On the Implementation of Interior Point Decomposition Algorithms for Two-Stage Stochastic Conic Programs

Sanjay Mehrotra * M. Gökhan Özevin †

October 31, 2005

Industrial Engineering and Management Science Technical Report 2005-04

Abstract

In this paper we develop a practical primal interior decomposition algorithm for two-stage stochastic programming problems. The framework of this algorithm is similar to the framework in Mehrotra and Özevin [17, 18] and Zhao [30], however their algorithm is altered in a simple yet fundamental way to achieve practical performance. In particular, this new algorithm weighs the log-barrier terms in the second stage problems differently than the theoretical algorithms analyzed in [17, 18, 30]. We give a method for generating a suitable starting point; selecting a good starting barrier parameter; a heuristic for first stage steplength calculation without performing line searches; and a method for adaptive addition of new scenarios over the course of the algorithm. The decomposition algorithm is implemented to solve two-stage stochastic conic programs with recourse whose underlying cones are cartesian products of linear, second order, and semidefinite cones. The performance of primal decomposition method is studied on a set of randomly generated test problems as well as a two-stage stochastic programming extension of Markowitz portfolio selection model. The computational results show that an efficient and stable implementation of primal decomposition method is possible. These results also show that in problems with large number of scenarios adaptive addition of scenarios can yield computational savings up to 80%.

Key Words: Stochastic Programming, Conic Programming, Semidefinite Programming, Benders Decomposition, Interior Point Methods, Primal-Dual Methods

*Corresponding Author, Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208, USA (mehrotra@northwestern.edu).

†ZS Associates, Evanston, IL 60201, USA (gokhan.ozevin@zsassociates.com). The research was supported in part by NSF-DMI-0200151 and ONR-N00014-01-1-0048. The work was performed while the author was at Northwestern.

1. Introduction

We consider the two-stage stochastic conic programming (TSSCP) problem in the form:

$$\begin{array}{rl} max & b^T x - \frac{1}{2} x^T G x + \rho(x) \\ s.t. & Dx = d, \\ & Ax + s_1 = c, \\ & s_1 \in \mathcal{K}_1, \end{array} \tag{1.1}$$

where

$$\rho(x) := \sum_{i=1}^{K} \pi_i \rho_i(x) \tag{1.2}$$

and

$$\begin{array}{rl} \rho_i(x) := \quad max & f_i^T y_i - \dfrac{1}{2} y_i^T H_i y_i \\ s.t. & R_i y_i + U_i x = z_i, \\ & W_i y_i + T_i x + s_{2,i} = h_i, \\ & s_{2,i} \in \mathcal{K}_{2,i}. \end{array} \tag{1.3}$$

The symmetric cones $\mathcal{K}_1$ and $\mathcal{K}_{2,i}$ are Cartesian products of non-negative orthant, second-order cones, and the cones of positive semidefinite matrices. By $\mathcal{K}_1^+$ and $\mathcal{K}_{2,i}^+$ we represent the interior of these cones. The size of these cones in the first stage and second stage problems may be different. The columns of submatrices of A, W_i, and T_i corresponding to the semidefinite cones are vectorizations of symmetric matrices. In other words, the cone of semidefinite matrices are considered as a set of vectors to comply with the above standard form of TSSCP. The matrices G and H_i are symmetric positive semidefinite.

Faybusovich [6, 7] presented a unified description of interior point algorithms for linear, second order cone, and semidefinite programming using the theory of Euclidian Jordan algebra. We also describe our interior decomposition algorithms using the Jordan algebra operations. For a brief introduction of Jordan algebra operations see [2, 26, 27].

Let us define the following feasibility sets:

$\mathcal{F}^0 := \{x \mid Dx = d, Ax + s_1 = b,\ s_1 \in \mathcal{K}_1\}$,
$\mathcal{F}_{2,i}(x) := \{y_i \mid R_i y_i + U_i x = z_i, W_i y_i + s_{2,i} = h_i - T_i x,\ s_{2,i} \in \mathcal{K}_{2,i}\}$,
$\mathcal{F}_i^1 := \{x \mid \mathcal{F}_{2,i}(x) \neq \emptyset\}$,
$\mathcal{F}^1 := \{\cap_{i=1}^K \mathcal{F}_i^1\} \cap \mathcal{F}^0$ and
$\mathcal{F} := \{(x, s_1) \times (y_1, s_{2,1}, \ldots, y_K, s_{2,K}) \mid Dx = d, Ax + s_1 = b,\ s_1 \in \mathcal{K}_1;\ R_i y_i + U_i x = z_i,$
$W_i y_i + s_{2,i} = h_i - T_i x,\ s_{2,i} \in \mathcal{K}_{2,i}\}$.

Here $\mathcal{F}_{2,i}(\cdot)$ is the second stage feasibility set parameterized by the first stage decision vector x, $\mathcal{F}^1$ is the set of first stage feasible solutions for which all second stage problems are also feasible, and $\mathcal{F}$ is the set of feasible solutions for the extensive formulation of TSSCP.

Mehrotra and Özevin [18] presented a decomposition based primal interior algorithm for the two-stage stochastic semidefinite programming problem with recourse. In their setting the first stage cone $\mathcal{K}_1$ and the second stage cones $\mathcal{K}_{2,i}$ are $p \times p$ and $r \times r$ symmetric positive semidefinite matrices, respectively. Mehrotra and Özevin [18] show that starting from a well centered first stage solution (starting barrier parameter μ^0) a short-step path-following interior point algorithm working on the first stage variables requires $O(\sqrt{p + rK}) \ln(\mu^0/\mu^*)$ first stage interior point iterations to obtain a well centered solution for barrier parameter μ^*, where K is the number of scenarios. Their analysis extends the earlier results of Zhao [30] for the two-stage linear stochastic programming problems, and the results of Mehrotra and Özevin [17] for the quadratic case. The work of Zhao [30], and Mehrotra and Özevin [17, 18] provides a framework for implementing primal interior decomposition algorithms for two-stage stochastic programming problems. Several simplifying assumptions were made by Zhao [30] and Mehrotra and Özevin [17, 18] while analyzing their algorithms. The purpose of this paper is to develop a more practical primal interior decomposition algorithm for two stage stochastic conic programs.

The basic steps of a primal interior decomposition algorithm are as follows:

A Basic Primal Interior Decomposition Framework (Algorithm 1)

Initialization.
Let x^1 be a starting point, $\mu^1 > 0$ a starting barrier parameter, and μ^* be the desired termination value of the barrier parameter. Also, let $\beta > 0, \gamma \in (0, 1)$, and $\theta > 0$ be suitable scalar parameters.

Step 1.

Step 1.1. Solve all second stage centering problems for the current x and μ.

Step 1.2. Using the second stage solutions compute the first stage Newton direction.

Step 1.3. Compute the local norm $\delta(\mu, x)$ of the Newton direction Δx as a measure of distance from the current point x to the first stage μ-center.

Step 1.4. Update first stage solution as $x := x + \theta \Delta x$. If $\delta(\mu, x) \leq \beta$ go to Step 2, otherwise go to Step 1.1.

Step 2. If $\mu \leq \mu^*$ (or some other termination criterion is satisfied) stop, otherwise reduce $\mu = \gamma\mu$ and go to Step 1.1.

Note that a variant of Algorithm 1 that reduces μ rapidly (long-step variant) may iterate several times in the loop Step 1.1 - Step 1.4 after reducing the value of μ in Step 2. In the short-step method this loop is executed only once, but this method reduces μ slowly. The iterates in the loop Step 1.1 - Step 1.4 are called the inner iterations of Algorithm 1. These iterations constitute most of the work in Algorithm 1. Theoretical values for parameters β, γ, θ, and the choice of $\delta(\mu, x)$ are given in [30, 17, 18] in the context of proving polynomial time convergence of short-step and long-step methods.

This paper develops a different primal interior decomposition algorithm than the one analyzed in [30, 17, 18]. While the algorithm analyzed in [30, 17, 18] has a better worst case theoretical complexity than the one proposed in this paper, computational experience suggests that the new proposed algorithm gives better performance in practice. The proposed algorithm differs from the algorithms in [30, 17, 18] in Step 1.1. In particular, it defines the second stage centering problems differently. We will present this algorithm and compare it with the algorithms in [30, 17, 18] in Section 2. We have some probabilistic/heuristic justifications for the superior performance of the proposed algorithm, which we intend to document elsewhere.

In the primal decomposition framework (for algorithms in [30, 17, 18], as well as the new algorithm) we need to address several issues to develop practical efficient implementations. These issues are:

(i) (Initialization). We need to find a suitable feasible first stage solution and ensure the feasibility of all second stage problems corresponding to this solution. If TSSCP satisfies a full recourse assumption, i.e., all second stage problems are feasible for all first stage solutions, then feasibility of second stage is not an issue. However, this assumption may not hold in general.

(ii) (Steps 1.1, 1.2). The assumption of the availability of exact solutions of the second stage centering problems in [30, 17, 18] considerably simplifies the convergence analysis. However, solving second stage problems exactly is not possible. The practical implications of working with inexact solutions on first stage iterations and algorithmic robustness are not clear.

(iii) (Step 1.3). It is not clear how closely the algorithm should follow the first stage central path (value of parameter β) for it to be practical and robust. Following the central path too closely would unnecessarily increase computational efforts, whereas if we ignore the central path the algorithm may not converge.

(iv) (Step 1.4). The theoretical analysis assumes taking fixed steps along the Newton direction. This is conservative since taking larger steps give faster convergence. We need to find a good way to select the first stage step-length θ without performing line searches. Performing a line search over the barrier objective (defined in Section 2) is not practical since an

evaluation of this function for a given μ and x requires solutions of all second stage problems which is equivalent to computing a new first stage Newton direction.

(v) (Step 1.5). A practical value of γ.

(vi) (Adaptive Addition of Scenarios). The primal decomposition interior point methods for two-stage stochastic programming are appealing because the computation of the first stage Newton direction decomposes across second stage scenarios. This naturally allows the possibility of increasing the number of second stage scenarios as the algorithm progresses. The analysis in [30, 17, 18] is for a fixed number of scenarios. Note that adding a significant number of new scenarios modifies the recourse function and shifts the first stage central path. A theoretical analysis of this shift is challenging. However, adding scenarios adaptively as the algorithm progresses has the potential of saving computational efforts, especially when the number of scenarios is very large. How can adaptive addition of scenarios be implemented in practice?

(vii) (Warm-Start). Is it possible to use solutions from previously solved second stage problems to warm-start new related second stage problems? There are two situations where a warm-start might be possible. The first possibility is when x and/or μ changes in Steps 1.4 and 1.5 in a pre-existing scenario. The second possibility is when scenarios are added adaptively, and a centered solution of a new scenario is desired for the current x and μ.

In stochastic programming we have an additional issue of determining the number of scenarios (when the uncertain parameters have continuous distribution, or have a huge finite support) required to ensure a desired quality of solution. This is achieved through generating lower and upper bounds, and a statistical analysis around these bounds. Methods for generating such bounds are discussed in Linderoth, *et. al.* [13], and are beyond the scope of this paper. In this paper we will assume that the problem has a given large number of scenarios.

Within the context of the proposed practical algorithm we provide resolution of (i-vii) to various extent. We give numerical results for the proposed implementation strategies. Computational results are presented in Sections 4–7. These results are obtained on an extension of the classical Markowitz portfolio optimization problem, and a set of randomly generated conic programming test problems that have linear, second order, and semidefinite cones. The test problems are described in Section 3. Section 4 focuses on various algorithmic parameters and line search in the primal decomposition algorithm. Section 5 presents results on various characteristics of the algorithm exhibited by the problems we solved. Section 6 develops a methodology for adaptive addition of Scenarios, and Section 7 discusses results obtained using scenario addition and warm-start during scenario addition. The computational results were obtained using MATLAB version 6.5.1 on a IBM T42 using Pentium M 1.7 GHz with 1GB of RAM. The entire code is written using MATLAB's internal libraries.

2. Primal Interior Decomposition Algorithms

In this section we describe the algorithm we have found to work better in practice. We compare this algorithm with the algorithms presented and analyzed in Zhao [30], and Mehrotra and Özevin [17, 18]. The algorithm of [30, 17, 18] is described in Section 2.1. In Section 2.2 we describe the proposed practical algorithm, and in Section 2.3 we compare the two algorithms computationally.

2.1 Zhao [30], Mehrotra and Özevin [17, 18] Primal Decomposition Algorithm

In [30, 17, 18] the authors redefine the recourse function in (1.2) as

$$\rho(x) := \sum_{i=1}^{K} \bar{\rho}_i(x),$$

where

$$\begin{aligned} \bar{\rho}_i(x) := \quad & max \quad \bar{f}_i^T y_i - \frac{1}{2} y_i^T \bar{H}_i y_i \\ & s.t. \quad (y_i, s_{2,i}) \in \mathcal{F}_{2,i}(x), \end{aligned} \tag{2.1}$$

$\bar{f}_i = \pi_i f_i$, and $\bar{H}_i = \pi_i H_i$. They consider the log-barrier problem associated with (2.1):

$$\begin{aligned} \bar{\rho}_i(\mu, x) := \quad & max \quad \bar{f}_i^T y_i - \frac{1}{2} y_i^T \bar{H}_i y_i + \mu \ln \det(s_{2,i}) \\ & s.t. \quad (y_i, s_{2,i}) \in \mathcal{F}_{2,i}(x). \end{aligned} \tag{2.2}$$

The first stage log-barrier problem in [30, 17, 18] is defined as:

$$\begin{aligned} max \quad \bar{\eta}(\mu, x) := \quad & b^T x - \frac{1}{2} x^T G x + \bar{\rho}(\mu, x) + \mu \ln \det(s_1) \\ s.t. \qquad & (x, s_1) \in \mathcal{F}^1, \end{aligned} \tag{2.3}$$

where

$$\bar{\rho}(\mu, x) := \sum_{i=1}^{K} \bar{\rho}_i(\mu, x).$$

For a given x and μ, the optimality conditions for (2.2) are:

$$\begin{aligned} & R_i y_i + U_i x = z_i, \\ & W_i y_i + T_i x + s_{2,i} = h_i, \\ & W_i^T \bar{\lambda}_i + R_i^T \bar{\gamma}_i + \bar{H}_i y_i = \bar{f}_i, \\ & P(\bar{\lambda}_i^{1/2}) s_{2,i} = \mu \iota_{2,i}, \\ & \bar{\lambda}_i \in \mathcal{K}_{2,i}^+, \ s_{2,i} \in \mathcal{K}_{2,i}^+. \end{aligned} \tag{2.4}$$

Let $x(\mu) := argmax\{\bar{\eta}(\mu, x)\}$ for a given $\mu > 0$. We refer to $x(\mu)$ as the first stage μ-center and to the trajectory $\{x(\mu),\ \mu > 0\}$ as the first stage central path. We denote the unique solution of (2.4) for any given $x \in \mathcal{F}_i^1$ and $\mu > 0$ by $(y_i(\mu, x),\ s_{2,i}(\mu, x),\ \bar{\lambda}_i(\mu, x),\ \bar{\gamma}_i(\mu, x)),\ i = 1, \ldots, K$, and call this solution the second stage μ-center. The μ-centers for different values of μ form the second stage central path for a given x as μ decreases from ∞ to zero.

A brief summary of the key Jordan algebra operations is as follows. In (2.4) the matrix $P(\lambda_i^{1/2})$ is the *Jordan quadratic presentation* of the vector $\lambda_i^{1/2}$ and $\iota_{2,i}$ is the *identity solution* of the cone $\mathcal{K}_{2,i}$ [6, 7]. The vector $\lambda_i^{1/2}$ is the *square root* (in the Jordan algebra sense) of the dual multiplier λ_i, that is, $\lambda_i^{1/2}$ is a unique vector such that $\lambda_i = P(\lambda_i^{1/2})\iota$. For any given symmetric cone $\mathcal{K}$, and a given $\lambda \in \mathcal{K}$, the *identity solution* ι of $\mathcal{K}$ is a unique element of $\mathcal{K}$ such that $L(\lambda)\iota = \lambda$ for all $\lambda \in \mathcal{K}$, where $L(\lambda)$ is the multiplication by λ linear operator of the cone $\mathcal{K}$. Furthermore, the vector λ^{-1} is the unique inverse of $\mathcal{K}^+$ such that $P(\lambda)\lambda^{-1} = \lambda$. The vector $\lambda^{1/2}$ is the *square root* of $\lambda \in \mathcal{K}$ such that $L(\lambda^{1/2})\lambda^{1/2} = \lambda$. The gradient and Hessian of the function $\ln\det(\lambda)$ has the form:

$$\nabla \ln\det(\lambda) = \lambda^{-1},\ \lambda \in \mathcal{K}^+,$$

$$\nabla^2 \ln\det(\lambda) = -P(\lambda^{-1}),\ \lambda \in \mathcal{K}^+.$$

The form of $P(\lambda)$ in case of primitive symmetric cones as well as Cartesian products of primitives are given in [26]. The gradient and Hessian of the barrier objective $\bar{\eta}(\mu, x)$ are calculated as follows:

$$\nabla\bar{\eta}(\mu, x) = b - Gx + \sum_{i=1}^{K}(T_i^T\bar{\lambda}_i(\mu, x) + U_i^T\bar{\gamma}_i(\mu, x)) - \mu A^T s_1^{-1}, \tag{2.5}$$

$$\nabla^2\bar{\eta}(\mu, x) = -G + \sum_{i=1}^{K}(T_i^T\nabla\bar{\lambda}_i(\mu, x) + U_i^T\nabla\bar{\gamma}_i(\mu, x)) - \mu A^T P(s_1^{-1})A. \tag{2.6}$$

The quantities $\nabla\bar{\lambda}(\cdot)$, and $\nabla\bar{\gamma}_i(\cdot)$ are calculated as follows. Differentiating the optimality conditions (2.4) with respect to x we obtain

$$\begin{aligned}
&R_i\nabla y_i = -U_i,\\
&W_i\nabla y_i + \nabla s_{2,i} = -T_i,\\
&W_i^T\nabla\bar{\lambda}_i + R_i^T\nabla\bar{\gamma}_i + \bar{H}_i\nabla y_i = 0,\\
&P(\bar{\lambda}_i^{1/2})\nabla s_{2,i} + P(s_{2,i}^{1/2})\nabla\bar{\lambda}_i = 0.
\end{aligned} \tag{2.7}$$

Solving (2.7) we get

$$\begin{aligned}
&\nabla\bar{\gamma}_i = (R_i Z_i^{-1} R_i^T)^{-1}(U_i - R_i Z_i^{-1} W_i^T Q_i T_i),\\
&\nabla y_i = -Z_i^{-1}(R_i^T\nabla\bar{\gamma}_i + W_i^T Q_i T_i),\\
&\nabla s_{2,i} = -(T_i + W_i\nabla y_i),\\
&\nabla\bar{\lambda}_i = -Q_i\nabla s_{2,i},
\end{aligned} \tag{2.8}$$

where

$$Q_i := P(\bar{\lambda}_i^{1/2})P(s_{2,i}^{-1/2}) \text{ and } Z_i := W_i^T Q_i W_i + \bar{H}_i.$$

For a fixed μ, the solution $(x(\mu), s_1(\mu); y_i(\mu,\ x(\mu)), s_{2,i}(\mu,\ x(\mu)),\ i = 1, \ldots, K)$ of (2.2–2.3) is same as the maximizer of the log-barrier function

$$b^T x - \frac{1}{2}x^T G x + \sum_{i=1}^{K}(\bar{f}_i y_i - \frac{1}{2}y_i^T \bar{H}_i y_i) + \mu \ln\det(s_1) + \mu \sum_{i=1}^{K} \ln\det(s_{2,i}) \tag{2.9}$$

over the set $\mathcal{F}$.

In the basic primal decomposition framework (Algorithm 1) the first stage Newton direction Δx (Step 1.2) is the optimal solution of

$$\begin{array}{ll} \max & \nabla\bar{\eta}(x)^T\Delta x + \frac{1}{2}\Delta x^T \nabla^2\bar{\eta}(x)\Delta x \\ s.t. & D\Delta x = 0. \end{array} \tag{2.10}$$

In Step 1.3, the local norm of the Newton step calculated at x is given by

$$\delta(\mu, x) = \sqrt{-\frac{1}{\mu}\Delta x^T \nabla^2\bar{\eta}(\mu, x)\Delta x}.$$

For two-stage stochastic semidefinite programs Mehrotra and Özevin [18] show that the log-barrier recourse function $\bar{\eta}(\mu, x)$ is a strongly μ-self-concordant function, and forms a strongly self-concordant family. This is used to establish convergence for short and long-step path following algorithms as discussed in the introduction.

2.2 A Practical Primal Decomposition Algorithm

We now describe a more practical primal decomposition algorithm within the framework of Algorithm 1. In this algorithm we work directly with the form of the second stage problem (1.3) without scaling the second stage objectives as is done in (2.1). Let us consider the log-barrier problem associated with (1.3):

$$\begin{array}{rll} \rho_i(\mu, x) := & max & f_i^T y_i - \frac{1}{2}y_i^T H_i y_i + \mu \ln\det(s_{2,i}) \\ & s.t. & (y_i, s_{2,i}) \in \mathcal{F}_{2,i}(x). \end{array} \tag{2.11}$$

The first stage log-barrier problem is now defined as:

$$\begin{array}{rll} max & \eta(\mu, x) := & b^T x - \frac{1}{2}x^T G x + \mu \ln\det(s_1) + \rho(\mu, x) \\ & s.t. & x \in \mathcal{F}^1, \end{array} \tag{2.12}$$

where

$$\rho(\mu, x) := \sum_{i=1}^{K} \pi_i \rho_i(\mu, x). \tag{2.13}$$

We can show that for a fixed μ, the solutions $(x(\mu), s_1(\mu); y_i(\mu, x(\mu)), s_{2,i}(\mu, x(\mu)),\ i = 1, \ldots, K)$ of (2.11 – 2.12) are same as the maximizer of the log-barrier function

$$b^T x - \frac{1}{2} x^T G x + \sum_{i=1}^{K} \pi_i (f_i y_i - \frac{1}{2} y_i^T H_i y_i) + \mu \ln det(s_1) + \mu \sum_{i=1}^{K} \pi_i \ln \det(s_{2,i}) \tag{2.14}$$

over $\mathcal{F}$.

Note that in contrast to (2.9), in (2.14) the second stage log-barrier terms are scaled with the corresponding probabilities. Hence, when solving the second stage problems to compute the Newton direction for a given x, solutions are computed for a smaller value of μ (scaled by π_i). We do not have an obvious analog of the result that $\bar{\eta}(\mu, x)$ is a strongly μ-self-concordant function for the function $\eta(x, \mu)$. A weaker worst case theoretical bound is expected when following the analysis in [18]. However, a probabilistic analysis shows that better convergence results are possible under an appropriate probabilistic assumption. A detailed analysis is beyond the scope of this paper, and would be treated elsewhere.

The algebraic development of the computation of Newton direction for (2.12) is similar to the development for (2.3). For a given x and μ, the optimality conditions of (2.11) are

$$\begin{aligned} & R_i y_i + U_i x = z_i, \\ & W_i y_i + T_i x + s_{2,i} = h_i, \\ & W_i^T \lambda_i + R_i^T \gamma_i + H_i y_i = f_i, \\ & P(\lambda_i^{1/2}) s_{2,i} = \mu \iota_{2,i}, \\ & \lambda_i \in \mathcal{K}_{2,i}^+,\ s_{2,i} \in \mathcal{K}_{2,i}^+, \end{aligned} \tag{2.15}$$

where $\iota_{2,i}$ is defined as before. The gradient and Hessian of the barrier objective $\eta(\mu, x)$ are calculated as follows:

$$\nabla \eta(\mu, x) = b - Gx - \sum_{i=1}^{K} \pi_i (T_i^T \lambda_i(\mu, x) + U_i^T \gamma_i(\mu, x)) - \mu A^T s_1^{-1}, \tag{2.16}$$

$$\nabla^2 \eta(\mu, x) = -G - \sum_{i=1}^{K} \pi_i (T_i^T \nabla \lambda_i(\mu, x) + U_i^T \nabla \gamma_i(\mu, x)) - \mu A^T P(s_1^{-1}) A. \tag{2.17}$$

In the basic primal decomposition framework (Algorithm 1) the first stage Newton direction (Step 1.2) is the optimal solution of:

$$\begin{aligned} \max \quad & \nabla \eta(x)^T \Delta x + \tfrac{1}{2} \Delta x^T \nabla^2 \eta(x) \Delta x \\ s.t. \quad & D \Delta x = 0, \end{aligned} \tag{2.18}$$

	Section 2.1 Alg.		Section 2.2 Alg.	
problem	itr	convergence	itr	convergence
MCR1-100	318	No	22	Yes
MCR2-100	236	No	27	Yes
MCR3-100	290	Yes	22	Yes
MCR4-100	254	No	29	Yes
MCR5-100	215	No	38	Yes
MCR6-100	323	Yes	35	Yes

Table 1: Comparison of Theoretical and Practical Algorithms ($\beta = 1,\ \nu = 1,\ \gamma = 0.1$; using line search)

which can be calculated by solving the KKT system associated with (2.18):

$$\begin{aligned} &\nabla^2\eta(x)\Delta x - D^T\tau = -\nabla\eta(x) \\ &D\Delta x = 0. \end{aligned} \tag{2.19}$$

In Step 1.3, the local norm of the Newton step calculated at x becomes

$$\delta(\mu, x) = \sqrt{-\frac{1}{\mu}\Delta x^T \nabla^2\eta(\mu, x)\Delta x}.$$

2.3 Performance Comparison of Basic Primal Decomposition Algorithms

The computations of Newton direction in Sections 2.1 and 2.2 were implemented using the same starting point, using identical parameter settings, and the same subroutines. These settings are discussed in the next section in the context of the algorithm in Section 2.2. We ensured that same relative precision is achieved when solving the second stage KKT systems (2.4) and (2.15) to compute the second stage solutions in their respective problems. Table 1 gives results on a set of 100 scenario test problems. These problems are described in Section 3.2. Columns "itr" give the number of inner iterations in both algorithms, and columns "convergence" indicates if convergence to eight digits of accuracy was achieved for these problems when the algorithms terminate. A failure to converge to a solution with eight digits of accuracy is indicated by "NO." Various reasons for this failure are explained in Section 4.2. Superiority of the algorithm in Section 2.2 is obvious.

3. Test Problems

We studied the performance of our decomposition algorithm on randomly generated two-stage stochastic conic programs and a specific stochastic programming problem arising from a two-stage extension of the classical Markowitz's mean-variance model. We now describe

generation of these test problems.

3.1 An Application from Finance: Two-Stage Extension of Markowitz's Mean Variance Model

The seminal mean-variance model of Markowitz [15] provides a quantitative framework for establishing a balance of the risk and return characteristics of various asset classes. In this model return on the portfolio is measured by the expected value of the portfolio return, and the associated risk is quantified by the variance of the portfolio return.

Consider a portfolio that invests in n assets over a single period. Denote by $x \in \mathbb{R}^n$ the portfolio vector, and by $\tilde{r} \in \mathbb{R}^n$ the random vector of asset returns over a given horizon. The mean-variance setting assumes that $\tilde{r}$ has a multivariate normal distribution with mean $\bar{r}$ and covariance Q. The corresponding mean-variance problem is given by $min\{w^T Q w : \bar{r}^T w \geq \rho, e^T w = 1\}$. By minimizing the portfolio variance by varying the level of expected return ρ we can derive the so called mean-variance efficient frontier.

We formulate a two-stage extension of the static single-period Markowitz model to address these issues. Let us first introduce some notation in preparation to formulating this model. This model can be viewed as an alternative to the robust optimization model proposed by Goldfarb and Iyengar [10] in incorporating uncertainty in the covariance matrix. The current time is denoted by t_1. At time t_1 after having observed the return vector r_0 the investor forms a portfolio and at time $t_2 > t_1$ he can revise his portfolio. The revised portfolio is kept until t_3. The problem is to decide a portfolio at time t_1 in anticipation of its revision at time t_2. The evolution of the asset return vector and the covariance matrix is approximated with K scenarios, which we index by $i = 1, \ldots, K$.

We define the following decision variables and parameters:

w_1 := vector of first stage portfolio positions,
r_1 := vector of asset returns in the first period,
$\bar{r}_1$:= expected asset returns in the first period,
$\bar{r}_1^{min}$:= lower bound for the expected asset returns in the first period,
Q_1 := covariance of asset returns in the first period,
$w_{2,i}$:= vector of second stage portfolio positions under scenario i,
$r_{2,i}$:= vector of asset returns in the second period under scenario i,
$\bar{r}_{2,i}$:= expected asset returns in the second period under scenario i,
$\bar{r}_{2,i}^{min}$:= minimum required return at the end of the second period under scenario i,
$Q_{2,i}$:= covariance of asset returns in the second period under scenario i.

We have the following two-stage extension of the basic mean-variance model

$$\begin{aligned} min \quad & {w_1}^T Q_1 w_1 + \sum_{i=1}^{K} \frac{1}{K} w_{2,i}^T Q_{2,i} w_{2,i} \\ s.t. \quad & \bar{r}_1^T w_1 \geq \bar{r}_1^{min}, \\ & e^T w_1 = 1, \\ & \bar{r}_{2,i}^T w_{2,i} \geq \bar{r}_{2,i}^{min}, \ i = 1, \ldots, K, \\ & e^T w_{2,i} = 1, \ i = 1, \ldots, K, \\ & \|w_1 - w_{2,i}\| \leq \tau_{2,i}, \ i = 1, \ldots, K. \end{aligned} \tag{3.1}$$

In the above model, the magnitude of the variation between first and second stage solutions is bounded by the constraints (3.1). In practice an investor would avoid large changes between the optimal first and second stage portfolio positions since such changes can lead to prohibitive amounts of transaction costs. Moreover, limiting the variation between the first and second period portfolio positions would make first period optimal portfolio composition less sensitive to perturbations in the problem parameters. Alternatively, the model may be viewed as a way of finding a 'centroid' solution when multiple estimates of r and Q exist. In both respects, one may prefer our model to the single-period robust portfolio optimization model introduced in Goldfarb and Iyengar [10], since robust models tend to be pessimistic.

For the test problems generated for this paper we use a multivariate GARCH model to describe the return process:

$$\begin{aligned} r_t &= \phi_0 + \phi_1 r_{t-1} + \epsilon_t, \ \epsilon_t \sim N(0, Q_t) \\ Q_t &= C + A * H_{t-1} + B * Q_{t-1} \end{aligned} \tag{3.2}$$

where the symbol $*$ denotes the Hadamard product of two matrices and $H_t = [\epsilon_{i,t}\epsilon_{j,t}]_{i,j=1,\ldots,n}$.

We randomly generated the model parameters ϕ_0, ϕ_1, A, C, and B, as well as the starting asset returns vector r_1 and covariance Q_1. The matrices A, B and C are generated as symmetric positive semidefinite matrices to ensure that the forecasted conditional covariance matrices Q_t are positive semidefinite [12]. We calibrated the probability distributions from which we sample A, B and C such that the forecasted asset returns in general fall into the [0.5,1.5] range, volatilities vary around 10% and correlations take values between -0.5 and 0.5. These parameter ranges allow us to generate problems that are realistic, but also have significant volatility to test the algorithm.

We generate scenarios $(\bar{r}_{2,i}, Q_{2,i})$ using an n-dimensional Sobol sequence [5, 9], where n is the number assets in the portfolio. To obtain a set of K samples, $(\epsilon_{1,1}, \ldots, \epsilon_{1,K})$, of the residual vector ϵ_1, we start with generating the K n-dimensional Sobol points $(S_1, \ldots, S_K)$. Then, we set $\epsilon_{1,i} = Q_1^{1/2}\Phi^{-1}(S_i), i = 1, \ldots, K$, where Φ is the cumulative normal distribution function.

Finally, we generate $(r_{2,i}, Q_{2,i})$ using the following relationships:

$$\begin{aligned} r_{2,i} &= \phi_0 + \phi_1(r_1 + \epsilon_{1,i}), \\ Q_{2,i} &= C + A * H_{1,i} + B * Q_1,\ i = 1, \ldots, K. \end{aligned}$$

To generate the minimum required expected asset returns $\bar{r}_1^{min}$ and $\bar{r}_{2,i}^{min}$ we assume that the investor wants to beat a fixed-weight strategy $\bar{w}$. We randomly generate the vector $\bar{w}$ and then set $\bar{r}_1^{min} = r_1\bar{w}$ and $\bar{r}_{2,i}^{min} = r_{2,i}\bar{w},\ i = 1, \ldots, K$. In this paper, we report results on instances of this model for $n = 20,\ 30$, and 40 and for different values of $\tau_{2,i}$. The computational results in Section 7 indicate that smaller values of $\tau_{2,i}$ tend to generate second stage problems with which warm-start is harder.

3.2 Generation of the Random Test Problems

While two-stage conic extensions of Markowitz model provide an important application example, a more systematic study on the algorithmic behavior is possible on random test problems. We generate these test problems by taking discrete approximations of the following continuous stochastic program:

$$\begin{aligned} max \quad \eta(x) := \quad & b^T x + \rho(x) \\ s.t. \quad & Dx = d, \\ & Ax + s_1 = c, \\ & \|x\|_2 \leq \tau_1 \\ & s_1 \in \mathcal{K}_1, \end{aligned} \tag{3.3}$$

where

$$\rho(x) := E(\rho(x, \tilde{\omega})), \tag{3.4}$$

and for each realization $\omega \in \Omega$ of the random variable $\tilde{\omega}$

$$\begin{aligned} \rho(x, \omega) := \quad max \quad & f_\omega^T y_\omega \\ s.t. \quad & R_\omega y_\omega + U_\omega x = z_\omega, \\ & W_\omega y_\omega + T_\omega x + s_{2,\omega} = h_\omega, \\ & \|y_\omega\|_2 \leq \tau_2 \\ & s_{2,\omega} \in \mathcal{K}_{2,\omega}. \end{aligned} \tag{3.5}$$

The random test problems are generated by taking $\tilde{\omega}$ as a t-dimensional vector that is uniformly distributed over $[0, L]^t$. The first and second stage cones $\mathcal{K}_1$ and $\mathcal{K}_{2,i}$, both consist of a linear, a second order and two semidefinite cones. The 2-norm inequality constraint in (3.3–3.5) gives an additional second order cone. The data D, A, b for the first stage problems uses $\tilde{\omega}$ to generate entries in $[-L, L]$. We ensure that D has full row rank and that A has

full column rank. We then set $d = D\tilde{x}$ and $c = A\tilde{x} + \iota_1$, where ι_1 is the *identity element* of the cone $\mathcal{K}_1$. This ensures that the first stage has a feasible interior solution. The problem data $W(\omega)$ and $T(\omega)$ consists of 4 blocks corresponding to the 4 primitive blocks that make up $\mathcal{K}_{2,\omega}$. We assume that t entries in $f(\omega), R(\omega), U(\omega)$ and in each block of the second stage problem data $W(\omega)$ and $T(\omega)$ are random. We start with randomly generating the vector f^0 and the matrices R^0, U^0, W^0 and T^0. The data f^0, R^0, U^0, W^0 and T^0 is used as a base, which is further randomized to get the second stage scenarios. This is done as follows.

We discretize the continuous probability distribution of $\tilde{\omega}$ using a quasi Monte Carlo technique. In particular, we use a t-dimensional Sobol sequence to generate the second stage problems. Let $(S_1, \ldots, S_N)$ be the Sobol sequence approximating the t-dimensional uniform distribution. To construct N second stage problems we generate N samples, $\omega_1, \ldots, \omega_N$, of the random vector $\tilde{\omega}$ from this Sobol sequence using N points. We set $\omega_i = p_{\%} * L * (1 - 2S_i),\ i = 1, \ldots, N$. To obtain the second stage scenario data f_i, R_i, U_i, W_i and T_i, t elements are randomized in every block (each block corresponding to a cone) of f^0, R^0, U^0, W^0 and T^0 by adding entries of ω_i. Here $p_{\%}$ is the randomization (measured as %) of the base data. A larger value of $p_{\%}$ gives a greater randomization of the base data. Next, we generate a random vector $\tilde{y}$ of the same size as the second stage variable y_ω and set $z_i = R_i\tilde{y} + U_i\tilde{x}$ and $h_i = W_i\tilde{y} + T_i\tilde{x} + \iota_{2,i}$. We choose an appropriate value of τ_1 and $\tau_{2,i}$ to ensure that the feasible sets are bounded. This ensures the feasibility of TSSCP. In (3.3) and (3.5) 2-norm constraints on x and y_ω are given separately for clarity, however, in the rest of the paper they are considered as a second-order cone block in A, W_i, T_i and their slacks will be included as a second-order cone block in s_1 and $s_{2,i}$, respectively.

Table 2 gives problems generated using the above described method. For all problems $L = 2.5$ is used. This table gives information on the number of equality constraints (Column: Equality), the size of linear cones (Column: LINC), the size of second order cones (Column: SOC), the size of semidefinite cones (Column: SDC) and the dimension of the first and second stage variables x and y_i in each of the problems. Problems MCR1–6 are generated by taking a reasonable amount of randomness in the second stage ($p_{\%} = 5\%$), Problems MCL1–6 are generated by taking a large amount of randomness ($p_{\%} = 50\%$), and problems MCH1–6 are generated by taking a huge amount of randomness ($p_{\%} = 500\%$). For all of these problems the size of first and second stage cones are equal and the dimension of the Sobol sequence is two ($t = 2$). In each second stage problems, two elements f_i, R_i, U_i and two elements in each block of W_i and T_i (altogether 22 elements) are perturbed using the two elements of ω_i. The 2-norm constraint is not randomized.

Problems MCL13–MCL63 have dimensions similar to those of problems MCL1–MCL6, however, these problems were generated using a 33-dimensional Sobol sequence ($t = 33$) and setting $p_{\%} = 50\%$. Three dimensional subvectors of the 33-dimensional Sobol points (altogether 11 distinct subvectors) are used to perturb three elements of f_i, R_i, U_i and three elements in each block of W_i and T_i. Thus these problems are expected to be random in a

different way than MCL1–6.

Problems MCRS1–6 have increasing size of the second stage cone and decision variables y_i. Here Problem MCRS1 is identical to Problem MCR2. Problems MCRF1–6 have increasing size of the first stage cone and decision variable x. Here Problem MCRF1 is identical to Problem MCR2. These problems were generated using a 2-dimensional Sobol sequence ($t = 2$) and setting $p_{\%} = 5\%$. When referring to the randomly generated problems we use `name`–N, where N indicates the number of second stage scenarios in the instance.

4. Implementing the Primal Interior Decomposition Algorithm

In this section we discuss a practical implementation of primal interior decomposition algorithms. This implementation uses the knowledge that the underlying problem is a stochastic program. The central idea is to estimate the properties of a large-scale stochastic problem with the help of problems with few scenarios. This information is used to device implementation heuristics. In the rest of this paper we assume that the probabilities π_i (weights) in the TSSCP (1.1– 1.3) are $1/K$.

4.1 Initialization

We need an interior starting point that is feasible for the first stage, and for which all second stage problems have feasible interior solutions. For this purpose we consider the feasibility barrier centering problem

$$\max_{(x,s_1)\in\mathcal{F}^0, s_1\in\mathcal{K}_1^+} \ln\det(s_1)$$

and its KKT conditions:

$$\begin{aligned} &Dx = d, \\ &Ax + s_1 = c, \\ &D^T\tau + A^T\upsilon = 0, \\ &P(s_1^{1/2})\upsilon = \iota_1, \\ &s_1, \upsilon \in \mathcal{K}_1^+. \end{aligned} \tag{4.1}$$

Equations (4.1) are solved to a desirable accuracy. An infeasible primal-dual interior point method based on the description in [26, 27] is used. If such a solution is not found within a desirable accuracy then the problem is 'infeasible'. Otherwise, the iterate at which the primal-dual algorithm terminates is taken as a solution. Some details of this implementation are given in Section 4.2. The properties of interior point methods suggest that if an interior feasible solution is available, then the infeasible primal-dual interior method will provide such a solution [22, 11]. The solution of (4.1) may not provide a starting point for which all second stage problems are feasible if the problem does not has full recourse. We use an artificial variable (y_i^a) for the second stage problem to ensure a full recourse. The second stage problems with the artificial variable take the form:

Problem	Equality(F,S)	LINC(F,S)	SOC(F,S)	SDC(F,S)	Dim(x, y_i)
MCR1	3,3	2,2	[11, 5][11, 5]	[16, 16][16, 16]	10,10
MCR2	3,3	3,3	[21,10][21,10]	[16, 25][16, 25]	20,20
MCR3	3,3	5,5	[31, 15][31, 15]	[16, 36][16, 36]	30,30
MCR4	3,3	5,5	[51,25][51,25]	[36, 64][36, 64]	50,50
MCR5	3,3	10,10	[101, 50][101, 50]	[100, 81][100, 81]	100,100
MCR6	3,3	15,15	[151, 75][151, 75]	[100, 169][100, 169]	150,150
MCL1	3,3	2,2	[11, 5][11, 5]	[16, 16][16, 16]	10,10
MCL2	3,3	3,3	[21,10][21,10]	[16, 25][16, 25]	20,20
MCL3	3,3	5,5	[31, 15][31, 15]	[16, 36][16, 36]	30,30
MCL4	3,3	5,5	[51,25][51,25]	[36, 64][36, 64]	50,50
MCL5	3,3	10,10	[101, 50][101, 50]	[100, 81][100, 81]	100,100
MCL6	3,3	15,15	[151, 75][151, 75]	[100, 169][100, 169]	150,150
MCH1	3,3	2,2	[11, 5][11, 5]	[16, 16][16, 16]	10,10
MCH2	3,3	3,3	[21,10][21,10]	[16, 25][16, 25]	20,20
MCH3	3,3	5,5	[31, 15][31, 15]	[16, 36][16, 36]	30,30
MCH4	3,3	5,5	[51,25][51,25]	[36, 64][36, 64]	50,50
MCH5	3,3	10,10	[101, 50][101, 50]	[100, 81][100, 81]	100,100
MCH6	3,3	15,15	[151, 75][151, 75]	[100, 169][100, 169]	150,150
MCL13	3,3	2,2	[11, 5][11, 5]	[16, 16][16, 16]	10,10
MCL23	3,3	3,3	[21,10][21,10]	[16, 25][16, 25]	20,20
MCL33	3,3	5,5	[31, 15][31, 15]	[16, 36][16, 36]	30,30
MCL43	3,3	5,5	[51,25][51,25]	[36, 64][36, 64]	50,50
MCL53	3,3	10,10	[101, 50][101, 50]	[100, 81][100, 81]	100,100
MCL63	3,3	15,15	[151, 75][151, 75]	[100, 169][100, 169]	150,150
MCRS1	3,3	3,3	[21,10][21, 10]	[16, 25][16, 25]	20,20
MCRS2	3,3	3,5	[21,10][31,15]	[16, 25][16, 36]	20,30
MCRS3	3,3	3,5	[21,10][51, 25]	[16, 25][16, 36]	20,30
MCRS4	3,3	3,10	[21,10][101,50]	[16, 25][36, 64]	20,50
MCRS5	3,3	3,15	[21,10][151, 75]	[16, 25][100, 81]	20,100
MCRS6	3,3	3,25	[21,10][251, 200]	[16, 25][100, 169]	20,150
MCRF1	3,3	3,3	[21, 10][21,10]	[16, 25][16, 25]	20,20
MCRF2	3,3	5,3	[31,15][21,10]	[16, 36][16, 25]	30,20
MCRF3	3,3	5,3	[51, 25][21,10]	[16, 36][16, 25]	30,20
MCRF4	3,3	10,3	[101,50][21,10]	[36, 64][16, 25]	50,20
MCRF5	3,3	15,3	[151, 75][21,10]	[100, 81][16, 25]	100,20
MCRF6	3,3	25,3	[251, 200][21,10]	[100, 169][16, 25]	150,20

Table 2: Description of the random test problems

$$
\begin{aligned}
max \quad & f_i^T y_i - \frac{1}{2} y_i^T H_i y_i - M y_i^a \\
s.t. \quad & R_i y_i + U_i x = z_i, \\
& W_i y_i + T_i x + s_{2,i} - y_i^a \iota_{2,i} = h_i, \\
& y_i^a \geq 0, s_{2,i} \in \mathcal{K}_{2,i},
\end{aligned} \tag{4.2}
$$

where M is a sufficiently large constant to ensure that $y_i^a \to 0$, as a solution of TSSCP is approached. It is easy to construct a feasible solution of (4.2), which is taken as a starting point for the centering problem (2.11) defined for (4.2).

We still have the problem of identifying a proper value of M. We achieve this in the preprocessing phase. We experiment with an approximation of the original large-scale problem with a small number of scenarios (e.g., $N = 100$) with different values of M, which is taken as a multiple of the infinite norm of the problem data. The smallest value of M that successfully reduces y_i^a below 10^{-10} for all sampled problems is multiplied by a constant for later use. This constant is 10 for the experiments reported in this paper. This strategy allows us to work with values of M that are not unnecessarily large. Taking very large values of M may cause numerical difficulties. Although this approach is heuristic it has worked successfully in all our computations. One may think about adjusting the value of M dynamically over the course of the algorithm.

4.1.1 Initial Warm Start

When solving different small sampled problems in the beginning, the first stage solution from the first sampled problem is used to warm start all subsequent problems.

4.1.2 Selection of the Starting Barrier Parameter

Starting with a μ that is too small may take too many inner iterations to reach the initial centered point, whereas, choosing an unnecessarily large starting μ increases the number of outer iterations. We use information from the preprocessing phase to identify a suitable starting μ. When solving a sampled problem we identify the value of μ that gives only a single digit of accuracy in this problem. Solving this problem to optimality also gives us the magnitude of the optimal objective. Let us denote the objective identified in this step by z. Let ν be the order of the first stage cone $\mathcal{K}_1$. We let $\hat{\mu}^1 = z/\nu$. This value of $\hat{\mu}^1$ is expected to correspond to a solution with zero to one digit of accuracy in the objective value, which can be verified by building a confidence interval around the barrier function value. Next we try different values of μ $(0.1\hat{\mu}^1, 10\hat{\mu}^1)$ as the starting μ, and record the number of inner iterations required to solve the problem. We select a value of μ that requires least number of inner iterations to get an optimal solution starting from that value. The corresponding solution is used to start the main algorithm for TSSCP.

4.2 Solution of the Second Stage Centering Problems

Each inner iteration of our algorithm solves second stage centering problems defined after updating x and/or μ. It is neither necessary nor possible to exactly solve second stage centering problems. In the following (4.3) ensures sufficient feasibility in the solution, and (4.4) ensures proximity (controlled by parameter ν) to the central path.

$$\|R_i y_i + U_i x - z_i\| \leq \texttt{tol}_{\texttt{feas}},\ \|W_i y_i + T_i x + s_{2,i} - h_i\| \leq \texttt{tol}_{\texttt{feas}},$$
$$\|W_i^T \lambda_i + R_i^T \gamma_i + H_i y_i - f_i\| \leq \texttt{tol}_{\texttt{feas}} \tag{4.3}$$
$$\|P(\lambda_i^{1/2}) s_{2,i} - \mu\iota\| \leq \nu\mu. \tag{4.4}$$

We start from the solution of the second stage problem available at the end of the previous iteration. These starting points are no longer feasible or centered. We employ an infeasible primal-dual interior point method to solve the second stage centering problems. The primal-dual method implements the Nesterov-Todd [21, 28] scaling to symmetrize the KKT system, and take steps along the Newton direction for recentering. A second stage line search is performed for computing the step length as follows.

If a full Newton step (length 1) is feasible, we give priority to restoring primal and dual feasibility and take this step. When a full Newton step is not feasible, we choose a step-length that minimizes $\|\xi\| := \|\xi_{feas}, \xi_{cent}\|$, where $(\xi_{feas} = R_i y_i + U_i x - z_i, W_i y_i + T_i x + s_{2,i} - h_i, W_i^T \lambda_i + R_i^T \gamma_i + H_i y_i - f_i)$ is the vector of primal and dual infeasibilities, and $\xi_{cent} = (P(\lambda_i^{1/2}) s_{2,i} - \mu\iota)$ is the residual vector in the complementarity condition (4.4). The derivative of $\|\xi\|^2$ is a third order polynomial which has one or three real roots. Let ϱ denote the maximum step-length that we can take without violating the conic constraints on the primal slack $s_{2,i}$ and on the dual multipliers (λ_i) associated with the primal constraints. If there are no real roots less than ϱ, ϱ is the optimal step length. If there is only one real root less than ϱ, it is the optimal step length. If all three roots are real and less than ϱ, either the smallest or the largest root is the optimal step length. When ϱ is the optimal step length we take a slightly shorter step ($0.9^*\varrho$) to ensure that primal and dual iterates stay far away from the boundary. In the results reported in this paper no attempts are made to take different steps along primal and dual directions, and a predictor-corrector strategy is not used.

Table 3 gives computational results for Problem MCR3 with 100 second stage scenarios using $\gamma = 0.1$, and different combinations of β and ν. We use $tol_{feas} = 10^{-9}$ in condition (4.3). The values of β used for results in this table are chosen to ensure closeness to the first stage central path. Larger values of β do not produce stable performance as seen from the results reported in Tables 3 and 4. For these results the algorithm is terminated when the relative improvement in the objective value at two successive major iterations is less than eight digits. It usually corresponds to a value of μ between 10^{-7} to 10^{-8}. The number of inner iterations taken by the algorithm are given in Table 3 if a problem is successfully solved to eight digits of accuracy. For all runs reported in Table 3 we used a relatively accurate line search to

avoid any misinterpretations. In all the runs performing line search, the search is terminated when we have three digits of accuracy in the step length parameter. The iteration counts are indicated by "–" when the implementation fails to achieve eight digits of precision. There are several reasons for this failure. For some problems the Cholesky factorization method in MATLAB becomes unstable. For other problems the solution does not has eight digits of accuracy, even though the change in the objective between two successive major iterations is less than eight digits. This may be due to the fact that for $\beta = 5$ the neighborhood of the central path may be too wide to correctly measure termination based on either relative improvement or barrier parameter value criterion.

When warm starting from the previous solution, typically only one (occasionally two) second stage iteration is required. Furthermore, the number of inner iterations of the primal decomposition algorithm remain unchanged for ν in a very large range. Computationally we observe that typically the termination conditions (4.4) are satisfied for much smaller values of $\texttt{tol}_{\texttt{feas}}$ since Newton steps are very effective from a warm start solution. Furthermore, from Table 3 we observe that the average number of iterations required to recenter the second stage grows only slightly for smaller values of ν.

4.3 First Stage Centering and Barrier Reduction Rate

Table 4 gives computational results for Problem MCR3-100 for $\nu = 1$, and using different combinations of β and γ. The results in Table 4 show that the total number of inner iterations increase for large values of γ (μ is decreased slowly). Also when μ is reduced too aggressively the total number of inner iterations tend to increase indicating that recentering becomes more difficult in this case. In our experiments the choices $\gamma = 0.1$ and $\beta = 1$ give good results for all the problems.

4.4 First Stage Step Length

The analysis of long-step primal interior point algorithms, such as the one given in [18, 20, 30] is based on taking fixed-length steps along the Newton direction. The choice of this step length is given as $\theta = (1 + \delta)^{-1}$, where δ is the local norm of the Newton direction, as explained in Sections 2.1 and 2.2. This step length is usually very conservative, and there is a need to develop a step length selection strategy that is more efficient. The strategy of taking a constant step to the boundary, which works well for the primal-dual algorithms [14, 16] is not always stable in the current setting. Also, in practice we can not perform an elaborate line search with back-tracking, since each barrier function evaluation (or its derivative evaluation) requires solution of all second stage scenarios. This is almost equal to the cost of computing the first stage direction afresh. Hence, we need a heuristic that avoids unnecessary barrier function evaluations while not increasing the total number of inner iterations significantly.

The following step length selection strategy has worked well in our computational experi-

$\gamma = 0.1$	ν			
β	0.01	0.1	1	10
$(2-\sqrt{(3)})/2$	31 1.09	31 1.01	30 1.00	31 0.99
0.5	24 1.11	26 1.01	24 1.00	25 1.00
1	21 1.10	21 1.01	22 1.00	22 1.00
3	20 1.10	19 1.01	20 1.00	19 1.00
5	- -	- -	- -	- -

Table 3: Number of inner iterations and average iterations for recentering second stage problems for Problem MCR3-100 using accurate (3 digits) line search

$\nu = 1$	γ			
β	0.05	0.1	0.25	0.5
$(2-\sqrt{(3)})/2$	29 4.33	30 3.86	36 2.83	56 2.35
0.5	26 3.83	24 3.14	31 2.42	50 2.09
1	24 3.50	22 2.71	27 2.08	49 2.04
3	- -	20 2.43	26 2.00	26 1.04
5	- -	- -	25 1.92	25 1.00

Table 4: Number of inner iterations and average inner iterations per outer iteration for Problem MCR3-100 using accurate (3 digits) line search

ments: $\theta = \min\{\alpha(1+\delta)^{-1}, 1, 0.9\varrho\}$, where ϱ is the maximum step to the boundary of the feasible set, and α is a scalar constant. We take this step, and check the value of barrier function at the new point. This evaluation of the barrier function is combined with the computation of the new Newton direction, if the step is accepted. Hence, there is no efficiency loss in this function evaluation. We ensure that the barrier function has reduced sufficiently at the new point, which is almost always the case. In a few rare occasions, when the barrier function is not reduced sufficiently we can backtrack by aggressively reducing α. In our computations when this happened we simply set $\alpha = 1$, i.e., in this case we take $\theta = (1+\delta)^{-1}$. This has always reduced the barrier function to a desirable amount. The use of the parameter α is justified through empirical observations, which indicate that when an iterate is close to the boundary the optimum step length is a multiple of $(1+\delta)^{-1}$. Such results are shown in Table 5 for problem MCR4-100. We use $\alpha = 4$ in our actual computations. The results in Table 6 show that the choice of parameter α is important in reducing the total number of inner iterations. Note that the number of inner iterations are slightly larger for both small and large values of α. This is because for a small value of α the step is conservative, while for a large value of α usually the step is so large that we take 0.9ϱ as a step length instead of $\alpha(1+\delta)^{-1}$. Reasons for failure indicated in these tables are similar to those discussed before. While analyzing 100 scenario problems of different size, and randomness we find that the heuristic step-length strategy performs well, usually, within 10–20% of the results obtained from an 'exact' line search. The 'exact' line search is performed using bisection method and it is terminated when the interval of uncertainty is reduced to 10^{-3}. The worst offenders (about 25% worse) were problems MCRF14–MCRF6 (see Table 10) suggesting that further refinements may provide improvements.

5. Basic Behavior of the Primal Decomposition Algorithm

5.1 Comparison with a Direct Method

Since we can formulate TSSCP as a large conic programming problem (extensive formulation), one can use a primal-dual interior point method for solving the extensive formulation. SeDuMi [25] is a popular solver that can be used to directly solve the extensive formulations of the problem we generate. Table 9 gives a comparison of the total number of iterations taken by the primal decomposition algorithm (for $\alpha = 4, \beta = 1, \gamma = 0.1, \nu = 1$) and those taken by SeDuMi 1.05 [25] for an extensive formulation of the problem. In order to compare the performance of primal decomposition algorithm with that of SeDuMi, it is more appropriate to consider the number of inner iterations taken by the primal decomposition algorithm with the number of iterations taken by SeDuMi to achieve the same accuracy in the solution. SeDuMi implements Mehrotra's predictor-corrector method. The work required to compute one predictor step in SeDuMi is roughly equivalent to computing a Newton direction in an inner iteration of the primal decomposition algorithm. A comparison of cpu times will be misleading because of a variety of reasons. These codes are written using different linear algebra libraries (SeDuMi uses its C code, while we depend on MATLAB). Because

iter #	μ^k	$(1+\delta)^{-1}$	max feasible step-length	line search step-length	$\frac{\text{line search step-length}}{(1+\delta)^{-1}}$
1	1.00E+00	0.02	0.07	0.07	3.50
2	1.00E+00	0.21	1.65	1.46	6.95
3	1.00E+00	0.28	1.95	1.48	5.29
4	1.00E+00	0.5	4.14	1.38	2.76
5	1.00E-01	0.08	0.22	0.22	2.75
6	1.00E-01	0.21	0.77	0.6	2.86
7	1.00E-01	0.41	2.3	1.35	3.29
8	1.00E-01	0.56	10.59	2.08	3.71
9	1.00E-02	0.04	0.1	0.09	2.25
10	1.00E-02	0.13	0.37	0.35	2.69
11	1.00E-02	0.38	2.34	1	2.63
12	1.00E-02	0.64	17.21	1.4	2.19
13	1.00E-03	0.03	0.09	0.09	3.00
14	1.00E-03	0.2	0.43	0.38	1.90
15	1.00E-03	0.47	2.54	0.9	1.91
16	1.00E-03	0.85	146.46	1.1	1.29
17	1.00E-04	0.03	0.1	0.1	3.33
18	1.00E-04	0.42	1.42	0.73	1.74
19	1.00E-04	0.82	20.69	1.04	1.27
20	1.00E-05	0.03	0.1	0.1	3.33
21	1.00E-05	0.62	3.38	0.86	1.39
22	1.00E-06	0.03	0.1	0.1	3.33
23	1.00E-06	0.61	2.55	0.81	1.33
24	1.00E-07	0.03	0.1	0.1	3.33
25	1.00E-07	0.57	1.73	0.73	1.28
26	1.00E-08	0.03	0.1	0.1	3.33
27	1.00E-08	0.51	1.13	0.6	1.18

Table 5: Behavior of feasible and optimal first stage step-lengths for problem MCR4-100 ($\beta = 1,\ \nu = 1,\ \gamma = 0.1,$)

		β				
problem	α	$\frac{2-\sqrt{3}}{2}$	0.5	1	3	5
MCR1-100	2	37	36	35	25	-
	4	35	28	31	25	-
	∞	35	32	27	31	-
	line search	-	26	22	20	-
MCR2-100	2	34	31	31	-	-
	4	34	31	30	-	-
	∞	35	32	32	-	-
	line search	-	28	27	23	22
MCR3-100	2	-	31	25	-	-
	4	35	31	25	-	-
	∞	35	31	25	-	-
	line search	30	24	22	20	-
MCR4-100	2	48	40	39	29	-
	4	39	37	32	-	-
	∞	40	38	32	-	-
	line search	-	32	29	-	-
MCR5-100	2	62	56	54	46	-
	4	52	43	40	-	-
	∞	54	45	43	-	-
	line search	47	42	38	-	-
MCR6-100	2	71	55	51	43	-
	4	55	46	43	-	-
	∞	57	48	45	-	-
	line search	49	41	35	-	-

Table 6: Heuristic step-length selection: Number of inner iterations for problems MCR1-MCR6 for varying values of β, ($\gamma = 0.1,\ \nu = 1$)

$\nu = 1,\ \alpha = 4$	γ			
β	0.05	0.1	0.25	0.5
$(2-\sqrt{3})/2$	35 5.83	33 4.43	58 4.83	99 4.13
0.5	30 5.00	28 3.50	46 3.83	73 3.04
1	21 3.50	24 3.14	34 2.83	51 2.13
3	- -	- -	24 2.00	26 1.08
5	- -	- -	- -	- -

Table 7: Heuristic step-length selection: Number of inner iterations and average iterations taken for reoptimizing second stage problems for Problem MCR3-100

$\gamma = 0.1,\ \alpha = 4$	ν			
β	0.01	0.1	1	10
$(2-\sqrt{3})/2$	33 1.35	33 1.12	33 1.03	33 1.03
0.5	28 1.40	28 1.14	28 1.03	28 1.03
1	24 1.48	24 1.17	24 1.04	24 1.04
3	- -	- -	- -	- -
5	- -	- -	- -	- -

Table 8: Heuristic step-length selection: Number of inner iterations and average inner iterations per outer iteration for Problem MCR3-100

	Our Implementation		SeDuMi		
problem	objective	itr	primal obj	dual obj	itr
MCR1-100	1.631426527	31	1.631426587	1.631426582	22
MCR2-100	3.384065478	30	3.384065491	3.384065487	24
MCR3-100	204.2468748	25	204.2468761	204.2468760	22
MCR4-100	21.08956769	32	21.08956771	21.08956770	25

Table 9: Comparison of our implementation ($\gamma = 0.1, \beta = 1, \nu = 1$, and $\alpha = 4$) to SeDuMi

MATLAB is an interpreted language, it must interpret every line in a for loop every time it goes through it and this makes MATLAB very slow in handling loops. Furthermore, when solving problems we generate scenarios on the fly, while SeDuMi reads an extensive formulation which becomes memory intensive to generate and write in a text file to be readable by SeDuMi. We were unable to feed larger problems to SeDuMi for this reason. For the problems that were solved by both SeDuMi and the primal decomposition algorithm, results indicate that both algorithms correctly achieve eight digits of accuracy. SeDuMi took 10 to 30% fewer iterations. Recall that the corrector step in the predictor-corrector method reduces the total number of iterations by 20 to 50%. Hence, it appears that for a "Newton" step based algorithm the total number of iterations taken by the primal decomposition algorithm are comparable.

5.2 Performance with Increasing Problem Size

Table 10 gives the number of iterations required by the primal decomposition method with increasing size of cones in the problem (in short called *problem size*). The problems with increasing second stage scenario size are MCRS1–MCRS6, and those with increasing first stage scenario size are MCRF1–MCRF6. These results show that the number of inner iterations are not significantly affected by the size of the second stage problems, whereas, the number of inner iterations show an early upward trend but become stable as size of the first stage problem grows. This indicates that the complexity of the algorithm depends on the size of the first stage problems, however, the iteration growth is not very rapid, which is generally the case with interior point methods.

5.3 Performance of the Algorithm with Increasing Randomness

We generated problems with increasing amount of randomness in the problem data by varying the value of $p_{\%}$. Recall that for Problems MCR1–6 $p_{\%} = 5\%$, for Problems MCL1–MCL6 $p_{\%} = 50\%$, and for Problems MCH1–MCH6 $p_{\%} = 500\%$. We also generated problems MCL13–MCL63 where the dimension of the underlying random variable is 33 and $p_{\%} = 50\%$. Results for 100-scenario instances of these problems are given in Table 11. When comparing the number of inner iterations for these problems with those for problems MCR1–6 no significant trend is observed with increasing randomness. From these results infer that the number of inner iterations required by the algorithm is not dependent of the randomness in

Problem	heur	line search	Problem	heur	line search
MCRS1-100	30	27	MCRF1-100	30	27
MCRS2-100	27	25	MCRF2-100	30	28
MCRS3-100	23	22	MCRF3-100	31	30
MCRS4-100	22	21	MCRF4-100	46	37
MCRS5-100	24	21	MCRF5-100	51	40
MCRS6-100	32	27	MCRF6-100	46	36

Table 10: Number of inner iterations with increasing problem size ($\gamma = 0.1, \beta = 1, \nu = 1$, and $\alpha = 4$)

Problem	heur	line search	Problem	heur	line search	Problem	heur	line search
MCL1-100	27	23	MCH1-100	26	24	MCL13-100	29	24
MCL2-100	36	29	MCH2-100	31	26	MCL23-100	31	26
MCL3-100	24	22	MCH3-100	25	25	MCL33-100	26	24
MCL4-100	32	29	MCH4-100	31	29	MCL43-100	32	30
MCL5-100	40	39	MCH5-100	38	36	MCL53-100	43	38
MCL6-100	43	37	MCH6-100	39	38	MCL63-100	40	37

Table 11: Number of inner iterations as problem randomness increases ($\gamma = 0.1, \beta = 1, \nu = 1$, and $\alpha = 4$)

the problem data.

5.4 Performance of the Algorithm with Increasing Number of Scenarios

In Table 12 we give the number of inner iterations taken when solving Problems MCR3 and MCR4 with increasing number scenarios. Implementation details for problems with larger than 100 scenarios are discussed in Section 7. A comparison of these results shows no increasing trend in the required number of inner iterations with increasing number of scenarios.

	# scenarios							
	10		100		1,000		10,000	
Problem	heur	line search	heur	line search	heur	line search	heur	line search
MCR2	31	27	30	27	32	26	32	26
MCR3	25	21	25	22	25	22	25	22

Table 12: Number of inner iterations as number of scenarios increases ($\gamma = 0.1, \beta = 1, \nu = 1$, and $\alpha = 4$)

6. Adaptive Scenario Addition Heuristic

The primal decomposition algorithm naturally allows adaptive addition of scenarios as the algorithm progresses. It is important to explore this possibility since adaptive addition of scenario may lead to significant computational savings. Recall that when the number of scenarios is fixed (no adaptive scenario addition) we move along a fixed first stage central path as μ is reduced. However, in the adaptive algorithm the first stage central path shifts when new scenarios are added. As the results in Section 7 show, we need to be careful while adding scenarios adaptively. This is because if large number of scenarios are added for a small value of the barrier parameter, the shift in the primal central path caused by this scenario addition may result in a poorly conditioned problem. This, in turn, may require significantly more inner iterations to recenter or even cause numerical breakdowns.

In this section we develop a heuristic that addresses this issue. The key idea behind our heuristic is to add the number of scenarios in such a way that scenario addition and reducing the value of barrier parameter in the algorithm have similar impact on shifting the first stage center. For this purpose we device methods for estimating the change in the barrier objective value caused by scenario addition, and also suggest ways to measure the change in the barrier objective resulting from reducing μ. The assumptions made while developing this heuristic are justified through empirical evidence.

We add new scenarios when μ is reduced in Step 2 of Algorithm 1. Our heuristic aims to maintain

$$|\eta_{N_k}(\mu^k, x_{N_k}(\mu^k)) - \eta_{N_{k-1}}(\mu^k, x_{N_{k-1}}(\mu^k))| \approx \Delta\eta_{N_k}(\mu^k),$$

where $\eta_{N_k}(\mu^k, x_{N_k}(\mu^k))$ is the optimum objective of the sample-average log-barrier problem with N_k samples for $\mu = \mu^k$ and

$$\Delta\eta_{N_k}(\mu^k) := \eta_{N_k}(\mu^k, x_{N_k}(\mu^k)) - \eta_{N_k}(\mu^k, x_{N_k}(\mu^{k-1})). \tag{6.1}$$

Note that the sample-average function is a random variable when scenarios are generated randomly (Monte Carlo). We represent the corresponding random variable by $\hat{\eta}_N(\cdot,\cdot)$. Every randomly generated batch of N-samples yields a different realization $\eta_N(\cdot,\cdot)$ of $\hat{\eta}_N(\cdot,\cdot)$.

A justification of the heuristic idea is as follows. In the adaptive implementation of our algorithm, at the beginning of the k-th outer iteration we decrease the barrier parameter from μ^{k-1} to $\mu^k = \gamma\mu^{k-1}$, and simultaneously increase the number of scenarios from N^{k-1} to N^k. In general, both of these actions may increase the distance of current point x to the central path. Decreasing μ moves away the μ-center, whereas, increasing the number of scenarios may shift the central path. Since experimentally we know that it takes only a few iterations to recenter after decreasing μ in the case where the number of scenarios is fixed, by keeping the impact of adding number of scenarios of the same order as the impact of decreasing μ, we expect that the number of iterations required to converge to the new center with added

scenarios will be of the same order.

6.1 Confidence Interval Estimation

Various estimations required to build our heuristic are motivated by the following theorem from Shapiro [24].

Theorem 6.1 *Consider the stochastic program* $\inf_{x\in X} \int_\Omega g(x,\omega)P(d\omega)$ *and let* z^* *be its optimal objective. Let* $\hat{z}_N$ *denote the optimal objective of the sample average problem* $\inf_{x\in X} N^{-1}\sum_{i=1}^N g(x,\omega^i)$*, where* ω^i *are independent. Suppose* X *satisfies the following conditions:*
(i) $g(x,\cdot)$ *is measurable for all* $x \in X$*;*
(ii) there exists some function $f:\ \Omega \to \mathbb{R}$ *such that* $\int_\Omega |f(\omega)|^2 P(d\omega) < \infty$*, and*

$$|g(x_1,\omega) - g(x_2,\omega)| \le f(\omega)|x_1 - x_2|,$$

for all $x_1, x_2 \in X$*;*
(iii) for some $\bar{x} \in X$*,* $\int_\Omega g(\bar{x},\omega)dP(\omega) < \infty$*, and* $E\{g(x,\omega)\} = \int_\Omega g(x,\omega)dP(\omega)$ *has a unique minimizer* $x^* \in X$*.*
Then $\sqrt{N}(\hat{z}_N - z^*)$ *converges in distribution to normal distribution* $\Phi(0, Var[g(x^*,\xi)])$*, where* $Var[g(x^*,\omega)] = \int_\Omega g(x^*,\omega)^2 dP(\omega) - E\{g(x^*,\omega)\}^2$*.*

Recall that since the log-barrier recourse function $\eta_N(\cdot,\cdot)$ is concave, it has a unique maximizer. Thus it satisfies the uniqueness condition in (*iii*) of Theorem 6.1. The assumption that first and second stage problems are feasible and bounded ensures the remaining regularity conditions in Theorem 6.1. Hence, when the random variable governing the stochastic program has a continuous distribution and the second stage problems are generated using Monte-Carlo sampling, Theorem 6.1 applies. In this case, we have $Var[\hat{\eta}_N(\mu, x_N(\mu))] = \sigma^2(\mu, x(\mu))/N$, where $\sigma^2(\mu, x(\mu)) = \mathrm{Var}[\rho(\mu, x(\mu), \tilde{\omega})]$ [9]. Here, $\rho(\cdot,\cdot,\tilde{\omega})$ is the log-barrier counterpart of $\rho(\cdot,\tilde{\omega})$ in (3.4). Note also that we used the observation that the randomness of $\hat{\eta}_N(\mu, x_N(\mu))$ is due to the log-barrier recourse function. Theorem 6.1 shows that $\sqrt{N}(\hat{\eta}_N(\mu, x_N(\mu)) - \eta(\mu, x(\mu)))$ converges in distribution to a normal random variable with a mean of zero and variance of $\sigma^2(\mu, x(\mu))$.

For any batch of N samples, this result provides a statistical bound on $|\eta_N(\mu, x_N(\mu)) - \eta(\mu, x(\mu))|$ that holds with a certain confidence:

$$|\eta_N(\mu, x_N(\mu)) - \eta(\mu, x(\mu))| \lesssim z_\alpha \frac{s_N(\mu, x_N(\mu))}{\sqrt{N}}.$$

Here $s_N(\mu, x_N(\mu))$ is a N-sample approximation of the sample variance estimator of $\sigma^2(\mu, x(\mu))$, and the value of z_α provides the level of confidence we want in the bound. In particular, z_α satisfies $P\{\Phi(0,1) \le z_\alpha\} = 1 - \alpha$. The statistical and approximate nature of this bound is

indicated by using the symbol $\widetilde{\leq}$.

We use a quasi-Monte-Carlo (QMC) (Sobol' sequence) technique instead of Monte-Carlo to generate second stage sample since it is known that better variance convergence rates are possible by using QMC techniques. For example, the empirical results in Glasserman [9] and our own experiments suggest that Sobol' sequence gives much faster convergence than a Monte-Carlo sampling. Unfortunately, the convergence rate theory for these techniques is not yet fully developed in the stochastic programming setting. For example, we do not know the asymptotic distribution of $\hat{\eta}_N(\mu, x_N(\mu)) - \eta(\mu, x(\mu))$ for scenarios generated using Sobol' sequence, and if an analogue of Theorem 6.1 holds. Heuristically, we expect that when scenarios are generated using Sobol' sequence

$$|\eta_N(\mu, x_N(\mu)) - \eta(\mu, x_N(\mu))| \widetilde{\leq} z_\alpha RMS_N(\mu, x_N(\mu)),$$

where $RMS_N(\mu, x_N(\mu))$ represents the "root-mean-square" error of $\hat{\eta}_N(\mu, x_N(\mu))$, and z_α provides a confidence interval as before. The root-mean-square error is calculated by randomizing the Sobol' sequence as follows.

Let S_i represent the Sobol' sequence used to generate the second stage scenarios. To generate j-th set of N scenarios we generate a random (Monte-Carlo) vector U_j, uniformly distributed over the unit hypercube of appropriate dimension and generate a new sequence: $\tilde{S}_i^{\,j} = (S_i + U_j)\ mod\ 1,\ i = 1, \ldots, N$. The problem corresponding to $\tilde{S}_i^{\,j}$ is generated as described in Section 3.2. This process is repeated for $j = 1, \ldots m$ to generate m random problems each having N scenarios. Let $\eta_{N(j)}(\mu, x_{N(j)}(\mu))$ denote the log-barrier objective of the j-th problem for a given μ and $x_{N(j)}(\mu)$. Now

$$RMS_N(\mu, x_N(\mu)) := \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(\eta_{N(j)}(\mu, x_{N(j)}(\mu)) - \bar{\eta}_N)^2},\ \text{where } \bar{\eta}_N := \frac{1}{m}\sum_{j=1}^{m}\eta_{N(j)}(\mu, x_{N(j)}(\mu)). \tag{6.2}$$

Calculation of $RMS_N(\mu, x_N(\mu))$ for large N requires significant work which is not practical in an algorithmic setting. However, in QMC we expect RMS to decrease by $O(1/N^\kappa)$, where $1 \geq \kappa > .5$ [9, Chapter 5]. Note that for Monte-Carlo $\kappa = 0.5$. The knowledge of κ can allow us to approximate $RMS_N(\mu, x_N(\mu))$ for large N, by scaling $RMS_{N'}(\mu, x_{N'}(\mu))$ calculated for some a small value N'. However, even this computation may be expensive, since computation of $RMS_{N'}(\mu, x)$ requires several (m) replications of problems with small sample size. With this in mind in our implementation we perform the computation of κ once off-line, while we further approximate the computation of $RMS_N(\mu, x_N(\mu))$ by taking (possibly biased) estimation

$$RMS_N(\mu, x_N(\mu)) \approx \hat{\sigma}_N(\mu, x_N(\mu))/N^\kappa,$$

where

$$\hat{\sigma}_N^2(\mu, x_N(\mu)) := \frac{1}{N-1}\sum_{i=1}^{N}(\rho_i(\mu, x_N(\mu)) - \bar{\rho}_N)^2,\ \text{where } \bar{\rho}_N := \frac{1}{N}\sum_{i=1}^{N}\rho_i(\mu, x_N(\mu)). \tag{6.3}$$

Note that as a consequence of this approximation we now have

$$|\eta_N(\mu, x_N(\mu)) - \eta(\mu, x(\mu))| \stackrel{\sim}{\leq} z_\alpha \hat{\sigma}_N(\mu, x_N(\mu))/N^\kappa,$$

which is similar to the approximation for the Monte-Carlo case expect that we now have a tighter inequality due to $1/N^\kappa$. With the above discussion the problem of estimating a bound on $|\hat{\eta}_N(\mu, x_N(\mu)) - \eta(\mu, x(\mu))|$ is now reduced to the problem of estimating $\hat{\sigma}^2_N(\mu, x(\mu))$ and κ. The discussion in Section 6.2 shows that good approximations of $\hat{\sigma}_N(\mu, x_N(\mu))$ are available cheaply because of the empirically observed properties of the primal central path. Fortunately very accurate values of κ are not required since the adaptive scenario generation heuristic allows a lot of flexibility. The calculation of κ is discussed in Section 6.3.

6.2 Empirical Observations on the First Stage Central Path and the Variance of Second Stage Barrier Objectives

We empirically observe that $\hat{\sigma}_N(\mu, \cdot)$ does not change significantly with μ at solutions on (or near) the first stage central path. Furthermore, it also does not change significantly with increasing value of N. Hence, it is sufficient to compute $\hat{\sigma}_N(\mu, \cdot)$ for a large value of μ and a small sample size N' at $x_{N'}(\mu)$. These empirical observations are illustrated in Table 13. Results in this table are obtained from 100, 1,000 and 10,000-scenario approximations of problem MCR3. We solved each of these problems using the primal decomposition method. We calculated $\hat{\sigma}_N(\mu^k, \hat{x}_N(\mu^k))$ as defined in (6.3) at the approximate first stage μ-center $\hat{x}_N(\mu^k)$ in every outer iteration k of the primal decomposition algorithm.

In the above experiment for each outer iteration k, we also calculated the changes in the value of η_N to observe its behavior. From the results in Table 13 we observe that $\Delta\eta_N(\mu^k)$ scales proportionally to μ and does not change significantly as N increases. This suggests that it is sufficient to solve the sample average approximation to a low accuracy (zero or one digit) to estimate $\hat{\sigma}_{N'}(\mu, \cdot)$, and $\Delta\eta_N(\mu)$ for any μ. Note that this last property is due to the form of the objective in (2.12) (due to the weighting of the second stage barrier term by π_i), and it is not obviously shared by the objective function in (2.3), where the 'contribution' of the barrier grows as more scenarios are added.

6.3 Estimating Convergence Rate of the Root-Mean-Square Error

For problem MCR3 we experimented with N= 10, 50, 250, 1,000 and 5,000 scenario approximations of the TSSCP. We generated $m = 100$ batches of 5000 samples. We identified $\hat{x}^1_{10}(0.001)$, an approximate first stage μ-center of the first 10-scenario problem for $\mu = 0.001$ and then solved all the second stage problems in all batches setting $x = \hat{x}^1_{10}(0.001)$ and $\mu = 0.001$. Let $i(j)$ be the index of the i-th sample in j-th set with N samples. For $j = 1, \ldots, 100$ and $N = 10, 50, 250, 1,000$ and $5,000$ we calculated

$$\eta_{N(j)}(0.001, \hat{x}_{10}(0.001)) = b^T x - \frac{1}{2} x^T G x + \mu \ln \det(s_1) + \frac{1}{N} \sum_{i(j)=1}^{N} \rho_{i(j)}(\mu, x) \Bigg|_{\mu=0.001,\ x=\hat{x}^1_{10}(0.001)}.$$

N	μ^k	$\hat{\sigma}_N(\mu^k, \hat{x}_N(\mu^k))$	$\Delta\eta_N(\mu^k)$
100	1E+0	4.998	-
	1E-1	5.167	9.17E+00
	1E-2	5.210	1.08E+00
	1E-3	5.200	1.13E-01
	1E-4	5.198	1.14E-02
	1E-5	5.198	1.13E-03
	1E-6	5.198	1.14E-04
	1E-7	5.198	1.42E-05
1,000	1E+0	5.470	-
	1E-1	5.639	9.15E+00
	1E-2	5.666	1.08E+00
	1E-3	5.650	1.13E-01
	1E-4	5.647	1.16E-02
	1E-5	5.646	1.17E-03
	1E-6	5.646	1.14E-04
	1E-7	5.646	1.94E-05
10,000	1E+0	5.319	-
	1E-1	5.505	9.16E+00
	1E-2	5.530	1.07E+00
	1E-3	5.506	1.13E-01
	1E-4	5.496	1.15E-02
	1E-5	5.495	1.14E-03
	1E-6	5.495	1.13E-04
	1E-7	5.495	1.66E-05

Table 13: Empirical Properties of the First Stage Central Path

N	$RMS_N(0.001, \hat{x}^1_{10}(0.001))$
10	2.254958E-01
50	5.581944E-02
250	1.150173E-02
1,000	4.230485E-03
5,000	1.309438E-03

Table 14: Number of scenarios vs. RMS

Using these values we obtained the root-mean-square error $RMS_N(0.001, \hat{x}^1_{100}(0.001))$ for $N = 10, 50, 250, 1,000$ and $5,000$. Table 14 gives the root-mean-square errors for problem MCR3 calculated in this way. An average convergence rate of $\kappa = 0.84$ is observed. Note that a different choice of x may give a slightly different rate of convergence, and also one may think about building a confidence interval for the rate of convergence. We have not done this in our experiments, since a crude estimate is sufficient for us. In our numerical experiments on adaptive scenario addition, we used $\kappa = 0.85$ for all randomly generated problems, and $\kappa = 0.7$ for all Markowitz problems. The dimension of the Sobol' sequence used to generate Markowitz problems is greater (20 to 40 vs. 2 or 33 in the randomly generated problems), potentially resulting in a slower convergence rate.

6.4 A Heuristic for Adaptive Scenario Addition

Let μ^3 be the value of barrier parameter after three major iterations in the pre-processing phase and $\hat{x}(\mu^3)$ be a sufficiently well-centered first stage solution for $\mu = \mu^3$. From the discussion in Section 6.1 it follows that

$$\begin{aligned}
& |\eta_{N_k}(\mu^k, \hat{x}_{N_k}(\mu^k)) - \eta_{N_{k-1}}(\mu^k, \hat{x}_{N_{k-1}}(\mu^k))| \\
= \; & |\eta_{N_k}(\mu^k, \hat{x}_{N_k}(\mu^k)) - \eta(\mu^k, x(\mu^k)) + \eta(\mu^k, x(\mu^k)) - \eta_{N_{k-1}}(\mu^k, \hat{x}_{N_{k-1}}(\mu^k))| \\
\lessapprox \; & z_\alpha \left(\frac{\hat{\sigma}_{N_k}(\mu^k, \hat{x}_{N_k})}{N_k^\kappa} + \frac{\hat{\sigma}_{N_{k-1}}(\mu^k, \hat{x}_{N_{k-1}})}{N_{k-1}^\kappa} \right) \\
\lessapprox \; & z_\alpha \left(1 + \left(\frac{N^{k-1}}{N^k} \right)^\kappa \right) \frac{\hat{\sigma}_{N'}(\mu^3, \hat{x}(\mu^3))}{N_{k-1}^\kappa} \\
\lessapprox \; & 2 z_\alpha \frac{\hat{\sigma}_{N'}(\mu^3, \hat{x}(\mu^3))}{N_{k-1}^\kappa}.
\end{aligned}$$

We may now choose N^k such that

$$2 z_\alpha \frac{\hat{\sigma}_{N'}(\mu, \cdot)}{N_{k-1}^\kappa} \;\; = \;\; \Delta \eta_{N_k}(\mu^k). \tag{6.4}$$

Note that the heuristic calculation in (6.4) gives an estimation of the number of scenarios at iteration $(k-1)$ in anticipation of the impact of reduction in the barrier parameter.

Hence, the heuristic anticipates the future and loads sufficiently many scenarios in the current outer iteration so that the central path does not shift dramatically when we load new scenarios in the next outer iteration. In our computations we set $N' = 100$ and used $\Delta\eta_{N_k}(\mu^k) \approx \Delta\eta_{100}(\mu^3)\frac{\mu^k}{\mu^3}$.

7. Computational Results on Adaptive Scenario Addition and Warm-Start

After adding new scenarios we need to solve the corresponding second stage centering problem for the current value of x and μ. In order to exploit the solutions of the scenarios that are already in the problem, we may look for a scenario that is close to the scenario to be added. One possibility is to look for a scenario that is close in terms of problem data to the new scenario. A solution from the neighboring scenario may provide a good starting point. The Sobol' sequences that we used to generate our scenarios do not provide this neighborhood information a priori. Therefore, for all Sobol' point S_k we heuristically identified a neighboring point among from the points $S_1, \ldots, S_{k-1}$. We then used the final iterate from this neighboring scenario for the current value of x and μ as a starting solution for the new scenario. Let us denote this warm-start strategy as warm-start-1.

Our experience with warm-start-1 has been mixed. For easy problems, e.g., when the second stage problems have only linear constraints and linear objectives, we observed that the strategy of using solution from a nearby scenario works. However, for more difficult conic problems warm-start-1 exacerbates the performance. For these problems starting from a well-centered point of a near by scenario, especially at small values of μ, Newton iterations were not able to absorb the infeasibility within a practical number of iterations. In general, our experience is that such warm starts with conic and semidefinite problems are harder for the same amount of perturbation in the problem data.

We also experimented with an alternative approach (warm-start-2), which was more stable in the current setting. In this approach, once we complete the final inner iteration of the first outer iteration, we also solve all the second stage problems that are yet not included in the problem, for the starting $\mu = \mu^1$ and the latest first stage solution $x = x(\mu^1)$. For $\mu = \mu^1$, the warm-start is very efficient and each second stage problem can be solved within a few iterations. We store all these second stage solutions for later use. When a new scenario is added in a subsequent outer iteration k ($k > 1$), we warm-start its solution from the iterate stored for this scenario in the first outer iteration. We proceed in two steps: In the first step, we solve the new problem for $\mu = \mu^1$ and $x = x(\mu^k)$ and in the second step we reduce μ from μ^1 to μ^k.

7.1 Discussion on Randomly Generated Problems

Tables 15–20 present computational results for problems MCR3-10,000, MCL3-10,000, MCH3-10,000, and MCL33-10,000. Run 1 in these tables correspond to an implementation where

all scenarios are loaded from the beginning, Run 2 uses the scenario addition heuristic of Section 6 with $z_\alpha = 1.96$, Run 3 delays addition of these scenarios one major iteration forward, and Run 4 delays addition of scenarios two major iterations forward. Run 5 and Run 6 on MCR3-10,000 are identical to Run 1 and Run 2, except that the algorithm is terminated with a larger value of μ. These runs are included because for MCR3-10,000, using the information from $\hat{\sigma}_{N'}(\mu, \cdot)$, we estimate that a 10,000 scenario problem can produce a "statistical accuracy" (i.e., have a confidence interval that ensures that the objective is accurate to that degree) of about four digits. This accuracy is achieved for $\mu = 10^{-4}$. The additional major iterations, although solve the optimization problem more accurately, they do not improve the quality of the stochastic programming solution. To get an additional digit of "statistical" accuracy in the stochastic programs, we will require between 100,000 to one million scenarios. Solving problems of such sizes is not possible in the MATLAB environment.

We make the following observations based on results in Tables 15–16 for Problem MCR3-10,000. First, we observe that a well designed heuristic is necessary for gaining efficiency for adaptive scenario addition. Note that for problem MCR3-10,000 Run 3 required significantly more iterations and time than Run 2. Run 4, which was more aggressive in delaying scenario addition, could not even solve the problems. In this run at $\mu = 10^{-4}$, after additional scenarios were loaded, the algorithm failed to recenter within 50 inner iterations and terminated unsuccessfully. There is only four digits of accuracy in the solution at this value of μ. Second, the warm start for existing scenario (after an update of x and/or μ) was effective. However, addition of a new scenario took relatively more iterations. The computational savings in Run 2 over Run 1 are about 40%. This is because about half of the iterations in Run 2 are done with all the scenarios. When comparing the cpu times of Run 5 and Run 6 we observe that the cpu time savings are about 55%. Run 7, which aggressively delays scenario addition, performs worse that Run 6.

When analyzing the results in Tables 17– 22 for Problem MCL3-10,000 and MCH3-10,000, we find a repeat of the above observations. Although Run 4 managed to optimize the problem MCL3-10,000, it took about 60% more time compared to Run 1. Run 4 for problem MCH3-10,000 was just as efficient as Run 2. Note that more scenarios are added sooner for problems MCL3-1000 and MCH3-10,000 since they have larger values of $\hat{\sigma}_{N'}(\mu, \cdot)$. Interestingly, for these problems delaying scenario addition by one or two major iterations did not cause the algorithm to fail although it increased the number inner iterations. We think this may be because for a larger value of μ it is easier to recover proximity to the first stage central path. However, for a smaller value of μ the recovery to the central path is harder since in this case the iterates are closer to the boundary of the first stage feasible set.

We observe that the average number of inner iterations taken when adding new scenarios in Run 1 increase slightly from MCR3-10,000, to MCL3-10,000, to MCH3-10,000. This shows that the ability to warm start for an existing scenario depends on the value of $\hat{\sigma_{N'}}(\mu, \cdot)$. As $\hat{\sigma_{N'}}(\mu, \cdot)$ decreases, for a fixed number of scenarios the distance between a given scenario and

its closest neighbor decreases. Same is true if the number of scenarios were to increase. In both cases, quality of final iterates of neighboring scenarios as starting point increases.

7.2 Discussion on Two-Stage Stochastic Markowitz Problems

We solved the two-stage Markowitz problem with 20, 30 and 40 securities setting $\tau_{2,i} = 0.05$ and a 40-security instance of the problem setting $\tau_{2,i} = 0.25$. Decreasing the upper bound $\tau_{2,i}$ on the variation of the portfolio decomposition makes the two-stage Markowitz problem increasingly harder and eventually infeasible.

In Table 24 we report results for the $\tau_{2,i} = 0.05$ case. We performed two runs for each problem. In Run 1, we loaded all scenarios from the beginning and in Run 2 we added them adaptively following the strategies given in Table 23. For all problems, Run 2 uses the scenario addition heuristic of Section 6 with $z_\alpha = 1.96$. With $\tau_{2,i} = 0.05$, warm-start-1 strategy was not stable. This is similar to the experience described in the previous section. Therefore, we performed addition of scenarios according to warm-start-2 strategy. Since scenario additions require about 7 second stage iterations and the algorithm converges to optimality taking considerably less inner iterations than on the randomly generated test problems, here the adaptive scenario addition does not yield significant computational savings.

In Table 26, we report results for $\tau_{2,i} = 0.25$, where warm-start-1 strategy worked successfully. Our experience in this case was positively different. In Run 1 we loaded all scenarios in the first outer iteration. In Run 2 scenarios are added adaptively as discussed in Section 6 with $z_\alpha = 1.96$. In Runs 3, 4, and 5 we added scenarios by warm-starting from the final iterate of a neighboring scenario at latest value of μ. Run 4 delays addition of scenarios one major iteration forward. Run 5 is very aggressive and delays addition of scenarios to the final two outer iterations. In Run 2 computational savings over Run 1 are about 30%. Due to more efficient addition of scenarios, savings in Run 3 increase by about 40%. Finally, Run 5 achieves 80% computational savings over Run 1. Also, note that even in Run 5, addition of scenarios was absorbed without taking additional inner iterations. All runs achieved optimality after 15 inner iterations. We estimate that with $\tau_{2,i} = 0.25$, 15,000-scenario instance of the two-stage Markowitz problem with 40 securities can produce a "statistical accuracy" of about four digits. So, the relative easiness of the second stage problems are not because scenarios are closer to each other in terms of problem data. We think that the reason lies in the geometric simplicity of the problem that allows absorption of infeasibilities easily even when starting from neighboring scenario solutions for small values of μ.

Recall that the objective of the two-stage Markowitz problem is a concave quadratic function. We remark that our implementation successfully handled these quadratic terms without any numerical issues.

In summary, our experience suggests that adaptive addition of scenarios can be a very effective way of achieving computational savings, particularly when warm-starting the second

outer itr	μ	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
1	1E+01	10,000	100	100	100	10,000	100	100
2	1E-01	10,000	100	100	100	10,000	100	100
3	1E-02	10,000	225	100	100	10,000	225	100
4	1E-03	10,000	3,300	225	100	10,000	3,300	225
5	1E-04	10,000	10,000	3,300	225	10,000	10,000	10,000
6	1E-05	10,000	10,000	10,000	3,300			
7	1E-06	10,000	10,000	10,000	10,000			
8	1E-07	10,000	10,000	10,000	10,000			

Table 15: Scenario addition strategies for MCR3-10,000 ($p_{\%} = 5\%,\ \hat{\sigma}_{100}(10^{-3}, \hat{x}(10^{-3})) = 5.20$)

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7
# inner itr	25	27	41	70	16	18	21
# new sce add itr	10,016	62,926	75,923	-	10,016	62,926	68,866
# 2nd stage recenter itr	260,000	154,675	184,971	168,016	166,400	58,388	72,045
# new sce adds	10,000	10,000	10,000	-	10,000	10,000	19,900
# 2nd stage recenters	250,000	154,575	184,800	167,875	160,000	58,350	71,900
final objective	204.622445	204.622445	204.622444	204.671617	204.621714	204.621671	204.620192
run time	3,238	2,043	2,526	1,971	2,141	970	1,323
avg new sce add itrs	1.00	6.29	7.59	-	1.00	6.29	6.89
avg recenter iters	1.04	1.00	1.00	1.00	1.04	1.00	1.00
infeasibility	5.54E-12	5.16E-12	2.42E-10	-	-	-	3.37E-10

Table 16: Computational Performance: MCR3-10,000

outer itr	μ	Run 1	Run 2	Run 3	Run 4
1	1E+01	10,000	100	100	100
2	1E-01	10,000	100	100	100
3	1E-02	10,000	850	100	100
4	1E-03	10,000	10,000	850	100
5	1E-04	10,000	10,000	10,000	850
6	1E-05	10,000	10,000	10,000	10,000
7	1E-06	10,000	10,000	10,000	10,000
8	1E-07	10,000	10,000	10,000	10,000

Table 17: Scenario addition strategies for MCL3-10,000 ($p_{\%} = 50\%,\ \hat{\sigma}_{100}(10^{-3}, \hat{x}(10^{-3})) = 16.57$)

	Run 1	Run 2	Run 3	Run 4
# inner itr	25	25	26	60
# new sce add itr	10,041	53,314	65,392	76,466
# 2nd stage recenter itr	260,744	153,385	124,517	370,784
# new sce adds	10,000	10,000	10,000	10,000
# 2nd stage recenters	250,000	153,250	124,400	370,650
final objective	205.620644	205.620643	205.620643	205.620643
run time	2,699	1,983	1,776	4,349
avg new sce add itrs	1.00	5.33	6.54	7.65
avg recenter iters	1.04	1.00	1.00	1.00
infeasibility	5.52E-12	5.73E-12	5.92E-12	5.52E-12

Table 18: Computational Performance: MCL3-10,000

outer itr	μ	Run 1	Run 2	Run 3	Run 4
1	1E+01	10,000	100	100	100
2	1E-01	10,000	900	100	100
3	1E-02	10,000	10,000	900	100
4	1E-03	10,000	10,000	10,000	900
5	1E-04	10,000	10,000	10,000	10,000
6	1E-05	10,000	10,000	10,000	10,000
7	1E-06	10,000	10,000	10,000	10,000
8	1E-07	10,000	10,000	10,000	10,000

Table 19: Scenario addition strategies for MCH3-10,000 ($p_{\%} = 500\%,\ \hat{\sigma}_{100}(10^{-3}, \hat{x}(10^{-3})) = 156.20$)

	Run 1	Run 2	Run 3	Run 4
# inner itr	28	27	29	33
# new sce add itr	17,643	52,783	65,881	77,425
# 2nd stage recenter itr	314,253	207,982	186,794	174,610
# new sce adds	10,000	10,000	10,000	10,000
# 2nd stage recenters	280,000	194,000	174,400	156,600
final objective	232.336743	232.336743	232.336743	232.336743
run time	3,223	2,490	2,373	2,321
avg new sce add itrs	1.76	5.28	6.59	7.74
avg recenter iters	1.12	1.07	1.07	1.12
infeasibility	8.73E-09	9.80E-09	9.70E-09	6.38E-09

Table 20: Computational Performance: MCH3-10,000

outer itr	μ	Run 1	Run 2	Run 3	Run 4
1	1E+01	10,000	100	100	100
2	1E-01	10,000	100	100	100
3	1E-02	10,000	1,500	100	100
4	1E-03	10,000	10,000	1,500	100
5	1E-04	10,000	10,000	10,000	1,500
6	1E-05	10,000	10,000	10,000	10,000
7	1E-06	10,000	10,000	10,000	10,000
8	1E-07	10,000	10,000	10,000	10,000

Table 21: Scenario addition strategies for MCL33-10,000 ($p_{\%} = 50\%,\ \hat{\sigma}_{100}(10^{-3}, \hat{x}(10^{-3})) = 24.96$)

	Run 1	Run 2	Run 3	Run 4
# inner itr	26	26	29	81
# new sce add itr	34,777	96,439	108,506	119,561
# 2nd stage recenter itr	280,745	165,344	157,244	495,655
# new sce adds	10,000	10,000	10,000	10,000
# 2nd stage recenters	270,000	165,200	157,000	494,300
final objective	204.112473	204.112473	204.112472	204.112473
run time	3,049	2,417	2,435	5,946
avg new sce add itrs	3.48	9.64	10.85	11.96
avg recenter iters	1.04	1.00	1.00	1.00
infeasibility	5.45E-12	5.75E-12	4.18E-10	5.19E-12

Table 22: Computational Performance: MCL33-10,000

stage problems from a nearby scenario is possible. However, it is also important to balance scenario addition with changes in the barrier function along the central path.

8. Conclusions

We have given a practical primal decomposition algorithm that follows the primal central path in the first stage. At each iteration, using approximate primal and dual solutions of the second stage barrier problems, it generates gradient and Hessian information for the first stage problem and takes a step along the Newton direction in the primal space. Several problems inherent in the context of primal algorithms were resolved using a preprocessing phase, and heuristics were developed for line search and scenario addition. These heuristics are based on empirically observed properties of the central path. A rigorous theoretical justification of these observed properties is a topic of future research.

Numerical experiments were conducted on a set of randomly generated problems, and a two stage extension of Markowitz's basic portfolio optimization model. Numerical experience suggests that we can solve the second stage centering problem approximately without compromising the performance of the decomposition algorithm. This experience also suggests that we need to follow the primal central path closely to develop stable implementations. Our results show that significant computational savings are possible for primal decomposition algorithms by adaptive addition of scenarios. These computational savings increase if it is possible to warm-start solution of newly added second stage centering problems from the solution of a neighboring problem. The possibility of using primal predictor-corrector methods, and integrating warm-starts with scenario generation may further improve the performance of primal decomposition methods, and should be explored in the future.

		$dim\ y_i = 20$		$dim\ y_i = 30$		$dim\ y_i = 40$	
itr	μ	Run 1	Run 2	Run 1	Run 2	Run 1	Run 2
1	1E-04	25,000	100	20,000	100	15,000	100
2	1E-05	25,000	100	20,000	100	15,000	100
3	1E-06	25,000	500	20,000	300	15,000	250
4	1E-07	25,000	12,500	20,000	7,820	15,000	5,200
5	1E-08	25,000	25,000	20,000	20,000	15,000	15,000
6	1E-09	25,000	25,000	20,000	20,000	15,000	15,000
7	1E-10	25,000	25,000	20,000	20,000	15,000	15,000
8	1E-11	25,000	25,000	20,000	20,000	15,000	15,000
9	1E-12	25,000	25,000	20,000	20,000	15,000	15,000

Table 23: Scenario addition strategies for two-stage Markowitz problems with 20,30, and 40 securities; $\tau_{2,i} = 0.05$, $\hat{\sigma}_{100}(10^{-7}, \hat{x}(10^{-7})) = 6.65 \times 10^{-5}, 4.92 \times 10^{-5}, 3.68 \times 10^{-5}$, for N=20, 30, and 40, respectively.

	$dim\ y_i = 20$		$dim\ y_i = 30$		$dim\ y_i = 40$	
	Run 1	Run 2	Run 1	Run 2	Run 1	Run 2
# inner itr	17	21	18	21	18	18
# new sce add itr	78,997	178,461	26,075	140,021	15,861	101,362
# 2nd stage recenter itr	524,696	450,853	411,641	305,371	291,613	151,852
# new sce adds	25,000	49,900	20,000	20,000	15,000	15,000
# 2nd stage recenters	425,000	277,400	340,000	224,760	255,000	146,300
final objective	-2.07640524E-03	-2.07640524E-03	-1.88081179E-03	-1.88081179E-03	-1.68518238E-03	-1.68518238E-03
run time	1,932	1,896	1,634	1,518	1,754	1,211
avg new sce add itrs	3.16	7.14	1.30	7.00	1.06	6.76
avg recenter iters	1.23	1.63	1.21	1.36	1.14	1.04
infeasibility	2.60E-14	6.88E-14	4.29E-14	4.00E-14	9.41E-09	9.61E-09

Table 24: Computational Performance: with 20,30, and 40 securities; $\tau_{2,i} = 0.05$

itr	μ	Run 1	Run 2	Run 3	Run 4	Run 5
1	1E-04	15,000	100	100	100	100
2	1E-05	15,000	100	100	100	100
3	1E-06	15,000	175	175	100	100
4	1E-07	15,000	4,150	4,150	175	100
5	1E-08	15,000	15,000	15,000	4,150	100
6	1E-09	15,000	15,000	15,000	15,000	100
7	1E-10	15,000	15,000	15,000	15,000	100
8	1E-11	15,000	15,000	15,000	15,000	15,000
9	1E-12	15,000	15,000	15,000	15,000	15,000

Table 25: Scenario addition strategies for with 40 securities; $\tau_{2,i} = 0.25,\ \hat{\sigma}_{100}(10^{-7}, \hat{x}(10^{-7})) = 2.9610^{-5}$

	Run 1	Run 2	Run 3	Run 4	Run 5
# inner itr	15	15	15	15	15
# new sce add itr	15,331	129,572	15,009	15,009	15,013
# 2nd stage recenter itr	281,249	129,320	159,122	129,320	46,470
# new sce adds	15,000	15,000	15,000	15,000	15,000
# 2nd stage recenters	225,000	114,050	143,850	114,050	46,200
final objective	-1.57070630E-03	-1.57070630E-03	-1.57070630E-03	-1.57070630E-03	-1.57070630E-03
run time	1,430	1,058	886	742	278
avg new sce add itrs	0.61	8.64	1.00	1.00	1.00
avg recenter iters	1.25	1.13	1.11	1.13	1.01
infeasibility	4.43E-14	4.45E-14	3.79E-14	4.35E-14	4.60E-14

Table 26: Computational Performance: with 40 securities; $\tau_{2,i} = 0.25$

Bibliography

[1] F. Alizadeh and S. H. Schmieta (1997), "Optimization with semidefinite, quadtratic and linear constraints," Technical Report, RUTCOR, Rutgers University.

[2] F. Alizadeh, D. Goldfarb (2003), "Second-order cone programming," *Mathematical Programming* 95, 3-51.

[3] O. Bahn, O. du Merle, J.-L. Goffin and J.P. Vial (1995), "A cutting plane method from analytic centers for stochastic programming," *Mathematical Programming* 69, 45-73.

[4] J. R. Birge and F. Louveaux (1997), *Introduction to Stochastic Programming*, (Springer-Verlag).

[5] P. Bratley and B. Fox (1988), "Algorithm 659: implementing Sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software* 14, 88-100.

[6] L. Faybusovich (1997), "Euclidean Jordan algebras and interior-point algorithms," *Positivity* 1, 331-357.

[7] L. Faybusovich (1997), "Linear systems in Jordan algebras and primal-dual interior-point algorithms," *Journal of Computational and Applied Mathematics* 86, 149-175.

[8] T.J. Flavin and M.R. Wickens, "Tactical asset allocation: a multivariate GARCH approach," Technical Report, (2000).

[9] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, (Springer, 2003)

[10] D. Goldfarb and G. Iyengar (2003), "Robust portfolio selection problems," *Mathematics of Operations Research*, 28(1), 1-38

[11] M. Kojima, M. Shida, and S. Shindoh (1995), "Global and local convergence of predicotr-corrector infeasible-interior-point algorithms for semidefinite programs," Research Reports on Information Sciences B-305, Department of Information Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan.

[12] O. Ledoit, P. Santa-Clara and M. Wolf (2003), "Flexible multivariate GARCH modeling with an application to international stock markets," *Review of Economics and Stattistics* 85(3) 735-747.

[13] J. Linderoth, A. Shapiro, S. Wright (2002), "The empirical behavior of sampling methods for stochastic programming", published electronically in: www.optimization-online.org.

[14] I.J. Lustig, R.E. Marsten, and D. Shanno (1992), "On implementing Mehrotra's predictor - corrector interior point method for linear programming", SIAM J. Optimization 2, 435-449

[15] H. M. Markowitz, "Portfolio selection," *Journal of Finance* 46, (1952) 469-477.

[16] S. Mehrotra (1992), "On the implementation of a primal-dual interior point method," *SIAM J. Optimization*, 2(4), 575-601.

[17] S. Mehrotra and M. G. Özevin (2006), "Decomposition-Based Interior Point Methods for Two-Stage Stochastic Convex Quadratic Programs with Recourse," (to appear in *Operations Research*).

[18] S. Mehrotra and M. G. Özevin (2004), "Two-stage stochastic semidefinite programming and interior point based decomposition methods: theory," Industrial Engineering and Management Science technical report 2004-16, Northwestern University, Evanston, IL 60208.

[19] R. O. Michaud (1998), *Efficient asset management: a practical guide to stock portfolio management and asset allocation*, Financial Management Association survey and synthesis series, (HBS).

[20] J. E. Nesterov and A. S. Nemirovsky (1994), *Interior-point polynomial algorithms in convex programming*, Studies in Applied Mathematics, SIAM.

[21] J. E. Nesterov and M.J. Todd (1998), "Primal-Dual Interior-Point Methods for Self-Scaled Cones," *SIAM Journal on Optimization* 8, 324-364.

[22] F.A. Potra and R. Sheng (1998), "A superlinearly convergent primal-dual infeasible-interior-point algorithm for semidefinite programming," SIAM J. Optimization, 8(4), 1007-1028.

[23] J. Renegar (2001), *A Mathematical view of Interior-Point Methods in Convex Programming*, MPS-SIAM Series on Optimization, SIAM.

[24] A. Shapiro (1991), "Asymptotic analysis of stochastic programs," *Annals of Operations Research* 30, 169-186.

[25] J. Sturm (1999), "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software* 11-12, 625-653.

[26] J. Sturm (2002), "Implementation of interior point methods for mixed semidefinite and second order cone optimization problems," *Optimization Methods and Software* 17(6), 1105-1154.

[27] J. Sturm (2003), "Avoiding numerical cancellation in the interior point method for solving semidefinite programs," *Mathematical Programming* Series B, 95(2), 219-247.

[28] M. J. Todd, K. C. Toh, and R. H. Tütüncü (1998), "On the Nesterov-Todd Direction in Semidefinite Programming," *SIAM Journal on Optimization* 8, 769-796.

[29] R. M. Van Slyke and R. J. Wets (1969), "L-Shaped linear programs with applications to optimal control and stochastic linear programming," *SIAM Journal of Applied Mathematics* 17, 638-663.

[30] G. Zhao (2001), "A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs," *Mathematical Programming* Ser. A 90, 507-536.