NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ADAPTIVE SELECTIONS OF SAMPLE SIZE AND SOLVER ITERATIONS IN STOCHASTIC OPTIMIZATION WITH APPLICATION TO NONLINEAR COMMODITY FLOW PROBLEMS

by

David A. Vondrak

March 2009

Thesis Advisor:	Johannes O. Royset
Second Reader:	R. Kevin Wood

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Adaptive Selections of Sample Size and Solver Iterations in Stochastic Optimization with Application to Nonlinear Commodity Flow Problems		**5. FUNDING NUMBERS**	
6. AUTHOR(S) David A. Vondrak			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

We present an algorithm to approximately solve certain stochastic nonlinear programs through sample-average approximations. The sample sizes in these approximations are selected by approximately solving optimal-control problems defined on a discrete-time dynamic system. The optimal-control problem seeks to minimize the computational effort required to reach a near-optimal objective value of the stochastic nonlinear program. Unknown control-problem parameters such as rate of convergence, computational effort per solver iteration, and optimal value of the program are estimated within a receding horizon framework as the algorithm progresses. The algorithm is illustrated with single-commodity and multi-commodity network flow models. Measured against the best alternative heuristic policy we consider for selecting sample sizes, the algorithm finds a near-optimal objective value on average up to 17% faster. Further, the optimal-control problem also leads to a 40% reduction in standard deviation of computing times over a set of independent runs of the algorithm on identical problem instances. When we modify the algorithm by selecting a policy heuristically in the first stage (only), we improve computing time, on average, by nearly 47% against the best heuristic policy considered, while reducing the standard deviation across the independent runs by 55%.

14. SUBJECT TERMS Nonlinear Stochastic Optimization, Optimal Control, Dynamic Programming, Network Commodity Flow, Sample Average Approximation, Projected Gradient Method			**15. NUMBER OF PAGES** 57
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ADAPTIVE SELECTIONS OF SAMPLE SIZE AND SOLVER ITERATIONS IN STOCHASTIC OPTIMIZATION WITH APPLICATION TO NONLINEAR COMMODITY FLOW PROBLEMS

David A. Vondrak
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 2009

Author: David A. Vondrak

Approved by: Johannes O. Royset
Thesis Advisor

R. Kevin Wood
Second Reader

Robert F. Dell
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

We present an algorithm to approximately solve certain stochastic nonlinear programs through sample-average approximations. The sample sizes in these approximations are selected by approximately solving optimal-control problems defined on a discrete-time dynamic system. The optimal-control problem seeks to minimize the computational effort required to reach a near-optimal objective value of the stochastic nonlinear program. Unknown control-problem parameters such as rate of convergence, computational effort per solver iteration, and optimal value of the program are estimated within a receding horizon framework as the algorithm progresses. The algorithm is illustrated with single-commodity and multi-commodity network flow models. Measured against the best alternative heuristic policy we consider for selecting sample sizes, the algorithm finds a near-optimal objective value on average up to 17% faster. Further, the optimal-control problem also leads to a 40% reduction in standard deviation of computing times over a set of independent runs of the algorithm on identical problem instances. When we modify the algorithm by selecting a policy heuristically in the first stage (only), we improve computing time, on average, by nearly 47% against the best heuristic policy considered, while reducing the standard deviation across the independent runs by 55%.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Optimization of stochastic programs is challenging in part because there is no closed-form solution, and because solution algorithms tend to be computationally intensive. The difficulty arises because the objective function and/or constraint functions are given by expectations that cannot be evaluated exactly. Hence, approximations are usually employed in optimization algorithms to estimate such functions. One class of such algorithms uses sample-average approximations within a nonlinear approximating problem (AP). However, there is no straightforward means to determine a sample size for use in these algorithms. Most algorithms resort to using heuristic policies for determining an appropriate sample size. In addition to sample size, because the AP is nonlinear, a suitable number of iterations of a nonlinear programming solver must also be selected for solving it. Most often, the number of solver iterations is selected prior to calculations beginning. It is often difficult in practice to select sample sizes and number of solver iterations that balances accuracy and computational effort.

We develop a discrete-time dynamic system, the optimal control of which determines a policy for sample size and a number of solver iterations for use in the AP. The optimal-control problem seeks to minimize the computational effort required to reach a near-optimal objective value of the stochastic nonlinear program. The optimal-control problem is approximately solved within a receding horizon framework, allowing repeated estimation of unknown parameters.

Empirical studies are performed on nonlinear single and multiple-commodity network-flow problems on a grid constructed of 50 nodes with 134 arcs. The optimal-control problem selects sample sizes and solver iterations that lead to near-optimal objective values in less time than alternative heuristic policies in all instances tested. Measured against the best alternative policy we consider for selecting sample sizes, the algorithm finds a near-optimal objective value on average up to 17% faster. Further, the optimal-control problem also leads to a 40% reduction in standard deviation of computing times over a set of independent runs of the algorithm on identical problem

instances. The unknown parameters in the optimal-control problem may be poorly estimated prior to the first stage of the algorithm, which may result in a poor policy for the first stage. When we modify the algorithm by selecting a policy heuristically in the first stage (only), we improve computing time, on average, by nearly 47% against the best heuristic policy considered, while reducing the standard deviation across the independent runs by more than one-half.

ACKNOWLEDGMENTS

I wish to thank Professor Johannes Royset for his unconditional guidance and unwavering patience during the completion of this research. Without his tireless efforts in keeping me focused and constant reminders in properly explaining notation, none of this would have been precise. His expert mind bears the responsibility for this research and thanks to his attentive mentorship; I was able to complete this fascinating journey.

I would also like to extend my deepest gratitude to Professor Kevin Wood for his dedicated efforts to ensure my writing was meaningful. He graciously accepted the challenge of editing my sometimes incoherent babble, and made my words more eloquent then they otherwise would have been.

Finally, I wish to thank my devoted wife, Terri, for her immeasurable support and endless patience in the midst of this personal adversity. Without her personal sacrifice, this work would not have been accomplished. To my daughters, Emily, Peytin, and Madelyn, thank you for enduring weeks without dad so that I could finish the work that kept me from you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Optimization of stochastic programs has become a focus of much research over the past decade. These problems are of particular interest in part because they have no closed-form solutions and solution algorithms tend to be computationally intensive. The difficulty arises because the objective function and/or constraint functions are given by expectations that cannot be evaluated exactly. Hence, approximations are usually employed to estimate such functions. Approximations may provide precise, high-fidelity estimates with high computational cost, or may provide low fidelity with less computational cost. This research focuses on finding a balance between these two important factors.

Numerous design and planning applications require the optimization of stochastic-programming problems. In aerodynamics, various problems arise in the optimization of a three-dimensional wing design (Alexandrov et al., 2001). In one civil-engineering discipline, problems arise from structural optimization of bridges or support structures subject to failure probability constraints (Polak and Royset, 2007; Royset and Polak, 2007). Such problems may seek to optimize the cross-sectional dimensions of a support column consisting of a material of particular yield strength subjected to various bending moments. Structural loading of the support column weighs heavily on design considerations due to inherent failure probabilities of the materials. Many additional examples of stochastic-programming problems can be found in the areas of logistics and supply-chain planning. Numerous papers discuss solution methodologies for optimal design of supply-chain management including, but not limited to productions lines, consumer demand, availability of raw materials, warehousing and transportation of goods. In Santoso et al. (2003), a proposed stochastic-programming model and solution algorithm are used to compute high-quality solutions to large-scale stochastic supply chain design problems. Poojari et al. (2006) presents a method to solve discrete resource

allocation problems in the presence of future uncertainties for supply-chain planning and Sox and Muckstadt (1997) describe a finite-horizon stochastic optimization model for a stochastic lot-scheduling problem.

One approach to approximately solve optimization problems defined in terms of expectations is to use Sample Average Approximations (SAA), e.g., (Ruszczynski and Shapiro, 2007). That is, one or more *approximating problems* (APs) are solved, problems that replace each expectation with a standard sample average approximation. The AP may be viewed as a deterministic mathematical program, a nonlinear mathematical program in our case. A very large sample size in the sample-average approximation will provide a precise estimate of the expected-value function, but the computational effort required to solve the AP for this sample size may be far too high. As an alternative, this research examines the computational effort associated with solving a sequence of APs where an efficient sample size is chosen at each stage of the sequence. Coarse approximations are made early and, as the calculations evolve, adaptive adjustments to the sample size are made, increasing the precision of the results. This adaptive control strategy is intuitively appealing as gains towards optimality come initially at low computational cost through coarse approximations, and fidelity is increased as larger samples are used near the completion of the algorithm.

In addition to finding an efficient sample size for use with calculations, we are also concerned with selecting an efficient number of iterations for the chosen nonlinear programming (NLP) solver used in solving various instance of the AP. The number of solver iterations has a profound impact on computational cost as well. Our approach will determine an efficient number of iterations for the NLP solver to carry out along with an efficient sample sizes for defining the APs as the overall algorithm progresses.

B. LITERATURE REVIEW

Stochastic programming solution methodologies incorporate both mathematical-programming techniques and statistically motivated approximations. These approximations are generally internal or external in nature. The distinction is found in the management of sample selection. With external methodologies (Royset and Polak,

2007), samples are taken before the solution algorithm begins (or they could be taken before the algorithm begins), and no additional sampling is performed during the optimization process. In internal methods (Higle and Zhao, 2004), samples are intrinsic to the iterative solution process, performed whenever the algorithm requires the estimation of expectations. An alternative to an external sampling approach with a fixed sample size is to vary the sample size during the calculations using closed-loop or open-loop techniques (Polak and Royset, 2007). With a closed-loop technique, the sample size is adapted during the calculations. For example, if the objective value in a given iteration falls below a moving floor, the sample size may be increased. On the other hand, in an open-loop technique, sample sizes are preassigned (Polak and Royset, 2007). In most cases, the sample sizes increase as iterations progress towards an optimal solution.

He and Polak (1990) describe a method for handling progressively finer stages of discretization for semi-infinite optimization problems, which uses a precision-adjustment strategy relevant to the present work. They formulate an auxiliary optimization problem that determines the number of solver iterations and precision level at different stages of the calculations so that overall computing time is minimized approximately. The present work is motivated by this study.

C. RESEARCH GOAL

The goal of this research is to find an efficient way of selecting sample size and number of solver iterations for use in solving a nonlinear stochastic program through the solution of a sequence of APs that use sample average approximations. We formulate a discrete-time optimal-control problem that we solve approximately to obtain a precision-adjustment policy for determining the sample size and number of solver iterations. This policy makes coarse approximations in the early stages of the problem and, as progression to the optimal objective value is achieved, the sample size increases to ensure convergence to a locally optimal solution. The optimal-control problem is formulated with the objective to minimize the amount of computational work necessary to reach a locally optimal solution to the AP.

D. STRUCTURE OF THESIS AND CHAPTER OUTLINE

This thesis is organized into five chapters including the Introduction. Chapter II provides a discussion on the formulation of a nonlinear stochastic problem, an outline of the algorithm we intend to use to solve stochastic nonlinear network flow problems and introduces our precision-adjustment problem. Chapter III introduces the methodology we propose to use for selecting an efficient sample size and number of solver iterations in the solution algorithm; stopping criterion are also discussed. Chapter IV formulates and solves two types of stochastic nonlinear network flow problems. It compares our algorithm's efficiency with the efficiency of similar algorithms that use heuristic sample-size and solver-iteration policies. Chapter V summarizes the research and presents the main findings and insights.

II. PROBLEM DEFINITION AND FORMULATION

This chapter formulates a stochastic nonlinear program and presents a conceptual algorithm we use as our solution approach. Further, it describes the precision-adjustment problem we face within the conceptual algorithm.

A. PROBLEM FORMULATION

This section describes the formulation of the "expected-value problem" and an approximating problem which will be used as a surrogate to approximately solve the expected-value problem.

1. Expected-value Problem

Consider the optimization of, say, an engineering design or logistics problem defined by the random function $F(x,\omega)$, where $x \in X \subset ¡^n$ is a vector of continuous decision variables and ω is a vector of continuous random variables defined on the probability space (Ω, F, P). This situation results in a difficult stochastic optimization problem of the form:

$$\mathbf{EP:} \quad \min_{x \in X \subset ¡^n} \{ f(x) := \mathrm{E}[F(x,\omega)] \}, \tag{2.1}$$

where X is a compact subset of $¡^n$ representing feasible solutions of the problem, and E is the expectation taken with respect to the known probability distribution P of the random vector ω. We assume that, for every $x \in X$, the expected-value function is well defined and smooth (continuously differentiable). Further, we denote the optimal value of (2.1) as f^* with a set of optimal solutions denoted by X^*. Further we indicate a set of ε-optimal solutions by X_ε^*, i.e., for any $\varepsilon \geq 0$

$$X_\varepsilon^* := \{ x \in X \mid f(x) - f^* \leq \varepsilon \}. \tag{2.2}$$

Because the distribution of $F(x,\omega)$ is unknown, we cannot compute the expectation in closed form, so we approximate the expected-value function by the

sample-average estimator discussed below. Uncertainty may also be introduced in the constraint functions defining *X*; however, our research focuses only on uncertainty in the objective function.

2. Approximating Problem

To approximate $f(x) = \mathrm{E}[F(x,\omega)]$ we use a sample average defined by:

$$f_N := \frac{1}{N}\sum_{j=1}^{N} F(x,\omega^j), \tag{2.3}$$

where $\omega^1, \omega^2, ..., \omega^N$, is a sample of size N consisting of independent, identically distributed (iid) random variables. Moreover, we define an approximating problem (AP):

$$\textbf{AP:} \quad \min_{x \in X} f_N(x), \tag{2.4}$$

with an optimal value denoted by f_N^*. We refer to (2.1) and (2.4) as the "expected-value" and "approximating problems," respectively.

The expected-value function, for instance, may be represented by the function depicted by the solid line in Figure 1. This function, as stated above, cannot be computed in closed form, so we approximate the function's form by the sample average as shown in Figure 1. As discussed in Section 3 below, we expect the sample average to take the form of the expected-value function as the sample size approaches infinity. It is known that solving the AP with an appropriate sample size provides a reasonable approximation to the solution to EP (Ruszczynski and Shapiro, 2007).

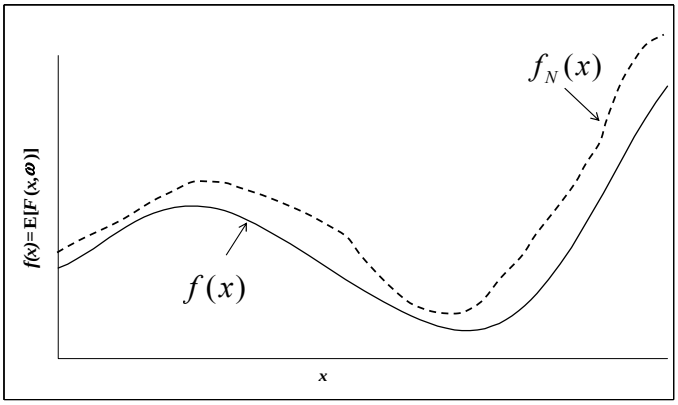

Figure 1. Expected Value function compared to Sample Average function

3. Properties of Sample Average

Monte Carlo simulation can be used to generate an iid sample of ω's of size N to obtain $F(x,\omega^1),F(x,\omega^2),...,F(x,\omega^N)$ for use in (2.3). It is well known that $f_N(x)$ is an unbiased estimator of $f(x)$ i.e.,

$$\begin{aligned}&\mathrm{E}[f_N(x)]=f(x),\\&\mathrm{Var}[f_N(x)]=\sigma^2(x)/N,\end{aligned}$$

where

$$\sigma^2(x)=\mathrm{Var}(F(x,\omega)).$$

Because the generated sample is iid and given rather weak general assumptions, it follows from the Law of Large Numbers (LLN) that $f_N(x)$ converges pointwise to $f(x)$ with probability one, as $N\to\infty$ (Ruszczynski and Shapiro, 2007). Therefore, we would expect the optimal value and optimal solution of the AP to converge to those of EP, as $N\to\infty$. This is made precise in the following proposition.

Proposition 1. (Prop. 4.2; Ruszczynski and Shapiro, 2007). If the pointwise LLN holds, i.e., $f_N(x)$ *converges to* $f(x)$ *uniformly on X, with probability one as* $N\to\infty$, then f_N^* converges to f^* with probability one, as $N\to\infty$.

Moreover, if $\mathrm{E}[F(x,\omega)^2]<\infty$, then it follows under weak assumption from the central limit theorem (CLT) that $\sqrt{N}(f_N(x)-f(x))\Rightarrow Y_x$, where $\Rightarrow$ denotes convergence in distribution and Y_x is a zero-mean normal random variable with variance $\sigma^2(x)$ (Ruszczynski and Shapiro, 2007). Hence, for a large N, $f_N(x)$ is approximately normally distributed with mean $f(x)$, and variance $\sigma^2(x)/N$. Figure 2 illustrates this result.

We know that

$$\min_{x\in X}\mathrm{E}\left[f_N(x)\right]\geq\mathrm{E}\left[\min_{x\in X}f_N(x)\right],$$

and, because $f_N(x)$ is an unbiased estimator of $f(x)$, it follows that $\mathrm{E}\left[f_N^*\right] \le f^*$. That is, the AP's optimal objective value is a downward-biased estimate of the optimal objective value of the EP.

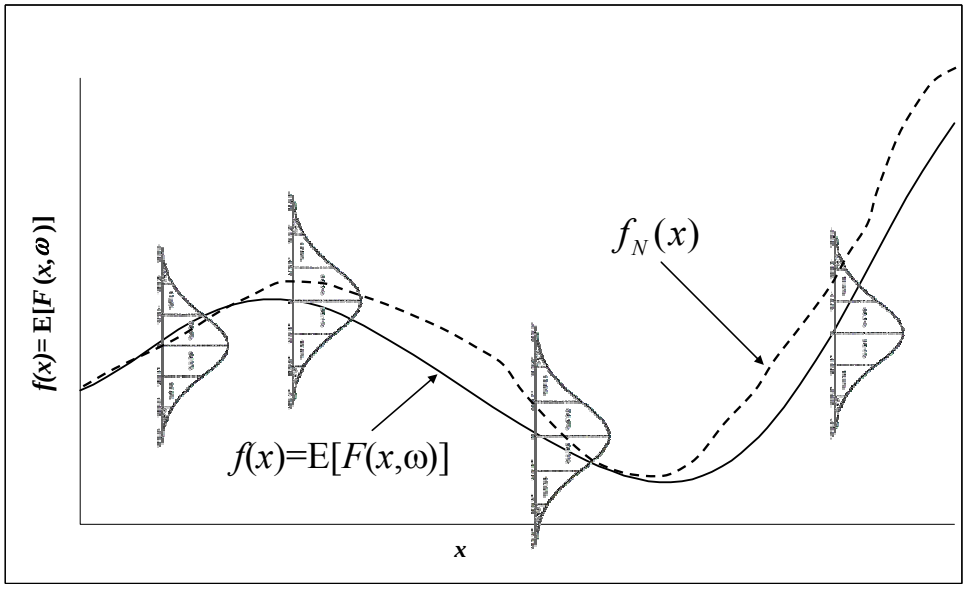

Figure 2. Approximate Normal Distribution of $f_N(x)$

B. ALGORITHM

We consider a conceptual algorithm of the following form for approximately solving EP.

Conceptual Algorithm (CA)

Data: Optimality tolerance $\varepsilon \ge 0$; initial point $x_0^1 \in X$.

Step 1: Set stage counter $k = 1$.

Step 2: Determine N_k and n_k.

Step 3: Carry out n_k iterations of a solver applied to:

$\min_{x \in X} f_{N_k}(x)$, started with x_0^k using the N_k found at Step 2.

Step 4: If $x_{n_k}^k \in X_\varepsilon^*$, then Stop. Else set $x_0^{k+1} = x_{n_k}^k$, replace k by k+1 and go to Step 2.

We refer to Steps 2-4 as a "stage." CA uses a single set of sample realizations for each stage, but all samples are independent between stages. For purposes of this research, we have chosen the Projected Gradient Method (PGM) as our nonlinear programming solver (for example, see Polak 1997, p. 66). However, it is worth mentioning that any linearly convergent nonlinear programming solver may be used. Our goal is to determine N_k and n_k for each stage that approximately minimizes the total computational effort required by CA to reach a near-optimal objective value to EP. We refer to the rule for selecting N_k and n_k as a "policy."

C. PRECISION-ADJUSTMENT PROBLEM

We define the task of selecting the appropriate N_k and n_k for each stage of the CA as the Precision-Adjustment Problem (PAP). We formulate PAP as a particular discrete-time optimal-control dynamic program, which is discussed in the next chapter. PAP is solved approximately using dynamic programming to obtain a precision-adjustment policy for determining N_k and n_k. This policy adaptively adjusts the sample size and number of solver iterations between stages. PAP is formulated to minimize the amount of computational work necessary to reach a near-optimal objective value of the stochastic nonlinear program. After carrying out n_k solver iterations, we abandon further progress to the locally optimal solution for the current AP, and use this iterate as a warm start for the next stage of CA.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

This chapter describes how we control the precision within the conceptual algorithm and how we estimate the parameters for the precision adjustment. Further, the chapter presents the criterion by which we stop the conceptual algorithm (in Step 4).

A. PRECISION ADJUSTMENT CONTROL

This section presents a discrete-time dynamic system, the optimal control of which will determine a sample size N_k and a number of solver iterations n_k to be used for each stage of CA.

1. Controlling Sample Size

The dynamic system described in Royset (2009) lays the foundation for this research. Following his approach, we first identify the asymptotic distributions of the progress made by CA. We assume that the solver used in Step 3 of CA is uniformly linearly convergent (see Royset, 2009), i.e., *there exists a* $\theta \in (0,1)$ *such that* $f_N(P_N(x)) - f_N^* \le \theta(f_N(x) - f_N^*)$ *for all* $x \in X$ *and* $N \in ¥$, where $¥ = \{1,2,3,...\}$ and $P_N(x)$ is the iterate found after carrying out one iteration of the solver used in the CA starting from x with sample size N. We refer to θ as the rate-of-convergence coefficient. We let $P_{N_k}^{n_k}(x)$ denote the iterate from the solver found at stage *k*, after n_k iterations with a sample of size N_k starting from *x*. It follows from the assumption of linear convergence and from the optimality of $f_{N_k}^*$, that for any $x \in X$,

$$f_{N_k}^* \le f_{N_k}(P_{N_k}^{n_k}(x)) \le f_{N_k}^* + \theta^{n_k}(f_{N_k}(x) - f_{N_k}^*)$$

with probability one. Moreover, based on rather weak assumptions and theorems introduced in Royset (2009), if N_k and n_k are large, $f_{N_k}(x_{n_k}^k)$ is approximately distributed as

$$\mathcal{N}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k), \tag{3.1}$$

where $\mathcal{N}(\mu, \sigma)$ represents a normally distributed random variable with mean μ and variance σ. Figure 3 illustrates a possible approximate distribution of $f_{N_k}(x_{n_k}^k)$ as stated from this conclusion.

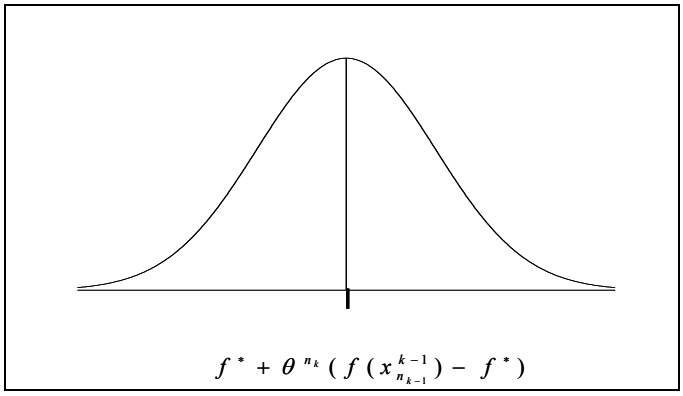

Figure 3. Approximate Normal Distribution of $f_{N_k}(x_{n_k}^k)$

Further, Royset (2009) shows that we can heuristically approximate the distribution $f(x_{n_k}^k)$ with truncation at f^* to account for the fact that $f(x) \geq f^*$ for all $x \in X$. Therefore our approximate distribution for $f(P_{N_k}^{n_k}(x_{n_{k-1}}^{k-1}))$ conditional on $x_{n_{k-1}}^{k-1}$ is:

$$\bar{\mathcal{N}}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k, f^*), \tag{3.2}$$

where $\bar{\mathcal{N}}$ denotes a truncated normally distributed random variable based on $\mathcal{N}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k)$, with a truncation threshold f^*. Figure 4 illustrates this distribution. As discussed in Subsection 2, we can control this distribution by changing N_k and n_k.

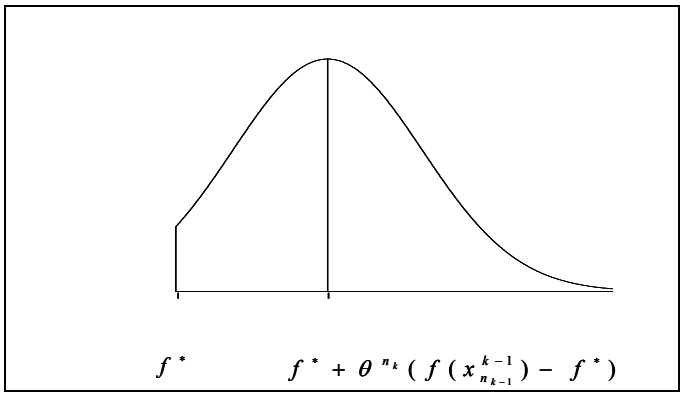

Figure 4. Approximate Truncated Normal Distribution for $f(x^k_{n_k})$

Since (3.2) approximately holds for any *k*, we can use that expression to estimate the quality of a solution generated by the CA after any number of stages given a selection of sample sizes and number of iterations. The situation is illustrated in Figure 5, where the $f(x^0_{n_0})$ and a selection of (N_1, n_1) determines the probability distribution of $f(x^1_{n_1})$. Given an outcome of that distribution, a selection of (N_2, n_2), gives the distribution of $f(x^1_{n_1})$, etc.

The values for f^*, θ, and $\sigma^2(x^*)$ in the distribution (3.2) are unknown, however. Since $x^{k-1}_{n_{k-1}}$ is known at the beginning of the k^{th} stage, we can estimate $f(x^{k-1}_{n_{k-1}})$, and we construct estimation schemes as shown in Subsection B below to estimate f^*, θ, and $\sigma^2(x^*)$. We now develop a dynamic program as discussed in Royset (2009).

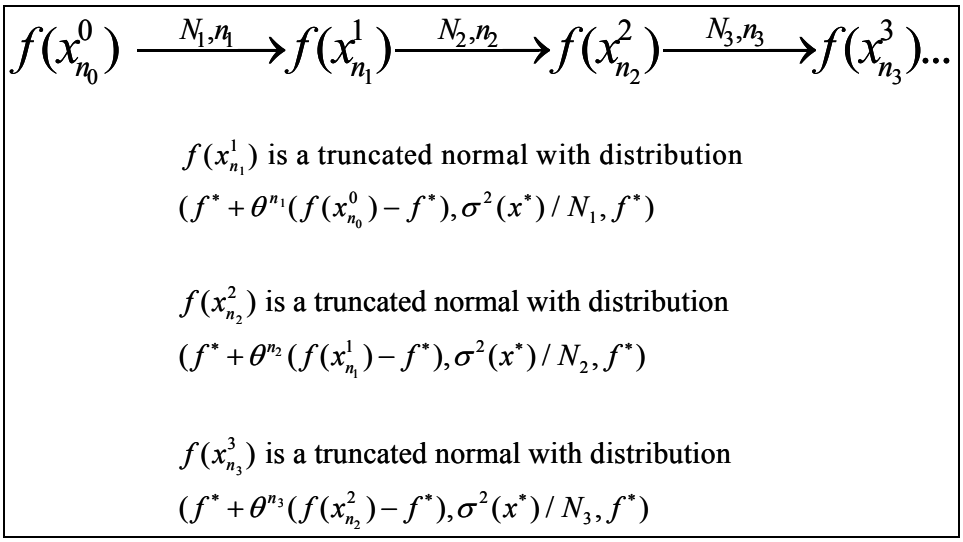

Figure 5. Truncated normal distributions of function values

2. Dynamic Program

Beginning at stage k, we define estimates of $f(x_{n_{k-1}}^{k-1}), f^*, \theta$, and $\sigma^2(x^*)$ as r_f, r^*, r_θ, and r_{σ^2} respectively and as in Royset (2009), use (3.2) as a basis for a model of $f(x_{n_l}^l),\ l = k,\ k+1,\ k+2,\ldots$. We let $f_l,\ l = k,\ k+1,\ k+2,\ldots$ denote our estimates of $f(x_{n_l}^l),\ l = k, k+1, k+2,\ldots$. Controls are defined as $(N_k, n_k),\ (N_{k+1}, n_{k+1}),\ (N_{k+2}, n_{k+2}),\ldots$ and the dynamic equation for the state f_l is

$$f_{l+1} = \bar{\mathcal{N}}(r^* + r_\theta^{n_l}(f_l - r^*), r_{\sigma^2}/N_l, r^*),\ l = k,\ k+1,\ k+2,\ldots \tag{3.3}$$

with initial condition $f_k = r_f$ and where the equality indicates equality in distribution.

We define $c(N,n)$ to be the computational cost of carrying out n iterations of the solver applied to the AP with a sample of size N, where the terminal state is characterized by $c(1,0) = 0$. Terminal states discussed in Royset (2009) are defined by X_ε^*, and are translated for our case as

$$T := \{\xi \in \text{¡} \mid \xi - r^* \le \varepsilon\}.$$

The set of feasible controls $R(\xi)$ are defined as: If $\xi \in T$, then $R(\xi) = \{(1,0)\}$; otherwise, $R(\xi) = ¥ \times ¥$. We seek an admissible stationary policy $\tau : ¡ \to ¥ \times ¥ \cup \{0\}$ with $\tau(f_l) \in R(f_l)$ which minimizes the computational cost by evaluating a planning horizon of feasible controls (N_k, n_k), (N_{k+1}, n_{k+1}), $(N_{k+2}, n_{k+2}), \ldots$. The sample-size control-problem stated in Royset (2009) which accomplishes this task is

$$J_{k,\tau}(r_f, r^*, r_\theta, r_\sigma) := \limsup_{s \to \infty} E\left[\sum_{l=k}^{s} c(\tau(f_l))\right],$$

subject to constraints (3.3). Here E is the expectation with respect to the disturbances on the stages k, $k + 1,\ldots, s$ due to the truncated normal distribution in (3.3) Finally, we define the *surrogate sample-size control problem* by

$$\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma): \quad \inf_{\tau} J_{k,\tau}(r_f, r^*, r_\theta, r_\sigma). \tag{3.4}$$

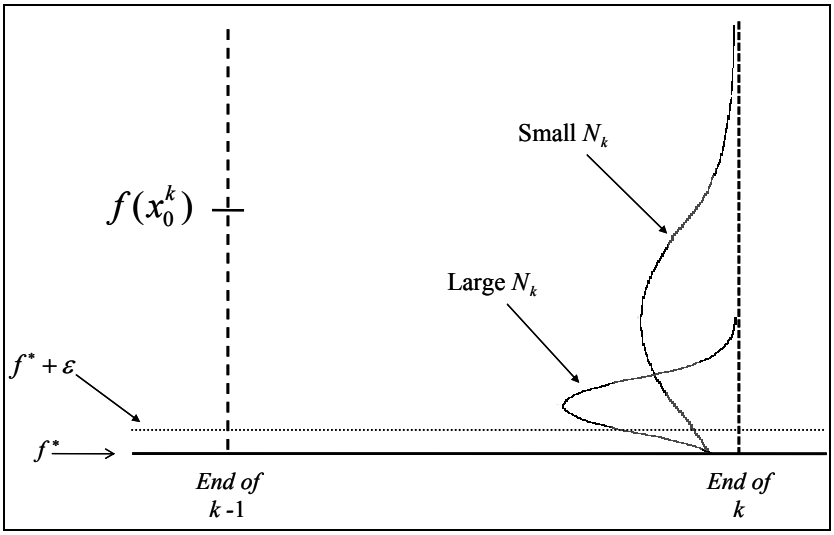

Figure 6. Representative illustration of dynamic program

Now, assume we have estimated $f(x_0^k) = f(x_{n_{k-1}}^{k-1})$ at the beginning of stage k; see Figure 6. We wish to choose a pair (N_k, n_k) that will be computationally efficient so that at the end of the k^{th} stage shown in Figure 5 we have progressed toward f^* at a "reasonable" computational cost. As shown by the dotted line in Figure 5, we define a

tolerance ε to determine a range of near-optimal objective values that are acceptable for selection as f^*. Further, we assume that the computational cost of the k^{th} stage is defined as $c(N_k, n_k) = N_k n_k$. Then, from (3.2), we see that selections of N_k and n_k have varying effects. A small N_k increases the variability in the truncated distribution whereas a large N_k compresses the distribution, reducing the variability. The number of solver iterations also impacts the cost by affecting the mean of the truncated distribution in (3.2). As n_k increases, the mean of the distribution moves closer to f^*, but also increases the computational cost. The optimal policy of $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$ balances the computational cost of selecting large N_k and n_k and the likelihood that the CA reaches a near-optimal objective value in the current stage. Hence, we expect that policy to be reasonably efficient even though it is based on several approximations.

To solve $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$ approximately, we discretize the state space and the truncated normal distribution and then apply backward recursion to the resulting dynamic program. We refer to the policy computed in this manner as Model-Predictive Control (MPC). We adopt the discretization technique, as in Royset (2009), on a planning horizon of 10 future stages. A general depiction of the discretization scheme applied to Figure 6 is shown in Figures 7 and 8. An example of a relatively low cost selection of N_k and n_k is shown in Figure 7. Here, a relatively small N_k with a choice of small n_k provides little gains toward f^*. Although progress is made toward f^*, additional stages of CA are necessary arrive at a near-optimal value.

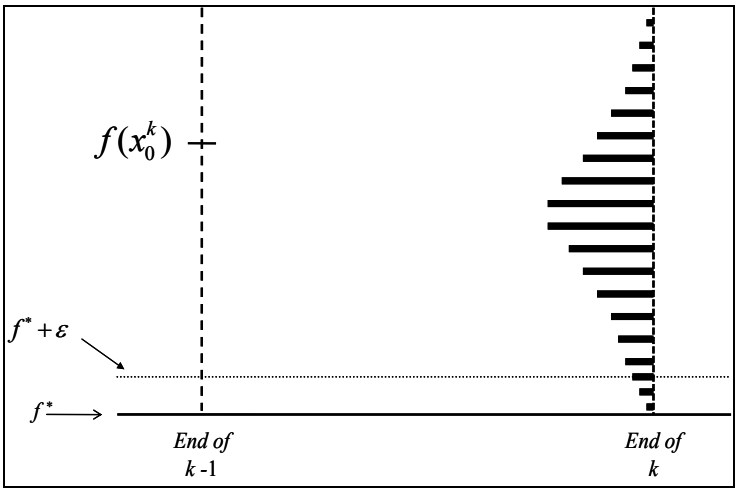

Figure 7. Discretization of truncated normal with small N_k and n_k

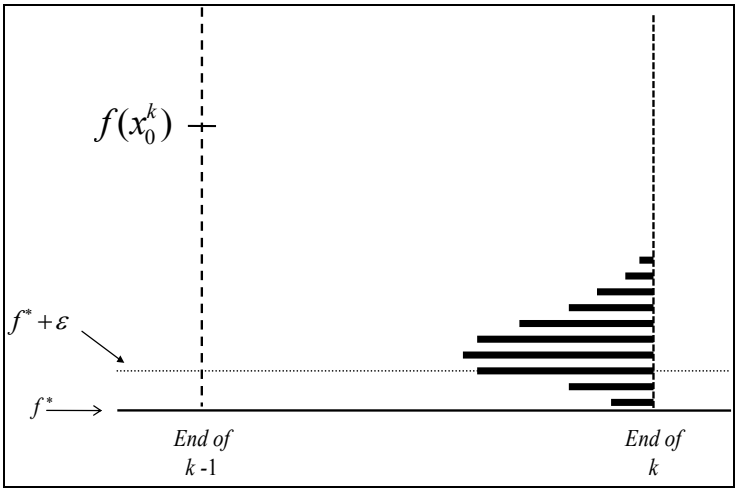

Figure 8. Discretization of truncated normal with large N_k and n_k

Figure 8 depicts a situation where a large N_k and a large n_k provide a near-optimal objective value close to f^* but at a high computational price. For this situation, future stages of CA may not be necessary in order to achieve a near-optimal objective value.

B. PARAMETER ESTIMATION

We rely on estimates of the parameters $f(x_{n_{k-1}}^{k-1}), f^*, \theta,$ and $\sigma^2(x^*)$ in $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$ as we describe next.

1. Estimating Variance

Upon completion of n_k iterations of the solver with a sample of size N_k in stage *k*, the set of iterates $\{x_i^k\}_{i=0}^{n_k}$ and function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ are known. We use this information to estimate parameters r_f, r^*, r_θ, and r_{σ^2}. First, we discuss our estimate for r_{σ^2}. We let $\hat{\sigma}_{k+1}^2$ denote our estimate of $\sigma^2(x^*)$ for stage *k* + 1 and set it equal to the sample variance at the last iterate, i.e.,

$$r_{\sigma^2} = \hat{\sigma}_{k+1}^2 := \frac{1}{N_k - 1}\sum_{j=1}^{N_k}(F(x_{n_k}^k, \omega^j) - f_{N_k}(x_{n_k}^k))^2 .$$

2. Estimation of Rate-of-Convergence Coefficient and Optimal Objective Value

Next, we determine r_θ, the estimate of the rate-of-convergence coefficient θ of the nonlinear programming solver used in Step 3 of CA. We adopt the method in He and Polak (1990) to estimate θ, but modify it slightly, adding an exponential smoothing step to avoid large changes in the estimate. As the estimate of θ is updated after each stage, we let $\hat{\theta}_k$ be the estimate of θ available at the beginning of stage *k* and set $r_\theta = \hat{\theta}_k$ in $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$.

As in Royset (2009), the (biased) estimate of f^* at the beginning of the k^{th} stage is computed by a weighted average of estimates of $f_{N_l}^*$, $l = 1, 2, ..., k-1$, and is denoted by $\hat{f}_k^*$. Unlike the approach used by Royset, $\hat{f}_k^*$ is computed with a fixed, conservative estimate of the rate of convergence denoted by $\overline{\theta}$. The meaning of "conservative" is discussed in more detail in the explanation of Step 6 of Subroutine PE below.

Subroutine PE(*k*, $\hat{\theta}_k$ $\hat{f}_k^*$, $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$**)**

Parameters: Exponential smoothing parameter $\psi \in (0,1)$, conservative estimate $\overline{\theta} \in (0,1)$ of rate of convergence θ and tolerance $\varepsilon_\theta > 0$.

Input data: Previous stage index k; estimates $\hat{\theta}_k$ and $\hat{f}_k^*$; function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ from stage k.

Output: Returns estimates $\hat{\theta}_{k+1}$ and $\hat{f}_{k+1}^*$.

Step 1: Set $\hat{\theta} = \hat{\theta}_k$.

Step 2: Estimate the minimum of $f_N(x_{n_k}^k)$ by:

$$b = \frac{1}{n_k} \sum_{i=0}^{n_k-1} \frac{f_{N_k}(x_{n_k}^k) - \hat{\theta}^{n_k-i}(f_{N_k}(x_i^k))}{1-\hat{\theta}^{n_k-i}}$$

Step 3: Solve least-square problem:

$$(a,b) = \arg\min_{a,b} \sum_{i=0}^{n_k} (\log(f_{N_k}(x_i^k) - b) - i\log a - b)^2$$

Step 4: If $|\hat{\theta} - a| < \varepsilon_\theta$, set $\hat{\theta} = a$ and go to Step 5. Else, set $\hat{\theta} = a$ and go to Step 2.

Step 5: Set $\hat{\theta}_{k+1} = \psi\hat{\theta} + (1-\psi)\hat{\theta}_k$

Step 6: Compute conservative estimate of the minimum value of $f_N(x_{n_k}^k)$

$$\hat{m}_k := \min_{i=0,1,\ldots n_k-1} \frac{f_{N_k}(x_{n_k}^k) - \bar{\theta}^{n_k-i} f_{N_k}(x_i^k)}{1-\bar{\theta}^{n_k-i}}$$

Step 7: Estimate f^*:

$$\hat{f}_{k+1}^* := \frac{N_k}{\sum_{i=1}^{k} N_k} \hat{m}_k + \frac{\sum_{l=1}^{k-1} N_l}{\sum_{l=1}^{k} N_l} \hat{f}_k^*$$

Step 8: Return $\hat{\theta}_{k+1}$ and $\hat{f}_{k+1}^*$

From our assumption of a uniformly linearly convergent solver, it follows that $f_{N_k}^* \geq (f_{N_k}(x_{n_k}^k) - \theta^{n_k-i} f_{N_k}(x_i^k))/(1-\theta^{n_k-i})$ for all i = 0, 1, 2, …, $n_k - 1$. Step 2 averages these lower-bounding estimates with the current estimate of θ and uses this as an estimate of $f_{N_k}^*$. With this estimate, we compute an estimate of the rate-of-convergence coefficient in a least-squares sense. That is, we use log-linear regression to compute the rate-of-convergence coefficient to best fit the sequence $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$. We define a tolerance for computing the rate-of-convergence coefficient, denoted by ε_θ, which

determines when to terminate the regression technique. Next, we perform exponential smoothing so as not to let the estimate of θ vary too much from one stage to another. The value $\hat{m}_k$ in Step 6 of Subroutine PE is guaranteed to be a lower bound on $f^*_{N_k}$ only if $\overline{\theta} \geq \theta$. Hence, we recommend $\overline{\theta}$ be set to a value close to 1. Since $\hat{f}^*_{k+1}$ is simply the weighted average of $\hat{m}_l$, $l = 1, 2, ..., k$, and $\mathrm{E}[f^*_N] \leq f^*$ (see Chapter II, Section A, Subsection 3), we therefore find that $\hat{f}^*_{k+1}$ is a lower bound on f^*, on average, when $\overline{\theta} \geq \theta$.

C. STOPPING CRITERION

We could implement a stopping criterion based upon a hypothesis test of Karush-Kuhn-Tucker (KKT) conditions as developed in Ruszczynski and Shapiro (2007). As stated in Ruszczynski and Shapiro (2007), suppose that the feasible set *X* is defined by equality and inequality constraints in the form

$$X := \{x \in \mathbb{R}^n : g_i(x) = 0,\ i = 1, ..., q;\ g_i(x) \leq 0,\ i = q+1, ...p\},$$

where the $g_i(x)$ are smooth deterministic functions. If only equality constraints are present and the gradient vectors $\nabla g_i(\hat{x})$, $i = 1, ..., q$ are linearly independent, then the hypothesis test of KKT conditions can be based upon an asymptotically noncentral chi-square distribution. If the assumption of linearly independent gradient vectors cannot be met, a degenerate solution is presented due to redundancy in the constraint functions. In the case of both equality and inequality constraints, a similar result is available (see Ruszczynski and Shapiro, 2007), which also relies on the linear independence of gradients of active constraints. Since such linear-independence assumptions may often fail in practical application, we have chosen to adopt an approach using optimality gaps as defined by Mak et al. (1999), and similarly by Ruszczynski and Shapiro (2007). We base our stopping criterion upon this approach with a small modification.

With the iterate $x_{n_k}^k$ found at the completion of stage *k*, we again estimate $f(x_{n_k}^k)$ as before, with $f_{N^*}(x_{n_k}^k)$ using a new independent sample of size N^*. Here we elect to use a large sample size to obtain an accurate approximation of $f(x_{n_k}^k)$. Calculation effort is not substantially increased by this procedure as we are not performing an optimization. From the central limit theorem, a probabilistic upper bound on f^* is approximately normally distributed with mean $f(x_{n_k}^k)$ and variance $\sigma^2(x_{n_k}^k)/N^*$ for large N^*.

The modification of the method described in Mak et al. (1999) occurs in our construction of a lower bound as described in Subroutine PE. While Mak et al. (1999) use the average of the optimal values of a set of APs, we construct a lower bound on f^* by averaging lower bounds on optimal values of the APs for each stage. Our method tends to be more conservative as it is based upon an assumption of a rate of convergence. From Royset (2009), we see that our lower bound is approximately normally distributed with mean f^* and variance $\sigma^2(x^*)/\sum_{l=1}^k N_l$, for large $N_1, N_2, ..., N_k$ and $n_1, n_2, ..., n_k$. Hence the inequality

$$\text{Prob}\left(f(x_{n_k}^k) \le f^* + \varepsilon\right) \ge \Phi\left(\frac{\hat{f}_{k+1}^* + \varepsilon - f_{N^*}(x_{n_k}^k)}{\sqrt{\hat{\sigma}_{k+1}^2 / N^* + \hat{\sigma}_{k+1}^2 / \sum_{l=1}^k N_l}}\right) \tag{3.5}$$

holds approximately. We therefore stop the calculations when the right-hand side in (3.5) exceeds a selected confidence level δ, typically 0.95 or 0.99.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. COMPUTATIONAL STUDY

This chapter presents a computational study of two network-flow problems to test how well Model-Predictive Control reduces computation times compared to alternative policies for selecting sample sizes. We define two network-flow problems with random congestion and present results of Model-Predictive Control as compared to heuristic control of sample sizes.

A. SINGLE-COMMODITY NETWORK FLOW

To develop a single-commodity flow problem (SCF) for testing, we consider the generic congestion model for single-commodity flows as described by Ahuja et al., (1993, p. 651), but modify it to include random congestion. Here, the generic model has a nonlinear objective function of the form

$$\min_{x \in X} \sum_{(i,j) \in A} \frac{x_{ij}}{M_{ij} - x_{ij}}, \tag{4.1}$$

where M_{ij} is the nominal capacity of arc (i, j) and x_{ij} is the flow of a single commodity on arc (i, j). For review of commodity flow and congestion modeling we refer the reader to pages 109-124 in Hearn et. al. (2001), and to Marcotte and Nguyen (1998) and Bergendorf et al. (1997). We first present an SCF problem and then advance to a multi-commodity flow problem. Even though the SCF problem is a special case of the multiple-commodity flow problem, we present SCF first, due to the relative ease of explaining this simpler problem.

We consider a graph $G = (N, A)$, where N and A are sets of nodes and arcs, respectively. The specific graph considered in this study is shown in Figure 9. This grid network can flow commodity left-to-right, north-to-south and south-to-north, but not right-to-left. The test-problem grid has 50 nodes and 134 arcs. The start node, denoted by s, is the supply source and the terminal node, t, is the demand sink. Individual costs associated with the flow across an arc are assigned as random numbers from a normal

distribution with parameters that will be specified, and we define a congestion parameter on each arc by generating a log-normally distributed random variable, with parameters that will be specified.

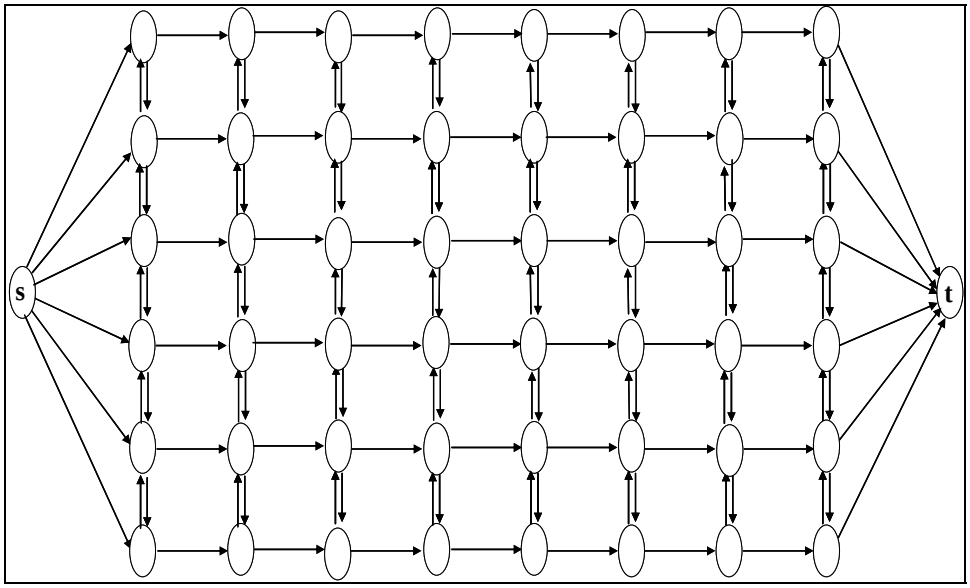

Figure 9. Transportation Grid Network

We formulate an SCF problem as follows:

Indices

i, j nodes, $i, j \in N$

(i, j) arcs $(i, j) \in A$

Data

M_{ij} capacity of arc (i, j).

C_{ij} unit cost to ship commodity on arc (i, j).

μ_{ij} mean of log-normal random variable denoting congestion on arc (i, j).

σ_{ij}^2 variance of log-normal random variable denoting congestion on arc (i, j).

D demand of commodity.

Random Variables

ω_{ij} congestion parameter on arc (i, j); this is a log-normal random variable, with mean μ_{ij} and variance σ_{ij}^2.

Decision Variables

x_{ij} amount of commodity shipped on arc (i, j).

Mathematical Formulation

$$\min \ \mathrm{E}\left[\sum_{(i,j)\in A} \frac{C_{ij} x_{ij}}{(1+\omega_{ij}) M_{ij} - x_{ij}} \right] \tag{4.2}$$

$$\text{s.t.} \ \sum_{j:(j,i)\in A} x_{ji} - \sum_{i:(i,j)\in A} x_{ij} = \begin{cases} -D & \text{if } i = s \\ 0 & \text{if } i \in N \setminus \{s,t\} \\ D & \text{if } i = t \end{cases}$$

$$0 \le x_{ij} \le M_{ij}, \qquad \forall \ i, j$$

We note that the expectation in the objective function can be computed by evaluating $|A|$ one-dimensional integrals, and thus a simpler method for solving this model is available. However, this model serves as a simple example to illustrate our solution approach, which applies to more general situations.

We assign 500 units of supply at node s, with a corresponding 500 units of demand at node t. Arc capacities are chosen as $M_{ij} = 100$ for all (i, j). Based on preliminary numerical experiments, we find that $\bar{\theta} = 0.993$ is sufficient to obtain lower bounds on $f_{N_k}^*$ in Step 6 of Subroutine PE.

B. MULTI-COMMODITY NETWORK FLOW

For testing, we also consider a congestion problem for a multi-commodity flow problem in a transportation network (MCF), and reuse the grid network described in Subsection A. The formulation is as follows:

Indices

i, j nodes, $i, j \in N$

(i, j) arcs $(i, j) \in A$

p commodity, $p \in \{1, 2, ..., P\}$

Data

M_{ij} capacity of arc (i, j).

C_{ij}^{p} unit cost to ship commodity p on arc (i, j).

μ_{ij} mean of log-normal random variable denoting congestion on arc (i, j).

σ_{ij}^{2} variance of log-normal random variable denoting congestion on arc (i, j).

D^{p} demand of commodity p.

Random Variables

ω_{ij} congestion parameter on arc (i, j); this is a log-normal random variable, with mean μ_{ij} and variance σ_{ij}^{2}.

Variables

x_{ij}^{p} amount of commodity p shipped on arc (i, j).

Mathematical Formulation

$$\min \quad \mathrm{E}\left[\sum_{(i,j)\in A} \frac{\sum_{p} C_{ij}^{p} x_{ij}^{p}}{(1+\omega_{ij}) M_{ij} - \sum_{p} x_{ij}^{p}} \right] \tag{4.3}$$

$$\text{s.t.} \quad \sum_{j:(j,i)\in A} x_{ji}^{p} - \sum_{i:(i,j)\in A} x_{ij}^{p} = \begin{cases} -D^{p} & \text{if } i = s \\ 0 & \text{if } i \in N \setminus \{s, t\}, \ \forall\, p \\ D^{p} & \text{if } i = t \end{cases}$$

$$\sum_{p} x_{ij}^{p} \le M_{ij}, \quad \forall \ \ i, j$$

$$0 \le x_{ij}^{p} , \qquad \forall \ i, j, p$$

For this problem, arc capacity is increased to $M_{ij} = 150$ for all (i, j) to allow for increased flow from additional commodities. We consider two commodities with supplies at *s* equal to 500 and 300 and demands at *t* equal to 500 and 300, respectively. In MCF, preliminary experimentation shows that $\overline{\theta} = 0.997$ tends to provide a valid lower bound of $f_{N_k}^{*}$ in Step 6 of Subroutine PE and we adopt that value for $\overline{\theta}$.

C. COMPUTATIONAL STUDY

For this computational study, we apply the parameters described next to both SCF and MCF. We use the PGM nonlinear-programming algorithm with Armijo step size rule; for example, see, Polak (1997, p. 67) and Bertsekas (1999, p. 31). The quadratic direction-finding problem in the PGM is solved using LSSOL (Gill et al., 1986) as implemented in TOMLAB 7.0 (Holmstrom, 2008). We use parameters $\alpha = 0.5$ and $\beta = 0.8$ in the Armijo step-size rule and in Subroutine PE use an exponential smoothing parameter $\psi = 1/3$ and tolerance $\varepsilon_{\theta} = 0.0001$.

For stopping criterion, we draw a new independent sample of size $N^{*} = 10000$ to evaluate $f_{N^{*}}(x_{n_k}^{k})$ and use a stopping confidence level of $\delta = 0.95$. We use $(\mu_{ij}, \sigma_{ij}^{2}) = (3, 4)$ for all (i, j) as parameters for the log-normal distributed random variable ω_{ij} representing congestion. Arc costs C_{ij} and C_{ij}^{p} are generated from a normal distribution of random numbers with mean 80 and standard deviation 20. Additionally, we set the relative optimality tolerance to 0.01 for use in calculations, i.e., $\varepsilon = 0.01 \hat{f}_{k+1}^{*}$ on stage *k*.

For comparison studies, we consider two versions of Model-Predictive Control; MPC1 and MPC2. In MPC1, Model-Predictive Control is applied to all stages of the conceptual algorithm, including the first stage. We find empirically that MPC might have poor control in the initial stage when the parameters estimated in $\mathbf{S\text{-}SSCP}_k(r_f, r^{*}, r_{\theta}, r_{\sigma})$ are inferior estimates. Hence, we also consider MPC2, where

Model-Predictive Control is used from the second stage of the conceptual algorithm. The first stage uses a predetermined sample size and number of solver iterations. We examine three choices for the first-stage policy resulting in the following three cases of MPC2:

1. *MPC2a*. $n_1 = 450,\ N_1 = 100$, and remaining stages use MPC.
2. *MPC2b*. $n_1 = 600,\ N_1 = 100$, and remaining stages use MPC.
3. *MPC2c*. $n_1 = 900,\ N_1 = 100$, and remaining stages use MPC.

Choices of n_1 for these cases of MPC2 are determined by solving $\bar{\theta}^{n_1} = 0.01$, $\bar{\theta}^{n_1} = 0.05$, and $\bar{\theta}^{n_1} = 0.1$, where $\bar{\theta}$ is the conservative rate-of-convergence coefficient as discussed in Chapter 3, Section B, Subsection 2.

As a basis for comparison, we consider the following heuristic policies:

1. *Fixed policy*. Predetermine N_k and n_k and keep fixed throughout each stage of CA.
2. *Additive policy*. Predetermine n_k and add a predetermined number to N_k at the beginning of each stage of CA (i.e., $N_{k+1} = 50 + N_k$).
3. *Multiplicative policy*. As in additive policy, predetermine n_k, and adjust sample size by a multiplicative factor at the beginning of each stage (i.e., $N_{k+1} = 1.2N_k$).

We use an initial sample size $N_1 = 10$ for all heuristic policies, except for the fixed policy for which $N_k = 0.5 \cdot N^* = 5000$ for all *k*. In order to use the PGM within CA, we must first find an initial feasible solution for both SCF and MCF to start the calculations. We do so by formulating and solving the following linear program in the case of SCF:

SCF-LP: $\min \sum_{(i,j)\in A} C_{ij}x_{ij}$

$$\text{s.t.} \quad \sum_{j:(j,i)\in A} x_{ji} - \sum_{i:(i,j)\in A} x_{ij} = \begin{cases} -D & \text{if } i = s \\ 0 & \text{if } i \in N \setminus \{s,t\} \\ D & \text{if } i = t \end{cases}$$

$$0 \le x_{ij} \le M_{ij}, \quad \forall\ i,j$$

and the following linear program in the case of MCF:

MCF-LP: $\min \sum_{(i,j)\in A} \sum_{p} C_{ij}^{p} x_{ij}^{p}$

$$\text{s.t.} \quad \sum_{j:(j,i)\in A} x_{ji}^{p} - \sum_{i:(i,j)\in A} x_{ij}^{p} = \begin{cases} -D^{p} & \text{if } i = s \\ 0 & \text{if } i \in N \setminus \{s,t\}, \ \forall\ p \\ D^{p} & \text{if } i = t \end{cases}$$

$$\sum_{p} x_{ij}^{p} \le M_{ij}, \quad \forall\ i,j$$

$$0 \le x_{ij}^{p}, \quad \forall\ i,j,p$$

We implement our network-flow problems in Matlab Version 7.7.0 on a desktop computer running Windows XP with 3.73 GHz processor speed and 3.25 GB of RAM. SCF-LP and MCF-LP are solved to find an initial feasible solution for both SCF and MCF using the linear programming solver linprog in the optimization toolbox.

For comparison studies, we record the computing time of CA using each of the MPC policies with the computing time for each of the other policies considered. We evaluate each of the heuristic policies with a different predetermined control on the number of solver iterations. Evaluations are run with n_k for all k set at 5, 25, 50, 75 and 100 iterations. In the additive approach, the sample size is increased by 100 at the beginning of every stage. For the multiplicative approach, two separate heuristics are considered. First, we evaluate the policy with an adjustment to sample size as $N_{k+1} = 1.5N_k$ for all k and then increase the adjustment control on sample size to $N_{k+1} = 2N_k$ for all k.

The results summarized in Table 1 provide average computing times over 20 runs of the CA with standard deviations for the SCF problem. The first column lists the individual policies mentioned above for determining (N_k, n_k). The second and third columns give the average and standard deviations, respectively, of the total computational times to reach a near-optimal objective value in the SCF problem.

	SCF Computational Times (sec.)	
Policy	avg	st dev
MPC1	23.17	3.60
MPC2a, $n_1 = 450$, $N_1 = 100$	14.87	2.67
MPC2b, $n_1 = 600$, $N_1 = 100$	18.38	2.48
MPC2c, $n_1 = 900$, $N_1 = 100$	24.77	3.15
Fixed, $n = 5$	443.30	2.48
Fixed, $n = 25$	638.12	42.84
Fixed, $n = 50$	605.40	50.24
Fixed, $n = 75$	614.86	38.27
Fixed, $n = 100$	631.84	50.60
Additive, $n = 5$	398.07	16.78
Additive, $n = 25$	87.58	7.78
Additive, $n = 50$	49.26	6.76
Additive, $n = 75$	41.54	6.81
Additive, $n = 100$	40.81	9.08
Mult 1.5, $n = 5$	> 1100	-----
Mult 1.5, $n = 25$	81.95	13.54
Mult 1.5, $n = 50$	31.20	6.31
Mult 1.5, $n = 75$	29.85	7.85
Mult 1.5, $n = 100$	27.95	5.93
Mult 2.0, $n = 5$	> 1100	-----
Mult 2.0, $n = 25$	> 1100	-----
Mult 2.0, $n = 50$	77.06	7.52
Mult 2.0, $n = 75$	35.98	7.34
Mult 2.0, $n = 100$	32.56	8.29

Table 1. Average and standard deviation of computing times of CA with Model-Predictive Control (MPC) policies and heuristic policies applied to SCF.

In Table 1, the best heuristic policy with respect to computational time is the multiplicative policy with a multiplicative factor of 1.5 and $n_k = 100$ for all k. In comparison, MPC1 finds a near-optimal objective value nearly five seconds faster and

does so with a 39.3% reduction in standard deviation of computational time over the 20 independent runs. MPC2a improves further still, offering a reduction in computing time of nearly 47%. Additionally, the standard deviation between the independent runs drops 55% as compared to the best heuristic policy. The computational times recorded do not reflect the time required to determine the Model-Predictive Control, i.e., to solve approximately $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$. We elect to exclude this time because for large, real-world problems, computing times for the minimization calculations and checking stopping criterion are expected to be considerably larger than computing times for determining (N_k, n_k).

Several policies considered for SCF return results that are costly regarding computing times. In those cases, we terminate the calculations after 1100 seconds and do not compute averages: see rows 12, 17, and 18 of Table 1. For the policies with times greater than 1100 seconds, the relatively small number of solver iterations is not sufficient to make substantial gains towards f^*. In these cases, N_k grows quite large and computing time suffers from the large sample size. For each of the problems, N_k is limited to 400,000 to avoid exhausting computer memory, and in each of these cases, the sample size grows to this limit.

The results summarized in Table 2 for MCF, provide average computing times over 20 runs of the CA, along with standard deviations.

Policy	MCF Computational Times (sec.) avg	std dev
MPC1	40.70	7.21
MPC2a, $n_1 = 450$, $N_1 = 100$	26.46	4.22
MPC2b, $n_1 = 600$, $N_1 = 100$	28.84	4.35
MPC2c, $n_1 = 900$, $N_1 = 100$	37.57	2.95
Fixed, $n = 5$	> 1000	-----
Fixed, $n = 25$	891.65	47.96
Fixed, $n = 50$	905.67	61.53
Fixed, $n = 75$	905.11	58.92
Fixed, $n = 100$	891.89	57.60
Additive, $n = 5$	612.40	41.05
Additive, $n = 25$	143.12	14.81
Additive, $n = 50$	79.86	9.91
Additive, $n = 75$	66.74	10.75
Additive, $n = 100$	54.86	7.24
Mult 1.5, $n = 5$	> 1000	-----
Mult 1.5, $n = 25$	151.96	33.66
Mult 1.5, $n = 50$	48.66	8.86
Mult 1.5, $n = 75$	44.41	7.51
Mult 1.5, $n = 100$	48.12	12.71
Mult 2.0, $n = 5$	> 1000	-----
Mult 2.0, $n = 25$	> 1000	-----
Mult 2.0, $n = 50$	142.55	46.12
Mult 2.0, $n = 75$	61.95	16.52
Mult 2.0, $n = 100$	52.64	11.67

Table 2. Average and standard deviation of computing times of CA with Model-Predictive Control (MPC) policies and heuristic policies applied to MCF.

As in SCF, a number of the policies make the sample size grow until it hits the limit of 400,000, thereby affecting overall computing times. In these cases we terminate the calculations after 1000 seconds and do not compute averages.

The best heuristic policy for MCF is again a multiplicative policy. However in this larger problem, computational time is best when $n_k = 75$ for all k: compare this to SCF, where computational time is best when $n_k = 100$ for all k. MPC1 improves on this computational time by nearly four seconds and does so with essentially the same variability of computational time between runs as the best heuristic policy.

We see that modifying MPC in the first stage gives further computational savings. Policy MPC2a shows an improvement of 40% in overall computing time, on average, and improves the standard deviation of computing time by almost 44% over the 20 independent runs. These results indicate that while the MPC typically provides a "good" policy for selecting (N_k, n_k), the estimates of parameters in $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$ for the first stage are rather poor and a heuristic policy may be better in that stage.

To verify that the stopping test (3.5) does not cause premature termination of CA, we compute a lower bound on f^* as described in Mak et. al., (1999). Specifically, we run the PGM on the AP with N = 10000 until that algorithm stalls and record the last function value. This is an estimate of f_N^*. We repeat this process 30 times. By the central limit theorem, the average of these function values is approximately normal and provides a lower bound $\underline{f}$ on f^*. We find that in all 160 runs of the MPC policies, the probability that the last solution found is no worse than (1 + 0.01) $\underline{f}$ is essentially 1.0. Hence, the stopping test (3.5) is rather conservative, as zero unsatisfactory solutions is well within the $0.05 \cdot 160 = 8$ expected when $\delta = 0.95$.

MPC1 solves each of the network-flow problems faster than any of the heuristic policies considered. Additional reductions in computing time are gained with MPC2 where the first stage of the CA is primarily used to estimate parameters in $\mathbf{S\text{-}SSCP}_k(r_f, r^*, r_\theta, r_\sigma)$ for $k \geq 2$.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

This thesis develops an efficient optimization algorithm for approximately solving stochastic nonlinear programming problems whose objective functions are sample-average approximations. We demonstrate improvement in computation times by approximately solving a discrete-time optimal-control problem to select a policy of well-balanced sample sizes and number of solver iterations for each stage of the algorithm. This policy, referred to as the Model-Predictive Control policy (MPC), is compared against alternative heuristic policies for selecting sample sizes and solver iterations. MPC approximately solves a single-commodity network-flow problem up to 17% faster, on average, than the best heuristic policy. Furthermore, the optimal-control problem provides a 40% reduction in standard deviation of computing times over a set of independent runs of the algorithm on identical problem instances. When we fix the number of solver iterations in the first stage and then proceed with MPC, we improve the computing time, on average, by nearly 47% and reduce the standard deviation between runs by more than one half.

The application of the discrete-time optimal-control problem to a larger multi-commodity network-flow problem shows an 8.4% improvement, on average, in computational time over the best heuristic policy with essentially the same variation of overall computational time between the 20 independent runs. With the first-stage modification to Model-Predictive Control, we improve computing time by 40%, on average, compared to the same heuristic policy and reduce standard deviation between runs by 44%.

The algorithm developed to solve nonlinear stochastic programs shows considerable promise and offers significant potential for further study.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows*. Prentice-Hall, Upper Saddle River, NJ.

Alexandrov, N. M., Lewis, R. M., Gumbert, C. R., Green, L. L., and Newman, P. A. (2001). Approximation and model management in aerodynamic optimization with variable-fidelity models. *Journal of Aircraft*, 38(6), 1093-1101.

Bergondorff, P., Hearn, D. W., and Ramana, M. (1997). Congestion toll pricing of traffic networks, *Network Optimization*, Pardalos, P. M., Hearn, D. W., and Hager, W. W. (Eds.), *Lecture Notes in Economics and Mathematical Systems*, 450, 51-71.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, Belmont, MA.

Gill, P. E., Hammarling, S. J., Murray, W., Saunders, M. A., and Wright, M. H. (1986). LSSOL 1.0 user's guide. Technical Report SOL-86-1, System Optimization Laboratory, Stanford University. Stanford, CA.

He, L. and Polak, E. (1990). Effective diagonalization strategies for the solution of a class of optimal design problems. *Institute of Electronics and Electrical Engineers Transactions on Automatic Control*, 35(3), 258-267.

Hearn, D. W., Yildirim, M. B., Bai, L., and Ramana, M. (2001). Computational methods for congestion toll pricing models. *Proceedings of IEEE Conference on Intelligent Transportation Systems*, 257-262.

Higle, J. L. and Zhao, L. (2004). Adaptive and nonadaptive samples in solving stochastic linear programs: a computational investigation. *Stochastic Programming E-Print Series*, (accessed on November 2008) http://dochost.rz.hu-berlin.de/spes/.

Holmstrom, K., and Systems Optimization Laboratory Stanford and UC San Diego. (2008). TOMLAB Optimization Environment, TOMLAB/SOL v7.0. (accessed on February 2009) http://tomopt.com/tomlab/.

Mak, W. K., Morton, D. P., and Wood, R. K. (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24, 47-56.

Marcotte, P., and Nguyen, S. (1998). *Equilibrium and Advanced Transportation Modeling*. Kluwer Academic Publishers, New York, NY.

Polak, E. (1997). *Algorithms and consistent approximations*. Springer-Verlag, New York, NY.

Polak, E. and Royset, J. O. (2007). Efficient sample sizes in stochastic nonlinear programming. *Journal of Computational and Applied Mathematics*.

Poojari, C., Lucas, C. and Mitra, G. (2008). Robust solutions and risk measures for a supply chain planning problem under uncertainty. *Journal of the Operations Research Society*, 59(1), 2-12.

Royset, J. O. (2009). Adaptive control of sample size in stochastic optimization. Available from author.

Royset, J. O. and Polak, E. (2007). Extensions of stochastic optimization results toproblems with system failure probability functions. *Journal of Optimization Theory and Application*, 133(1), 1-18.

Royset, J. O. and Polak, E. (2004). Implementable algorithm for stochastic optimization using sample average approximations. *Journal of Optimization Theory Applications*, 122(1), 157-184.

Ruszczynski, A. and Shapiro, A. (2007). Lectures on stochastic programming. *Stochastic Programming Resources*, (accessed on November 2008) http://www.stoprog.org/index.html?resources.html.

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

Shapiro, A. (2000). Stochastic programming by Monte Carlo simulation methods. *Stochastic Programming E-Print Series*, (accessed on November 2008) http://dochost.rz.hu-berlin.de/spes/.

Shapiro, A. and Homem-de-Mello, T. (1998). A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81, 301-325.

Sox, C. R. and Muckstadt, J. A. (1997). Optimization-based planning for the stochastic lot-scheduling problem. *Institute of Industrial Engineers Transactions*, 29(5), 349-357.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Johannes O. Royset
   Naval Postgraduate School
   Monterey, California

4. R. Kevin Wood
   Naval Postgraduate School
   Monterey, California

5. Robert F. Dell
   Naval Postgraduate School
   Monterey, California