Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

DISCRETE STOCHASTIC OPTIMIZATION USING LINEAR INTERPOLATION

Honggang Wang
Bruce W. Schmeiser

School of Industrial Engineering
Purdue University
West Lafayette, IN 47907, U.S.A.

ABSTRACT

We consider discrete stochastic optimization problems where the objective function can only be estimated by a simulation oracle; the oracle is defined only at the discrete points. We propose a method using continuous search with simplex interpolation to solve a wide class of problems. A retrospective framework provides a sequence of deterministic approximating problems that can be solved using continuous optimization techniques that guarantee desirable convergence properties. Numerical experiments show that our method finds the optimal solutions for discrete stochastic optimization problems orders of magnitude faster than existing random search algorithms.

1 INTRODUCTION

Many methods have been developed to solve discrete stochastic optimization (DSO) problems as discussed in (Andradóttir 1998, Fu et al. 2005, Swisher et al. 2000). Most of the approaches for large-size DSO problems seek to find the optimum through the discrete space using random search.

1.1 Problem Statement

We consider discrete optimization problems of the form

(**IP**) find $\underline{x}^*$ such that $\underline{x}^* \in \underset{\underline{x}\in\mathbb{X}}{\operatorname{argmin}}\ g(\underline{x})$

where g is a real-valued function on $\mathbb{X}$, $\mathbb{X} \subseteq \mathbb{Z}^d$ and $\mathbb{Z}^d$ is the set of d-dimensional integer vectors.

The discrete DSO problems of interest have the form of **IP** with the following three assumptions:

Assumption 1. *The function g is bounded on any closed subset of $\mathbb{X}$.*

Assumption 2. *There is a simulation oracle that returns $Y(\underline{x},\underline{\omega})$, for every $\underline{x} \in \mathbb{X}$ and the $\underline{\omega}$ is the random vector defined on some probability space, such that $\overline{Y}(\underline{x}) = \sum_{i=1}^{k} Y(\underline{x},\underline{\omega_i})/k \to g(\underline{x})$ almost surely, as $k \to \infty$, $\forall\, \underline{x} \in \mathbb{X}$.*

Assumption 3. *There exists an optimum set $\mathbb{M}^*$, such that $\mathbb{M}^* \neq \emptyset$ and $\forall \underline{x}^* \in \mathbb{M}^*$, $g(\underline{x}^*)$ is finite.*

We also need following defintions to define our problem statement.

Definition 1. *The local neighborhood of an integer point $\underline{x} \in \mathbb{X}$ is $N(\underline{x}) = \{\underline{x}' : \underline{x}' \in \mathbb{X}, ||\underline{x}' - \underline{x}|| \leq 1\}$, where $\mathbb{X}$ is the feasible set of function g and $||\underline{x}||$ is the Euclidean norm of vector $\underline{x}$.*

Definition 2. *$\underline{x}^* \in \mathbb{X}$ is called a local optimizer of a function g in a feasible region $\mathbb{X}$, if $g(\underline{x}') \leq g(\underline{x})$ $\forall \underline{x}' \in N(\underline{x}^*) \cap \mathbb{X}$.*

Definition 3. *The optimal set $\mathbb{M}^*$ of a function g is the set of all local optimizers of g over the feasible set $\mathbb{X}$.*

We want to design algorithms to find one of the local optimizers $\underline{x}^*$ in the optimal set $\mathbb{M}^*$ of objective function g over set $\mathbb{X}$, given a simulation oracle $Y(\underline{x},\underline{\omega})$ provided that the sample mean of $Y(\underline{x})$ consistently estimates $g(\underline{x})$ for all $\underline{x} \in \mathbb{X}$. Thus, we define our problem statement in a stochastic form of **IP**:

(**ISP**) find an $\underline{x}^*$ that minimizes g locally, provided that a simuluation oracle Y consistently estimates g.

1.2 Background and Motivation

In many practical applications of stochastic optimization, the objective function is usually estimated by an expensive simulation process and the problem **IP** itself is NP-hard. For instance, consider a classical (s,S) inventory problem (Pichitlamken and Nelson 2003, Koenig and Law 1985), where s is the reorder point and S is the order-up-to level. Assume that the level of inventory of units is periodically reviewed, we want to find the inventory levels of units

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

of products for a manufacturing plant such that it yields the maximum expected profits. If the production times are truncated normal distributions and the demand times and quantities are distributed exponentially and uniformly respectively, then the problem cannot be solved analytically.

The objective function is not available analytically but can be evaluated with noise by simulating the stochastic system. The simulation evaluation, however, is computationally expensive and returns only the estimates of the function with limited sample size. One wants to limit the number of function evaluations while get the function estimates at a high-level accuracy. A continuous search can save computing iterations with the guided improving directions. Can we search the discrete space "continuously" by creating gradient-type directions?

1.3 Proposed Method

We propose a continuous search method with linear interpolation for DSO problems. Instead of random search, we search the optimal solutions through a constructed continuous space. Piecewise-linear interpolation creates a continuous response surface that agrees with g at all feasible points in $\mathbb{X}$. We use retrospective optimization approach to solve the constructed continuous stochastic problem with nonlinear derterministic optimization tools.

Kriging (Biles et al. 2007, Huang et al. 2006) is another approach to stochastic optimization via simulation that relies on interpolation of the response surface.

2 CONSTRUCTION OF CONTINUOUS RESPONSE USING LINEAR INTERPOLATION

A continuous response surface that connects all integer response points of **ISP** reveals the curvature information of its objective function g. This curvature of g often helps search algorithms mover faster toward the optimum of g. Simplex interpolation creates piecewise-linear continuous surface fast and offers nice properties.

2.1 Continuous Relaxation

To do continuous search in the solution space we create a continuous response surface for the discrete problem **IP**. Nonetheless the objective function in **IP** or in **ISP** is often defined at $\underline{x} \in \mathbb{X}$ in the discrete space. Doing continuous relaxation by dropping the integral constraints on these variables is typically not applicable because a fractional number of machines or workers does not make any sense in real world. A half of machine or 1.4 workers or machines are not defined in our simulation model.

Although the function g is not defined at non-integer points, we like to be able to search on these fractional points, since that provides us the possibility using the faster and more-reliable continuous search methods to solve discrete optimization problems. We want to define, for instance, what $\underline{x} = \{4.6, 2.3, 0.8\}$ means in an integer parameter setting problem. How can 4.6 machines be simulated in the simulation model? Some type of continuous interpolation is required on the integral points. Coupling with data interpolation, we create a continuous function $f : \mathbb{Q} \to \mathbb{R}, \mathbb{Q} \subseteq \mathbb{R}^d$ and $\mathbb{Q} \bigcap \mathbb{Z}^d = \mathbb{X}$, such that $f(\underline{z}) = g(\underline{z}),\ \forall \underline{z} \in \mathbb{X}$. The original integer problem is relaxed in the sense that the domain of the objective function of **IP** is now a connected subset $\mathbb{Q}$ in Euclidean space $\mathbb{R}^d$. This relaxation is different from the usual continuous relaxation, since except the grid integral points all the continuous points do not have any real values and physical meanings in our problems. The function values are created at fractional points only for the purpose of continuous search.

2.2 Multiple-Dimensional Continuous Interpolation

In multiple-dimensional Euclidean space, interpolating data on grid points is an easy way to create a continuous function with desirable features through the whole space. Data interpolation provides a continuous function that agrees with true function values g at every integral point $\underline{x} \in \mathbb{X}$. The extreme points in the domain of the interpolated function are always integral point in $\mathbb{X}$.

We can use multi-linear interpolation (Weiser and Zarantonello 1988) on the entire unit cube, however, the dimensionality issue implies that d-cube interpolation may consume a lot of computing resource, as the number of vertices of a unit hyper-cube is 2^d. A simplex interpolation is a faster alternative, because it requires only $d+1$ vertices for calculating the function value at any point $\underline{x}$ in that simplex.

2.3 Simplex Piecewise-Linear Interpolation

We use the method described in (Weiser and Zarantonello 1988) and (Rovatti et al. 1998) to interpolate the integer-ordered data on every feasible simplex defined by integer vertices in $\mathbb{X}$. The interpolation partitions the real set $\mathbb{Q}$ to a collection of closed simplices $\{S_i\}_{i \in K}$, where S_i denotes a unit simplex defined by neighboring integer points in $\mathbb{X}$. We define the simplex-interpolation routine as follows.

Routine SI
Given: A point $\underline{x} \in \mathbb{Q}$ and a function g on $\mathbb{X}$;
Return: $f(\underline{x})$.

(i) Locate the unit cube containing $\underline{x}$ by identifying the "lower left" vertex of the cube, which is $\lfloor \underline{x} \rfloor$;
(ii) Let $\underline{x}' = \underline{x} - \lfloor \underline{x} \rfloor$, a point in the unit cube;
(iii) Sort the components of $\underline{x}'$ to obtain $\underline{x}'^{p(1)} \geq \underline{x}'^{p(2)} \geq \ldots \geq \underline{x}'^{p(d)}$. As a permutation of $\{1, \ldots, d\}$, $\{p(1), p(2), \ldots, p(d)\}$ defines a simplex enclosing

$\underline{x}$ with the vertices $\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_d$, where $\underline{x}_0 = \lfloor \underline{x} \rfloor$, $\underline{x}_i = \underline{x}_{i-1} + \underline{e}_i$, and $\underline{e}_i$ is the unit vector with the $p(i)$th element equal to 1, for $i = 1, 2, \ldots, d$;

(iv) construct the function f using

$$f(\underline{x}) = \sum_{i=0}^{d} \left[g(\underline{x}_i)(\underline{x}'^{p(i)} - \underline{x}'^{p(i+1)}) \right], \tag{1}$$

where $\underline{x}'^{p(0)} = 1$ and $\underline{x}'^{p(d+1)} = 0$. For a fixed simplex, the integral components $\lfloor \underline{x} \rfloor$ of a design point $\underline{x}$ are constant integers; thus Equation 1 is a linear function of $\underline{x}$.

Replacing the deterministic function values $g(\underline{x}_i)$ in (1) with the stochastic simulation oracle $Y(\underline{x}_i, \underline{\omega})$, for $i = 0, 1, \ldots, d$, we obtain the stochastic form to estimate the function value $f(\underline{x})$ at any point $\underline{x} \in \mathbb{Q}$. We extend $Y(\underline{x})$ to the real set $\mathbb{Q}$ based on simplex-interpolation routine SI

$$Y(\underline{x}, \underline{\omega}) = \sum_{i=0}^{d} \left[Y(\underline{x}_i, \underline{\omega})(\underline{x}'^{p(i)} - \underline{x}'^{p(i+1)}) \right]. \tag{2}$$

From Assumption 2, Equation (2) allows the consistent estimation of $f(\underline{x})$ at any $\underline{x} \in \mathbb{Q}$ using the sample average $\overline{Y}_k(\underline{x})$.

2.4 Constructed Continuous Response Surface

We present some of the properties of simplex interpolation discussed in (Weiser and Zarantonello 1988) without proofs.

Proposition 1. $f(\underline{x}) = g(\underline{x}), \forall \underline{x} \in \mathbb{X}$.

Proposition 2. *The constructed function f is continuous and piecewise linear on its domain $\mathbb{Q}$.*

Proposition 3. *Given a point $\underline{x}$ in $\mathbb{R}^d$, the simplex enclosing $\underline{x}$ created by the interpolation is uniquely identified.*

Proposition 4. *The set of created simplices is a partition of the domain $\mathbb{Q}$ of f.*

The simplex-interpolation also provides following proposition:

Proposition 5. *An integer optimum of g is also an optimal point of f.*

With these properties provided by the simplex interpolation, the **ISP** problem is converted to a continunous stochastic optimization (CSO) problem with desirable characteristics:

(**CSP**) Find a local optimizer $\underline{x}^*$ of function f over the feasible set $\mathbb{Q}$, where f can be consistently estimated by the simulation oracle $Y(\underline{x}, \underline{\omega})$, where $\underline{\omega}$ is the random effects in the simulation model, $\mathbb{Q} \subseteq \mathbb{R}$, and $\mathbb{Q} \cap \mathbb{Z}^d = \mathbb{X}$.

3 RETROSPECTIVE OPTIMIZATION FRAMEWORK FOR STOCHASTIC PIECEWISE-LINEAR OPTIMIZATION PROBLEMS

With the construction of continuous response surface using simplex interpolation, we create a class of continuous stochastic optimization (CSO) problems of the form **CSP**. The objective functions of these CSO problems are continuous and piecewise linear on their domains. The classical Stochastic Approximation (Kushner and Clark 1978) approach cannot be applied to these CSO problems due to the nondifferentiability of objective functions at the boundary points. Although CSO problems generally are not continuously differentiable, the piecewise linearity makes them tractable due to the developed theories of nondifferentiable analysis. In particular if CSO is convex, there are many convex optimization tools adaptable for stochastic settings, e.g. stochastic subgradient methods (Shor 1985) and conjugate gradient search (Hiriart-Urruty and Lemarechal 1993).

The Retrospective Optimization (RO, also called sample average approximation method) approach offers us a general setting in that we can use deterministic nonlinear optimization tools for solving a convergent sequence of approximating problems of **CSP**. The sequence of the sample path problems converges to **CSP** w.p.1 as the sample size goes to infinity under the Assumption 2.

3.1 The Sample-Path Approximation Approach

Healy and Schruben (1991) propose the retrospective approach for stochastic optimization. The fundamental idea of the RO algorithms is simple. It generates a sample of the random inputs and uses the sample average of system observations to approximate f in **CSP**. With a fixed sample of random vectors, **CSP** becomes deterministic. One can solve the approximating problem with deterministic optimization tools.

As discussed in (Spall 2005), retrospective optimization approach has some noticeable advantages: (a) there are many powerful deterministic search methods applicable and robust optimization tools for handling complex constraints; (b) for a fixed sample path, it is possible to conduct a large number of iterations of deterministic search at nominal cost.

Chen and Schmeiser (1994) propose the retrospective approximation in the context of stochastic root finding. Instead of solving one sample-path problem, they generate and solve a sequence of sample-path problems $\{\mathbf{P}_k\}$ with increasing sample sizes $\{m_k\}$ and decreasing sequence of tolerance errors. Jin and Schmeiser (2003) develop a retrospective optimization algorithm to solve continuous stochastic optimization problems with general constraints. Shapiro and Wardi (1996) analyze the convergence properties of a gradient-based implementation of RO method. Kleywegt et al. (2001) use the RO approach to solving a

stochastic knapsack problem. They have studied the convergence rates, stopping rules, and computational complexity of their RO algorithms.

3.2 The Sample-Path Approximation of Problem CSP

Recall that we are interested in solving a CSO problem of the form **CSP**: find a local optimizer $\underline{x}^*$ of function f over $\mathbb{Q}$, where f can be estimated by $Y(\underline{x},\underline{\omega})$, $\underline{\omega}$ is the random effects in the simulation model, the connected set $\mathbb{Q}\subseteq\mathbb{R}^d$, and $\mathbb{Q}\cap\mathbb{Z}^d=\mathbb{X}$.

We put problem **CSP** in the RO setting by randomly generating an independent and identically distributed (i.i.d.) sample of m_k realizations of random vector $\underline{\omega}$. The sample path function is denoted as $\hat{f}_k(\underline{x},\xi_k)=\sum_{i=1}^{m_k}Y(\underline{x},\underline{\omega}_i)/m_k$ and a deterministic counterpart problem of the kth sample path is defined as follows.

($\mathbf{P}_k$) Find a local optimal solution $\underline{x}_k^*$ of $\hat{f}_k(\underline{x},\xi_k)$ over a feasible set $\mathbb{Q}\subseteq\mathbb{R}^d$, where $\xi_k=\{\underline{\omega}_1,\ldots,\underline{\omega}_{m_k}\}$ is independently generated realizations of random vector that is fixed at any feasible point for this particular sample path.

The constructed objective function $\hat{f}_k$ of $\mathbf{P}_k$ is piecewise linear, differentiable almost everywhere (nondifferentiable on a set of Lebesgue measure zero), and consequently the generalized gradient (Clarke 1990) is well defined at every feasible point. These properties allow us to adopt nonsmooth optimization techniques such as bundle methods (Kiwiel 1985) and r-algorithm (Shor 1985) to solve the problem $\mathbf{P}_k$.

Bundle methods basically are descent or ascent methods using bundles of information, including function values and generalized gradients of multiple trial points, to calculate the improving direction and the distance to search. For piecewise-smooth functions, bundle methods have been proved to converge to a local optimum and many numerical results show that they are probably the most promising algorithms for optimizing general nonsmooth nonconvex functions (Haarala, Miettinen, and Mäkelä 2007).

Shor (1985) created a method using operation of space dilation in the direction of the difference of two successive almost-gradients, called r-algorithm. Under some weak conditions r-algorithm guarantees the local convergence for piecewise-smooth functions.

There are other methods like sampling gradient method, quasi-newton BFGS, modified interior point method, etc, that may solve our sample path problems $\mathbf{P}_k$. But they may not be efficient for $\mathbf{P}_k$ due to nonsmoothness and nonconvexity of $\hat{f}_k$.

3.3 Retrospectively Optimizing Problem CSP

RO algorithms recursively solve sample path problems for the estimates of $\underline{x}_k^*$. The algorithms need to define rules for creating a sequence of sample path problems. Typical settings for a RO algorithm include: determining the increasing sample sizes $\{m_k\}$; a strategy for sampling $\xi_k=\{\underline{\omega}_1,\ldots,\underline{\omega}_{m_k}\}$ across iterations; calculation of a starting solution for each $\mathbf{P}_k$; determining a decreasing sequence of error tolerance $\{\varepsilon_k\}$ for sample path problems $\{\mathbf{P}_k\}$. In addition, a deterministic method must be chosen to solve each sample path problem $\mathbf{P}_k$ before we develop an efficient retrospective optimization method.

Seeking an exact solution for each $\mathbf{P}_k$ is time consuming and not necessary since $\mathbf{P}_k$ itself is an approximating problem of true problem **CSP**. Thus approximated ε_k-solutions for those sample path problems are desirable.

4 ALGORITHM DESIGN

4.1 Introduction

The classic steepest-decent method is easy to implement for optimizing differentiable functions to find stationary points. Our problem in **CSP** is not differentiable everywhere thus the usual gradient-based method may not work. We design an algorithm using generalized gradient to extend the smooth descent search for nonsmooth problems. Instead of the usual gradient we use one arbitrary subgradient in the set of subdifferentiable to calculate the descent direction. A generalized-gradient algorithm has the global convergence for convex problems in general, but it solves the inventory problem quite efficiently.

Besides the calculation of descent direction, selection of step sizes is important for the performance of most gradient-type numerical search algorithms. Due to the discreteness of the original problem **IP**, the step sizes for the problem **CSP** is not helpful if they get too small. We are searching for the optimal integer solution not the exact continuous solution to **CSP**. Thus we are not quite interested in some specific points in the domain $\mathbb{Q}$ of problem **CSP**, but the optimal region or simplex in $\mathbb{Q}$ is particularly concerned.

An increasing sequence of sample sizes are determined to generate sample-path problems $\{\mathbf{P}_k\}$. Suppose the initial sample size is m_0, we increase the sample size by multiplying a constant $c>1$ but $c<2$ so that it increases but not too fast. In the early search, with a small sample size m_k, $\mathbf{P}_k$ can be solved quickly; while within later iterations, with a large sample size but a better starting solution and information from past sample paths, $\mathbf{P}_k$ still can be efficiently solved.

We want to save the computing time by evaluating few function values. The simplex linear interpolation approach allows us to evaluate the function at any continuous point by simulating $(d+1)$ integer points. For a high dimensional problem computing time for those function evaluations is still quite much. Especially at the early stage of solving process with a small sample size, we can estimate a function value with partial number of vertices of the associated simplex, i.e. estimating $f(\underline{x})$ with fewer than $(d+1)$ integer vertices of the simplex enclosing $\underline{x}$. Approximating $f(\underline{x})$ with a partial

number of simplex vertices associated is more reasonable if $\underline{x}$ is close to the boundary of this simplex.

4.2 A Linear Interpolation Search Algorithm Using Generalized Gradient

We implement a simplex-interpolation search algorithm based on the descent method with inexact line search for the problem **CSP**.

Algorithm LIS0
Given: a starting point $\underline{x}_0 \in \mathbb{Q}$, initial sample size m_0, the initial step size s_0, a sample size multiplier $c > 1$, and the simulation oracle returning $\overline{Y}(\underline{x}, \underline{\omega})$ for all $\underline{x} \in \mathbb{Q}$ and random vector $\underline{\omega}$.
Find: an optimal design point $\underline{x}^*$ that minimizes $g(\underline{x})$ locally over $\mathbb{X}$.

Step 0 Set the iteration counter $k = 0$;
Step 1 Generate random inputs $\xi_k = \{\underline{\omega}_1, \ldots, \underline{\omega}_{m_k}\}$, independent of $\xi_0, \ldots, \xi_{k-1}$;
Step 2 Search:
- (i) Simulate at $\underline{x}_k$ to obtain $\widehat{f}(\underline{x}_k) = \overline{Y}_{m_k}(\underline{x}_k, \xi_k)$, and the estimate $\underline{\hat{v}}$ of the generalized gradient $\underline{v}$ at $\underline{x}_k$;
- (ii) Set $i = 0$;
- (iii) Set $\underline{x}_{\text{new}} = \underline{x}_k$;
- (iv) Search in direction $\underline{\hat{v}}$:
  - (a) Set $i = i + 1$;
  - (b) Set $s = 2^{i-1} \times s_0$;
  - (c) Set $\underline{x}_{\text{old}} = \underline{x}_{\text{new}}$;
  - (d) Set $\underline{x}_{\text{new}} = \underline{x}_{\text{old}} + s \times \underline{\hat{v}}/||\underline{\hat{v}}||$;
  - (e) If $\underline{x}_{\text{new}} \notin \mathbb{Q}$, project $\underline{x}_{\text{new}}$ back to $\mathbb{Q}$;
  - (f) Simulate at $\underline{x}_{\text{new}}$ with sample size m_k to calculate $\hat{f}(\underline{x}_{\text{new}}) = \overline{Y}_{m_k}(\underline{x}_{\text{new}}, \xi_k)$;
  - (g) If $\hat{f}(\underline{x}_{\text{new}}) < \hat{f}(\underline{x}_{\text{old}})$, go to (a);
- (v) Set $\underline{x}_k = \underline{x}_{\text{old}}$;
  If $i > 1$, go to Step 2;

Step 3 Iterate:
- (i) Set $k = k + 1$;
- (ii) Set sample size $m_k = c \times m_{k-1}$;
- (iii) Set $\underline{x}_k = \underline{x}_{k-1}$;
- (iv) Go to Step 1;

The above algorithm produces a sequence of continuous solutions $\{\underline{x}_k\}$ that estimate the true optimal solution of **CSP** better as the sample size m_k goes larger. We can round $\underline{x}_k$ to the closest integer solution or pick the best integer solution from the last RO iteration to get an estimate of the optimal integer solution of **ISP**.

5 NUMERICAL EXPERIMENTS

We apply our algorithm to optimize the assemble-to-order system studied in (Hong and Nelson 2006). The performance of our linear-interpolation search method is compared with that of COMPASS by several numerical tests.

5.1 An Assemble-to-Order Inventory Problem

We study the exactly same instance of assemble-to-order problem as presented in (Hong and Nelson 2006). The inventory system has eight items $I_1, I_2, \ldots, I_8$ and five customers. Customers arrive in Poisson times with different rates, $\lambda_1, \lambda_2, \ldots, \lambda_5$, and each of them orders a set of required key items and optional items. Customer arrives to check the stock of all of key items; if they are in stock the customer buys the key items plus the available optional items, otherwise leaves without an order. Each item has sale profit p_i, holding cost h_i and inventory capacity C_i, $i = 1, 2, \ldots, 8$. The production time for item I_i is normally distributed with mean μ_i and variance σ_i^2 truncated at 0, $i = 1, 2, \ldots, 8$.

In each simulation replication we set a warm-up period of 20 units of time and then average profits over the rest of time. We want to find the reorder inventory levels $\{x_1, \ldots, x_8\}$ to maximize the expected total profit.

5.2 Experimental Results

Within each sample path, our algorithm uses the common random numbers and the same sample size for all the design points visited. The sequence of sample-path problems increase sample sizes by 10% each time and use the random initial seed values. We run 10 independent macroreplications with random starting solutions and different initial seeds using our LIS0 algorithm. Figure 1 shows 10 independent simulation runs of our algorithm. The dot-dash line shows the performance of COMPASS solving the same problem. Figure 2 (Hong and Nelson 2006) shows the performance of COMPASS compared to those of simulated annealing and random search. In the inventory example, for the same quality of solution COMPASS requires roughly one-hundred times the computing effort of LISO. COMPASS solves a transient problem but it cannot optimize the steady-state systems. Our algorithm solves the steady-state assemble-to-order problem with similar results to those of the transient case presented here.

Our LIS0 algorithm finds the optimal solution faster and more efficiently especially for problems with large feasible sets. Figure 3 shows its search path for an assemble-to-order problem with a feasible set of integer vectors $[1, 1000]^8$ which is much larger than the previous instance of problem. Random search methods like COMPASS can be quite slow. Our algorithm finds the optimal solution of problems having

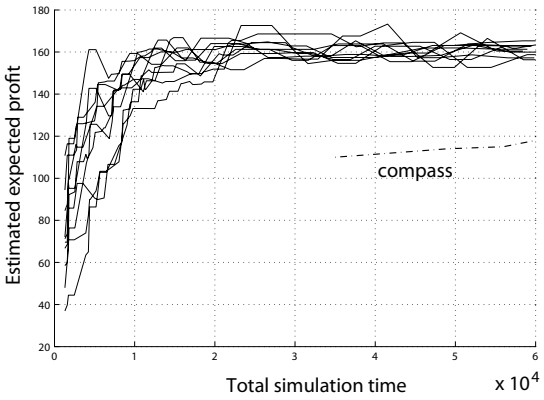

Figure 1: Optimizing an assemble-to-order system

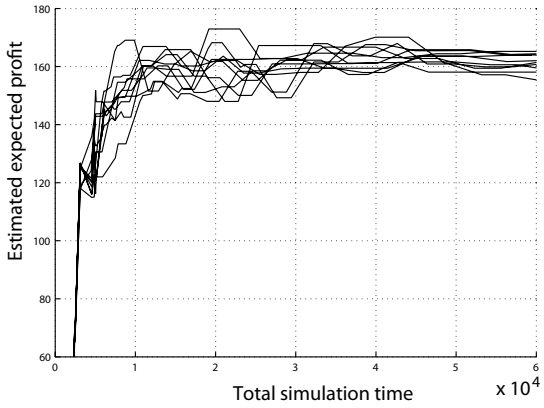

Figure 3: Optimizing an large assemble-to-order system

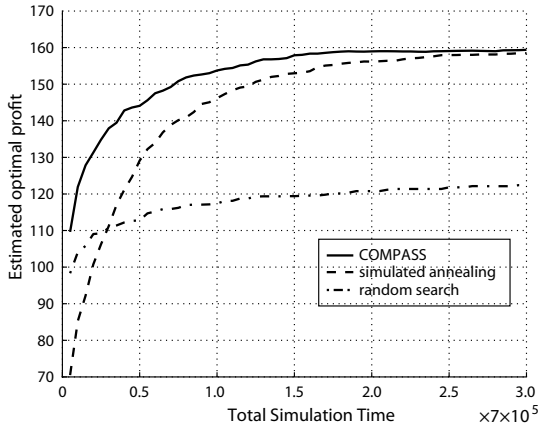

Figure 2: Performance of COMPASS (Hong and Nelson 2006) compared to random search and simulated annealing

large solution space with a small increase of computing time.

6 SUMMARY

The proposed LIS0 algorithm using linear interpolation gives promising results in our numerical experiments. Convergence analysis and new algorithm design are our future work.

REFERENCES

Andradóttir, S. 1998. A review of simulation optimization techniques. *Proceedings of the 1998 Winter Simulation Conference*:151–158.

Biles, W., J. Kleijnen, and I. Nieuwenhuyse. 2007. Kriging metamodeling in constrained simulation optimization: an explorative study. *Proceedings of the 2007 Winter Simulation Conference*:355–362.

Chen, H., and B. Schmeiser. 1994. Retrospective approximation algorithms for stochastic root finding. *Proceedings of the 1994 Winter Simulation Conference*:255–261.

Clarke, F. 1990. *Optimization and nonsmooth analysis*. Society for Industrial and Applied Mathematics, Philadelphia.

Fu, M., F. Glover, and J. April. 2005. Simulation optimization: a review, new developments, and applications. *Proceedings of the 2005 Winter Simulation Conference*:83–95.

Haarala, N., K. Miettinen, and M. Mäkelä. 2007. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Math. Program.* 109 (1): 181–205.

Healy, K., and L. Schruben. 1991. Retrospective simulation response optimization. *Proceedings of the 1991 Winter Simulation Conference*:901–906.

Hiriart-Urruty, J., and C. Lemarechal. 1993. *Convex analysis and minimization algorithms I and II*. Springer-Verlag.

Hong, L. J., and B. L. Nelson. 2006. Discrete optimization via simulation using compass. *Operations Research* 54 (1): 115–129.

Huang, D., T. Allen, W. Notz, and N. Zeng. 2006. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization* 34 (3): 441–466.

Jin, J., and B. Schmeiser. 2003. Simulation-based retrospective optimization of stochastic systems. *Proceedings of the 2003 Winter Simulation Conference*:543–547.

Kiwiel, K. 1985. *Methods of descent for nondifferentiable optimization*. Springer-Verlag.

Kleywegt, A., A. Shapiro, and T. Homem-De-Mello. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optimization* 12 (2): 479–502.

Koenig, L., and A. Law. 1985. A procedure for selecting a subset of size m containing the l best of k independent normal populations, with applications to simulation. *Comm. in Statist.: Simulation and Comput.* 14:719–734.

Kushner, H., and D. Clark. 1978. *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, New York.

Pichitlamken, J., and B. Nelson. 2003. A combined procedure for optimization via simulation. *ACM Trans. Modeling and Comput. Simulation* 13:155–179.

Rovatti, R., M. Borgatti, and R. Guerrieri. 1998. A geometric approach to maximum-speed n-dimensional continuous linear interpolation in rectangular grids. *IEEE Transactions on Computers* 47 (8): 894–899.

Shapiro, A., and Y. Wardi. 1996. Convergence analysis of gradient descent stochastic algorithms. *Journal on Optimization theory and application* 91 (2): 439–454.

Shor, N. 1985. *Minimization methods for non-differentiable functions*. Springer-Verlag.

Spall, J. 2005. *Introduction to stochastic search and optimization*, 425–432. John Willey & Sons.

Swisher, J., P. Hyden, S. Jacobson, and L. Schruben. 2000. A survey of simulation optimization techniques and procedures. *Proceedings of the 2000 Winter Simulation Conference*:119–128.

Weiser, A., and S. Zarantonello. 1988. A note on piecewise linear and multilinear table interpolation in many dimensions. *Mathematics of Computation* 50 (181): 189–196.

AUTHOR BIOGRAPHIES

HONGGANG WANG is a PhD candidate in Shool of Industrial Engineering at Purdue University. His research interests include discrete stochastic optimiztion via simulation, stochastic process and modeling, and stochastic optimization in supply chain management.

BRUCE W. SCHMEISER is a professor of Industrial Engineering at Purdue University. His research interests center on developing methods for better simulation experiments. He is a fellow of INFORMS and IIE, and has been active within the Winter Simulation Conference for many years.