence. After a predefined number of iterations, the value of λ is decreased if a large enough change in the lower bound was seen, and it is increased if some level of change was not seen. In this case, since access to the lower bound is explicitly prohibited by the private information constraints, the idea from Caprara et al. (1999) is implemented based on a change in the gap between the upper bound and lower bound. Let $\hat{\pi}^k = \sum_j^n \pi_j$ be the value found in iteration k. Then

$$\lambda^k = \begin{cases} 1.5\lambda^{k-1}, & \text{if } \frac{\hat{\pi}^{k-1}}{\hat{\pi}^{k-2}} < \gamma_1; \\ \lambda^{k-1}/2, & \text{if } \frac{\hat{\pi}^{k-1}}{\hat{\pi}^{k-2}} > \gamma_2; \\ \lambda^{k-1}, & \text{otherwise} \end{cases}$$

where $0 < \gamma_1 < \gamma_2 < 1$.

It should be noted that in this case one can consider splitting λ^k into λ_{ij}^k for each $j \in \alpha(i)$ and updating at each node i separately. At each iteration, the node has access to its own upper and lower bounds. In implementing the algorithm, one could allow each node to change its own λ value at each iteration based on the change in its lower bound or gap as described for the global λ value above. However, in this paper, this was not done since if any member in the cooperation became 'greedy' and had the power to choose its λ-updating scheme, it could artificially change its lambda value to gain an advantage over the other nodes.

E. NDC Algorithm

The subgradient algorithm, referred to as the nodal decomposition-coordination algorithm (or NDC algorithm for short), for finding the global solution to NSP-PI is now presented. In this implementation, a central authority referred to as the *central control* is used to calculate the $\sum_i \pi_i$ and $\sum_i s_i$ values and distribute them to all nodes. The algorithm is formally stated in Figure 5.

Recall that for linear programs, given a good choice of starting λ^0 and a strategy to update λ^k that satisfies the requirements mentioned in Section D, it can be guaranteed that, in theory, the algorithm will converge to the global optimal solution. However, it is well documented in the literature that finding such a "good" starting λ^0 and updating rule is very problem specific and no general rule that works well for all problems exists. Therefore, traditional Lagrangian relaxation implementations either choose a very small λ^0 value, which leads to long runtimes but optimal or very close to optimal solutions, or a larger value, which leads to faster run times but often falls short of optimality. For the computational experiments described later, the second approach was chosen.

One of the assumptions of this algorithm is that every "compromise" solution set in the NCPs is feasible. This feasibility can be maintained through artificial variables with carefully chosen penalties, but there is a particular danger in this framework that can result.

NDC Algorithm

Step 0. Initialization. Set $k = 0$ and choose λ^0, ξ and $\epsilon > 0$. For each $i = 1, \ldots, N$: (a) Set $\mu_{ij}(k) = 0$ for all $j \in \mathcal{N}(i)$ in CP$_i$; (b) Set $\hat{v}_{ij} = 0$ for all $j \in \mathcal{N}(i)$ in CP$_j$.

Step 1. Solve the coordinated problems and pass information. For $i = 1, \ldots, N$: Solve CP(i) (3.10). Pass requests (u_{ij}) to neighboring nodes $j \in \mathcal{N}(i)$.

Step 2. Solve the non-coordinated problems. For $i = 1, \ldots, N$, use $(\hat{v}_{ij})$ to set $(\bar{z}_{ij})$ values in NCP(i) (3.7), and then solve NCP(i).

Step 3. Solve the coordinated problems and pass information. For $i = 1, \ldots, N$: Set $(\bar{z}_{ij})$ in CP(i) and solve CP(i); calculate π_i and s_i; share π_i and s_i with central control.

Step 4. Share information Calculate the sum $\hat{\pi}^k = \sum_i \pi_i$ and $\sum_i s_i$ at central control. Pass values to nodes $i = 1, \ldots, N$.

Step 5. Termination. If $\hat{\pi}^k - \hat{\pi}^{k-1} < \epsilon$, stop. Report the solution to NCP(i) as the best implementable solution found.

Step 6. Pass information and perform updates. Send $\hat{\pi}^k$ to all nodes. Update λ^k, if necessary. For $i = 1, \ldots, N$ calculate $\mu_{ij}(k+1)$ for all $j \in \mathcal{N}(i)$. Set $k \leftarrow k + 1$ and go to Step 2.

Fig. 5. The Nodal Decomposition-Coordination Algorithm

When penalties imposed on the relaxed constraints grow too quickly due to a λ^0 or ξ that is too large, the algorithm can converge to a suboptimal point. High values for μ and ξ at each node make it too expensive for the node to choose any solution vector that differs from the compromise proposals. At each iteration, the nodes simply agree to supply the compromise solution $\hat{z}$. This solution may be unimplementable because it can cause artificial variables to enter the basis at termination. In this case, initialization of the penalties should be adjusted such that the penalties grow at a slower rate or the artificial variables' penalty values in NCP should be adjusted.

F. A Supply Chain Application

Computational experiments on the NDC Algorithm were performed using an application from the supply chain setting. The problem studied is the supply chain inventory coordination problem (SCICP) with private information restrictions. For this problem, the nodal structure in Figure 4 takes the form of a supply chain where each node represents a facility that receives raw materials from other nodes (called the node's *suppliers*) and sends a finished good to another node (called the node's *buyer*.) Thus, a finished good from one node is a raw material to another.

Figure 6 gives a graphical representation of the supply chain system being studied. The left side shows a global view of the supply chain. The supply chain shown shows three levels of nodes, which will be referred to as *echelons*. The arrows represent the flow of finished good products from supplier nodes to buyer nodes, and the bottom box represents the demand from the consumer.

The right half of Figure 6 shows an exploded view of the internal processes of an arbitrary node j. The *Work-in-process* oval represents the decisions made concerning the value-added processes on the raw materials, which flow in from the supplier nodes at the top. The

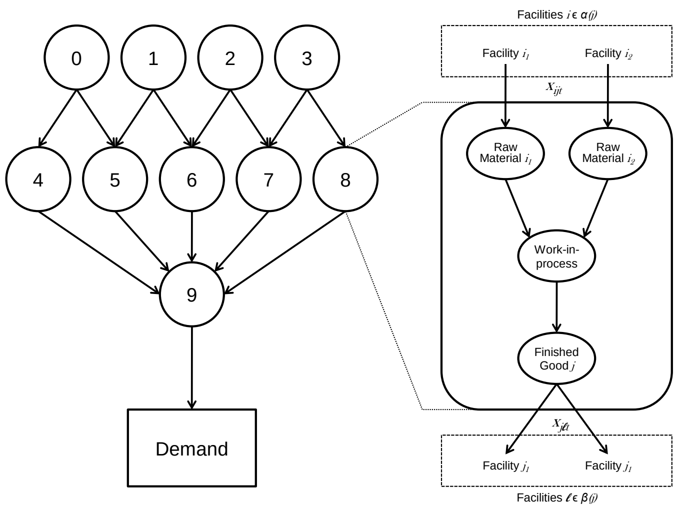

Fig. 6. Graphical representation of a supply chain

Raw Material and *Finished Good* ovals represent the decisions regarding inventory levels of raw materials and finished goods at the node in each time period. The rounded rectangle surrounding the nodal representation emphasizes that the parameters and decisions governing the operations carried out within the node are private information. The following notation will be used to formulate SCICP on this supply chain structure.

Indices

i:	Index for facilities (or good types)
j:	Index for facilities (or good types)
ℓ:	Index for facilities (or good types)
t:	Index for planning periods
n:	Index for last facility in the supply chain
ω:	Index for demand outcome of a multivariate random variable $\tilde{\omega}$

Sets

$\mathcal{T}$:	Set of planning periods
$\mathcal{N}$:	Set of facilities (or good types) in the system
$\alpha(j)$:	Set of immediate predecessors (suppliers) of facility j
$\beta(j)$:	Index denoting the immediate successors (buyers) of facility j
$\mathcal{P}(j)$:	Set of parameters that are private to facility j
$\tilde{\omega}$:	Set of all possible (outcomes) scenarios ω

Parameters and Functions

$S^R_{ijt}(x)$:	Cost for facility j to order x units of raw materials from facility i in period t
$S^F_{jt}(y)$:	Cost for facility j to produce y units of finished product in period t
$H^R_{ijt}(x)$:	Holding cost at facility j for x units of raw material i at the end of period t
$H^F_{jt}(y)$:	Holding cost at facility j for y units of finished goods j at the end of period t
m_{ij}:	Amount of material i required to make one unit of good j at facility j
$d_t(\omega)$:	Demand for the end product in period t under scenario ω
M:	a very large number; should be defined to be greater than all possible objective costs realizable by the system
h^-_{nt}:	Cost to outsource a unit of end product in period t after demand realization
h^+_{nt}:	Inventory cost for a unit of end product in period t after demand realization
h^I_{nt}:	Cost to replace inventory used to satisfy demand which was not met by the first-stage decision.

Decision Variables

I^R_{ijt}:	Raw material inventory type i held at facility j at end of period t
I^F_{jt}:	Finished good inventory at facility j at the end of period t
x_{ijt}:	Amount of raw material i to send from facility i to facility j in period t ($x_{ij} = [x_{ij1}, ..., x_{ij\|\mathcal{T}\|}]^\top$, $x_j = [x_{1j}, ..., x_{\|\mathcal{N}\|j}]^\top$: and $x = [x_1, ..., x_{\|\mathcal{N}\|}]^\top$)
y_{jt}:	Production quantity of finished goods to make at facility j in period t ($y_j = [y_{j1}, ..., y_{j\|\mathcal{T}\|}]^\top$ and $y = [y_1, ..., y_{\|\mathcal{N}\|}]^\top$)
$z^-_{nt}(\omega)$:	Excess demand not satisfiable by the first-stage decision that is to be satisfied by outsourcing under scenario ω
$z^I_{nt}(\omega)$:	Excess demand not satisfiable by the first-stage decision that is to be satisfied by pulling stock from inventory under scenario ω
$z^+_{nt}(\omega)$:	Extra units of stock allocated in the first-stage that are to be kept in inventory under scenario ω

In describing the flow of materials in this system, it is important to note that each node can have multiple buyer nodes within the supply chain but produces only one “finished good” product. This finished good can be thought of as a subassembly required by the next node in a production system to create the next subassembly, thus, a finished good from one node becomes a raw material to its buyer. At the final node, where the demand is viewed, the finished good is referred to as the “final product.” It is assumed that the cost for transporting a final product from one node to a buyer node is absorbed by the buyer node.

Based on the above notation, the SCICP can be stated as the following two-stage SMIP:

$$\text{SCICP}: \text{Min} \sum_{j \in \mathcal{N}} \sum_{t \in \mathcal{T}} \Big\{ \sum_{i \in \alpha(j)} (S^R_{ijt}(x_{ijt}) + H^R_{ijt}(I^R_{ijt})) + S^F_{jt}(y_{jt}) + H^F_{jt}(I^F_{jt}) \Big\} + E[f(x_n, \tilde{\omega})] \quad (3.18a)$$

$$\text{s.t.} \quad I^R_{ij(t-1)} + x_{ijt} - I^R_{ijt} - m_{ij} y_{jt} = 0, \quad \forall j \in \mathcal{N}, i \in \alpha(j), t \in \mathcal{T} \quad (3.18b)$$

$$I^F_{j,t-1} + y_{jt} - I^F_{jt} - \sum_{\ell \in \beta(j)} x_{j\ell t} = 0, \quad \forall j \in \mathcal{N}, t \in \mathcal{T} \quad (3.18c)$$

$$S^R_{ijt}(\cdot), S^F_{jt}(\cdot), H^R_{ijt}(\cdot), H^F_{jt}(\cdot), \quad m_{ij} \in \mathcal{P}(j), \forall j \in \mathcal{N}, i \in \alpha(j), t \in \mathcal{T} \quad (3.18d)$$

$$I^R_{ijt}, I^F_{jt}, x_{ijt}, x_{j\ell t}, y_{jt} \geq 0, \quad \forall j \in \mathcal{N}, i \in \alpha(j), \ell \in \beta(j), t \in \mathcal{T} \quad (3.18e)$$

$$I^R_{ijt}, I^F_{jt}, x_{ijt}, x_{j\ell t}, y_{jt} \text{ Integer}, \quad \forall j \in \mathcal{N}, i \in \alpha(j), \ell \in \beta(j), t \in \mathcal{T} \quad (3.18f)$$

where, for a particular scenario (outcome) $\omega \in \tilde{\omega}$ of $\tilde{\omega}$ for facility n (last facility, $i = n$) :

$$f(x_n, \omega) = \text{Min} \sum_{t \in \mathcal{T}} \big\{ h^-_{nt} z^-_{n,t}(\omega) + h^+_{nt} z^+_{nt}(\omega) + h^I_{nt} z^I_{nt} \big\} \quad (3.19a)$$

$$\text{s.t.} \quad z^-_{nt}(\omega) + z^I_{nt}(\omega) - z^+_{nt}(\omega) \geq d_t(\omega) - x_{n\ell t}, \quad \ell \in \beta(n), \forall t \in \mathcal{T} \quad (3.19b)$$

$$- z^I_{nt}(\omega) \geq -I^F_{nt}, \quad \forall t \in \mathcal{T} \quad (3.19c)$$

$$z^-_{nt}(\omega), z^I_{nt}(\omega), z^+_{nt}(\omega) \geq 0, \quad \forall t \in \mathcal{T} \quad (3.19d)$$

$$z^-_{nt}(\omega), z^I_{nt}(\omega), z^+_{nt}(\omega) \text{ Integer}, \quad \forall t \in \mathcal{T} \quad (3.19e)$$

The objective function (3.18a) prescribes finding the minimum production and inventory costs, plus the expected cost regarding inventory recourse decisions. Constraint (3.18b) describes the balance between the inventory of on-hand of raw materials, order quantities and production quantities, while constraint (3.18c) describes the balance between the inventory on hand of finished goods/sub-assemblies, production quantities, and outbound flow of finished goods/sub-assemblies. Constraint (3.18d) describes the private data restrictions, that is, it enforces that certain data parameters are only known by the facility where those parameters are set. Constraint (3.18e) enforce the non-negativity restrictions on the decision variables and constraint (3.18f) enforces integrality restrictions.

In the subproblem, the objective function (3.19a) prescribes finding the minimum costs associated with accounting for the randomness in the demand for a given scenario. Constraint (3.19b) describes the balance between demand realization, first-stage production decision, emergency product acquisition, and inventory variables. In this model the first-stage inventory decisions offer their inventory to satisfy future demand. The model assumes that there is a cost associated with unsatisfied demand for the end product (e.g. outsourcing or last minute production at some higher cost). Constraint (3.19c) ensures that no more inventory than is available can be assigned to make up for excess demand in any period. Finally, constraints (3.19d) enforce the non-negativity requirements on the decision variables and constraint (3.19e) enforces integrality restrictions.

In SCICP, the objective cost functions in (3.18a) are intentionally left in a general format to stress the adaptability of the NDC algorithm to different types of supply chains. Two classes of cost functions will now be discussed. The first is a pure linear cost function and the second is a mixed-integer cost function. For both cases, the following shared notation is used:

Cost Parameters

C_{ij}^R:	Setup cost at facility i for ordering from facility j
c_{ij}^R:	Per unit cost at facility i for ordering from facility j
h_{ij}^R:	Per unit holding cost at facility i to hold raw material of type j from one time period to the next
C_i^F:	Fixed production setup cost at facility i.
c_i^F:	Per unit setup cost at facility i.
h_i^F:	Per unit holding cost at facility i to hold one unit of its own finished product/subassembly from one time period to the next

1. Linear Cost Functions

Linear cost functions are the most straightforward value functions, and the ones considered in the test instances solved in this paper. For this case, the value functions given in (3.18a) of SCICP are:

$$S_{ijt}^R(x_{ijt}) = c_{ijt}^R x_{ijt}$$

$$S_{it}^F(y_{it}) = c_{it}^F y_{it}$$

$$H_{ijt}^R(I_{ijt}^R) = h_{ij}^R I_{ijt}^R$$

$$H_{jt}^F(I_{jt}^F) = h_j^F I_{jt}^F$$

Incorporating the value functions for the linear case into SCICP yields the following model:

$$\begin{aligned}\text{SCICP}_L: \text{Min} \sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}}\{ \sum_{j\in\beta(i)} (C^R_{ijt}x^B_{ijt} + c^R_{ijt}x_{ijt} + h^R_{ijt}I^R_{ijt}) + C^F_{it}y^B_{it} + c^F_{it}y_{it} + h^F_{it}I^F_{it}\} \\ + E[f(x_n,\tilde{\omega})] \\ \text{s.t.}\quad Constraints\ (3.18b) - (3.18e)\end{aligned}$$

where, for a particular demand realization ω of $\tilde{\omega}$ for the facility n (last facility, $i = n$), $f(x,\omega)$ is as given in formulation (3.19).

Decomposing the problem assuming a linear objective function is a direct application of the decomposition laid out in section (D). For each facility $i \in \mathcal{N}$ the following value function describes the objective function for that node, and in the case of node n, describes the first-stage objective portion:

$$V_i = \sum_{t\in\mathcal{T}}\{ \sum_{j\in\beta(i)} (c^R_{ijt}x_{ijt} + h^R_{ijt}I^R_{ijt}) + c^F_{it}y_{it} + h^F_{it}I^F_{it}\}.$$

2. Fixed Charge Cost Functions

In the case where placing an order and producing a product for a given time period requires some setup cost, SCICP is a mixed-binary problem whose objective function includes a fixed charge. Using the notation given in F, the value functions in the objective (3.18a) of SCICP becomes:

$$S^R_{ijt}(x_{ijt}) = \begin{cases} C^R_{ijt} + c^R_{ijt}x_{ijt} & \text{if } x_{ijt} > 0 \\ 0 & \text{otherwise}\end{cases}$$

$$S^F_{it}(y_{it}) = \begin{cases} C^F_{it} + c^F_{it}y_{it} & \text{if } y_{it} > 0 \\ 0 & \text{otherwise}\end{cases}$$

$$H^R_{ijt}(I^R_{ijt}) = h^R_{ij} I^R_{ijt}$$

$$H^F_{jt}(I^F_{jt}) = h^F_j I^F_{jt}$$

In practice, it is common that the linear parts $c^R_{ijt}x_{ijt}$ and $c^F_{it}y_{it}$ in the above cost functions are ignored. In this case, setting $c^R_{ijt} = 0$ and $c^F_{it} = 0$ gives the desired model.

To implement $S^R_{ijt}(x_{ijt})$ and $S^F_{it}(y_{it})$, it is necessary to define two new decision variables:

$$x^B_{ijt} = \begin{cases} 1 & \text{if } x_{ijt} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$y^B_{it} = \begin{cases} 1 & \text{if } y_{it} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Incorporating these refined value functions into SCICP yields the following model:

$$\text{SCICP}_{FC} : \text{Min} \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} \Big\{ \sum_{j \in \beta(i)} (C^R_{ijt}x^B_{ijt} + c^R_{ijt}x_{ijt} + h^R_{ijt}I^R_{ijt}) + C^F_{it}y^B_{it} + c^F_{it}y_{it} + h^F_{it}I^F_{it} \Big\}$$

$$+ \; E[f(x_n, \tilde{\omega})] \tag{3.21a}$$

$$\text{s.t. } Constraints\ (3.18b) - (3.18e)$$

$$x_{ijt} - Mx^B_{ijt} \leq 0 \quad \forall i \in \mathcal{N}, j \in \beta(i), t \in \mathcal{T} \tag{3.21b}$$

$$y_{it} - My^B_{it} \leq 0 \quad \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{3.21c}$$

$$x^B_{ijt}, y^B_{it} \in \mathbb{B} \tag{3.21d}$$

In SCICP$_{FC}$ constraint (3.21b) assigns a binary variable that equals one if raw materials j are ordered in period t at facility i. Constraint (3.21c) assigns a binary variable that equals one if good i is produced in period t. Finally, constraint (3.21d) gives the binary restrictions on the additional decision variables. The value of M must be chosen such that it is large enough to allow for the binary variable being set to 1 to maintain feasibility of the constraint for all possible values of $x_{n,j,t}$ and $y_{n,t}$.

G. Experiments

In order to test the efficacy of the NDC Algorithm, a number of random test instances of the SCICP problem were created. Each instance includes two forms: a stochastic problem in which the demand is represented by a set of random variables, and a deterministic version in which the demand is taken to be the expected value of those random variables. The stochastic version will be referred to as the *stochastic problem* and the expected value problem will be referred to as the *EV problem*. Generation of random instances for testing will be discussed, followed by a discussion of the results.

1. Instance Generation

The algorithm was tested on a number of randomly generated test instances. The instances have different sizes and all parameters were randomly generated. In the instances, m_{ij} was sampled from a uniform distribution between one and five and costs were sampled from truncated normal distributions. The cost for ordering was sampled using a mean of 100 and standard deviation of 30 and the cost for production was sampled using a mean of 200 and standard deviation of 50. Inventory holding costs per time period for raw materials were sampled from a truncated normal distribution with mean 40 and standard deviation of 15 and for finished goods from a truncated normal distribution with mean 50 and standard deviation of 15. As mentioned in Section D, in order to get a solution for each of the non-coordinated problems, an artificial variable was added and penalized in the objective. This penalty value was calculated by finding an upper bound of the cost to manufacture

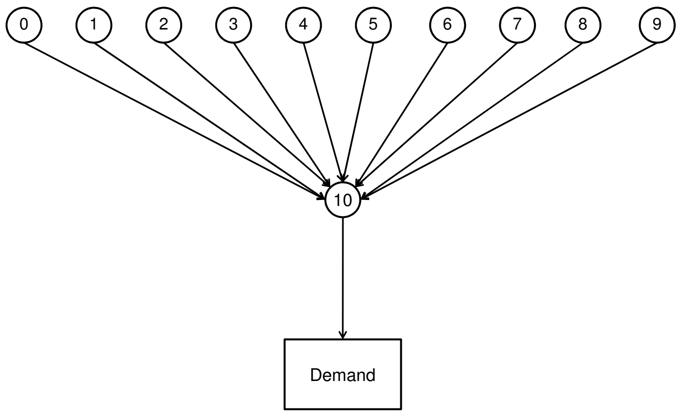

Fig. 7. Graphical representation of instances sized 10-1

one finished good at the top echelon of nodes and propagating that cost through the system based on the number of raw materials used to create the lower tiers' finished goods. These parameters were decided upon through experimentation.

The instances are named according to their size. Each instance includes a number of echelons which ranged from two to four. Each of the instances was created over a six-time period time horizon. For example, an instance referred to as having size 5-10-1 would have five nodes in the top echelon, 10 intermediate nodes, and a final node which sees the demand. The three test sizes chosen were 10-1, 10-20-1 and 10-15-10-1. For the interested reader, the structure of the each of the instances is shown graphically in Figures 7 - 9. Each instance was generated with three uniformly-distributed independent outcomes of the random demand in each time period.

For stochastic programs with discrete probability distributions, it is possible to formulate an equivalent single large scale IP called the *deterministic equivalent problem* (DEP).

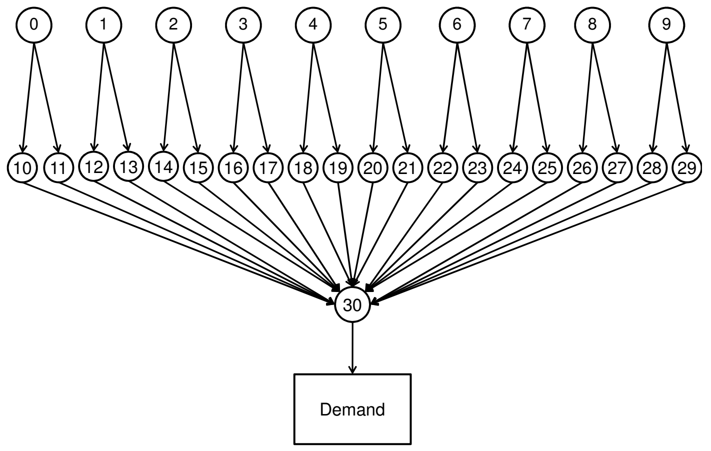

Fig. 8. Graphical representation of instances sized 10-20-1

This is accomplished by explicitly handling the expectation in SCICP (3.18) by including decision variables for each realization of the random variables in the formulation. For this study the DEPs were formulated for both the coordinated and non-coordinated problems in the last node and solved directly in CPLEX.

Traditional optimization methods would begin by taking the average of the random demand and forming a deterministic expected value (EV) IP for the last node. To illustrate the increase in complexity between the EV and DEP formulations, consider the instances of size 10-15-10-1. The EV problems for the last node would contain 330 variables and 198 constraints while the DEP problems contained 13452 variables and 8946 constraints. To justify this increase in problem size and complexity, the effect of using the EV solution in favor of the stochastic problem solution was simulated by calculating the *value of the stochastic solution* (VSS). In stochastic programming, the VSS is usually calculated using optimal values to a stochastic mixed-integer program's EV problem and its recourse ver-

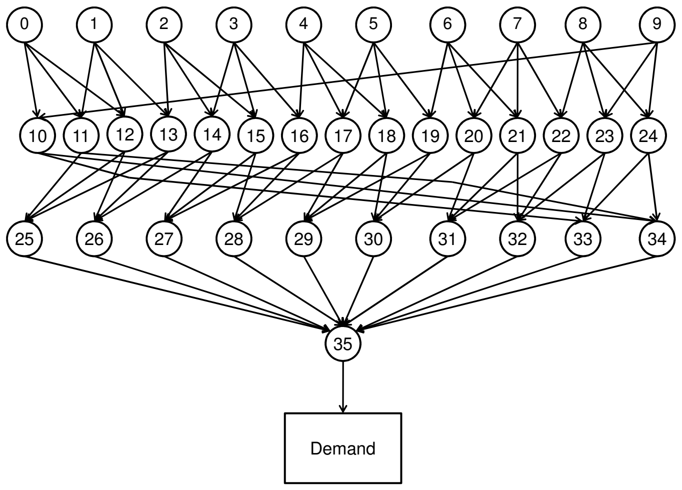

Fig. 9. Graphical representation of instances sized 10-15-10-1

sion. To accomplish this, the private information constraints were relaxed, which allowed for a single *stochastic global problem* to be formulated. Expected values for the demand were then calculated and the *deterministic global problem* was formulated. By solving the deterministic global problem and fixing the optimal solution vector in the global stochastic problem, a measure of the expected cost of using the expected values for the demand could be determined. The VSS was then calculated by subtracting the value obtained by solving the global stochastic problem without fixing any variables. It must be emphasized that these two global problems were formulated for illustrative purposes only. In practice, neither of the problems can be formulated, as doing so violates the private information constraints.

2. Computational Results

The NDC Algorithm was implemented using C++ and the CPLEX 12 Callable Library (IBM ILOG CPLEX, 2009) and tests were run on a Dell X5355 computer with Intel(R)

323 Xeon(R) X processors at 2.66 GHz each with 12.0 GB of RAM. The inputs for the algorithm were an IP model describing the non-coordinated problem at each node and an IP descibing the coordinated problem at each node, both in MPS file format. For the SIP nodes, the MPS files were large scale stochastic programs in their deterministic equivalent form. In addition to the nodal problems a text file to facilitate the negotiation between nodes was also created.

As private information restrictions are essential to the motivation of the NDC Algorithm, it is important to consider the information that was necessary to achieve this negotiation. The global information contained in the file included the total number of nodes and the number of time periods. For each node, the information contained in the file was which nodes in the chain were its suppliers, which nodes in the supply chain were its buyers, and what variables in each of that node's subproblems represented the coordination variables (proposed order and purchase quantities.)

As described in Section D, the linearisation of the absolute values in equations (3.9) added two variables to the formulation for each coordination variable. In this implementation of the algorithm, those variables were also listed as inputs. However, these variables were only used to facilitate the passing of order and delivery proposals between a node and its neighbor; thus the integrity of private information at each node was preserved. In practice, each node would solve its own problem, and would internally be able to calculate the discrepancies between its own optimal solution and the coordination proposals corresponding to its neighbors' order requests and delivery proposals.

For each of the three test sizes, the results for six replications are reported. For each instance, two versions were created: one which maintained the private data restrictions, and another which ignored those constraints in order to find the best possible solution for comparison purposes. Relaxing the private information constraints allowed the creation of a single large scale stochastic mixed-integer program. The objective value solution to these

Table II. Computational results size 10-1

Instance	**1**	**2**	**3**	**4**	**5**	**6**
Iterations	1398	1024	925	1014	1261	910
Runtime (sec)	3600	3600	3600	3600	3600	3600
NDC Solution	65.754	49.164	39.160	54.642	44.744	61.136
Global solution	65.135	48.563	38.579	53.975	44.207	60.624
Gap %	0.95%	1.24%	1.51%	1.24%	1.21%	0.84%
VSS	8.638	5.902	9.112	9.017	6.295	9.164
VSS %	13.26%	12.15%	23.62%	16.71%	14.24%	15.12%

Table III. Computational results size 10-20-1

Instance	**1**	**2**	**3**	**4**	**5**	**6**
Iterations	915	1093	833	1022	902	768
Runtime (sec)	3600	3600	3600	3600	3600	3600
NDC Solution	239.325	220.463	270.933	241.905	251.947	243.543
Global solution	235.650	216.925	262.037	237.581	244.048	234.646
Gap	1.56%	1.63%	3.39%	1.82%	3.24%	3.79%
VSS	28.131	21.061	34.639	25.826	15.395	14.291
VSS %	11.94%	9.71%	13.22%	10.87%	6.31%	6.09%

is reported as the *Global solution* while the objective value corresponding to the problem with private information restrictions is reported as the *NDC solution*. Solution values have been scaled down by a factor of 10^6 for convenience.

Rows one and two of the Tables II - IV give the number of iterations of the NDC algorithm performed and the runtime of the algorithm, respectively. For all instances, a time limit of one hour was imposed and all instances ran for the full duration. Preliminary computational results had the initial values of ξ and λ (chosen in step 0 of the NDC algorithm) set higher and some of the smaller instances were terminating in less than an hour. However, it was found that the solution quality was improved when these initial values were set lower and the algorithm was allowed to run the full hour. The results reported here reflect

Table IV. Computational results size 10-15-10-1

Instance	**1**	**2**	**3**	**4**	**5**	**6**
Iterations	987	1023	1016	1038	1047	1026
Runtime (sec)	3600	3600	3600	3600	3600	3600
NDC Solution	1883.568	1725.142	1622.519	1720.257	1743.123	1833.221
Global solution	2162.723	1985.647	1834.955	1968.002	2054.538	2144.210
Gap	10.33%	9.36%	4.14%	5.67%	3.97%	5.21%
VSS	279.155	260.505	212.436	247.745	311.415	310.989
VSS %	14.82%	15.10%	13.09%	14.40%	17.87%	16.96%

that approach.

Information describing the output from solving the problems is given in rows three through five of each table. Table II gives the information for the instances of size 10-1. The results show that the gap between the NDC algorithm's solution and the global solution averages 1.16%. It is important to emphasize that this gap would not be computable in practice because the private information requirements make formulation of the global problem impossible. For testing purposes, these constraints were relaxed in order to provide evidence supporting NDC's ability to find good solutions. For the instances of size 10-20-1, the average solution gaps from the global solution were found to be 2.57% and for size 10-15-10-1, the average was 6.45%. Since protecting private information while seeking close-to-optimal solutions was the goal of this study, these gaps were deemed reasonable.

Rows six and seven give the VSS calculations as described in the previous subsection. For the instances of size 10-1, the VSS averaged 10.85% over the (determinstic) EV problem, for size 10-20-1 this gap was 9.69%, and for the largest size, the average VSS gap was 15.37%. These values give two insights. First, they give a clear reason for including probability distributions to model uncertain parameters in optimization models: if a decision maker uses absolute values to describe the problem data, he can expect his decisions to result in an objective value that is at least 10 - 15% higher than optimal. This phenomenon is expected in these types of problems, as the solution to a stochastic integer program generally gives a more robust solution than its corresponding EV problem. The second insight it provides supports the use of NDC with random data over the current standard practice. In practice, solving a nodal decision problem such as SCICP would begin by finding the expected values of the demand and then relaxing, at least partially, the private information restrictions. Depending on the solution strategy employed, a solution would be generated whose objective would be at least as far from optimality as the VSS difference. The NDC algorithm allows a decision maker to formulate a more realistic model of the system by the inclusion of random parameters and beat the optimality gap reported by VSS solution, in most cases by a wide margin.

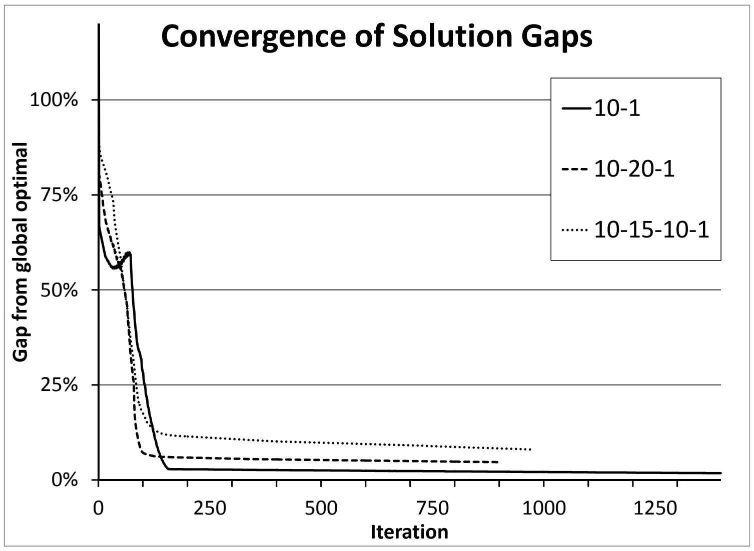

Fig. 10. Scaled convergence plot of three instances

Some insight into the behavior of the algorithm can also be gained by viewing a plot showing the convergence of the algorithm. As traditional upper and lower bounds are not readily available in the NDC algorithm, Figure 10 instead displays the difference between the upper and lower bounds at each iteration (which were calculated as $\hat{\pi}$ in Step 4 of the algorithm.) The gaps have been converted into percentages from their respective global optimal solution for comparison purposes and are plotted against the iteration number. Each trendline represents the first instance given in Tables II-IV.

An initial version of the NDC algorithm was implemented without the regularization term found in the coordinated subproblem. In contrast to the fairly smooth convergence rate seen in Figure 10, the solution gap suffered from solution oscillations that resulted in a graph that showed gaps that spent hundreds of iterations oscillating between two values for the gap. This phenomenon was particularly bad for the largest instances of size 10-15-10-1. In this version of NDC, note that the solutions start very far from the optimal solution, but in less than a quarter of the iterations performed reach a point where the gaps slowly converge toward the optimal global solution.

H. Conclusion

This paper presents a nodal decomposition-coordination method for stochastic programs where the decisions are distributed among multiple stakeholders, all of whom have restrictions on the data known to them about the other stakeholders. Further, no omnipotent entity with access all information on the system exists. Despite these private information restrictions, optimal or close to optimal decisions are required. The algorithm presented here exploits the properties of Lagrangian relaxation and subgradient optimization to allow for each stakeholder to solve its own subproblems and coordinate with other stakeholders without violating privacy restrictions on the problem data. A computational study on a supply chain inventory coordination problem shows that the proposed methodology can obtain solution values that are within 10% of the optimal solution without violating private information restrictions. In addition, the results reveal that stochastic solutions outperform the corresponding expected value solutions.

Solving stochastic integer programs is a very hard prospect, and most algorithms are very dependent on the structure of the problem to guarantee tractability and convergence. For this work, the stochastic data was used to formulate the deterministic equivalent problem and then solved using an integer programming solver directly. The number of random variable realizations for each time period was limited in order to make the subproblems solvable within a reasonable amount of time. Thus extensions to this work include the development and implementation of decomposition algorithms for solving the nodal stochastic integer programs. Finally, this work is applicable to other important applications in addition to supply chain logistics such as homeland security applications, which often carry with them inherent security constraints that are often difficult to resolve. Other applications might include transportation of goods over long distances which includes loading and offloading to different modes of transportation along the way which are run by different firms.

CHAPTER IV

SCENARIO FENCHEL DECOMPOSITION

A. Introduction

In two-stage stochastic integer programming (SIP) a decision must be made here-and-now (first-stage) before future uncertainty is realized. Future uncertainty is modeled using a probability distribution of the random variables. A recourse decision (second-stage) is made after the uncertainty is resolved. A two-stage SIP formulation can be given as follows:

$$\begin{aligned} \text{SIP2: } \text{Min } & c^\top x + \mathbb{E}[f(\tilde{\omega}, x)] \\ \text{s.t. } & Ax \geq b \\ & x \in X. \end{aligned} \tag{4.1}$$

In problem SIP2, x denotes the first-stage decision vector, $c \in \Re^{n_1}$ is the first-stage cost vector, $b \in \Re^{m_1}$ is the first-stage righthand side, $f(\tilde{\omega}, x)$ is the recourse function with $\tilde{\omega}$ being a multivariate random variable, and $\mathbb{E}$ denotes the mathematical expectation operator satisfying $\mathbb{E}[|f(\tilde{\omega}, x)|] < \infty$ for all $x \in \{Ax \geq b, x \in X\}$. $A \in \Re^{m_1 \times n_1}$ is the first-stage constraint matrix, and X defines binary restrictions on some components of x.

In this work, it is assumed that the underlying probability distribution of $\tilde{\omega}$ is discrete with a finite number of realizations (scenarios) k and corresponding probabilities $p^k,\ k = 1, 2, \ldots, K$. Thus for a given scenario k, the recourse function $f(k, x)$ is given by the following second-stage mixed-integer program (MIP):

$$\begin{aligned} f(k, x) = \text{Min } & q^{k\top} y^k \\ \text{s.t. } & W^k y^k \geq h^k - T^k x \\ & y^k \in Y. \end{aligned} \tag{4.2}$$

In formulation (4.2), $y^k \in \Re^{n_2}$ is the recourse decision variable vector, $q^k \in \Re^{n_2}$ is the recourse cost vector, $W^k \in \Re^{m_2 \times n_2}$ is the recourse matrix, $T^k \in \Re^{m_2 \times n_1}$ is the technology matrix, and $h^k \in \Re^{m_2}$ is the righthand side. The set Y defines binary restrictions on some components of y^k. Now SIP2 can also be stated as the following deterministic equivalent problem (DEP), which will be used in the scenario decomposition:

$$\begin{aligned} \text{DEP: Min } & c^\top x + \sum_{k=1}^{K} p^k q^{k\top} y^k \\ \text{s.t. } & Ax \geq b \\ & T^k x + W^k y^k \geq h^k, \; k = 1, 2, \ldots, K \\ & x \in X, \; y^k \in Y, \qquad k = 1, 2, \ldots, K. \end{aligned} \tag{4.3}$$

DEP is a large-scale MIP, which makes a decomposition approach a necessity for most practical sized problems. Decomposition approaches for SIP2 traditionally fall under one of two categories: *stage-wise* decomposition or *scenario-wise* decomposition. Stage-wise decomposition strategies are usually based on Benders decomposition (Benders, 1962) or L-shaped decomposition (Slyke and Wets, 1969). This paper relies on the scenario-wise decomposition strategy for SIP2 that uses Fenchel cutting planes Boyd (1994a). Solving a scenario decomposed SIP2 is difficult because traditional methods require solving the subproblem MIPs several times. In addition, the existence of a duality gap introduced by the decomposition due to dualizing some of the constraints, makes optimality hard to guarantee outside of a branch-and-bound scheme.

The contribution of this work is a new solution methodology for SIP that decomposes SIP2 scenario-wise via Lagrangian relaxation, uses progressive hedging Rockafellar and Wets (1991) to solve the LP relaxation, and Fenchel cutting planes to solve the scenario subproblems. We refer to this methodology as *scenario Fenchel decomposition* or simply, SFD. Because Fenchel cutting planes are generally expensive to compute, this approach is suitable for SIP with special structure such as knapsack constraints, which can be exploited by the Fenchel cutting plane approach. The SFD algorithm is derived and tested on SIP2 with multiple knapsack constraints. Knapsack constraints appear in many applications of SIP such as investment planning (Carraway et al., 1993; Henig, 1990), transportation and scheduling (Kleywegt and Papastavrou, 1998, 2001) and chance constrained SIP (Claro and Sousa, 2010).

The rest of the paper is organized as follows: Section B describes related work on solutions strategies for SIP2 and a review of the literature involving Fenchel cuts. Section C reviews preliminaries on scenario decomposition while Section D lays out the framework for generating Fenchel cuts for SIP2. The SFD algorithm is formally described in Section E and preliminary computational results reported in Section F. Finally, a summary of the paper is given in Section G.

B. Related Work

This section begins by discussing briefly scenario-wise decomposition approaches and follows it up with a summary of related work involving Fenchel cutting planes. Scenario-wise decomposition strategies for SIPs use variable splitting procedures on the first-stage decision variables to create *nonanticipativity* constraints, and then relax them using Lagrangian relaxation. As described previously, two-stage SIPs describe two sets of decisions and it is important that the first-stage solution be the same for all scenarios. Nonanticipativity constraints are a way of enforcing the first-stage solution to be the same for all the scenarios before the realization of the random variables in the second-stage. Nonanticipativity constraints are then relaxed using Lagrangian relaxation and algorithms are developed which seek to find a solution to the Lagrangian dual.

Caroe and Schultz (1999) proposed dual (scenario) decomposition for SIPs. In their approach, the Lagrangian multipliers associated with the relaxed nonanticipativity constraints are updated from one iteration to the next by a subgradient optimization approach. While several approaches are valid, the bundle method of Kiwiel (1990) is a popular choice in such a framework. While solving stochastic linear programs (SLPs) using the Lagrangian dual was a well-known strategy for SLPs before Caroe's work, it is important to note that since the dual decomposition algorithm relies on the Lagrangian dual, it may not be possible to find the (integer) optimal solution to SIP2 because of the existence of the duality gap for the Lagrangian dual of an IP (Nemhauser and Wolsey, 1999). For this reason, Carøe implements a branch-and-bound procedure within his algorithm to force finite convergence to the integer optimal. Another formal treatment of the dual decomposition algorithm can be found in Louveaux and Schultz (2003).

Rockafellar and Wets (1991) develop the progressive hedging algorithm (PHA) for scenario decomposed SLPs. In each iteration, the algorithm sets a single target nonanticipative value based on the previous iteration's solution. While the convergence of PHA subproblems to a nonanticipative first-stage solution is not based on Lagrangian duality, the form of the subproblem is similar to the dual decomposition subproblem and only differs by the inclusion of a quadratic term penalizing moving too far from the nonanticipative value from one iteration to the next. The PHA algorithm has been successfully applied to SLPs for different applications such as operation planning problems in hydrothermal power systems (Santos et al., 2009).

A number of papers exist which use PHA to find good solutions for SIPs. Haugen et al. (2001) apply PHA to the SIPs for the lot sizing problem. The authors exploit the quick convergence of PHA to a (possibly nonoptimal) integer solution, and then fix those integer values in the DEP and use LP techniques to find their final solution. The authors conclude that their algorithm is able to find good solutions to their test instances quickly, and note that solving their PHA subproblems exactly only adds to the computational effort and does not improve solution quality.

Watson et al. (2010) use a variant of PHA to find good solutions for two-stage stochastic programs with chance constraints. The authors perform two relaxations, first on the chance constraints, and then on the nonanticipativity constraints. The authors present their algorithm as a heuristic and conclude that their algorithm finds acceptable solutions to some network flow problems and aircraft sustainability planning problems. Watson and Woodruff (2010) develop a number of innovations to PHA for the stochastic resource allocation problem. The authors admit the inability to reach optimality for problems with discrete variables, so instead present their improvements as a heuristic for finding good solutions. The paper addresses how to choose algorithmic parameters that tend to perform well in PHA and suggest methods for improving convergence and detecting cycling.

Escudero et al. (2011) apply a Lagrangian dual approach to the stochastic set packing problem (SSPP) with randomly distributed objective coefficients. The form of SSPP that the authors solve is a knapsack constrained problem with a risk measure minimizing the risk of any scenario performing under some specified threshold. The authors develop an algorithm that returns good feasible solutions with smaller gaps than can be achieved by solving using a brute force approach.

Other general knapsack constrained stochastic programs have received attention in the literature. Carraway et al. (1993) develop an algorithm for solving a single stage knapsack problem in which the returns are described by normal random variables. Their approach uses dynamic programming and branch-and-bound to guarantee optimality, and computational experiments are shown to outperform other approaches. Kleywegt and Papastavrou (1998) introduce the dynamic and stochastic knapsack problem (DSKP.) DSKP is a multistage stochastic problem in which items arrive into the system according to a Poisson arrival process. A decision is made when an item arrives to be accepted or rejected into the knapsack and rejecting an item incurs a penalty. Several forms of the problem are presented and properties of the problems are derived for the case where the weights are all identical. In Kleywegt and Papastavrou (2001) the authors relax the identical weight assumption and develop properties regarding the structure of the problem.

Henig (1990) study an investment problem where a limited quantity of funds is to be invested and the objective is to maximize the chance that a target value is achieved. The authors present a solution approach using dynamic programming for problems with normally distributed benefits. Extensions to mean risk and mean variance measures are also discussed. A multiobjective metaheuristic for solving mean-risk stochastic knapsack problems is presented in Claro and Sousa (2010). The authors suggest the multiobjective approach in order to relieve difficulties associated with the incorporation of mean risk objectives into discrete optimization problems. Computational experiments are reported that show that their metaheuristic produces good solutions.

This paper develops a new PHA approach combined with Fenchel cutting planes. Fenchel cutting planes are a class of deep cutting planes derived using Fenchel duality in convexity theory (Rockafellar, 1997) that take advantage of the maximum separation/minimum distance duality. Fenchel cuts were first suggested in Boyd (1994a), and a number of characteristics are derived in Boyd (1994b), Boyd (1995). The most important results from Boyd's work are that Fenchel cutting planes are facet defining and that the use of Fenchel cuts in a cutting plane approach yields an algorithm with finite convergence. He also highlights the fact that generating a Fenchel cut for binary programs is computationally expensive in general; therefore, problems with special structure are desirable to achieve fast convergence. Computational experiments demonstrating the effectiveness of Fenchel cuts are presented in Boyd (1993) for knapsack polyhedra and in Boyd (1994b) for pure binary problems.

Since Boyd's pioneering work, only a few authors have adopted Fenchel cutting planes in their work. In Sáez (2000), the author uses Fenchel cutting planes to improve the bounds obtained from Lagrangian relaxation solutions to MIPs. The author also shows that Fenchel cutting planes always dominate Lagrangian cuts. More recently, Ramos and Sáez (2005) use Fenchel cutting planes to solve deterministic capacitated facility location problems. The authors again compare the use of Fenchel cuts to Lagrangian cuts in finding good relaxation bounds for their problem.

Ntaimo (2011) adapt the cuts in Boyd (1993) for two-stage SIPs under a stage-wise decomposition setting. The author derives two forms of the cuts: one on the (x, y) variable space, and another that derives the cuts in the y space and then lifts them to the (x, y) variable space. Computational experiments are run using both forms of the cuts in a Benders decomposition framework, and the results suggest that Fenchel cuts outperform the disjunctive decomposition algorithm (Sen and Higle, 2005) for some large-scale instances.

C. Preliminaries

This paper merges the decomposition and solution approach of PHA with a cutting plane approach using Fenchel cutting planes. First, SIP2 is decomposed into the necessary scenario decomposed subproblems for use in a PHA framework. First, the x variables are split once for each scenario. This introduces K new vectors of variables x^k, $k = 1, 2, \ldots, K$ to the problem. For the first-stage constraints $Ax^k \geq b$, the constraint set is duplicated K times, once for each new x variable. The substitution of x^k for x is straightforward in the second-stage constraints. In the objective function, adding the copies x^k, $k = 1, 2, \ldots, K$ can be thought of as moving the x variables to the second-stage, where the probability measure acts on the objective coefficients in the same way that it does the coefficients of the second-stage variables.

The other step is adding nonanticipativity constraints to the problem. After creating the duplicate x^k, $k = 1, 2, \ldots, K$ variables to SIP2, it becomes apparent that an optimal solution to the new problem is only feasible to SIP2 if all variables x^k, $k = 1, 2, \ldots, K$ are equal. As described in Caroe (1998), there are several valid choices in how to achieve nonanticipativity. Some choices for nonanticipativity constraints for SIP2 (4.3) are:

$$\begin{aligned} & x^k - x^{k+1} = 0, \qquad k = 1, \ldots, K-1 \\ & x^K - x^1 = 0 \end{aligned} \tag{4.4a}$$

$$x^k - \sum_{s=1}^{K} p^s x^s = 0, \quad k = 1, \ldots, K \tag{4.4b}$$

$$\begin{aligned} x^k - x^s = 0, \qquad & k = 1, \ldots, K; \\ & s \in \{1, 2, \ldots, K\}. \end{aligned} \tag{4.4c}$$

The nonanticipativity constraints given in (4.4a) represent a cyclic representation of the constraints, (4.4b) represents nonanticipativity in the expectation, and (4.4c) achieves nonanticipativity by choosing a single subproblem and enforcing its solution across all subproblems. The scenario decomposed problem can be stated as follows:

$$\begin{aligned} \text{DEP2: Min } & \sum_{k=1}^{K} p^k \left(c^\top x^k + q^{k\top} y^k \right) \\ \text{s.t. } & Ax^k \geq b, && k = 1, 2, \ldots, K \\ & T^k x^k + W^k y^k \geq h^k, && k = 1, 2, \ldots, K \\ & x^k - \sum_{s=1}^{K} p^s x^s = 0, && k = 1, 2, \ldots, K \\ & x^k \in X, \ y^k \in Y, && k = 1, 2, \ldots, K. \end{aligned} \tag{4.5}$$

Performing Lagrangian relaxation on the nonanticipativity constraints in DEP2 yields a problem whose constraint set is separable into K subproblems. For purposes of generating Fenchel cutting planes, it is useful to define the feasible region of each MIP subproblem separately, as follows:

$$Z^k := \{x^k, y^k | \quad Ax^k \geq b \\ T^k x^k + W^k y^k \geq h^k \\ x^k \in X,\ y^k \in Y\}. \tag{4.6}$$

Performing the variable substitution $\bar{x}_t = \sum_{s=1}^{K} p^s x^s$ and using the notation for Z^k from (4.6) gives the following Lagrangian relaxation subproblem, for $k = 1, \ldots, K$:

$$\begin{aligned} D^k(\lambda) : \text{Min}\, & p^k[c^\top x^k + q^{k\top} y^k + \lambda^k \left(x^k - \bar{x}_t\right)] \\ \text{s.t. } & x^k,\ y^k \in Z^k. \end{aligned} \tag{4.7}$$

The Lagrangian relaxation of DEP2 can then be expressed as

$$D(\lambda) = \sum_{k=1}^{K} D^k(\lambda) \tag{4.8}$$

and the Lagrangian dual with respect to the nonanticipativity constraints is then given as:

$$\underset{\lambda \text{ free}}{\text{Max}}\ D(\lambda). \tag{4.9}$$

As mentioned previously, the PHA subproblem differs from the dual decomposition subproblem by the inclusion of a quadratic term in the objective that hedges against choosing a solution that is too far from the current incumbent solution. At iteration t, let $\bar{x}_t$ denote the current incumbent solution. The PHA subproblem at iteration t is formally given as follows:

$$\begin{aligned} \text{P}^k(\lambda^k) : \text{Min}\, & p^k[c^\top x^k + q^{k\top} y^k + \lambda^k \left(x^k - \bar{x}_t\right) + \frac{\rho^k}{2}\left(x^k - \bar{x}_t\right)^2] \\ \text{s.t. } & x^k, y^k \in Z^k \end{aligned} \tag{4.10}$$

where ρ^k is a parameter chosen to keep x^k close to $\bar{x}_t$ from one iteration to the next. In our approach, the LP relaxations of $P^k(\lambda^k)$ will be solved by PHA. Let the sets X_{LP} and Y_{LP}

denote X and Y, respectively, with the integer requirements relaxed. The LP-relaxation of the PHA subproblem feasible region $Z^k(\lambda^k)$ yields the following feasible region:

$$\begin{aligned} Z^k_{LP} := \{x^k, y^k | \quad & Ax^k \geq b \\ & T^k x^k + W^k y^k \geq h^k \\ & x^k \in X_{LP},\ y^k \in Y_{LP}\} \end{aligned} \tag{4.11}$$

and the PHA subproblem can be expressed as

$$\begin{aligned} \mathrm{P}^k_{LP}(\lambda^k) = \text{Min}\ & p^k[c^\top x^k + q^{k\top} y^k + \lambda^k \left(x^k - \bar{x}_t\right) + \frac{\rho^k}{2}\left(x^k - \bar{x}_t\right)^2] \\ \text{s.t.}\ & x^k,\ y^k \in Z^k_{LP}. \end{aligned} \tag{4.12}$$

PHA can be used to solve a scenario decomposed SLP that has been reformulated as in (4.12). The algorithm is a proximal point method (Rockafellar and Wets, 1991) that computes a new target point at each iteration based on the first-stage solutions of all the subproblems. PHA is formally stated in Figure 11.

PHA Algorithm

Step 1: Choose an initial $\bar{x}_0$, step sizes ρ_0^k and multipliers λ_0^k and $\epsilon > 0$. Set PHA iteration counter $t = 0$.

Step 2: Solve $\mathrm{P}_{LP}^k(\lambda^k)$ $\forall k$ (4.12). Let the solutions be $(\hat{x}_t^k, \hat{y}_t^k, \hat{\lambda}_t^k)$. Update $\bar{x}_{t+1} = \sum_{k=1}^{K} p^k \hat{x}_t^k$.

Step 3: Update $\lambda_{t+1} = \lambda_t + \rho^k(\hat{x}_t^k - \bar{x}_{t+1})$. If $\sum_{k=1}^{K} p^k ||\hat{x}_t^k - \bar{x}_t|| < \epsilon$, $(\hat{x}_t^k, \hat{y}_t^k, \hat{\lambda}_t^k)$ is optimal for $P_{LP}^k(\lambda^k)$; Report optimal solution variables $\hat{x}_t^k$ and $\hat{y}_t^k$ and corresponding objective $\sum_{k=1}^{K} P_{LP}^k(\lambda^k)$. Otherwise increment t. Go to Step 2.

Fig. 11. The PHA Algorithm

One limitation of PHA is that convergence is only guaranteed for SLP. This paper seeks to use the convergence results of PHA for SIP2 by incorporating Fenchel cutting planes into the algorithm. In the following section, Fenchel cuts are developed for (4.12) with the goal of recovering the convex hull of scenario problem (4.10) in the neighborhood of the optimal solution.

D. Scenario Fenchel Decomposition

Scenario Fenchel decomposition (SFD) combines the PHA algorithm with a cutting plane method exploiting Fenchel cuts. Fenchel cutting planes are chosen because they are capable of recovering faces of the convex hull of binary programs, which is the structure inherent in SIP2. The goal is to construct the convex hull of integer points in the neighborhood of the optimal solution so that by solving (4.12) with enough Fenchel cuts added, the optimal solution can be found without having to use branch-and-bound to guarantee optimality. To begin, the foundations for Fenchel cutting planes on scenario decomposed subproblems are developed followed by a description of how the cuts can be derived.

Finding an integer solution to SIP2 via PHA is a hard prospect since each of the K subproblems are IPs and therefore the existence of a duality gap from dualizing the nonanticipativity constraints in general requires implementing a branch-and-bound scheme to guarantee optimality. Instead of working with the IP subproblems directly, the method described here seeks the optimal solution by a cutting plane approach on the LP relaxations of the subproblems. Cutting plane approaches are useful for solving IPs since the optimal solution to an LP over the convex hull of an IP coincides with the optimal solution to the IP. To achieve this, consider using Fenchel cutting planes to recover (at least partially) the convex hull of integer points of $\mathrm{P}^k(\lambda)$. Fenchel cutting planes, which are shown in Boyd (1993) to be deep cuts, will be used to recover integer solutions to the IP subproblems without ever solving them as IPs. The Fenchel cuts described below are derived for the feasible regions Z^k_{LP} (4.11).

The SFD algorithm derived here will employ the PHA algorithm in order to find a nonanticipative solution for the K subproblems $\mathrm{P}^k_{LP}(\lambda^k)$ and Fenchel cuts will be used to separate non-integer solutions from Z^k_{LP}. Starting from the LP relaxations of the scenario subproblems $\mathrm{P}^k_{LP}(\lambda^k)$, Fenchel cuts will now be given for each scenario subproblem with the goal of recovering the convex hull of feasible integer points in the neighborhood of the optimal solution to $\mathrm{P}^k(\lambda^k)$. The Fenchel cuts presented here for $\mathrm{P}^k_{LP}(\lambda^k)$ are adapted from Ntaimo (2011), where they are used to solve stochastic programs under stage-wise decomposition. Fenchel cuts are adapted here for scenario subproblems.

Let $(\hat{x}^k, \hat{y}^k, \hat{\lambda}^k)$ be a solution to $\mathrm{P}^k_{LP}(\lambda^k)$ and let $CONV(\cdot)$ denote the convex hull of feasible integer points for an IP. A Fenchel inequality is constructed in such a way as to find a hyperplane that separates the non-integer point $(\hat{x}^k, \hat{y}^k)$ to a hyperplane passing through a point in $CONV(\mathrm{P}^k(\lambda^k))$ without cutting off any feasible integer points in $CONV(\mathrm{P}^k(\lambda^k))$. The following theorem describes the criteria which guarantees the existence of such a Fenchel inequality.

THEOREM IV.1. *Let $(\hat{x}^k, \hat{y}^k) \in Z^k_{LP}$ be given. Define $g(k, \alpha^k, \beta^k) = Max\ \{\alpha^{k\top} x^k + \beta^{k\top} y^k \mid (x^k, y^k) \in CONV(Z^k_{IP})\}$ and let $\delta(k, \alpha^k, \beta^k) = \alpha^{k\top}\hat{x}^k + \beta^{k\top}\hat{y}^k - g(k, \alpha^k, \beta^k)$. Then there exists vectors α^k and β^k for which $\delta(k, \alpha^k, \beta^k) > 0$ if and only if $(\hat{x}^k, \hat{y}^k) \notin CONV(Z^k_{IP})$.*

The proof is given in Boyd (1994a) and is omitted from this paper. The result of Theorem IV.1 is that given a solution $(\hat{x}^k, \hat{y}^k)$ to P^k_{LP}, if $\delta(k, \alpha^k, \beta^k) > 0$, then there exists a valid inequality that will separate $(\hat{x}^k, \hat{y}^k)$ from the integer-feasible region Z^k. The inequality derived in such a way is of the form $\alpha^{k\top} x^k + \beta^{k\top} y^k \leq g(k, \alpha^k, \beta^k)$ and is called a Fenchel inequality. When generating a Fenchel inequality, it is desirable to maximize the distance between $(\hat{x}^k, \hat{y}^k)$ and the hyperplane $\alpha^{k\top} x^k + \beta^{k\top} y^k \leq g(k, \alpha^k, \beta^k)$ without cutting off any integer points in $CONV(Z^k_{IP})$. This requires maximizing $\delta(k, \alpha^k, \beta^k)$. The following corollary is useful to ensure that this maximization is possible.

COROLLARY IV.2. *The function $\delta(k, \alpha^k, \beta^k)$ is piecewise linear and concave.*

Proof. Piecewise linearity is obvious as $\delta(k, \alpha^k, \beta^k)$ is a linear function of α^k and β^k. Concavity can be shown by applying the definition:

$\lambda\delta(k, \alpha^k_1, \beta^k_1) + (1-\lambda)\delta(k, \alpha^k_2, \beta^k_2) =$

$\lambda\alpha^k_1\hat{x}^k + \lambda\beta^k_1\hat{y}^k - \lambda g(k, \alpha^k_1, \beta^k_1) + (1-\lambda)\alpha^k_2\hat{x}^k + (1-\lambda)\beta^k_2\hat{y}^k - (1-\lambda)g(k, \alpha^k_2, \beta^k_2) =$

$(\lambda\alpha^k_1 + (1-\lambda)\alpha^k_2)\hat{x}^k + (\lambda\beta^k_1 + (1-\lambda)\beta^k_2)\hat{y}^k - \lambda g(k, \alpha^k_1, \beta^k_1) - (1-\lambda)g(k, \alpha^k_2, \beta^k_2) \leq$

$(\lambda\alpha^k_1 + (1-\lambda)\alpha^k_2)\hat{x}^k + (\lambda\beta^k_1 + (1-\lambda)\beta^k_2)\hat{y}^k - \lambda g(k, \lambda\alpha^k_1 + (1-\lambda)\alpha^k_2, \lambda\beta^k_1 + (1-\lambda)\beta^k_2)$ $\square$

While any (α^k, β^k) that gives a positive $\delta(k, \alpha^k, \beta^k)$ will provide a valid Fenchel inequality, finding such an (α^k, β^k) requires a search of the (α^k, β^k) space constrained to a convex set $\Pi^{\alpha,\beta}$. Maximizing the function $\delta(k, \alpha^k, \beta^k)$ provides such a search and returns the deepest cutting plane possible. This maximization provides a maximal Fenchel inequality, which is a Fenchel cutting plane (or Fenchel cut) and separates $(\hat{x}^k, \hat{y}^k)$ from Z^k_{LP}. To generate a Fenchel cut, a solution to the following optimization problem is required:

$$\delta^k = \max_{(\alpha^k,\beta^k)\in\Pi^{\alpha,\beta}} \left\{ {\alpha^k}^\top \hat{x}^k + {\beta^k}^\top \hat{y}^k - g(k, \alpha^k, \beta^k) \right\}, \tag{4.13}$$

where the maximization is done over a linearly defined domain $\Pi^{\alpha,\beta}$ and

$$g(k, \alpha^k, \beta^k) = \max_{(x^k,y^k)\in CONV(Z^k_{IP})} \left\{ {\alpha^k}^\top x^k + {\beta^k}^\top y^k \right\}. \tag{4.14}$$

Once found, the Fenchel cut separating the non-integer point $(\hat{x}^k, \hat{y}^k)$ from CONV(Z^k_{IP}) is:

$${\alpha^k}^\top x^k + {\beta^k}^\top y^k \leq g(k, \alpha^k, \beta^k) \tag{4.15}$$

Note that the cut (4.15) passes through a point in $CONV(Z^K_{IP})$ (found in 4.14), and is a facet of $CONV(S^k)$.

Solving (4.13) is not a trivial task. As a result of Corollary IV.2, it is suggested in Boyd (1994a) that generalized programming or a proximal ascent procedure such as subgradient optimization can be used to solve (4.13) and generate a Fenchel cut. For this work, a generalized programming method based on Benders decomposition is used. The method uses a master problem (given below) to construct a linear approximation of the subproblem space while the subproblem returns feasible integer points from Z^k_{IP} (4.6).

Fenchel Cut Generation Subroutine (FCG)

Step 0. Initialization.

Let $(\hat{x}^k, \hat{y}^k)$ $\forall k$ be given. Set $t = 0$, $\ell^0 = -\infty$, $u^0 = \infty$, and choose $\epsilon' > 0$ and $(\alpha_0^k, \beta_0^k) \in \Pi^{\alpha,\beta}$.

Step 1. Solve Subproblem and Compute Lower Bound.

Use (α_t^k, β_t^k) to form and solve subproblem (4.17) to get solution (x_t^k, y_t^k) and objective value $g(k, \alpha_t^k, \beta_t^k)$. Let $d_t = (\hat{x}^k - x_t^k)^\top \alpha_t^k + (\hat{y}^k - y_t^k)^\top \beta_t^k$. Set $\ell^{t+1} =$ Max $\{d_t,\ \ell^t\}$. If ℓ^{t+1} is updated, set incumbent solution $\delta^k = d_t$ and

$(\alpha^{k*}, \beta^{k*}, g(k, \alpha^{k*}, \beta^{k*})) = (\alpha_t^k, \beta_t^k, g(k, \alpha_t^k, \beta_t^k))$.

Step 2. Solve Master Problem.

Use $(\hat{x}^k, \hat{y}^k)$ and subproblem (4.17) solution (x_t^k, y_t^k) to form and add constraint (4.16b) to master program. Solve master program to get an optimal solution $(\theta^t, \alpha_t^k, \beta_t^k)$. Set $u^{k+1} =$ Min$\{\theta^t,\ u^t\}$. If $u^{t+1} - \ell^{t+1} \leq \epsilon'$, stop and declare incumbent solution ϵ'-optimal. Otherwise, set $t = t + 1$ and go to Step 1.

Fig. 12. The Fenchel Cut Generation Subroutine

$$\delta_\tau^k = \underset{\alpha^k, \beta^k \in \Pi^{\alpha,\beta}}{\text{Max}} \quad \theta \tag{4.16a}$$

$$\text{s.t.} \quad -\theta + (\hat{x}^k - x_t^k)^\top \alpha^k + (\hat{y}^k - y_t^k)^\top \beta^k \geq 0,\ t = 1, \cdots, \tau. \tag{4.16b}$$

Given an optimal solution $(\theta_t, \alpha_t^k, \beta_t^k)$ to (4.16) at iteration t, (x_t^k, y_t^k) is the optimal solution to the following subproblem:

$$g(k, \alpha_t^k, \beta_t^k) = \text{Max } \alpha_t^{k\top} x^k + \beta_t^{k\top} y^k \quad \text{s.t. } (x^k, y^k) \in CONV(Z_{IP}^k). \tag{4.17}$$

Adopting a Benders decomposition framework, a method for generating Fenchel cuts is stated in Figure 12.

E. Decomposition Algorithm

SFD Algorithm

Step 1: Initialization.

Set SFD iteration counter $\tau = 0$.

Step 2: Find nonanticipative solution.

Solve PHA, which returns solutions $(\hat{x}_\tau^k, \hat{y}_\tau^k)$ $\forall k$.

Step 3: Check Integrality.

If $(\hat{x}_\tau^k, \hat{y}_\tau^k) \in Z_{IP}$ $\forall k$, end. Report $(\hat{x}_\tau^k, \hat{y}_\tau^k)$, $k = 1, 2, \ldots, K$ as optimal.

Step 4: Add Fenchel Cuts.

For each k such that $(\hat{x}_\tau^k, \hat{y}_\tau^k) \notin \{0,1\}^{n_1+n_2}$: Use FCG to obtain δ_τ^k, α_τ^k and β_τ^k. If $\delta_\tau^k = 0$ $\forall k$, terminate; $(\hat{x}_\tau^k, \hat{y}_\tau^k) \in CONV(Z_{IP}^k)$. Otherwise for $k = 1, 2, \ldots, K$ if $\delta_\tau^k > 0$, add Fenchel cut $\alpha_\tau^{k\top} x^k + \beta_\tau^{k\top} y^k \leq g(k, \alpha_\tau^k, \beta_\tau^k)$ to $\text{P}_{LP}^k(\lambda)$, which separates $(\hat{x}_\tau^k, \hat{y}_\tau^k)$ from $CONV(Z_{IP}^k)$. Set $\tau = \tau + 1$ and go to Step 2.

Fig. 13. The Scenario Fenchel Decomposition Algorithm

The SFD algorithm is formally stated in Figure 13. The algorithm adds Fenchel cutting planes to $\mathrm{P}^k(\lambda^k)$ in order to recover the convex hull of integer points in the neighborhood of the optimal solution. This methodology allows the convergence properties of PHA for SLPs to apply to SIP2. The integer optimum can be found provided the convex hull of $\mathrm{P}^k(\lambda^k)$ (in the neighborhood of the optimal solution) can be found, which adding a sufficient number of Fenchel cuts will accomplish.

In order to prove that SFD is a tractable algorithm, it is useful to notice that the algorithm consists of two main parts: finding a nonanticipative first-stage solution to the LP relaxations using PHA, and generating a Fenchel cut using FCG. Assuming these two steps terminate finitely, SFD will also terminate finitely. Recall that a Fenchel cut for a scenario subproblem describes a facet of the convex hull of that subproblem. Since by definition the convex hull must have a finite number of facets, generating the facets one at a time will necessarily have a finite number of steps. The following lemma and theorem provide the necessary finite termination properties for FCG and PHA.

LEMMA IV.3. *Assume $Z^k \neq \emptyset$ and in Step 0 of FCG, $\Pi^{\alpha,\beta}$ is the unit sphere of an arbitrary norm. Then FCG terminates in a finite number of iterations.*

Lemma IV.3 follows from Theorem 3.2 in Boyd (1995) where the author shows that when $\Pi^{\alpha,\beta}$ is defined by the unit sphere of an arbitrary norm, only a finite number of Fenchel cuts describing the same face can be generated.

THEOREM IV.4. *PHA applied to a SLP terminates in a finite number of iterations with an optimal solution.*

Theorem (IV.4) is proven in numerous places in the literature (Birge and Louveaux (1997), Ruszczyński and Shapiro (2003), Rockafellar and Wets (1991)) and so the proof is omitted from this paper. Finite termination of SFD is guaranteed by Lemma IV.3 and Theorem IV.4. SFD is essentially an algorithm which applies PHA to the linear relaxations of a set of scenario decomposed subproblems. In each iteration, the feasible space of the subproblems contract toward the convex hull of optimal integer points to SIP2. Formal treatment of this claim is given in Theorem IV.5.

THEOREM IV.5. *SFD, applied to SIP2* (4.1)*, terminates with a solution* $(\hat{x}, \hat{y}) \in CONV(SIP2)$ *whose optimal objective value coincides with the optimal objective value to SIP2.*

Proof. Optimality of the subproblems $P^k_{LP}(\lambda^k)$ in each step of PHA is guaranteed by Theorem IV.4. Further, generating a Fenchel cut by FCG terminates in a finite number of steps by Lemma IV.3 so generating a Fenchel cut for each subproblem whose solution is non-integer can be done in a finite amount of time. Therefore, each iteration of SFD terminates finitely.

Since at each iteration in which an integer solution to a subproblem is not found by PHA a Fenchel cut describing a face of the subproblem is added, and since there are a finite number of faces to the convex hull of an IP, SFD will terminate in a finite number of iterations. Thus, the nonanticipative solution returned in the final iteration of PHA $(\hat{x}^k, \hat{y}^k) \in CONV(SIP2)$. Since IP theory states that the optimal objective value to an IP P coincides with the optimal objective of optimizing over the convex hull of P, equality of the objective returned by PHA to the optimal integer solution of SIP2 is also guaranteed. □

Theorem IV.5 guarantees that SFD will converge to a solution in $CONV$(SIP2) in a finite number of iterations. However, in practice the computational effort required to find the optimal solution may be too great to justify solving to optimality. In FCG, a Fenchel cut is generated on the (x, y) space based on the current $(\hat{x}, \hat{y})$ solution. It was observed that multiple iterations of SFD may be required before a significant change in $\bar{x}_t$ was seen. This situation occurs when a Fenchel cut generated in an iteration of SFD is deep in the

y-space but not in the x-space. Once $\bar{x}$ changes, some of these cuts become redundant. For this reason, limiting either the run time or number of iterations is recommended for large scale problems. In the computational study, presented below, a two hour run time limit was imposed and the optimality gap remaining at termination is reported.

F. Computational Results

This section reports computational results to demonstrate the ability of SFD to find solutions for several two-stage SIPs. SFD was implemented in C++ using the CPLEX 12.1 Callable Library (IBM ILOG CPLEX, 2009) and Microsoft Visual Studio 2010. In each iteration of SFD, the CPLEX Quadratic optimizer was used to solve PHA and the updating rules given in Step 2.3 of SFD were implemented as stated to update the PHA objective penalties. Initial PHA parameters ρ were chosen as suggested in Watson and Woodruff (2010), where the authors suggest using the c values from SIP2 (4.1). Computational experiments were run on a DELL Optiplex GX620 3.0 GHz computer with 3.5 GB of RAM. Benchmark results are compared with generating the DEP (4.3) and solving using the CPLEX MIP solver. The design of experiments is explained in the next subsection 1 followed by the computational results, which are reported in subsection 2.

1. Design of Experiments

Since generating a Fenchel cut requires solving an IP (usually multiple times), exploiting special problem structures can yield significant computational advantages over solving a general IP. For this reason, SFD was tested on randomly generated SIPs with pure binary decisions variables and multiple knapsack constraints in both the first- and second-stage. As noted in Ntaimo (2011), this choice of problems allows for the FCG subroutine procedure to be simplified in the following way. At the beginning of the FCG subroutine, SFD provides solution $(\hat{x}^k, \hat{y}^k)$. For each element of $(\hat{x}^k, \hat{y}^k)$ which is 0, the corresponding α^k or β^k can be set to 0 for the duration of the FCG.

The problems each have a 10 knapsack constraints in the first-stage and 20 knapsack constraints in the second-stage and each scenario has equal probability of occurrence. The instance data were randomly generated using the uniform distribution. Knapsack weights were generated by sampling from $uniform(2, 8)$. Objectives function values were generated similar to Watson and Woodruff (2010) in that the first-stage costs were chosen to be much higher than the second-stage costs. Cost functions for the first-stage variables were sampled from $uniform(400, 650)$. Second-stage costs were sampled from $uniform(6, 16)$. In order to generate tight knapsack constraints, the righthand side value for each constraint was generated by finding the maximum knapsack weight (W_{max}) for the constraint and sampling from $uniform(2 + 2W_{max}, 4W_{max})$.

Computational experiments were run on four test sets. Each test set had a constant number of variables and four subsets in which the number of scenarios was varied. Each of the subsets contain five randomly generated instances. The problem names follow a naming convention that describes the problem size. The first numeral of the name describes the number of first-stage variables, the second describes the number of second-stage variables, the third describes the number of scenarios.

2. Results

We now report results from computational experiments performed using the SFD algorithm. In Watson and Woodruff (2010), the authors note that the parameter ρ need not stay constant throughout the PHA procedure. Initial testing on the problem instances showed that reducing the quadratic term ρ whenever the objective improvement slowed offered computational advantages. For many instances, it was observed that a good first-stage scenario solution could be found that coincided with one large subset of the scenarios, and another solution was good for another subset of the scenarios. In each iteration of SFD, PHA can have difficulty moving too far from the target $\bar{x}_t$ solution when ρ is too large. This results in SFD performing non-improving iterations of PHA. Initial tests showed that reducing ρ whenever objective improvement slows alleviated this issue for the instances.

Table V. Results of test set 1

	CPLEX		**SFD**					
Instance	**Time**	**Gap %**	**Time**	**Gap %**	**FCG %**	**PHA %**	**Iters**	**# FCs**
knaps.10.20.25.a	7200.0	0.71	2568.4	0.00	18.50	81.50	71	1216
knaps.10.20.25.b	7200.0	1.05	3467.4	0.00	12.44	87.56	88	1194
knaps.10.20.25.c	7200.0	0.67	6091.3	0.00	12.52	87.48	87	1137
knaps.10.20.25.d	7200.0	0.71	3098.8	0.00	16.02	83.98	72	1329
knaps.10.20.25.e	7200.0	0.74	3728.0	0.00	11.64	88.36	61	1079
Average	**7200.0**	**0.78**	**3790.8**	**0.00**	**14.22**	**85.78**	**76**	**1191**
knaps.10.20.50.a	7200.0	2.30	7200.0	0.61	10.56	89.44	46	2261
knaps.10.20.50.b	7200.0	6.19	7200.0	0.05	15.97	84.03	57	2561
knaps.10.20.50.c	7200.0	3.76	7200.0	0.02	18.86	81.14	57	2471
knaps.10.20.50.d	7200.0	5.01	7200.0	0.03	16.48	83.52	53	2332
knaps.10.20.50.e	7200.0	6.26	7200.0	0.03	15.76	84.24	63	2628
Average	**7200.0**	**4.70**	**7200.0**	**0.15**	**15.53**	**84.47**	**55**	**2451**
knaps.10.20.75.a	7200.0	5.28	7200.0	0.31	23.12	76.88	44	3231
knaps.10.20.75.b	7200.0	5.76	7200.0	0.44	21.33	78.67	42	3131
knaps.10.20.75.c	7200.0	6.76	7200.0	0.64	22.25	77.75	45	3347
knaps.10.20.75.d	7200.0	5.67	7200.0	0.72	24.00	76.00	44	3266
knaps.10.20.75.e	7200.0	4.62	7200.0	0.58	20.84	79.16	42	3142
Average	**7200.0**	**5.62**	**7200.0**	**0.54**	**22.31**	**77.69**	**43**	**3223**
knaps.10.20.100.a	7200.0	5.94	7200.0	0.57	34.96	65.04	43	4290
knaps.10.20.100.b	7200.0	6.83	7200.0	1.19	39.98	60.02	45	4500
knaps.10.20.100.c	7200.0	6.80	7200.0	0.51	41.97	58.03	48	4781
knaps.10.20.100.d	7200.0	7.80	7200.0	0.58	40.60	59.40	43	4291
knaps.10.20.100.e	7200.0	8.05	7200.0	0.77	38.25	61.75	45	4499
Average	**7200.0**	**7.08**	**7200.0**	**0.72**	**39.15**	**60.85**	**45**	**4472**
knaps.10.20.150.a	7200.0	9.39	7200.0	2.30	50.65	49.35	40	5986
knaps.10.20.150.b	7200.0	6.00	7200.0	1.13	47.76	52.24	39	5804
knaps.10.20.150.c	7200.0	9.39	7200.0	3.44	53.62	46.38	42	6300
knaps.10.20.150.d	7200.0	11.25	7200.0	2.97	51.14	48.86	38	5695
knaps.10.20.150.e	7200.0	7.68	7200.0	1.90	50.10	49.90	41	6149
Average	**7200.0**	**8.74**	**7200.0**	**2.35**	**50.65**	**49.35**	**40**	**5987**
knaps.10.20.200.a	7200.0	9.31	7200.0	4.27	56.73	43.27	36	7185
knaps.10.20.200.b	7200.0	10.18	7200.0	4.10	44.15	55.85	33	6600
knaps.10.20.200.c	7200.0	8.19	7200.0	4.23	37.85	62.15	29	5800
knaps.10.20.200.d	7200.0	12.23	7200.0	4.91	45.07	54.93	31	6200
knaps.10.20.200.e	7200.0	8.44	7200.0	3.92	46.33	53.67	32	6400
Average	**7200.0**	**9.67**	**7200.0**	**4.28**	**46.02**	**53.98**	**32**	**6437**

Computation of the optimality gap for SFD required special attention. Since SFD is a Lagrangian relaxation technique SFD provides an upper bound solution at the end of every iteration. A lower bound on the optimal solution is not readily available except on the

Table VI. Results of test set 2

	CPLEX		**SFD**					
Instance	**Time**	**Gap %**	**Time**	**Gap %**	**FCG %**	**PHA %**	**Iters**	**# FCs**
knaps.20.30.25.a	7200.0	1.68	7200.0	0.07	33.08	66.92	92	2208
knaps.20.30.25.b	7200.0	1.40	7200.0	0.28	19.90	80.10	65	1615
knaps.20.30.25.c	7200.0	2.53	7200.0	0.38	30.12	69.88	78	1950
knaps.20.30.25.d	7200.0	1.72	7200.0	0.34	26.91	73.09	73	1825
knaps.20.30.25.e	7200.0	1.43	7200.0	0.02	26.34	73.66	81	1938
Average	**7200.0**	**1.75**	**7200.0**	**0.22**	**27.27**	**72.73**	**78**	**1907**
knaps.20.30.50.a	7200.0	4.17	7200.0	1.54	30.92	69.08	49	2450
knaps.20.30.50.b	7200.0	2.34	7200.0	1.03	27.23	72.77	47	2340
knaps.20.30.50.c	7200.0	3.57	7200.0	2.10	29.21	70.79	42	2100
knaps.20.30.50.d	7200.0	2.33	7200.0	1.23	24.87	75.13	41	2050
knaps.20.30.50.e	7200.0	3.15	7200.0	1.53	29.47	70.53	47	2349
Average	**7200.0**	**3.11**	**7200.0**	**1.49**	**28.34**	**71.66**	**45**	**2258**
knaps.20.30.75.a	7200.0	5.96	7200.0	2.12	37.47	62.53	37	2769
knaps.20.30.75.b	7200.0	5.20	7200.0	2.24	49.82	50.18	38	2840
knaps.20.30.75.c	7200.0	5.21	7200.0	2.64	32.97	67.03	35	2622
knaps.20.30.75.d	7200.0	5.66	7200.0	2.68	38.31	61.69	33	2475
knaps.20.30.75.e	7200.0	5.41	7200.0	2.51	29.70	70.30	38	2850
Average	**7200.0**	**5.49**	**7200.0**	**2.44**	**37.65**	**62.35**	**36**	**2711**
knaps.20.30.100.a	7200.0	5.95	7200.0	3.06	43.96	56.04	34	3400
knaps.20.30.100.b	7200.0	7.24	7200.0	3.66	53.22	46.78	32	3200
knaps.20.30.100.c	7200.0	6.85	7200.0	2.91	61.25	38.75	31	3100
knaps.20.30.100.d	7200.0	6.48	7200.0	2.90	41.81	58.19	32	3192
knaps.20.30.100.e	7200.0	6.96	7200.0	3.63	66.67	33.33	33	3300
Average	**7200.0**	**6.70**	**7200.0**	**3.23**	**53.38**	**46.62**	**32**	**3238**
knaps.20.30.150.a	7200.0	10.26	7200.0	5.77	61.03	38.97	25	3750
knaps.20.30.150.b	7200.0	7.94	7200.0	4.64	68.80	31.20	25	3750
knaps.20.30.150.c	7200.0	10.15	7200.0	5.54	68.97	31.03	29	4350
knaps.20.30.150.d	7200.0	9.22	7200.0	6.60	55.82	44.18	24	3600
knaps.20.30.150.e	7200.0	8.32	7200.0	5.37	57.57	42.43	27	4050
Average	**7200.0**	**9.18**	**7200.0**	**5.59**	**62.44**	**37.56**	**26**	**3900**
knaps.20.30.200.a	7200.0	10.98	7200.0	7.48	50.91	49.09	21	4200
knaps.20.30.200.b	7200.0	9.53	7200.0	7.33	48.28	51.72	22	4400
knaps.20.30.200.c	7200.0	11.56	7200.0	7.91	54.90	45.10	23	4600
knaps.20.30.200.d	7200.0	12.22	7200.0	5.11	56.47	43.53	26	5200
knaps.20.30.200.e	7200.0	10.18	7200.0	7.02	66.46	33.54	24	4800
Average	**7200.0**	**10.89**	**7200.0**	**6.97**	**55.40**	**44.60**	**23**	**4640**

rare occasion that SFD returns an integer solution (which would coincide with the optimal solution.) For this reason, a rounding heuristic was required to compute a lower bound for SFD whenever optimality was not achieved. This lower bound was found in all of the instances to be greater than or equal to the lower bound returned at termination of solving the DEP with CPLEX. The lower bound solutions can be found in Appendix A.

The tables are organized as follows: the time required to solve the DEP directly using the CPLEX MIP solver and optimality gap remaining are given in the first two columns and four columns are used to describe the solution time using SFD. Note that there are two main steps in the SFD algorithm. First, SFD finds a nonanticipative solution for the linear relaxations of the subproblems (with added Fenchel cuts), and then the algorithm generates a Fenchel cut for each subproblem that returns a non-integer solution. In the tables, the first of the SFD columns gives the total runtime for the SFD algorithm, the second gives the gap remaining at termination and the final two columns give the proportion of time spent generating Fenchel cuts and solving PHA iterations, respectively. The bolded rows in each each table give the average of the columns for the preceding five replications.

At termination, SFD was able to beat the CPLEX gap for all of the instances reported. From Table V, note that SFD was able to find the optimal solution to the 25 scenario subproblems in about an hour on average, while CPLEX was unable to solve any of the instances to optimality in the two hour time limit. Additionally, for the 50 scenario and 100 scenario problems, the optimality gap left by SFD was under 1% for all but one instance, which was significantly smaller than the CPLEX gaps of 4.7% and 7%, respectivly. The relationship between the CPLEX gap and the SFD gap continues throughout Tables VI - VIII, with each solution method seeing an increase in their gaps as the size of the problems was increased. The fact that SFD outperformed solving the DEP directly for all of the test instances is promising, as SFD uses only Fenchel cutting planes to seek integer solutions while the CPLEX solver has a host of cutting planes and computational efficiencies at its disposal.

The tests reported in Tables V - VIII provide insights into the performance of SFD as the number of scenarios increase. In Table V, SFD had no gap remaining for the 25 scenario problem, and a slight increase was seen when doubling the number of scenarios to 50, and again to 100. The change from the 100 scenario problems to the 200 scenario problems increased the gap by more than 5 times, which is interesting since the test sets in Tables VI - VIII seem to have a relatively linear relationship between the size of the

Table VII. Results of test set 3

	CPLEX		SFD					
Instance	**Time**	**Gap %**	**Time**	**Gap %**	**FCG %**	**PHA %**	**Iters**	**# FCs**
knaps.30.40.25.a	7200.0	2.33	7200.0	0.49	50.93	49.07	88	2191
knaps.30.40.25.b	7200.0	2.00	7200.0	0.44	41.65	58.35	89	2225
knaps.30.40.25.c	7200.0	2.13	7200.0	0.82	37.40	62.60	81	2017
knaps.30.40.25.d	7200.0	1.91	7200.0	0.58	49.06	50.94	91	2255
knaps.30.40.25.e	7200.0	1.75	7200.0	0.75	45.33	54.67	97	2409
Average	**7200.0**	**2.02**	**7200.0**	**0.62**	**44.87**	**55.13**	**89**	**2219**
knaps.30.40.50.a	7200.0	4.23	7200.0	3.42	37.44	62.56	42	2100
knaps.30.40.50.b	7200.0	2.15	7200.0	2.50	36.27	63.73	46	2300
knaps.30.40.50.c	7200.0	3.88	7200.0	2.66	40.05	59.95	39	1950
knaps.30.40.50.d	7200.0	3.33	7200.0	2.27	43.56	56.44	51	2550
knaps.30.40.50.e	7200.0	3.49	7200.0	2.90	48.76	51.24	45	2250
Average	**7200.0**	**3.42**	**7200.0**	**2.75**	**41.22**	**58.78**	**45**	**2230**
knaps.30.40.75.a	7200.0	5.42	7200.0	2.86	52.05	47.95	32	2400
knaps.30.40.75.b	7200.0	6.37	7200.0	3.15	52.85	47.15	36	2700
knaps.30.40.75.c	7200.0	6.14	7200.0	3.32	59.28	40.72	35	2625
knaps.30.40.75.d	7200.0	7.49	7200.0	4.51	50.10	49.90	34	2550
knaps.30.40.75.e	7200.0	6.39	7200.0	4.53	43.38	56.62	29	2175
Average	**7200.0**	**6.36**	**7200.0**	**3.67**	**51.53**	**48.47**	**33**	**2490**
knaps.30.40.100.a	7200.0	9.33	7200.0	5.60	74.18	25.82	28	2800
knaps.30.40.100.b	7200.0	6.79	7200.0	4.71	68.32	31.68	27	2700
knaps.30.40.100.c	7200.0	7.29	7200.0	4.10	70.84	29.16	23	2300
knaps.30.40.100.d	7200.0	8.03	7200.0	5.01	71.47	28.53	28	2800
knaps.30.40.100.e	7200.0	7.86	7200.0	5.32	55.79	44.21	25	2500
Average	**7200.0**	**7.86**	**7200.0**	**4.95**	**68.12**	**31.88**	**26**	**2620**
knaps.30.40.150.a	7200.0	9.35	7200.0	7.13	59.48	40.52	22	3300
knaps.30.40.150.b	7200.0	10.70	7200.0	7.62	71.75	28.25	24	3600
knaps.30.40.150.c	7200.0	10.03	7200.0	7.58	69.39	30.61	21	3150
knaps.30.40.150.d	7200.0	8.78	7200.0	6.47	67.07	32.93	22	3300
knaps.30.40.150.e	7200.0	7.58	7200.0	5.99	60.14	39.86	21	3150
Average	**7200.0**	**9.29**	**7200.0**	**6.96**	**65.57**	**34.43**	**22**	**3300**
knaps.30.40.200.a	7200.0	10.76	7200.0	7.25	49.38	50.62	20	4000
knaps.30.40.200.b	7200.0	13.87	7200.0	7.58	42.46	57.54	19	3800
knaps.30.40.200.c	7200.0	10.14	7200.0	7.08	47.45	52.55	18	3600
knaps.30.40.200.d	7200.0	9.47	7200.0	7.39	50.46	49.54	20	4000
knaps.30.40.200.e	7200.0	11.44	7200.0	7.61	46.08	53.92	18	3600
Average	**7200.0**	**11.14**	**7200.0**	**7.38**	**47.17**	**52.83**	**19**	**3800**

optimality gap and number of scenarios.

Another interesting trend exposed by increasing the number of scenarios is in the proportion of time spent generating Fenchel cuts versus the amount of time finding nonanticipative solutions to the LP relaxations. In general, SFD spent more time in the PHA section of the algorithm than in generating Fenchel cuts for instances with a small number of scenarios and the proportion evened out as the scenarios were increased. This is logical, since generating a FC requires solving an integer program several times, so when there are

Table VIII. Results of test set 4

	CPLEX		**SFD**					
Instance	**Time**	**Gap %**	**Time**	**Gap %**	**FCG %**	**PHA %**	**Iters**	**# FCs**
knaps.40.50.25.a	7200	2.75	7200	1.58	54.79	45.21	79	1975
knaps.40.50.25.b	7200	2.35	7200	1.40	55.95	44.05	87	2145
knaps.40.50.25.c	7200	1.46	7200	0.94	49.11	50.89	75	1805
knaps.40.50.25.d	7200	1.51	7200	0.80	56.08	43.92	91	2275
knaps.40.50.25.e	7200	0.82	7200	0.95	43.49	56.51	87	2129
Average	**7200**	**1.78**	**7200**	**1.13**	**51.88**	**48.12**	**84**	**2066**
knaps.40.50.50.a	7200	4.17	7200	4.18	43.05	56.95	40	2000
knaps.40.50.50.b	7200	3.92	7200	3.53	34.83	65.17	32	1600
knaps.40.50.50.c	7200	3.56	7200	2.62	52.37	47.63	46	2288
knaps.40.50.50.d	7200	3.54	7200	2.88	42.74	57.26	42	2100
knaps.40.50.50.e	7200	4.77	7200	3.89	46.95	53.05	38	1900
Average	**7200**	**3.99**	**7200.0**	**3.42**	**43.99**	**56.01**	**40**	**1978**
knaps.40.50.75.a	7200	7.07	7200	4.84	54.78	45.22	30	2250
knaps.40.50.75.b	7200	8.21	7200	6.26	39.68	60.32	27	2025
knaps.40.50.75.c	7200	7.88	7200	4.39	73.40	26.60	29	2175
knaps.40.50.75.d	7200	5.96	7200	4.08	49.91	50.09	29	2175
knaps.40.50.75.e	7200	7.07	7200	4.61	67.84	32.16	27	2025
Average	**7200**	**7.24**	**7200**	**4.84**	**57.12**	**42.88**	**28**	**2130**
knaps.40.50.100.a	7200	7.57	7200	5.57	71.55	28.45	24	2400
knaps.40.50.100.b	7200	7.67	7200	5.73	68.66	31.34	21	2100
knaps.40.50.100.c	7200	8.19	7200	6.18	67.42	32.58	25	2500
knaps.40.50.100.d	7200	7.33	7200	5.45	54.15	45.85	20	2000
knaps.40.50.100.e	7200	7.15	7200	5.13	68.02	31.98	24	2400
Average	**7200**	**7.58**	**7200**	**5.61**	**65.96**	**34.04**	**23**	**2280**
knaps.40.50.150.a	7200.0	9.44	7200.0	7.51	55.10	44.90	20	3000
knaps.40.50.150.b	7200.0	8.24	7200.0	6.96	67.78	32.22	16	2400
knaps.40.50.150.c	7200.0	9.74	7200.0	8.18	61.48	38.52	19	2850
knaps.40.50.150.d	7200.0	8.93	7200.0	6.98	67.91	32.09	18	2700
knaps.40.50.150.e	7200.0	10.41	7200.0	8.53	62.09	37.91	18	2700
Average	**7200.0**	**9.35**	**7200.0**	**7.63**	**62.87**	**37.13**	**18**	**2730**
knaps.40.50.200.a	7200	11.45	7200	8.30	29.67	70.33	15	3000
knaps.40.50.200.b	7200	9.96	7200	7.59	49.98	50.02	18	3600
knaps.40.50.200.c	7200	13.74	7200	7.46	56.06	43.94	17	3400
knaps.40.50.200.d	7200	11.54	7200	7.75	44.28	55.72	16	3200
knaps.40.50.200.e	7200	12.23	7200	7.58	52.78	47.22	14	2800
Average	**7200**	**11.78**	**7200**	**7.74**	**46.56**	**53.44**	**16**	**3200**

more (scenario) subproblems, more IPs must be solved. The exception to this trend was the last instance of test set 4 (Table VII) and all of test set 4 (Table VIII). For these instances, the proportion of time spent in PHA and FCG was relatively even. The reason for this seems to be the effect that increasing the number of variables has on the number of IPs that must be solved to generate a Fenchel cut on each scenario subproblem.

Another trend is revealed by considering the effect that increasing the number of variables has across problems instances with the same number of scenarios. Consider the

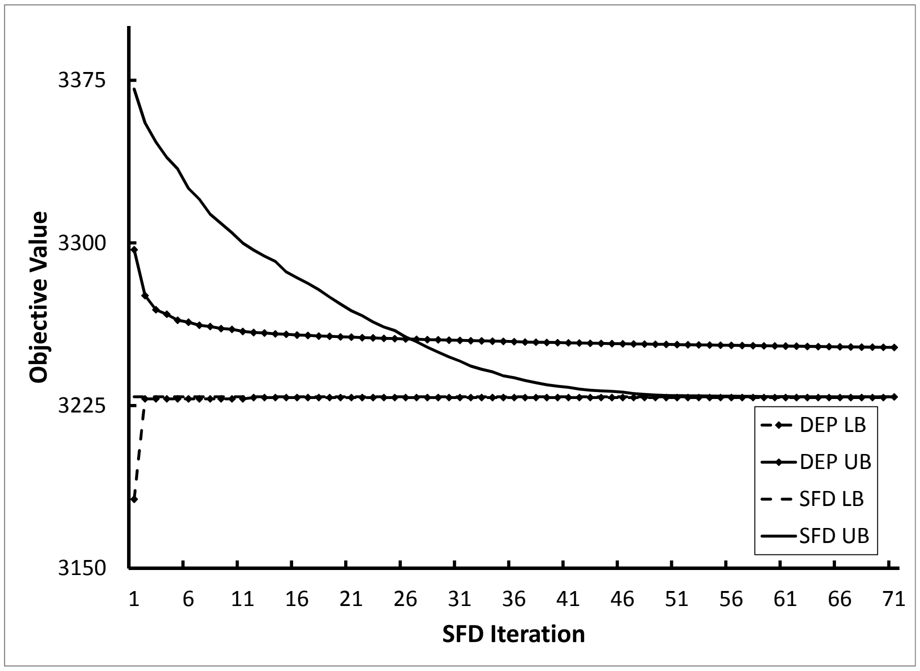

Fig. 14. Convergence plot for instance 10.20.25.a

output presented in Tables V - VIII for the problems with a set number of scenarios. On average, the gap in the SFD solution increases somewhat linearly as the problem size increases from one table to the next while the gaps returned by CPLEX from solving the DEP stay relatively constant as the number of variables increases. To gain insight into the rate of convergence of each solution method, consider the following graphs of the upper and lower bounds.

Figures 14, 15 and 16 display the upper and lower bounds for both solving the DEP directly using CPLEX (DEP UB and DEP LB) and using SFD (SFD UB and SFD LB). The

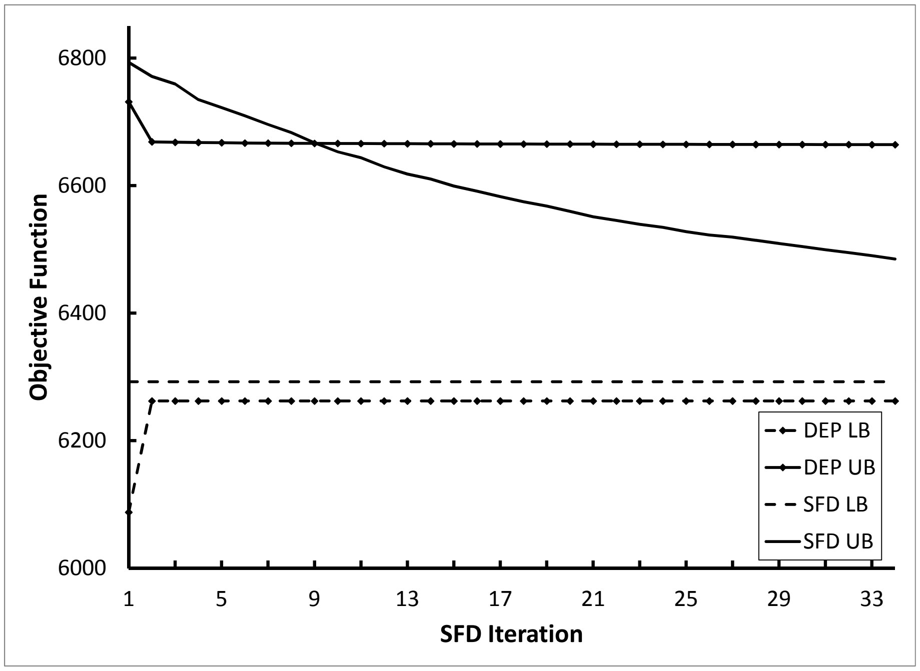

Fig. 15. Convergence plot for instance 20.30.100.a

figures use the iterations corresponding to SFD, and are not based on time. The CPLEX bounds corresponding to the SFD iterations were obtained by sampling the CPLEX output at regular intervals corresponding to the number of SFD iterations completed in the time limit. As seen in the figures, the CPLEX solver begins by adding several cuts to the LP relaxation problem, giving it a tighter upper bound in the first iteration, but then is slow to improve its upper bound after the first few iterations. The lower bounds from CPLEX have a similar trend: they find a good solution early in the branch-and-bound tree, but then are slow to improve upon that solution. By contrast, the upper bound for SFD shows a fairly linear convergence rate. As mentioned before, a lower bound for SFD was not available at every iteration, so the lower bound shown across all iterations is the result of applying the rounding heuristic on the solution returned in the final iteration of SFD. The graphs of convergence seem to suggest that SFD has a fairly reliable convergence rate toward the optimal solution, while solving the DEP can continue on for a very long time with no improvements to either the upper or lower bound.

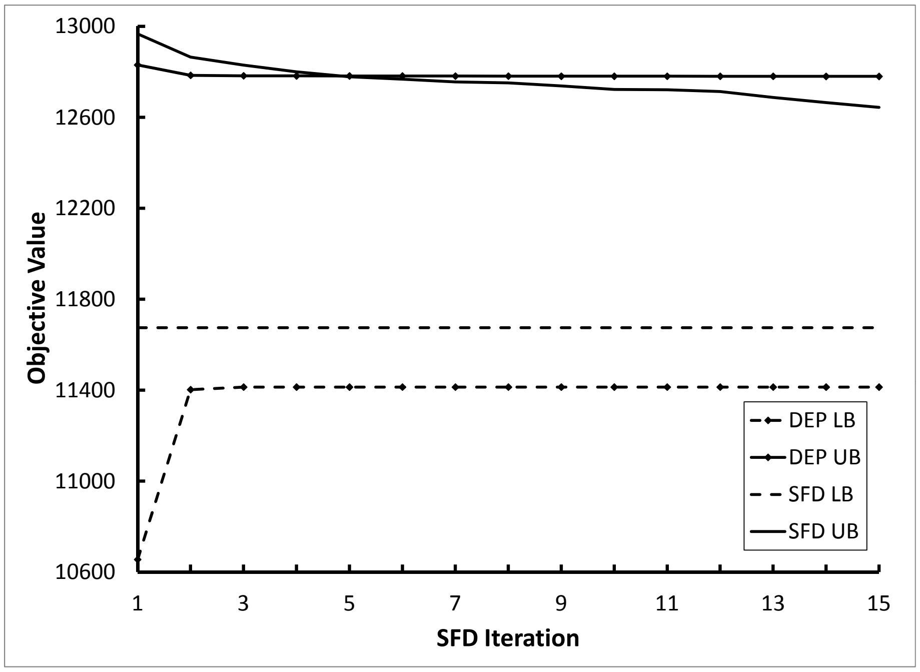

Fig. 16. Convergence plot for instance 40.50.200.a

To summarize, the results of the computational study suggest that SFD solves multi-dimensional (0-1) knapsack problems better than solving the DEP directly using CPLEX.

SFD seems to perform best on instances with a smaller number of variables in the first and second stages where it found optimal or very close to optimal solutions for many of the instances. For larger instances, SFD still outperforms solving the DEP, and trends in the data suggest that the gaps will continue to close at a faster rate than CPLEX given longer run times. This result is promising, as SFD is an algorithm which seeks to recover the convex hull using nothing but Fenchel cuts and does not have many of the state of the art advancements of a commercial solver like CPLEX. The inclusion of branch-and-bound on some or all of the variables or additional cutting planes could yield even better results. Exploration of these ideas is left as future work.

G. Conclusion

This paper introduces a new scenario decomposition method for solving stochastic (0-1) two-stage SIPs. The method uses Fenchel cutting planes on scenario decomposed SIPs in order to iteratively recover the convex hull of integer points in the neighborhood around the optimal solution. This approach allows for convergence results from PHA for SLPs to be applied to SIPs and a computational study was presented which provides empirical evidence to support this claim. This work opens up new avenues for future research. As mentioned in section 2 some of the instances solved using SFD required more time from PHA than in generating Fenchel cuts. Using Fenchel cuts with faster implementations of PHA or with other methods for solving scenario decomposed SIPs may yield better results. It was also noted that for the instances with a larger number of variables the proportion of time required for generating Fenchel cuts increased.

Continued work focuses on reducing the amount of time required to find nonanticipative first-stage solutions. One potion would be to implement Fenchel cuts within a branch-and-bound algorithm (such as the dual decomposition algorithm of Caroe and Schultz

(1999).) Branching on the first-stage variables in particular should help minimize the number of SFD iterations for cases when there is little or no change in the first-stage solution vector. In such a scheme the FCG subroutine would be useful in recover integral second-stage variables, whereas the branch-and-bound algorithm would provide integer first-stage variable values. Another option for recovering a nonanticipative solution would use the L-shaped method Slyke and Wets (1969). While the L-shaped method uses a stage-wise decomposition approach, a Fenchel cut generated on a scenario subproblem provides a cut suitable for the corresponding second-stage subproblem.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This dissertation investigated new subgradient-based solution approaches for two stage SMIPs. Stochastic programming, in particular stochastic programming with integer variables is a rich and challenging subset of the mathematical programming field that continues to challenge researchers. Research in this area is important because the inclusion of random parameters into mathematical programming problems yield solutions that hedge against unfavorable future events, which can lead to larger profits and smaller losses when extreme scenarios occur.

The first major contribution of this work was the development of a new method for decomposition and coordination of complex systems of decision makers under private information restrictions. Until the work described here, other authors focused on either relaxing the private information requirements in some way through the use of an intermediary or the solution methods were highly heuristic. The nodal decompositon-coordination (NDC) method can also be considered heuristic since optimality cannot be claimed (due to the duality gap from the Lagrangian relaxation approach) but it was demonstrated that NDC results still beat the still often used deterministic solution.

One of the practical challenges in the supply chain inventory coordination instances was guaranteeing feasibility of the noncoordinated problems. In the SCICP instances, artificial variables were included to enforce feasibility, which caused some numerical instability in the NDC iterations. The choice of the artificial variables and the step sizes dramatically affected the convergence rate of the algorithm. Testing NDC on applications without this infeasibility drawback could yield better results in terms of gap from the global optimal.

The NDC algorithm opens new avenues for future research. For this study, the number of realizations of the random demand was limited so that the DEP could be formulated and solved directly at each step. Implementing a more sophisticated SMIP algorithm to solve the SMIP subproblems would dramatically increase the ability of NDC to solve realistic sized instances and would allow for the inclusion of more random parameters describing the model. However, as discussed in Chapter II, choosing an algorithm for solving a SMIP is very dependent on where the random variables appear in the problem as well as the types of decision variables, so careful modelling of more realistic problem instances is required.

The second contribution of this thesis was the Scenario Fenchel Decomposition method, which uses Fenchel cutting planes in coordination with the progressive hedging algorithm to arrive at the optimal solution. The method was tested on multidimensional knapsack polyhedra in order to take advantage of the special structures that knapsack polyhedra afford the Fenchel cutting plane subroutine. Computational results demonstrate that the algorithm was able to beat attempts at solving the DEP directly.

The algorithm differs from other algorithms in literature. Instead of relying on an enumeration scheme such as branch-and-bound, the algorithm converges toward an optimal solution with the use of Fenchel cuts alone. While improvements may be realized by imposing branch-and-bound on some of the variables, that was not the approach here, and convergence of the algorithm was promising. The SFD algorithm demonstrates the ability of Fenchel cuts to recover the convex hull of scenario subproblems for SMIP.

Ongoing work on SFD focuses on improving the speed at which nonanticipative solutions are found. Computational experiments on the algorithm displayed that there are improvements to be made. The PHA algorithm took a much higher percentage of time than was expected by the authors, so new methods need to be explored such as using branch-and-bound on the first-stage variables. The use of branch-and-bound on the first-stage variables would also offer a convenient lower bound (feasible integer solution) whenever the branch and bound tree returns an integer first-stage decision vector, a phenomenon that is rare in the current SFD implementation. Another option for recovering nonanticipative first-stage decisions is solving the LP relaxed problem with the L-shaped method. Fenchel cuts can then be generated for the scenario subproblems and added to the second stage subproblems for the next L-shaped iteration.

Additionally, the algorithm has only been tested on randomly generated multidimensional knapsack constrained problems. Since knapsack constrained problems are common in applications such as transportation, scheduling, investment planning, and capacity expansion, an interesting extension would be to test the ability of SFD to solve some problems from those applications.

REFERENCES

Ahmed, S., Tawarmalani, M., and Sahinidis, N. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, **100**, 355–377.

Alonso-Ayuso, A., Escudero, L., Garin, A., Ortuno, M., and Perez, G. (2003). An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, **26**(1), 97–124.

Balas, E., Ceria, S., and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 integer programs. *Mathematical Programming*, **58**(3), 295–324.

Bazaraa, M., Sherali, H., and Shetty, C. (1993). *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, Singapore.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, **4**, 238 – 252.

Birge, J. and Louveaux, F. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, **34**(3), 384 – 392.

Birge, J. and Louveaux, F., editors (1997). *Introduction to Stochastic Programming*. Springer, New York.

Blair, C. and Jeroslow, R. (1982). The value function of an integer program. *Mathematical Programming*, **23**, 237–273.

Boyd, E. (1993). Generating fenchel cutting planes for knapsack polyhedra. *SIAM Journal on Optimization*, **3**(4), 734–750.

Boyd, E. (1994a). Fenchel cutting planes for integer programs. *Operations Research*, **42**(1), 53–64.

Boyd, E. (1994b). Solving 0/1 integer programs with enumeration cutting planes. *Annals of Operations Research*, **50**(1), 61–72.

Boyd, E. (1995). On the convergence of fenchel cutting planes in mixed-integer programming. *SIAM Journal on Optimization*, **5**(2), 421–435.

Cachon, G. and Lariviere, M. (2005). Supply chain coordination with revenue-sharing contracts: Strengths and limitations. *Management Science*, **51**(1), 30–44.

Caprara, A., Fischetti, M., and Toth, P. (1999). A heuristic method for the set covering problem. *Operations Research*, **47**(5), 730–743.

Caroe, C. (1998). *Decomposition in Stochastic Integer Programming*. PhD thesis, Department of Operations Research, University of Copenhagen, Denmark.

Caroe, C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, **24**(1-2), 37–45.

Caroe, C. and Tind, J. (1997). A cutting plane approach to mixed 0-1 sotchastic integer programs. *European Journal of Operational Research*, **101**, 306–316.

Caroe, C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, **83**, 451–464.

Carraway, R., Schmidt, R., and Weatherford, L. (1993). An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval Research Logistics*, **40**, 161–173.

Chen, H., Wang, F., and Zeng, D. (2004). Intelligence and security informatics for homeland security: Information, communication, and transportation. *IEEE Transactions on Intelligent Transportation Systems*, **5**(4), 329–341.

Chu, C. and Leon, V. (2008). Power-of-two single-warehouse multi-buyer inventory coordination with private information. *International Journal of Production Economics*, **111**(2), 562 – 574.

Chu, C. and Leon, V. (2009). Scalable methodology for supply chain inventory coordination with private information. *European Journal of Operational Research*, **195**(1), 262 – 279.

Claro, J. and Sousa, J. (2010). A multiobjective metaheuristic for a mean-risk stochastic knapsack problem. *Computational Optimization and Applications*, **46**, 427–450.

Cruijssen, F., Cools, M., and Dullaert, W. (2007). Horizontal cooperation in logistics: Opportunities and impediments. *Transportation Research Part E-Logistics and Transportation Review*, **43**(2), 129–142.

Dantzig, G. (1955). Linear programming under uncertainty. *Management Science*, **1**(3-4), 197–206.

Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, **8**(1), 101–111.

Dantzig, G. and Wolfe, P. (1961). The decomposition algorithm for linear programs. *Econometrica*, **29**(4), 767–778.

Escudero, L., Galindo, E., Garcia, G., Gomez, E., and Sabau, V. (1999). Schumann, a modeling framework for supply chain management under uncertainty. *European Journal of Operational Research*, **119**(1), 14–34.

Escudero, L., Landete, M., and Rodríguez-Chía, A. (2011). Stochastic set packing problem. *European Journal of Operational Research*, **211**, 232–240.

Fox, M., Barbuceanu, M., and Teigen, R. (2000). Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems*, **12**(2-3), 165–188.

Giannoccaro, I. and Pontrandolfo, P. (2004). Supply chain coordination by revenue sharing contracts. *International Journal of Production Economics*, **89**(2), 131–139.

Haneveld, W. and van der Vlerk, M. (1999). Stochastic integer programming: general models and algorithms. *Annals of operations research*, **85**, 39–57.

Haugen, K., Løkketangen, A., and Woodruff, D. (2001). Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *Eurpoean Journal of Operational Research*, **132**, 116–122.

Henig, M. (1990). Risk criteria in a stochastic knapsack problem. *Operations Research*, **38**(5), 820–825.

Higle, J. and Sen, S. (1991). Stochastic decomposition- an algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, **16**(3), 650–669.

IBM ILOG CPLEX (2009). *IBM ILOG CPLEX Callable Library version 12.1 C API Reference Manual*. International Business Machines Corporation, New York.

Jeong, I. and Leon, V. (2002a). Decision-making and cooperative interaction via coupling agents in organizationally distributed systems. *IIE Transactions*, **34**(9), 789–802.

Jeong, I. and Leon, V. (2002b). A distributed scheduling methodology for a two-machine flowshop using cooperative interaction via multiple coupling agents. *Journal of Manufacturing Systems*, **21**(2), 126–139.

Jeong, I. and Leon, V. (2003). Distributed allocation of the capacity of a single-facility using cooperative interaction via coupling agents. *International Journal of Production Research*, **41**(1), 15–30.

Jeong, I. and Leon, V. (2005). A single-machine distributed scheduling methodology using cooperative interaction via coupling agents. *IIE Transactions*, **37**(2), 137–152.

Kiwiel, K. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, **46**, 105–122.

Kleywegt, A. and Papastavrou, J. (1998). The dynamic and stochastic knapsack problem. *Operations Research*, **46**(1), 17–35.

Kleywegt, A. and Papastavrou, J. (2001). The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, **49**(1), 26–41.

Lageweg, B., Lenstra, J., Kan, A. R., and Stougie, L. (1985). Stochastic integer programming by dynamic programming. *Statistica Neerlandica*, **39**, 97–113.

Laporte, G. and Louveaux, F. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, **13**, 133–142.

Louveaux, F. V. and Schultz, R. (2003). Stochastic integer programming. In Ruszczyǹski, A. and Shapiro, A., editors, *Stochastic Programming*, volume **10** of *Handbooks in Operations Research and Management Science*, pages 213 – 266. Elsevier, Amsterdam, The Netherlands.

Mendonca, D. (2007). Decision support for improvisation in response to extreme events: Learning from the response to the 2001 world trade center attack. *Decision Support Systems*, **43**(3), 952–967.

Nemhauser, G. and Wolsey, L. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York.

Norkin, V., Ermoliev, Y., and Ruszczyński, A. (1998). On optimal allocation of indivisibles under uncertainty. *Operations Research*, **46**(3), 381–395.

Ntaimo, L. (2011). Fenchel decomposition for two-stage stochastic mixed-integer programming. *Journal of Global Optimization (Submitted)*.

Ntaimo, L. and Sen, S. (2008). A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, **40**(3), 299–319.

Petrovic, D., Roy, R., and Petrovic, R. (1999). Supply chain modelling using fuzzy sets. *International Journal of Production Economics*, **59**(1-3), 443–453.

Phillips Jr., C., Ting, T., and Demurjian, S. (2002). Information sharing and security in dynamic coalitions. In *SACMAT '02: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, pages 87–96, New York, NY, USA. ACM.

Ramos, M. and Sáez, J. (2005). Solving capacitated facility location problems by fenchel cutting planes. *The Journal of the Operational Research Society*, **56**(3), 297–306.

Rockafellar, R. (1976). Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, **1**(2), 97–116.

Rockafellar, R. (1997). *Convex Analysis*. Princeton University Press, New Jersey.

Rockafellar, R. and Wets, R. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, **16**(1), 119–147.

Ruszczyński, A. (1986). A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, **35**, 309–333.

Ruszczyński, A. (1995). On convergence of an augmented lagrangian decomposition method for sparse convex optimization. *Mathematics of Operations Research*, **20**(3), 634–656.

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

Sáez, J. (2000). Solving linear programming relaxations associated with lagrangean relaxations by fenchel cutting planes. *European Journal of Operational Research*, **121**(3), 609 – 626.

Santos, M., Silva, E., Finardi, E., and Goncalves, R. (2009). Practical aspects in solving the medium-term operation planning problem of hydrothermal power systems by using the progressive hedging method. *Electrical Power and Energy Systems*, **31**, 546–552.

Santoso, T., Ahmed, S., Goetschalckx, M., and Shapiro, A. (2005). A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, **167**(1), 96–115.

Schneeweiss, C. and Zimmer, K. (2004). Hierarchical coordination mechanisms within the supply chain. *European Journal of Operational Research*, **153**(3), 687–703.

Schultz, R. (1993). Continuity properties of expectation functions in stochastic integer programming. *Mathematics of Operations Research*, **18**, 578–589.

Sen, S. and Higle, J. (2005). The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Mathematical Programming*, **104**(1), 1 – 20.

Shapiro, A. (2003). Monte carlo sampling methods. In Ruszczynski, A. and A. Shapiro, editors, *Handbooks in Operations Research and Managment Science, Volume 10*, chapter 6, pages 353–425. Elsevier Science, Amsterdam, The Netherlands.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). *Lectures on Stochastic Programming*. SIAM, Philadelphia, Pennsylvania.

Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, **101**(1-2), 165–200.

Shultz, R., Stougie, L., and van der Vlerk, M. (1998). Solving stochastic programs with integer recourse by enumeration: a framework using gröbner basis reductions. *Mathematical Programming*, **83**, 229–252.

Slyke, R. V. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, **17**(4), 638–663.

Watson, J., Wets, R., and Woodruff, D. (2010). Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, **22**(4), 543–554.

Watson, J. and Woodruff, D. (2010). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, pages 1–16.

Wets, R. (1974). Stochastic programs with fixed recourse: the equivalent deterministic program. *SIAM Review*, **16**(3), 309–339.

Wollmer, R. (1980). Two stage linear programming under uncertainty with 0-1 first stage variables. *Mathematics Programming*, **19**, 279–288.

APPENDIX A

ADDITIONAL PROBLEM INFORMATION FOR SFD COMPUTATIONS

The problems studied in section F are randomly generated stochastic problems with multiple knapsack constraints. The problems are of the form:

$$\begin{aligned} \text{knaps: } \text{Min } & c^\top x + \mathbb{E}[f(\tilde{\omega}, x)] \\ \text{s.t. } & Ax \le b \\ & x \in \mathbb{B}^{n_1}, \end{aligned} \tag{A.1}$$

where, for a given scenario k, the recourse function $f(k, x)$ is given by the following second-stage mixed-integer program (MIP):

$$\begin{aligned} f(k,x) = \text{Min } & q^{k\top} y^k \\ \text{s.t. } & W^k y^k \le h^k - T^k x \\ & y^k \in \mathbb{B}^{n2}. \end{aligned} \tag{A.2}$$

Tables IX - XII are organized as follows: CPLEX LB denotes the best integer decision found while solving the DEP. SFD LB denotes the objective from the best integer decision after applying a rounding heuristic on the SFD solution. DEP Nodes denotes the number of nodes explored while attempting to solve the DEP using CPLEX. LP Gap is the gap between solving the linear relaxation of the DEP and the best known integer solution to the problem (taken from the lower bound calculations).

Table IX. Additional problem information from test set 1

Instance	CPLEX LB	SFD LB	DEP Nodes	LP gap
knaps.10.20.25.a	3229.0	3229.0	2799900	4.40%
knaps.10.20.25.b	3292.6	3292.6	3845800	4.94%
knaps.10.20.25.c	3158.7	3158.7	3768700	4.00%
knaps.10.20.25.d	3248.1	3248.1	3743100	3.85%
knaps.10.20.25.e	3276.2	3276.2	3207300	4.40%
Average			**3472960**	**4.32%**
knaps.10.20.50.a	4520.6	4520.6	2040600	7.66%
knaps.10.20.50.b	4447.3	4447.3	4761800	8.34%
knaps.10.20.50.c	4087.6	4087.6	3352100	8.49%
knaps.10.20.50.d	4070.9	4070.9	2170400	8.50%
knaps.10.20.50.e	4393.7	4396.6	2657400	8.84%
Average			**2996460**	**8.37%**
knaps.10.20.75.a	5141.0	5141.0	3753800	8.11%
knaps.10.20.75.b	5150.6	5153.9	2768400	8.92%
knaps.10.20.75.c	5241.7	5241.7	2791500	9.98%
knaps.10.20.75.d	4889.5	4891.3	858100	8.26%
knaps.10.20.75.e	5242.2	5242.2	3022100	6.76%
Average			**2638780**	**8.40%**
knaps.10.20.100.a	5932.9	5932.9	1966000	9.63%
knaps.10.20.100.b	6047.7	6060.6	539500	9.81%
knaps.10.20.100.c	6394.4	6425.0	2085700	9.37%
knaps.10.20.100.d	5759.0	5764.4	768200	10.72%
knaps.10.20.100.e	5826.7	5830.9	1648600	11.15%
Average			**1401600**	**10.13%**
knaps.10.20.150.a	7580.0	7765.8	240943	10.94%
knaps.10.20.150.b	8021.9	8028.7	518400	9.21%
knaps.10.20.150.c	8076.6	8116.1	390200	11.81%
knaps.10.20.150.d	7125.3	7246.7	954800	13.05%
knaps.10.20.150.e	7764.4	7829.4	317600	9.64%
Average			**484389**	**10.93%**
knaps.10.20.200.a	10224.6	10249.7	672900	11.23%
knaps.10.20.200.b	10087.0	10155.6	321000	11.91%
knaps.10.20.200.c	10085.6	10109.4	271310	10.89%
knaps.10.20.200.d	10085.9	10299.1	911900	13.21%
knaps.10.20.200.e	9962.3	10006.2	285600	10.58%
Average			**492542**	**11.56%**

Table X. Additional problem information from test set 2

Instance	**CPLEX LB**	**SFD LB**	**DEP Nodes**	**LP gap**
knaps.20.30.25.a	3484.3	3484.3	6577600	3.98%
knaps.20.30.25.b	3558.3	3558.3	7993700	3.93%
knaps.20.30.25.c	3530.8	3530.8	6278000	4.96%
knaps.20.30.25.d	3433.7	3433.7	3654000	4.51%
knaps.20.30.25.e	3552.3	3552.3	4606500	4.28%
Average			**5821960**	**4.33%**
knaps.20.30.50.a	4450.5	4454.3	3396700	6.86%
knaps.20.30.50.b	4636.4	4636.4	2237500	5.76%
knaps.20.30.50.c	4538.1	4538.1	4109300	6.86%
knaps.20.30.50.d	4467.7	4467.7	1962600	5.69%
knaps.20.30.50.e	4542.3	4542.3	4151400	6.89%
Average			**3171500**	**6.41%**
knaps.20.30.75.a	5421.3	5425.9	1418700	8.71%
knaps.20.30.75.b	5359.4	5359.4	775900	7.56%
knaps.20.30.75.c	5395.4	5395.4	2104600	7.28%
knaps.20.30.75.d	5483.7	5491.6	2034400	7.41%
knaps.20.30.75.e	5618.0	5618.0	1313600	8.43%
Average			**1529440**	**7.88%**
knaps.20.30.100.a	6289.8	6292.4	2007800	7.96%
knaps.20.30.100.b	6753.5	6767.2	1565700	9.47%
knaps.20.30.100.c	5919.7	5926.4	1594400	9.50%
knaps.20.30.100.d	6347.4	6351.1	1480800	8.92%
knaps.20.30.100.e	5888.4	5919.8	506000	9.67%
Average			**1430940**	**9.10%**
knaps.20.30.150.a	7966.1	8050.3	961700	12.34%
knaps.20.30.150.b	8702.9	8726.1	1198800	10.39%
knaps.20.30.150.c	8394.6	8478.1	372900	11.67%
knaps.20.30.150.d	8278.8	8335.1	759200	10.88%
knaps.20.30.150.e	8490.3	8496.0	899400	11.34%
Average			**838400**	**11.32%**
knaps.20.30.200.a	9846.5	9979.7	610100	11.41%
knaps.20.30.200.b	10523.7	10595.7	669800	10.60%
knaps.20.30.200.c	10271.4	10344.9	525900	12.52%
knaps.20.30.200.d	10898.3	11326.7	281000	10.05%
knaps.20.30.200.e	10566.3	10596.5	508000	11.67%
Average			**518960**	**11.25%**

Table XI. Additional problem information from test set 3

Instance	CPLEX LB	SFD LB	DEP Nodes	LP gap
knaps.30.40.25.a	3531.0	3531.0	3128400	5.08%
knaps.30.40.25.b	3569.5	3569.5	2297000	4.50%
knaps.30.40.25.c	3613.0	3613.0	2412000	4.40%
knaps.30.40.25.d	3544.4	3544.4	4380000	3.99%
knaps.30.40.25.e	3658.1	3658.1	2798900	4.44%
Average			**3003260**	**4.48%**
knaps.30.40.50.a	4799.6	4799.6	1997200	7.39%
knaps.30.40.50.b	4747.3	4747.3	2352800	6.04%
knaps.30.40.50.c	4651.3	4653.1	1803400	6.91%
knaps.30.40.50.d	4668.9	4669.0	2311800	6.48%
knaps.30.40.50.e	4687.5	4692.5	1082400	6.87%
Average			**1909520**	**6.74%**
knaps.30.40.75.a	5612.7	5617.6	887800	7.55%
knaps.30.40.75.b	5644.3	5651.6	1039800	8.19%
knaps.30.40.75.c	5543.0	5547.7	1138400	8.40%
knaps.30.40.75.d	5611.9	5615.5	1168500	9.71%
knaps.30.40.75.e	5728.2	5728.7	1230000	8.38%
Average			**1092900**	**8.44%**
knaps.30.40.100.a	6507.8	6542.5	751400	11.09%
knaps.30.40.100.b	6755.8	6759.8	980500	9.43%
knaps.30.40.100.c	6291.0	6307.5	780000	9.48%
knaps.30.40.100.d	6477.8	6485.0	845000	9.84%
knaps.30.40.100.e	6386.7	6423.0	591500	9.07%
Average			**789680**	**9.78%**
knaps.30.40.150.a	8819.5	8864.0	405300	10.98%
knaps.30.40.150.b	9003.0	9043.0	286600	12.80%
knaps.30.40.150.c	8999.0	9055.8	253200	12.61%
knaps.30.40.150.d	8761.3	8800.9	497800	10.63%
knaps.30.40.150.e	8766.2	8781.9	338700	9.49%
Average			**356320**	**11.30%**
knaps.30.40.200.a	11391.8	11567.4	391500	10.83%
knaps.30.40.200.b	10999.6	11439.6	352500	11.40%
knaps.30.40.200.c	10163.6	10317.8	239000	9.95%
knaps.30.40.200.d	10751.1	10796.7	243900	10.29%
knaps.30.40.200.e	10754.2	10986.5	259600	10.16%
Average			**297300**	**10.53%**

Table XII. Additional problem information from test set 4

Instance	**CPLEX LB**	**SFD LB**	**DEP Nodes**	**LP gap**
knaps.40.50.25.a	3700.3	3700.3	2254700	5.61%
knaps.40.50.25.b	3670.1	3670.1	2443800	4.90%
knaps.40.50.25.c	3665.8	3665.8	1919600	4.80%
knaps.40.50.25.d	3651.4	3651.4	1708400	3.95%
knaps.40.50.25.e	3687.5	3687.5	1771600	4.75%
Average			**2019620**	**4.80%**
knaps.40.50.50.a	4839.7	4839.8	1929000	8.33%
knaps.40.50.50.b	4725.2	4725.2	1749700	7.07%
knaps.40.50.50.c	4775.0	4777.0	1205900	7.12%
knaps.40.50.50.d	4713.1	4719.6	983200	6.37%
knaps.40.50.50.e	4820.7	4823.4	1491500	7.74%
Average			**1471860**	**7.32%**
knaps.40.50.75.a	5759.0	5766.2	1001400	8.61%
knaps.40.50.75.b	5708.9	5721.0	811600	9.85%
knaps.40.50.75.c	5582.1	5589.3	536400	9.73%
knaps.40.50.75.d	5631.4	5642.3	1001900	7.39%
knaps.40.50.75.e	5687.5	5689.1	1315000	9.72%
Average			**933260**	**9.06%**
knaps.40.50.100.a	6680.6	6709.4	770600	9.41%
knaps.40.50.100.b	6665.2	6672.3	838000	9.30%
knaps.40.50.100.c	6761.4	6766.9	565500	10.08%
knaps.40.50.100.d	6727.8	6743.5	640400	9.62%
knaps.40.50.100.e	6643.2	6655.5	664000	8.92%
Average			**695700**	**9.47%**
knaps.40.50.150.a	8928.8	8977.3	284100	11.00%
knaps.40.50.150.b	8675.0	8693.7	476200	9.91%
knaps.40.50.150.c	9028.2	9075.4	390000	11.38%
knaps.40.50.150.d	8714.9	8756.1	435900	10.39%
knaps.40.50.150.e	9224.9	9256.6	750800	11.80%
Average			**467400**	**10.90%**
knaps.40.50.200.a	11467.4	11674.9	200268	11.07%
knaps.40.50.200.b	11739.2	11850.0	327400	10.55%
knaps.40.50.200.c	11022.6	11500.3	229500	10.39%
knaps.40.50.200.d	11195.4	11445.0	237500	10.49%
knaps.40.50.200.e	10854.8	11222.5	221400	9.64%
Average			**243214**	**10.43%**

VITA

Name	Eric Benjamin Beier
Department	Industrial and Systems Engineering
Address	3131 TAMU College Station, TX, 77843-3131
Contact	eric.beier@gmail.com
Education	BS Industrial Engineering, Lamar University 2004 ME Industrial Engineering, Texas A&M University 2008 PhD Industrial Engineering, Texas A&M University 2011

The typist for this dissertation was the author.