NUMERICAL COMPARISONS OF BLACK BOX SIMULATION OPTIMIZATION METHODOLOGIES: RSM AND SPSA

(KARA KUTU BENZETİM MODELLERİNİ ENİYİLEYEN YÖNTEMLERİN NÜMERİK KARŞILAŞTIRILMALARI : RSM VE SPSA)

by

Makbule Rengin BURHANOĞLU, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Date of Submission : May 4, 2009

Date of Defense Examination: June 3, 2009

Supervisors : Asst. Prof. Dr. M. Ebru ANGÜN
Assoc. Prof. Dr. Esra ALBAYRAK

Committee Members: Assoc. Prof. Dr. Temel ÖNCAN
Assoc. Prof. Dr. Mehmet Mutlu YENİSEY
Asst. Prof. Dr. Cafer Erhan BOZDAĞ

ACKNOWLEDGEMENTS

I would like to thank Yrd. Doç. Dr. M. Ebru Angün and Doç. Dr. Esra Albayrak, for their guidance and support throughout the preparation of this study despite all difficult conditions around us.

I would also like to thank Rengin Burhanoğlu for her friendship of the past ten years, her permanent desire to be my colleague in every project at Galatasaray University and just for being herself any time.

I would also like to extend special thanks to my manager at work; Ayşe Yenidoğan who provided me with the time I needed to finalize this thesis, and who showed an endless patience for my mind divided in three for the past six months.

Finally, I would like to thank my family for their love and trust in me and my husband Yılmaz Cem Şenol.

SEDA EYİGÜN
JUNE 2009

ACKNOWLEDGEMENTS

I would like to thank my professors Yrd. Doç. Dr. M. Ebru ANGÜN and Doç. Dr. Esra Albayrak for their guidance and support throughout the preparation of this study.

I would also like to thank Doç. Dr. Orhan FEYZİOĞLU and my friends İlke BEREKETLİ and Emre YAMANGİL for their trust and support during my master study.

Finally, I would like to thank Seda EYİGÜN for her friendship of the past ten years, her permanent desire to be my colleague in every project at Galatasaray University and just for being herself any time, and my family for their love.

Makbule Rengin BURHANOĞLU
JUNE 2009

TABLE OF CONTENTS

LIST OF SYMBOLS

CFD: Central Finite Differencing

FDSA: Finite Difference

FFD: Forward Finite Differencing Stochastic Approximation

EOQ: Economic Order Quantity

IPA: Infinitesimal Perturbation Analysis

LR/SF: Likelihood Ratio/Score Function

MSR: Mean Squared Residual

PA: Perturbation Analysis

RSM: Response Surface Methodology

SA : Stochastic Approximation

SP: Simultaneous Perturbation

SPSA: Simultaneous Perturbation Stochastic Approximation

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

By the developments in computer technologies, there has been a significant increase on academic studies and practical applications about simulation optimization in the last decades. The major advantage of this augmentation, which also positively affects commercial software, is the opportunity of examiner a large number of design solutions and to obtain the values of various performance measures compared to the traditional methods.

Simulation optimization methods can be classified as white box methods and black box methods. While white box methods enabling the practitioners to estimate the random responses and their gradient vectors, black box methods use the simulation outputs of the random responses to approximate their gradient vectors. Although white box methods outperform black box methods, black box methods shall be preferred in a case which the knowledge of the stochastic structure of the system is not available to the practitioners.

In this study, performances of black box methods Response Surface Methodology and Simultaneous Perturbation Stochastic Approximation are compared numerically considering that the computer budget is limited and each simulation run is time-consuming. Comparisons are realized on the newsvendor problem, (*s*, *S*) inventory model and generic call center problem.

The computer program of RSM and SPSA algorithms are coded in *Matlab 7.6* for the newsvendor problem and (*s*, *S*) inventory model. The call center model is first designed in *Arena*, and then RSM and SPSA steps are applied manually based on the results get by this model.

RESUME

Grace aux developments à la technologie de l'enformatique, dans les dernières années, il exists un remarquable augmentation aux travoux académiques et pratiques sur l'optimisation de simulation. L'avantage le plus grand de cette augmentation, qui a aussi affecté positivement les nombres des logicielles commerciales, est de l'opportunité d'examiner nombreauses solutions de dessins et d'obtenir les valeurs de plusieurs mesures de performance quand on compare aux méthodes traditionelles.

Les méthodes d'optimisation de simulation peuvent etre classifiées comme des méthodes de boîte blanche et des méthodes de boîte noire. Les méthodes de boîte blanche permettent l'utilisateur d'estimer les critères de performance du modèle de simulation avec ses vecteurs gradyans, quand les méthodes de boîte noire permettent l'utilisateur seulement d'obtenir les estimations de gradyans des critères de performance en utilisant les données de la simulaiton. Bien que les méthodes de boîte blanche soient plus efficace que les méthodes de boîte noire, il est mieux de préférer les méthodes de boîte noire dans un cas où l'information de la structure probabilistic du system n'est pas disponible pour l'utilisateur.

Dans ce travail, on a comparé numériquement les performances de deux méthodes de boîte noire, 'Response Surface Methodology' et 'Simultaneous Perturbation Stochastic Approximation'. On a assumé que le budget de l'ordinateur est limité et chaque tour de la simulation consomme beaucoup de temps. On a comparé les methods sur le problème de vendeur de journal, sur le modèle d'inventaire (*s*, *S*) et sur le problème de centre d'appel.

Pour le problème de vendeur de journal, sur le modèle d'inventaire (*s*, *S*), le programme informatique de RSM et SPSA est codé dans *Matlab 7.6.* Le modèle de centre d'appel est designé dans *Arena*, puis RSM et SPSA est appliqué sur ce modèle.

ÖZET

Bilgisayar teknolojisindeki gelişmeler sayesinde, son yıllarda benzetim modellerinin eniyilemesi üzerindeki akademik çalışmalar ve pratik uygulamalarda belirgin bir artış görülmüştür. Piyasadaki ticari bilgisayar programlarının sayısını da olumlu yönde etkileyen bu artışın en büyük avantajı ise geleneksel metotlara göre daha fazla sayıda tasarım çözümü deneyebilme ve değişik performans ölçütlerini değerlendirebilme imkânı sağlamasıdır.

Benzetim modellerini eniyileyen yöntemler beyaz ve kara kutu yöntemler olarak ikiye ayrılır. Beyaz kutu yöntemleri, kullanıcıya benzetim modelinin performans kriterlerinin tahminlerinin yanında performans kriterlerinin gradyent tahminlerini de sağlarken, kara kutu yöntemleri yalnızca benzetim modelinin çıktılarını kullanarak performans kriterlerinin gradyent tahminlerini sağlar. Beyaz kutu yöntemlerinin, kara kutu yöntemlerine göre daha etkin olmasına karşın, ele alınan problemin yapısının bir genelleştirilmiş yarı Markov süreciyle modellenemeyeceği durumlarda kara kutu yöntemleri tercih edilmelidir.

Bu çalışmada, bilgisayarın benzetim modelleri için kullanacağı sürenin kısıtlı olduğu ve benzetim programının her bir adımının bilgisayarda çalıştırılmasının uzun sürdüğü varsayımları altında, kara kutu yöntemlerinden Response Surface Methodology ve Simultaneous Perturbation Stochastic Approximation'ın performansları nümerik olarak karşılaştırılmıştır. Karşılaştırmalar, klasik gazeteci çocuk problemi, (s, S) envanter modeli ve çağrı merkezi problemi üzerinde yapılmıştır.

Klasik gazeteci çocuk problemi ve (s, S) envanter modeli icin RSM ve SPSA programı *Matlab 7.6* kullanılarak kodlanmıştır. Çağrı merkezi problemi ise, *Arena*'da

modellenmiş, buradan çıkan sonuçlar kullanılarak RSM ve SPSA manuel olarak koşulmuştur.

1. INTRODUCTION

The term ‘simulation optimization’ has become widespread in both academical and practical studies. From the academical point of view, simulation optimization has become one of the new entries in the updated second edition of the ‘Encyclopedia of Operations Research and Management Science’ [1]. Furthermore, many surveys and panel discussions about the future of simulation optimization, and its methodologies and applications have been published; see Tekin and Sabuncuoglu [2]; Fu [3]; Andradóttir et al. [4]; Azadivar [5]; Andradóttir [6]; also all the Winter Simulation Conference proceedings, which are available online at the website [7]. From the practical point of view, optimization modules have been recently implemented in many commercial discrete-event simulation packages; Table 1.1 shows simulation packages and optimization methods incorporated into these packages [3].

We can explain one of the many reasons for the interest in simulation optimization, as follows. For problems that arise in practical applications, explicit mathematical formulations may be too restrictive; that is where simulation is relevant. Therefore, for many practical cases one cannot obtain an analytical solution through methods that require explicit mathematical formulations. Indeed, simulation optimization has led to the numerical solution of large-scale, real-world decision-making problems; see, for example, Azadivar and Truong [8], April et al. [9], and Martin and Schouwenaar [10].

In this thesis, we mainly consider stochastic simulation. Moreover, we focus on black box simulation optimization methods - which we will detail later in this chapter - because of their generality, and their ease of use and implementation. To illustrate their generality, we give the following references that apply black box simulation optimization methods to either deterministically or stochastically simulated systems in very diverse fields, such as engineering design [11] and ergonomic design of

workstations [12] - where both papers use deterministic simulation -, and air traffic control [13], production planning [14], and design of manufacturing cells [15] - where all three papers use stochastic simulation.

Table 1.1 An overview of commercial software packages used in simulation optimization

Optimization package (simulation platform)	*Vendor (URL)*	*Primary Search Strategies*
AutoStat (AutoMod)	Auto Simulations, Inc. (www.autosim.com)	Evolutionary, genetic algorithms
OptQuest (Arena, CrystalBall, et al.)	Optimization Technologies, Inc. (www.opttek.com)	Scatter search and tabu search, neural networks
OPTIMIZ (SIMUL8)	Visual Thinking International Ltd. (www.simul8.com)	Neural networks
SimRunner (ProModel)	PROMODEL Corp. (www.promodel.com)	Evolutionary, genetic algorithms
Optimizer (WITNESS)	Lanner Group, Inc. (www.lanner.com/corporate)	Simulated annealing, tabu search

In the rest of this thesis, we do not consider screening, which can be considered as stage zero (pre-processing phase) of any black box simulation optimization methods. For screening, we give the following brief description and references. Screening is the process of searching for the few truly important input variables among the great many potentially important input variables that affect a system's performance; for a recent reference, see Dean and Lewis [16]. Trocine and Malone [17] compares and contrasts screening methods in terms of efficiency, effectiveness, and robustness. As screening methods, classical factorial designs [18], two-stage group screening [19], sequential bifurcation [20, 21], and iterated fractional factorial designs [22] are considered.

More specifically, we focus on numerical comparisons of two black box methods called response surface methodology (RSM) and simultaneous perturbation stochastic

approximation (SPSA), respectively. RSM originated in Box and Wilson [23]. Myers and Montgomery [18], which is a classic textbook on RSM, gives the following general description, for a minimization problem. In the first stage of RSM, an experimental design is used to locally fit a first-order polynomial to the observed values of the random response; this fit uses linear regression. Then, a steepest descent search direction is estimated from the fitted first-order polynomial, and a number of steps are taken along this direction - until no additional decrease in objective is evident. This procedure is repeated until a first-order polynomial becomes an inadequate model, which is indicated when the gradient is not significantly different than a zero vector. In the second stage of RSM, a second-order polynomial is fitted locally, and this polynomial is minimized. Furthermore, canonical and ridge analyses are performed to determine the nature of the fitted objective function (i.e., convex, concave, or indefinite) and the nature of the estimated optimum (i.e., single optimum or multiple optima).

Some disciplines interpret RSM in a completely different way: RSM becomes a one shot approach that fits a single response surface, which is either a second-order polynomial or a kriging model, to the random response over the whole experimental area. Next, this nonlinear fitted model is optimized using a global optimization procedure; see Sacks et al. [24], Jones, Schonlau, and Welch [25], and Simpson et al. [26]. In this thesis, we do not consider the one-shot RSM, but we concentrate on the sequential version as recently described in Myers and Montgomery [18].

Stochastic approximation (SA) is an iterative technique that can be used to optimize both real systems and computer simulation of real systems. This technique was introduced in the 1950s by Robbins and Monroe [27] and Kiefer and Wolfowitz [28]. Later, a new SA algorithm, namely simultaneous perturbation SA (SPSA), was introduced and developed by Spall [29, 30, 31]. A recent, complete reference on SPSA is the monograph Spall [32].

Both RSM and SPSA are broadly applicable in the sense that they can be easily integrated into both stochastic and deterministic simulations, since RSM and SPSA do not necessarily exploit the stochastic structure of the simulated system. Not exploiting

the stochastic structure has the advantage of being very flexible. However, a disadvantage is that both RSM and SPSA may be computationally more expensive than the other methods that do take the stochastic structure into account.

RSM and SPSA have the two important features of black box approaches, namely generality and simplicity. Furthermore, unlike genetic algorithms that have a family of solution points at each iteration, both RSM and SPSA have a single solution point at each iteration along the search path. RSM has the following disadvantages compared to metaheuristics and SPSA (i.e., genetic algorithms, tabu search, and simulated annealing): *(i)* RSM is assumed to be a continuous optimization method, since RSM is similar to gradient-based approaches. Hence, unlike metaheuristics and SPSA, RSM is not suitable for discrete optimization; *(ii)* RSM may find a local optimum, as opposed to metaheuristics and SPSA that search for a global one.

The remainder of this chapter is organized as follows. Section 1.1 distinguishes between black box and white box approaches to gradient estimation through simulation, with the emphasis on black box approaches. Section 1.2 gives an overview of black box simulation optimization methods.

1.1 Black Box Approaches to Gradient Estimation

In this section, we differentiate between black box and white box approaches to gradient estimation through simulation. We emphasize black box approaches, since RSM and SPSA - the focus of this thesis - are black box methods. By definition, simulation is treated as a black box if the gradient estimates (and possibly higher order derivative estimates) through simulation are not available using either perturbation analysis (PA) - see Ho and Cao [33], Fu and Hu [34] - or likelihood ratio score function (LR/SF) - see Rubinstein and Shapiro [35].

Before presenting the common approaches to obtain black box gradient estimates through simulation, we introduce a general problem formulation for simulation optimization, as follows:

$$\min_{d \in \Theta} imize\, E_{\omega}\left[F(d,\omega)\right] \tag{1.1}$$

where E_{ω} is the expectation operator with respect to the simulation's seed vector ω, $F(d,\omega)$ is a random response estimated through simulation, d is $k \times 1$ vector of input variables, and Θ is the (explicitly or implicitly defined) feasible search space.

The most straightforward way for obtaining gradient estimates uses finite differences. Forward finite differencing (FFD) needs $k + 1$ simulation runs to obtain a single gradient estimate, i.e.; the i^{th} ($i = 1,\ldots, k$) component of the gradient estimate, say $\hat{g}_i$ at d is

$$\hat{g}_i(d) = \frac{\hat{F}(d + c_i e_i) - \hat{F}(d)}{c_i} \tag{1.2}$$

where $\hat{F}(d + c_i e_i)$ and $\hat{F}(d)$ are estimates of F in (1.1) at the two input vectors $d + c_i e_i$ and d, e_i is the unit vector in the i^{th} direction and c_i is a scalar.

Central finite differencing (CFD) conducts $2k$ simulation runs to obtain a single gradient estimate:

$$\hat{g}_i(d) = \frac{\hat{F}(d + c_i e_i) - \hat{F}(d - c_i e_i)}{2c_i} \tag{1.3}$$

where $\hat{F}(d - c_i e_i)$ is the estimate of F in (1.1) at the input vector $d - c_i e_i$. Obviously, CFD is computationally more expensive than FFD. On the other hand, the CFD estimators are less biased than the FFD estimators. In either case, however, a single gradient estimate requires $O(k)$ simulation replicates.

In RSM, we will use resolution-III designs to obtain gradient estimates. By definition, this design type gives unbiased estimators of the gradients of the random responses, provided that first-order polynomial approximations are adequate for these responses; see Kleijnen [36]. To obtain a single gradient estimate, resolution-III designs need only $k + 1$ simulation runs, rounded upwards to a multiple of four. Therefore, resolution-III designs are computationally more efficient than CFD; they require the same number of runs as FFD, but give smaller variances (standard errors) than FFD. For further comparisons of design of experiments schemes – including resolution-III designs – with FFD and CFD, we refer to Brekelmans et al. [37].

The method of simultaneous perturbation (SP) of Spall [38] avoids these $O(k)$ simulation replicates, and estimates a single gradient by perturbing in all directions simultaneously. To obtain a single gradient estimate, SP needs only two simulation runs, independent of k, as follows:

$$\hat{g}_i(d) = \frac{\hat{F}\left(d + c_i\hat{\Delta}\right) - \hat{F}\left(d - c_i\hat{\Delta}\right)}{2\hat{\Delta}_i} \tag{1.4}$$

where $\hat{F}(d + c_i\hat{\Delta})$ and $\hat{F}(d - c_i\hat{\Delta})$ are estimates of F in (1.1) at the two input vectors $d + c_i\hat{\Delta}$ and $d - c_i\hat{\Delta}$ and $\hat{\Delta} = \left(\hat{\Delta}_1, \ldots, \hat{\Delta}_i, \ldots, \hat{\Delta}_k\right)^T$ represents a realization of a vector, say Δ, of independent, identically distributed random perturbations satisfying certain conditions given in Spall [31]. In practice, the simplest and most popular distribution for Δ is the symmetric (scaled) Bernoulli distribution. The difference between the FFD/CFD estimators and the SP estimators is that the numerator, which involves the expensive simulation replicates, varies in the FFD/CFD estimates (see (1.2) and (1.3)), whereas the numerator is constant in the SP estimates, and it is the denominator involving the random perturbations that varies (see (1.4)). A difficulty encountered in implementing FFD, CFD, and SP is that the choice of the scalar c_i must balance between too much noise and too much bias; that is, in order for the bias to be small, it is necessary to let the scalar c_i be small. However, when c_i is small, the

FFD, CFD, and SP estimators usually have large variances; see Spall [39] for practical guidelines for choosing c_i in SP.

In Table 1.2 - again taken from Fu [3] - we compare white box approaches to gradient estimation such as infinitesimal perturbation analysis (IPA), which is the simplest form of PA, and LR/SF, as well as FFD, CFD, and SP. IPA and LR/SF require more knowledge about the simulated system than FFD, CFD, and SP. For example, LR/SF assumes the knowledge of a distribution that dominates the input distribution; see L'Ecuyer [40]. Under certain conditions, IPA and LR/SF provide gradient (and possibly higher order derivatives) estimators with desirable statistical properties such as unbiasedness and strong consistency through a single simulation run; for these conditions, see, for example, Glasserman [41] for PA, and Rubinstein and Shapiro [35] for LR/SF. LR/SF applies more generally than IPA; see L'Ecuyer [40]. However, Glynn proves that when both IPA and LR/SF yield unbiased and strongly consistent estimators, LR/SF's estimators have larger variances [42]. For further comparison of finite differences, IPA, and LR/SF, we refer to L'Ecuyer [40].

Table 1.2: Gradient estimation through simulation

Approach	*Number of simulations*	*Key feature(s)*	*Disadvantage*
IPA	1	highly efficient, easy to implement	limited applicability
LR/SF	1	assumes the knowledge of a distribution that dominates the input distribution	possibly high variance
SP	2	widely applicable	generally noisy
FFD	k+1	widely applicable	generally noisy
CFD	2k	widely applicable	generally noisy

1.2 Black Box Simulation Optimization Methods

In this section, we present several black box simulation optimization methods that we consider as alternatives to RSM and SPSA. We do not claim to give an exhaustive

literature survey on black box simulation optimization. We focus on methods that we think to be most widely used; also see Andradóttir et al. [4] and Boesel et al. [43], which are the panel discussions at the Winter Simulation Conferences in 2000 and 2001, respectively, and also Table 1.1, which shows the most popular optimization approaches among practitioners. See also Figure 1.1 - taken from Tekin and Sabuncuoglu - which classifies various simulation optimization techniques [2].

In the following, we do not consider statistical ranking, selection, and multiple comparison methods; see, for example, Goldsman et al. [44], or a more recent reference Boesel, Nelson, and Kim [45]. The primary difference between statistical ranking, selection, and multiple comparison methods and the methods described below is that the former methods evaluate exhaustively all members of a fixed and finite set of alternatives. However, the latter methods attempt to search efficiently through the feasible search space to find better solutions, because exhaustive search is impractical or impossible (i.e., the feasible search space can be unbounded or uncountable). Furthermore, we do not consider sample path optimization of Gürkan, Özge, and Robinson [46, 47], and random search methods (see, for example, Andradóttir [6]. The sample path methods use IPA to estimate gradients through simulation (hence, it is a white box approach). To the best of our knowledge, random search methods have been applied solely to discrete optimization problems.

In the following subsections, we shall summarize black box methods, namely the stochastic approximation method, genetic algorithms, tabu search, simulated annealing, and ordinal optimization.

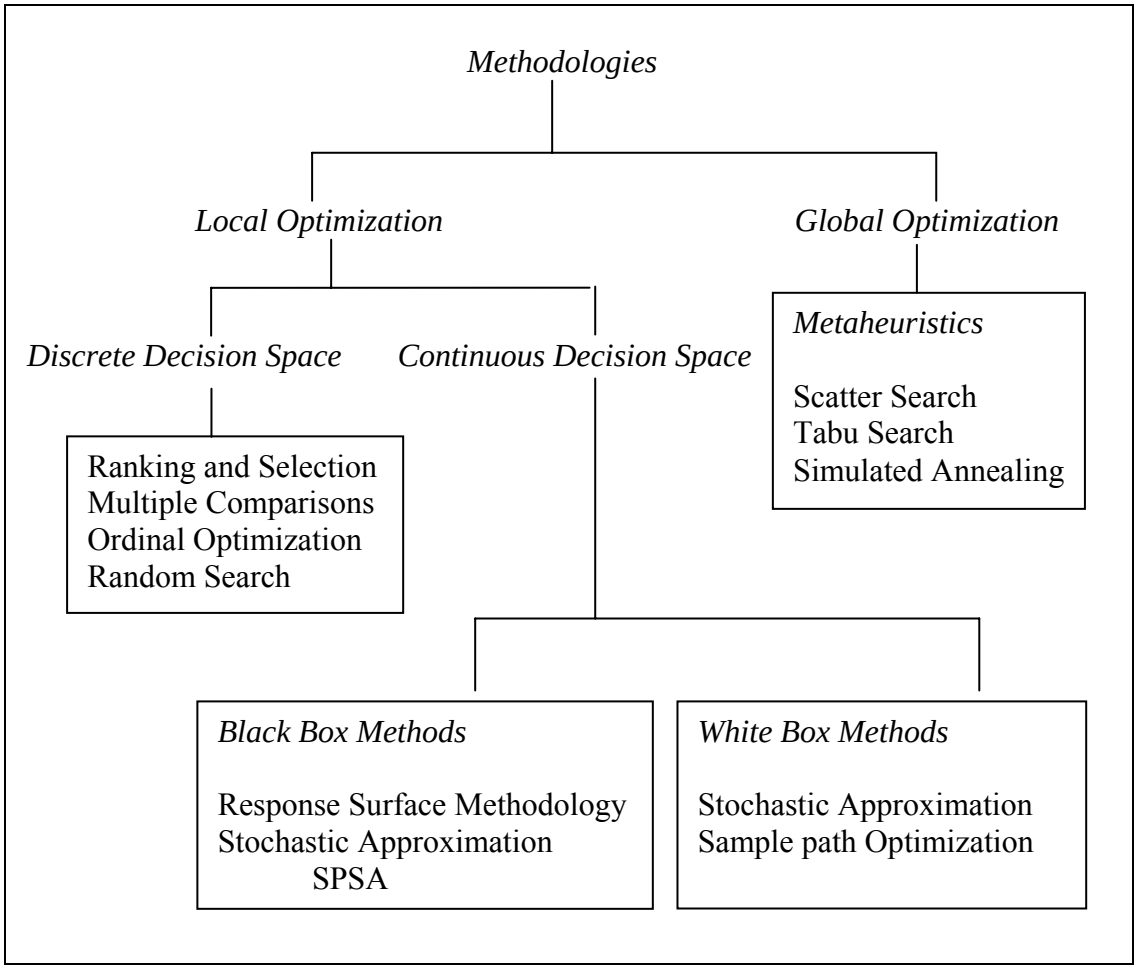

Figure 1.1: Simulation optimization methodologies

1.2.1 The Stochastic Approximation Method

The stochastic approximation method attempts to mimic the gradient search method in deterministic optimization, while taking into account the stochastic setting. Given the problem in (1.1), the general form of stochastic approximation is:

$$d_{n+1} = \Pi_{\Theta}\left(d_n - a_n \hat{g}\left(d_n\right)\right) \tag{1.5}$$

where Θ is closed and convex, Π_{Θ} denotes some projection back into Θ when the iteration leads to a point outside Θ (e.g., the simplest projection would be to return to the previous point), n denotes the iteration number, and $\{a_n\}$ is a sequence of step size

multipliers such that $\sum_{n=1}^{\infty} a_n = \infty$ and $\sum_{n=1}^{\infty} a^2{}_n = \infty$ [3]. If the finite differences (1.2) or (1.3) are used to obtain $\hat{g}$ in (1.5), then the procedure in (1.5) is called Kiefer-Wolfowitz's algorithm; see Kiefer and Wolfowitz [28]. If simultaneous perturbation in (1.4) is used to obtain $\hat{g}$ in (1.5), then it is called simultaneous perturbation stochastic approximation; see Spall [32, 38, 48], and Kleinman, Spall, and Naiman [49].

Under appropriate conditions, one can prove the convergence of the sequence $\{d_n\}$ to the local minimum with probability one, as *n* goes to infinity; for these conditions in case of SPSA, see Spall [31]. In practice, however, the performance of stochastic approximation is very much sensitive to the sequence $\{a_n\}$; see, for example, Fu [3]. Theoretically, a constant step size results in weak convergence - that is, convergence in distribution - which means that the iterates may oscillate around the local minimum. Yet, in practice, a constant step size results in much quicker convergence in the early stages of the method - unlike a step size decreasing at each step.

Some of the (recent) advances on stochastic approximation are as follows. Since stochastic approximation is a gradient search method, it generally finds a local optimum. Therefore, Maryak and Chin enhance SPSA to find the global optimum [50]. Furthermore, Gerencsér, Hill and Vagó [51] and Whitney, Solomon, and Hill [52] apply stochastic approximation to discrete optimization, although it has been usually used for continuous optimization.

1.2.2 Genetic Algorithms

Unlike stochastic approximation and RSM, evolutionary search strategies such as genetic algorithms work with a family of solution points - namely the population – rather than a single solution point. More importantly, the solution points in the current population interact to form the next population - also called the next generation. Important factors that affect the success of genetic algorithms are the selection procedure and the types of genetic operators. Selection can be done either deterministically or probabilistically - and is based on the fitness of a solution point,

where the fitness of a solution point corresponds to its objective function value. Two of the simplest (deterministic) selection procedures include keeping each generation at a constant number of the fittest solution points (survival of the fittest), or keeping only the offspring from reproduction (complete generational turnover). Genetic operators operate on a genetic representation (code) of a solution point, and are generally classified as crossover operators and mutation operators. The crossover operators take two solution points from the population that have relatively good fitnesses, and combine them to make two new solution points. The mutation operators take a single, well-performing solution point, and alter it slightly. Notice that the crossover operators distinguish genetic algorithms from other metaheuristics, such as simulated annealing and tabu search. For more details and references, see Michalewicz and Schoenauer, and Fouskakis and Draper [53, 54].

1.2.3 Tabu Search

Tabu search can be thought of as a variation on local search that incorporates two main strategies, namely adaptive memory and responsive exploration [55]. The features of these strategies modify the neighborhood of a solution point as the search progresses, and thus determine the effectiveness of the algorithm. In particular, the modification from which the method derives its name may forbid certain points (classifying them tabu); i.e., these tabu points do not belong to the current neighborhood of a solution point. Thus, for example, short-term memory can prevent the search from revisiting recently visited points, whereas long-term memory can encourage moves that have historically led to improvements (intensification) and moves into previously unexplored regions of the search space (diversification). For details and more references, see Glover, and Fouskakis and Draper [56, 54].

1.2.4 Simulated Annealing

Simulated annealing may be thought of as a variation on local search, in which the main idea for a minimization problem is to accept all downhill improving moves, but sometimes accept also uphill moves, where the acceptance probability of uphill moves

decreases to zero at an appropriate rate (this is the cooling schedule from which the method derives its name, in analogy with the physical annealing process, where the system seeks the lowest energy state) [57]. By accepting uphill moves, simulated annealing tries to avoid local minima. An attractive property of this algorithm is that - unlike genetic algorithms and tabu search - convergence can be rigorously proved in many settings; see, for example, Gutjahr and Pflug [58], and Alrefaei and Andradóttir [59]. On the other hand, the procedure has been found to converge relatively slow to a good solution point, compared to genetic algorithms and tabu search. For more details and references, we refer to Anandalingam [60], and Fouskakis and Draper [54].

1.2.5 Ordinal Optimization

In this section, we present the ordinal approach to simulation optimization|as opposed to the cardinal approach; see Ho, Sreenivas, and Vakili [61], and Ho et al. [62]. Note that both RSM and SPSA are cardinal approaches. Furthermore, it will become clear in this section that ordinal optimization is not meant to replace traditional cardinal optimization, but supplements it.

Ordinal optimization differs from cardinal optimization in two important ways: *(i)* the aim of ordinal optimization is not to look for the best but to find a solution that is good enough (goal softening), which has to be statistically defined; *(ii)* ordinal optimization focuses on approximating the order among the outputs of the given input vectors rather than accurately estimating the values of these outputs (and possibly their gradients and higher-order derivatives) at these input vectors.

Before detailing these two properties, we explain a general weakness related to simulation optimization problems, following Ho et al. [61]. Suppose that in (1.1), Θ is a huge, arbitrary, but finite search space. The standard approach to estimate $E_{\underline{\omega}}[F(d,\omega)]$ is

$$\overline{\widehat{F}}_N(d) = \frac{1}{N}\sum_{j=1}^{N} \widehat{F}(d,\widehat{\omega}_j) \tag{1.6}$$

where $\hat{\omega}_j$ is the j^{th} sampled value of the random vector ω and N is the number of simulation replicates (or the length of the simulation run). Now, the problem is that the confidence interval of (1.6) cannot improve faster than $1/\sqrt{N}$.

Ordinal optimization is based on two tenets:

i) Order converges exponentially fast, whereas (1.6) converges at the rate $1/\sqrt{N}$; see Dai, and Dai and Chen [63, 64]. This is intuitively reasonable: think of the problem of holding identically looking packages in each hand and trying to determine which package weighs more versus estimating the difference in weight between the two packages.

ii) Goal softening eases the computational burden of finding the optimum. In ordinal optimization, one settles for the set of good enough input vectors with high probability (e.g., any of the top *r* of the input vectors, 95% of the time).

Ordinal optimization has some common features with statistical ranking, selection, and multiple comparison methods: for example, relative ordering in ordinal optimization is in the same spirit as multiple comparisons; goal softening is in the same spirit as statistical ranking and selection. The primary difference is the scale; the former method deals with a very large search space, whereas Goldsman and Nelson suggests two to twenty input vectors for the latter methods [65]. Furthermore, ordinal optimization does not address questions such as the distance between the best and the rest, which is a cardinal notion in multiple comparison methods, or whether or not the observed order is a maximum likelihood estimate of the actual order.

Some advantages of ordinal optimization are simplicity (the procedure is easy to state and implement), generality (applicable to a very large class of problems), and efficiency (it is possible to select good input vectors with high probability in the presence of significant estimation errors). In some cases, however, obtaining a good enough subset of input vectors may not be very satisfactory. Then, ordinal optimization can be considered as an approach complementary to cardinal optimization (for example, as a

pre-processing step that narrows the search space). Moreover, once a statistically good solution is obtained through the ordinal approach, the final fine tuning to reach the true optimum can be accomplished via cardinal optimization.

Finally, there have been many applications of ordinal optimization to different problems such as communication networks [66], rare event simulation [67], resource allocation problems [62], and robot motion planning problem [68]. Additional applications and an interactive online demo can be found at the website: hrl.harvard.edu/~ho/DEDS [69].

2. RESPONSE SURFACE METHODOLOGY AND SIMULTANEOUS PERTURBATION STOCHASTIC APPROXIMATION

2.1 Introduction

With the increasing computational power of computers, simulation optimization has become a very active research area for the last two decades; see, for example, Jacobson and Schruben [70], Safizadeh [71], Azadivar [5], Andradóttir et al. [4], Fu [3], Tekin and Sabuncuoglu [2], and Fu, Glover and April [72]. Recent discussions of simulation optimization are Henderson and Nelson, and Kleijnen [73, 74].

There are a number of simulation optimization methods that can be classified as either white box or black box methods. White box methods such as sample path optimization assume that unbiased estimates of the gradients of the random responses are obtainable through a single run of simulation [75]. This gradient estimation is achieved through a form of perturbation analyses [41, 33, 76] or the likelihood ratio/score function method [35]. On the other hand, black box methods such as meta-heuristics, simultaneous perturbation stochastic approximation [32], and response surface methodology [77, 18] need only the simulation outputs of the random responses. When the unbiasedness assumption is satisfied, white box methods outperform black box ones; see, for example, L'Ecuyer, Giroux, and Glynn [78]. However, in many practical problems, this assumption is not satisfied; hence, black box methods are applicable.

This study aims at comparing SPSA and RSM numerically for simulation optimization problems when each simulation run is very time-consuming and the computational budget allows only a small number of runs. The ultimate purpose, however, is not to make definitive conclusions as to superiority, but simply to illustrate reasonably comparable performances on different types of simulation optimization problems. As

an example of time-consuming simulation, 36 to 160 hours of computer time are needed to simulate a crash model at Ford Motor Company; see the panel discussion Simpson et al. [79]. In such a case, it is important that the search reaches a vicinity of the true optima with the most improvement in the goal function using only a small number of runs. From that vicinity, a large number of runs can still be performed to estimate a local (or preferably global) optimal solution, but this is not within the scope of this study.

There are many articles in the literature related to this work. For example, Spall [31] and Chin [80] compare SPSA with the standard finite-difference SA (FDSA) and the random-directions SA, and Kleinman et al. compare SPSA with FDSA using bootstrapping examples [49]. Furthermore, Fu and Hill use discrete event dynamic systems to compare SPSA with FDSA [81]. Recently, Kleijnen and Wan [82] compare RSM with OptQuest (an add-on to simulation software such as Arena, ProModel, and Simul8) and Bashyam and Fu's perturbation analysis combined with the feasible directions method using an (*s*, *S*) inventory management problem [83]. However, all these authors assume that the computational budget enables them to compare the methods asymptotically, which is in contrast with the main assumption of this study (i.e., small number of runs). To the best of our knowledge, there is no work in the simulation optimization literature that compares RSM with SPSA using a small or large number of runs.

In this study, we consider examples of a classic simulation optimization problem; that is, there is a single, random response to be minimized. This problem can be formulated as

$$\underset{d \in R^k}{\text{minimize}}\, E_\omega \left[F_0 \left(d, \omega \right) \right] \tag{2.1}$$

where d is the vector of k input variables, ω is the simulation seed vector, F_0 is the random simulation response, and E_ω is the expectation operator with respect to ω. In (2.1), the mathematical form of F_0 is not known explicitly; only its simulation outputs

are available. However, it is assumed that F_0 is continuous and continuously differentiable on R^k.

Further, we consider examples of simulation optimization problem with stochastic constraints and box constraints, so (2.1) becomes:

$$\begin{aligned} &\min_{d \in R^k} imize\, E_\omega \left[F_0(d, \omega) \right] \\ &E_\omega \left[F_j(d, \omega) \right] \geq a_j \\ &l \leq d \leq u \end{aligned} \tag{2.2}$$

where l is the lower bound on d, u is the upper bound on d, a_j is the j^{th} component of the deterministic right-hand-side vector, and the $F_j(j = 0,...,r-1)$ are r random simulation responses (outputs). Again, we do not know these F_j explicitly, so we estimate their means through simulation. However, we again assume that these F_j are continuous and continuously differentiable on the feasible set defined by the inequalities in (2.2).

The remainder of this chapter is organized as follows. Sections 2.2 and 2.3 present literature surveys on SPSA and RSM, respectively. Sections 2.4 and 2.5 introduce the step-by-step implementation guidelines of SPSA and RSM for unconstrained problem type in (2.1) and constrained problem type in (2.2), respectively. The performances of these two methods will be compared on various example problems in the following chapter.

2.2 Literature Survey on SPSA

Stochastic approximation (SA) is an iterative technique that can be used to optimize both real systems or computer simulation of real systems. This technique is introduced in the 1950s by Robbins and Monroe [27], Kiefer and Wolfowitz [28], and Blum [84]. Later, a new SA algorithm, namely simultaneous perturbation SA (SPSA), is introduced

and developed by Spall [31]. A recent, complete reference on SPSA is the monograph Spall [32].

From the practitioners' point of view, SPSA has been very useful since it has been applied to diverse areas such as vessel traffic management [85], air traffic management [13], simulation optimization of complicated discrete-event dynamic systems [81, 86] image processing [87], optimization of fresh-food supply chains in uncertain environments [88], simulation optimization for revenue management of airlines with cancellations and overbookings [89], and industrial quality improvement [90].

There are a number of techniques for enhancing the performance of the basic SPSA algorithm. For example, Spall uses the average of several SP gradient estimations to reduce the noise effects [31]. Furthermore, Spall derives an adapted SPSA that emulates for stochastic problems the fast Newton-Raphson algorithm of deterministic optimization [38]. These two improvements can be applied at the expense of extra, timeconsuming simulation runs.

Sadegh and Spall consider the problem of choosing the best distribution for the perturbation vector [91]. On the basis of asymptotic distribution results, it is shown that the optimal distribution for the components of the perturbation vector is symmetric Bernoulli. This simple distribution has also proven effective in many finite-sample practical and simulation examples.

There are only a few articles which cope with problems with general inequality constraints such as Rezayat, Wang and Spall, and Sadegh [92, 93, 94]. The first two articles deal with the constraints using several types of penalty functions, and the last article uses a projection operator. Furthermore, Fu and Hill consider simple box constraints and implement the following projection algorithm: project x% ($0 < x < 100$) of the way - as measured in the parameter direction most violated – to the boundary [81].

Since SPSA uses only the local gradient information, the classic SPSA converges to a local optimal solution. Recently, Maryak and Chin generalizes SPSA to a globally convergent algorithm [95]. In the numerical studies, they find significantly beter performance of SPSA as a global optimizer than for the popular simulated annealing and genetic algorithm methods, which are often recommended for global optimization. Finally, SPSA is originally meant for continuous optimization problems. Hill changes this original algorithm so that it can also be applied to discrete optimization problems [96].

2.3 Literature Survey on RSM

Response surface methodology has originated from the work of Box and Wilson who apply RSM to find optimal operating conditions for chemical processes [23]. One of the earliest application of RSM to simulated systems is by Biles [97]. Recent Works on RSM are by Barton and Meckesheimer, and Kleijnen (2008) [98, 74].

Like SPSA, RSM is important for practitioners since it can be applied to data obtained from both real-life (non-simulated) systems and simulated systems [23]. The latter systems can be deterministic or stochastic; see, for example, Ben-Gal and Bukchin [12] for a case study of RSM optimization of a deterministic simulated system, and Irizarry, Wilson, and Trevino [99], and Yang and Tseng [100] for case studies of RSM optimization of stochastic simulated systems.

2.4 Guidelines for Unconstrained Problems

2.4.1 SPSA

We consider the basic version of SPSA for which Spall [32] provides the following step-by-step implementation guidelines and the MATLAB code available at <www.jhuapl.edu/SPSA/Pages/MATLAB.htm> [69].

Step 0: Initialize and select coefficients:

Set counter index $p = 0$. Pick an initial input vector $\hat{d}_0$. Pick nonnegative values for the coefficients a, c, A, α and γ to calculate the following gain sequences $\{a_p\}$ and $\{c_p\}$ as follows:

$$a_p = \frac{a}{(p+1+A)^{\alpha}} \text{ and } c_p = \frac{c}{(p+1)^{\gamma}} \tag{2.3}$$

This choice of $\{a_p\}$ and $\{c_p\}$ satisfies classic regularity conditions of the SA algorithm; i.e., $a_p \to 0$ and $c_p \to 0$ as $p \to \infty$, $\sum_{p=0}^{\infty} a_p = \infty$, $\sum_{p=0}^{\infty} (a_p / c_p)^2 < \infty$ [32].

Chin derives the asymptotically optimal values $\alpha = 1$ and $\gamma = 1/6$, but these values are not generally useful in practice since they require information on the random response F_0 and its gradients [101]. In practical applications, Spall recommends to set $\alpha = 0.602$ and $\gamma = 0.101$ [32]. Spall also suggests to set A at 10% or less of the maximum number of iterations, and to set c at a level approximately equal to the estimated standard deviation of F_0 [32].

Step 1: Generate the SP vector

Using parametric bootstrapping, generate a k-dimensional random perturbation vector Δ_p. The components of this Δ_p have to satisfy some statistical properties:

i) All k components have to be independent, identically and symmetrically distributed about zero with a finite variance,
ii) all k components have to have finite inverse moments [32].

The second property implies that the normal and uniform distributions can not be used to generate Δ_p, since these distributions do not have finite inverse moments.

Spall recommends to use a Bernoulli ± 1 distribution with probability of 1/2 for each ± 1 outcome to generate each component of Δ_p since this distribution is both theoretically valid and practically effective [32]. Hence, in the numerical examples, each component of Δ_p is generated through a symmetric Bernoulli ± 1 distribution.

Step 2: Perform simulation runs

Perform four runs to obtain one simulation output of the random response, denoted by $\hat{F}(.)$, at each of $\hat{d}_p + c_p\Delta_p$ and $\hat{d}_p - c_p\Delta_p$, and two simulation outputs of the random response at $\hat{d}_p$, using independent pseudorandom numbers. At this step, Spall needs only two runs, one at each of $\hat{d}_p + c_p\Delta_p$ and $\hat{d}_p - c_p\Delta_p$, to estimate an SP gradient at the next step [32]. The two extra runs at $\hat{d}_p$ are required to obtain point estimates of the sample mean and the sample variance of the expected objective function, denoted by $\bar{\hat{F}}_0\left(\hat{d}_p\right)$ and $\hat{S}_0^2\left(\hat{d}_p\right)$, where

$$\bar{\hat{F}}_0\left(\hat{d}_p\right) = \frac{\hat{F}_{0,1}\left(\hat{d}_p\right) + \hat{F}_{0,2}\left(\hat{d}_p\right)}{2} \tag{2.4}$$

$$\hat{S}_0^2\left(\hat{d}_p\right) = \left(\hat{F}_{0,1}\left(\hat{d}_p\right) - \bar{\hat{F}}_0\left(\hat{d}_p\right)\right)^2 + \left(\hat{F}_{0,2}\left(\hat{d}_p\right) - \bar{\hat{F}}_0\left(\hat{d}_p\right)\right)^2$$

The estimates are used in Step 4 to determine the next estimated iterate.

Step 3: Estimate gradient through SP

Estimate the gradient $g_p\left(\hat{d}_p\right)$ of the expected objective function with respect to *d* by the equation:

$$\hat{g}_p\left(\hat{d}_p\right)=\begin{bmatrix} \dfrac{\hat{F}\left(\hat{d}_p+c_p\Delta_p\right)-\hat{F}\left(\hat{d}_p-c_p\Delta_p\right)}{2c_p\Delta_{p,1}} \\ . \\ . \\ \dfrac{\hat{F}\left(\hat{d}_p+c_p\Delta_p\right)-\hat{F}\left(\hat{d}_p-c_p\Delta_p\right)}{2c_p\Delta_{p,k}} \end{bmatrix} \tag{2.5}$$

where $\Delta_{p,i}\left(i=1,...,k\right)$ is the i^{th} component of Δ_p. Observe that in (2.5), the numerator is the same for each component *i* so that only two simulation runs are necessary to obtain one SP gradient estimate, independent of the dimension *k* of the problem.

Step 4: Update the current estimated iterate

Estimate a candidate for the new current iterate by the following standard SA recursion:

$$\hat{d}_{p+1}=\hat{d}_p-a_p\hat{g}_p\left(\hat{d}_p\right) \tag{2.6}$$

At this step, Spall considers $\hat{d}_{p+1}$ in (2.6) as the new current estimated iterate [32]. However, due to the stochastic nature of the problem in (2.1), in this study $\hat{d}_{p+1}$ is considered as the new current estimated iterate if there is a statistically significant improvement in the estimated objective function with respect to $\hat{d}_p$. Following the approach in Kao, Chen and Tseng [102], the following hypothesis is tested via the two-sample Student t-test for independent samples:

$$\begin{aligned} H_0 &: E_\omega\left[F\left(\hat{d}_{p+1},\omega\right)\right]-E_\omega\left[F\left(\hat{d}_p,\omega\right)\right]\le 0 \\ H_1 &: E_\omega\left[F\left(\hat{d}_{p+1},\omega\right)\right]-E_\omega\left[F\left(\hat{d}_p,\omega\right)\right]>0 \end{aligned} \tag{2.7}$$

In order to test (2.7), perform two runs at $\widehat{d}_{p+1}$ to obtain point estimates $\overline{\widehat{F}}_0\left(\widehat{d}_{p+1}\right)$ and $\widehat{S}_0^2\left(\widehat{d}_{p+1}\right)$ as in (2.4). Then, an appropriate test statistic to use is the following standard Student t-statistic [103]:

$$t = \frac{\overline{\widehat{F}}_0\left(\widehat{d}_{p+1}\right) - \overline{\widehat{F}}_0\left(\widehat{d}_p\right)}{\sqrt{\frac{\widehat{S}_0^2\left(\widehat{d}_{p+1}\right)}{2} + \frac{\widehat{S}_0^2\left(\widehat{d}_p\right)}{2}}} \tag{2.8}$$

If the random simulation responses are assumed to be normally distributed, under H_0 this test statistic has a t-distribution with v degrees of freedom, where v is given by [103]:

$$v = \frac{\left(\frac{\widehat{S}_0^2\left(\widehat{d}_{p+1}\right)}{2} + \frac{\widehat{S}_0^2\left(\widehat{d}_p\right)}{2}\right)^2}{\left(\frac{\widehat{S}_0^2\left(\widehat{d}_{p+1}\right)}{2}\right)^2 + \left(\frac{\widehat{S}_0^2\left(\widehat{d}_p\right)}{2}\right)^2} \tag{2.9}$$

If the resulting p-value of the test does not exceed its pre-specified significance level, then H_0 in (2.7) is rejected, and $\widehat{d}_{p+1}$ in (2.6) is set as the new current estimated iterate. Otherwise, the old estimated iterate $\widehat{d}_p$ is not changed.

The test statistic in (2.8) requires some comments. Uunlike Kao et al. [102] who use a pooled variance estimate in (2.8), we do not assume common (homoscedastic) variances of the simulation outputs. For a discussion on how realistic the common variance assumption is, see Kleijnen [74]. Furthermore, we do not assume the normality of the simulation outputs. Nevertheless, the Student t-statistic in (2.8) is quite insensitive to nonnormality; see Kleijnen and many references therein [19].

Step 5: Iterate or terminate

If the stopping criterion is satisfied, then stop the SPSA procedure. Otherwise, set p+1 to p, and go to Step 1.

In the numerical examples, the stopping criterion is satisfied when the maximum number of runs is reached or there is no statistically significant improvement in the objective, whichever happens first.

2.4.2 RSM

Based on Kleijnen, Den Hertog, and Angün [104] and Angün et al. [105], this section provides the steps for the practical implementation of RSM as follows:

Step 0: Initialize

Input a maximum number of iterations. Pick an initial input vector $\widehat{d}_0$. Input a (fixed) user-specified experimental area, where the response F_0 is locally approximated by a first-order regression metamodel. Design local metamodel fitting experiment. Conduct simulation experiments at the input vectors (design points).

In the numerical examples, the maximum number of iterations is also used as the stopping criterion of RSM. Furthermore, the half-size of the experimental area at the p^{th} iteration is given by c_p in (2.3). Thus, considering $\widehat{d}_0$ as the center of the initial design, the high and low levels of the i^{th} input variable are given by, say, $\widehat{d}_{0,i} + c_0$ and $\widehat{d}_{0,i} - c_0$ respectively, where $\widehat{d}_{0,i}$ is the ith component of $\widehat{d}_0$. A resolution-III design is selected, since such a design gives unbiased estimators of the coeffcients of the first order regression metamodel with a small number of simulation runs, provided that the first-order metamodel is an adequate approximation; see Kleijnen [106].

Step 1: Fit a first-order regression metamodel

Approximate F in (2.1) by a first-order regression metamodel, where the regression coefficients are estimated through ordinary least squares:

$$\hat{\beta} = \left(X^T X \right)^{-1} X^T \hat{F}_0 \tag{2.10}$$

In (2.10), X is the design matrix augmented by a column of ones and $\hat{F}_0$ denotes the simulation outputs at the design points. Furthermore, estimate the variance of $\hat{F}$ through mean squared residual (MSR):

$$\hat{\sigma}^2 = \frac{\left(\hat{F}_0 - X\hat{\beta} \right)^T \left(\hat{F}_0 - X\hat{\beta} \right)}{N - (k+1)} \tag{2.11}$$

where N is the total number of local runs and $k + 1$ is the number of regression coeffi cients to be estimated.

Step 2: Estimate a search direction

Estimate the following adapted steepest descent direction, which is first derived in Kleijnen et al. [104]:

$$\hat{g} = -C^{-1}\hat{\beta}_{-0} \tag{2.12}$$

where $\hat{\beta}_{-0}$ equals $\hat{\beta}$ excluding the intercept, and C is the matrix obtained by deleting the first row and the first column of $(X^T X)^{-1}$. The search direction in (2.12) is shown to perform better than the ordinary estimated steepest descent search direction in Kleijnen et al. and Angün et al. [104, 105].

Step 3: Update the current estimated iterate

Similar to (2.6), the next estimated iterates are given by:

$$\hat{d}_{p+1} = \hat{d}_p - a_p \hat{g} \tag{2.13}$$

where $\hat{g}$ is now given by (2.12). The step size at the p^{th} iteration is given by a_p in (2.3).

Step 4: Select a resolution-III design and simulate at the design points

Consider $\hat{d}_p$ as the center of the current design, and c_p as the half-size of the experimental area. Conduct a total of *N* simulation runs at the design points.

Step 5: Iterate or terminate

If the maximum number of iterations is reached, stop the RSM procedure. Otherwise, set $p + 1$ to p, and go to Step 1.

2.5 Guidelines for Constrained Problems

Based on Angün et al. that describes the generalized version of RSM, which can handle stochastic constraints as well as deterministic box constraints, this section provides the the steps for the practical implementation of the SPSA and RSM [105]. The problem of handling constraints has not been considered often in the SPSA literature; see also Section 2.2. Therefore, we will use the same approach for SPSA as the approach in the generalized RSM with some modifications.

Before presenting the details of our heuristic, we briefly explain the general procedure for RSM. First, the *r* expected responses in (2.2) are locally approximated by *r* first-order polynomials. Next, these polynomials are used to estimate the search direction and a maximum step size. In this estimated search direction, a line search using two statistical tests is performed to determine an iterate better than the current iterate. At the end of this line search, the heuristic may fail to find a better iterate than the current one. In that case, the heuristic does not leave the current iterate. New first-order polynomials are fitted in a new local experimental area or in the old one - the latter happens when the line search fails to find a better iterate. Again a new search direction and a maximum step size are estimated, and the heuristic continues - as described above - until a stopping criterion is satisfied.

The two statistical tests in the line search test candidate iterates for feasibility and improvement in objective. Different from the tests in Angün et al. [105] which consider relative improvements, the tests in this study consider absolute improvements as in Kleijnen through the classic Student t-tests [107]. This approach is simpler than the one in Angüun et al., but requires one more expensive simulation run [105]. Since these tests are exactly the same for both SPSA and RSM, we will describe them before introducing the steps for SPSA and RSM.

Tests for improvement in objective and feasibility:

To decide whether a point has a better estimated objective, we test the null hypothesis in (2.7) (unconstrained case) through the t-statistic in (2.8). For SPSA, $\widehat{S}_0(.)$ is estimated through the sample variance estimator in (2.4), and for RSM it is estimated through the MSR estimator in (2.11). To decide whether the same point is feasible or not, we test the following null hypothesis for each constraint *j*:

$$H_0 : E_\omega\left[s_j\left(\widehat{d}_p, \omega\right)\right] \geq 0 \qquad (2.14)$$
$$H_1 : E_\omega\left[s_j\left(\widehat{d}_p, \omega\right)\right] < 0$$

through the following t-statistic, where s_j is the slack estimated through $\widehat{s}_j = a_j - \widehat{F}_j$:

$$t = \frac{\widehat{s}_j\left(\widehat{d}_p\right)}{\sqrt{\frac{\widehat{S}_j^2\left(\widehat{d}_p\right)}{2}}} \qquad (2.15)$$

Again for SPSA, $\widehat{S}_j(.)$ is estimated through (2.4), and for RSM through (2.11) after replacing $\widehat{F}_0$ by $\widehat{F}_j$. The degrees of freedom are $\nu = 1$ and $\nu = N - (k + 1)$ for SPSA and RSM, respectively. Now the null hypothesis in (2.14) is rejected if the resulting p-value of the test is less than the pre-specified significance level. For simplicity, we use

the same significance level, namely 10%, for both test for improvement in objective and test for feasibility.

2.5.1 Generalized RSM

First, we detail each step of the constrained RSM as follows.

Step 0: Initialize:
Input a fixed, user-specified number of simulation runs per line search, and a maximum total number of simulation runs for the whole simulation study. This maximum can be determined by the time and budget constraints of the expensive simulation study. For the whole study, initialize the number of simulation runs already executed to zero.

Input a fixed, user-specified size of the local experimental area, where the responses F_j in (2.2) are locally approximated by first-order polynomials. This local experimental area should lie within the global area determined by the box constraints in (2.2). The size of this local experimental area is clearly scale dependent, and there are no general guidelines to determine an appropriate size that would work in all applications; see the standard RSM textbooks by Myers and Montgomery, and Khuri and Cornell [18, 77]. Therefore, to determine an appropriate size, the users need to have insight into their application.

To fit first-order polynomials, RSM uses classic designs of resolution-III. The design points $\hat{d}_p$ are simulated to estimate their objectives $\hat{F}_0\left(\hat{d}_p\right)$ and their slacks $\hat{s}_j\left(\hat{d}_p\right)$. The initial iterate, say $\hat{d}_0$, is the feasible design point (among the N design points) that has the minimum objective $\hat{F}_0\left(\hat{d}_0\right)$ estimated through simulation.

The number of simulation runs already executed for the whole study is increased by N, which denotes the number of runs used for initialization.

Step 1: Fit first-order polynomials and estimate variances:

For each response *j*, approximate (2.2) by local first-order polynomials within the local experimental area using (2.10), and obtain point estimates $\hat{\sigma}_{j,j}$ for the variances through (2.11). Because the locally constant variance assumption may not hold globally, use only the most recent estimates $\hat{\sigma}_{j,j}$ in the heuristic.

Step 2: Estimate a search direction and a maximum step size:

Determine the search direction as follows:

$$\hat{g} = -\left(\hat{B}^T \hat{S}^{-2} \hat{B} + \hat{R}^{-2} + \hat{V}^{-2}\right)^{-1} \hat{\beta}_{0,-0} \tag{2.16}$$

where $\hat{S}$, $\hat{R}$ and $\hat{V}$ are diagonal matrices with as main-diagonal elements the current estimated slack vectors s, r, v > 0 and entries of $\hat{S}^{-2}$ are estimated through a single simulation run at $\hat{d}_p$, i.e., $\hat{s}_j^{-2} = \left(a_j - \hat{F}_j\right)^{-2}$. The last factor in (2.16), namely $-\hat{\beta}_{0,-0}$ is the estimated classic steepest-descent direction.

Assuming that the approximations hold globally, a maximum step size into the direction (2.16) can be obtained explicitly through $\lambda_{\max} = \max\{0, \min\{\lambda_1, \lambda_2, \lambda_3\}\}$ where

$$\lambda_1 = \min\left\{\left(c_h - \hat{\beta}_{h,-0}^T \hat{d}_p\right) / \hat{\beta}_{h,-0}^T \hat{g} : h \in \{1,..,r-1\}, \hat{\beta}_{h,-0}^T \hat{g} < 0\right\},$$

$$\lambda_2 = \min\left\{\left(u_i - \hat{d}_{p,i}\right) / \hat{g}_i : i \in \{1,...,k\}, \hat{g}_i > 0\right\},$$

$$\lambda_3 = \min\left\{\left(l_i - \hat{d}_{p,i}\right) / \hat{g}_i : i \in \{1,...,k\}, \hat{g}_i < 0\right\}.$$

To increase the probability of staying within the interior of the feasible region, take only *80%* of $\lambda_{\max}$ as the maximum step size λ.

Step 3: Estimate a line minimum:

This step is illustrated through the flowchart in Figure 2.1. Initialize the number of simulation runs already executed per line search to zero. Simulate at $\hat{d}_p + \lambda \hat{g}$ to $\hat{s}_j$, $\bar{\hat{F}}_0$ and $\hat{S}_j^2$.

At this point, the heuristic compares the current iterate $\hat{d}_p$ with the candidate iterate $\hat{d}_p + \lambda \hat{g}$ to determine the better iterate, where better means both feasible and lower objective. As mentioned in the beginning of this section, the heuristic tests the ratios of both the slacks and the objectives.

Determine the better of $\hat{d}_p$ and $\hat{d}_p + \lambda \hat{g}$. Denote the better by $\hat{d}_c$. Set its objective to $\hat{F}_{0,c}$. Increase the number of already executed simulation runs for both the whole study and the line search by one.

Next the interval $\left[\hat{d}_c, \hat{d}_c + \lambda \hat{g}\right]$ is systematically halved in the same search direction to estimate a line minimum. Such binary search is used since a better point may lie between $\hat{d}_c$ and $\hat{d}_c + \lambda \hat{g}$. Repeat the procedure each time with a new interval $\left[\hat{d}_c, \hat{d}_c + \lambda \hat{g}\right]$, until the fixed number of simulation runs per line search is reached. Then, set the slacks of the current estimated line minimum $\left(\hat{d}_c\right)$ to $\hat{s}_j\left(\hat{d}_c\right)$.

Step 4: Select a resolution-III design and simulate the design points:

The current design point $\hat{d}_c$ and the other design points form the N vertices of a k dimensional hypercube with the side length determined by the fixed size of the local experimental area. Simulate at the new design points. Increase the number of simulation runs already executed for the whole study by the number of new runs.

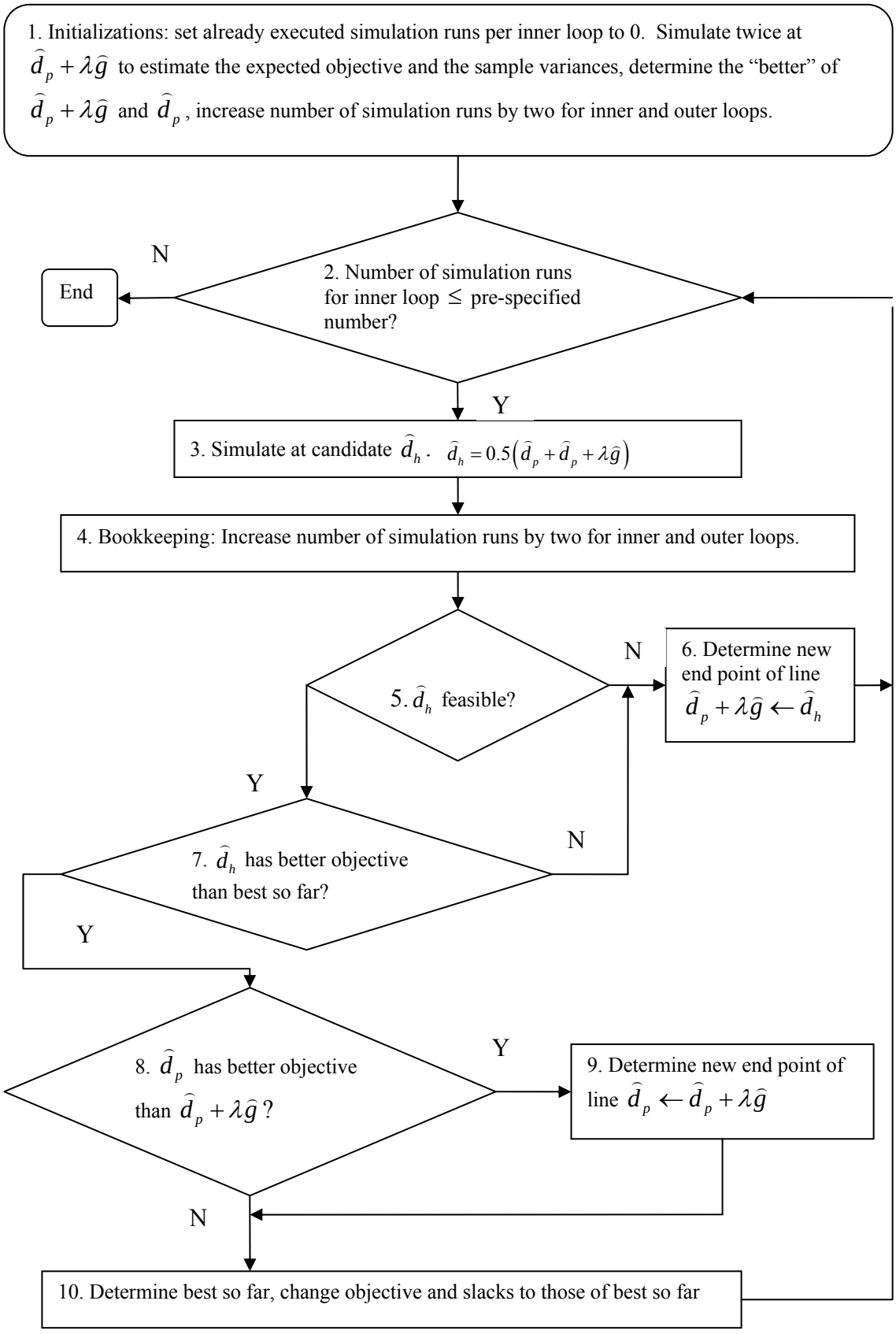

Figure 2.1: Estimating a line minimum

Step 5: Check the stopping criteria:

Stop the heuristic when the number of executed simulation runs for the whole study exceeds the maximum. Otherwise, set $p + 1$ to p, and go to Step 1.

2.5.2. Constrained SPSA

Step 0: Initialize and select coefficients:

Set again the counter index $p = 0$. Pick the same initial input vector $\hat{d}_0$ as in the generalized RSM. Pick nonnegative values for the coefficients c and γ to calculate $\{c_p\}$ through (2.3). We will use the same step size sequence as in the generalized RSM, hence we will not need $\{a_p\}$.

Step 1: Generate the SP vector:

Using parametric bootstrapping, generate a single *k*-dimensional random perturbation vector Δ_p through a symmetric Bernoulli ± 1 distribution.

Step 2: Perform simulation runs:

Perform four runs to obtain *r* simulation outputs of the random responses, denoted by $\hat{F}_j(.)$, at each of $\hat{d}_p + c_p\Delta_p$ and $\hat{d}_p - c_p\Delta_p$ and two simulation outputs for all *r* responses at $\hat{d}_p$. Again, the two extra runs at $\hat{d}_p$ are required to obtain the point estimate of the expected objective and the point estimates of the *r* sample variances of responses through (2.4). These estimates are used in the line search in Step 4 to determine the next estimated iterate.

Step 3: Estimate a search direction and a maximum step size:

Estimate the *r* gradients through (2.5), replacing $\hat{F}_0$ by $\hat{F}_j$. Then, estimate the search direction in (2.16). Now, the rows in $\hat{B}$ consist of the gradients of the constraints estimated through (2.5), the components of $\hat{S}$, $\hat{R}$ and $\hat{V}$ are computed as in the generalized RSM, and $\hat{\beta}_{0,-0}$ is the gradient of the expected objective estimated again

through (2.5). We will use the same step size in the constrained SPSA as in the generalized RSM.

Step 3: Estimate a line minimum:

This step is the same as in the generalized RSM, hence it is illustrated through the flowchart in Figure 2.1.

Step 4: Iterate or terminate:

If the stopping criterion is satisfied (i.e., the maximum number of simulation runs is reached), then stop the constrained SPSA procedure. Otherwise, set $p + 1$ to p, and go to Step 1.

3. APPLICATIONS

3.1. Introduction

In the following subsections, we compare the performance of SPSA with that of RSM. In these subsections, we consider the simplest version of the newsvendor problem as described in, for example, Silver, Pyke, and Peterson [108], the (*s*, *S*) inventory optimization with a service-level constraint which is originally investigated by Bashyam and Fu [83], and the call center originally simulated in Arena and fully specifed in Kelton, Sadowski, and Sadowski [109]. In the final subsection, we present conclusions from these numerical experiments.

We emphasize that the newsvendor problem enables us to control the intrinsic noise of the simulation; moreover, this example is computationally efficient compared with expensive simulation studies that may take weeks. Furthermore, we (the evaluators) know the true optimum, the binding constraints, etc.; therefore, this example provides controlled laboratory settings. The (*s*, *S*) inventory optimization with a service-level constraint and the call-center problem lay the ground for investigating the performances of SPSA and RSM in realistic simulation studies.

To measure the variability in the performances of SPSA and RSM over different sample paths - obtained by simulating with different seed vectors - we generate 1000 macro-replicates (in practice, analysts make a single macro-replicate if their simulation is expensive). At the end of each macro-replicate, we have an estimate $\widehat{d}$ of the true optimal solution and an estimate of the optimal objective value. We then sort these macro- replicates with respect to their deviations from the true optimum and with respect to their deviations from the initial objective value.

Angün et al. compared OptQuest - which is an optimization add-on to various popular simulation software packages - with RSM on a constrained `toy' problem and the call center problem, and found that neither RSM nor OptQuest considerably outperformed the other with respect to the estimated optimal objective value obtained in a relatively small number of simulation runs [105].

We conduct the experiments on a PC with Windows XP, Intel Celeron CPU of 2.40 GHz, and 512 MB RAM. We code the heuristic in Matlab 6.5.

3.2. Newsvendor Problem

We consider a single item inventory control problem with nonnegative, continuous de mand over a single period, which is the simplest unconstrained version of the newsvendor problem.

Let D denote the one period random demand, c the unit acquisition cost, $s > c$ the unit selling price, and $w < c$ the unit salvage value. The distribution of D and the parameters of its distribution are assumed to be known. Now the problem is to determine an optimal order-up-to quantity Q^* such that the following expected net profit function is maximized:

$$E\left[P,(Q)\right] = sE\left[\min\left(Q,D\right)\right] + wE\left[\left(Q-D\right)_+\right] - cQ \tag{3.1}$$

In (3.1), $\min\left(Q,D\right)$ and $\left(Q-D\right)_+ = \max\left(Q-D,0\right)$ correspond to the sold and the salvaged units, respectively, if Q units are ordered at the beginning of the period.

where F is the demand's cumulative distribution function, and $h = c - w$ and $b = s - c$ are the unit overage and underage costs, respectively. If F is strictly increasing, then F has an inverse, and the unique Q^* is given by

$$Q^* = F^{-1}\left(\frac{b}{b+h}\right) \tag{3.2}$$

The problem of determining Q^* and $E\left[P\left(Q^*\right)\right]$ becomes more interesting when there are significant demand uncertainty, and large overage and underage costs. The demand uncertainty is usually measured by the coefficient of variation of demand. In the following, we use uniform distribution for demands over [50, 3500] (coeffcient of variation =0.5611), and different values for a in (2.2) and β, where $\beta = b/(b+h)$.

We comment on Tables 3.1 and 3.2 in the final subsection.

Table 3.1: Variability of the estimated objectives over 1000 macro-replicates for the newsvendor problem: $a = 5$

		Deviations from the true optimal		*Percent deviations from the initial objective*	
		SPSA	RSM	SPSA	RSM
β=0.25	10^{th} quantile (best)	0.719	0.7583	37.54	18.29
	25^{th} quantile	0.7249	0.7604	34.64	17.25
	50^{th} quantile	0.7303	0.7628	32.00	16.11
	75^{th} quantile	0.7354	0.7649	29.52	15.06
	90^{th} quantile (worst)	0.7395	0.7670	27.48	14.06
β=0.50	10^{th} quantile (best)	0.8507	0.8716	37.35	18.14
	25^{th} quantile	0.8524	0.8722	35.84	17.64
	50^{th} quantile	0.8539	0.8728	34.42	17.03
	75^{th} quantile	0.8554	0.8734	0.3310	16.48
	90^{th} quantile (worst)	0.8566	0.8740	31.94	15.95
β=0.75	10^{th} quantile (best)	0.8420	0.8858	113.55	54.38
	25^{th} quantile	0.8460	0.8869	108.14	52.83
	50^{th} quantile	0.8504	0.8881	102.17	51.15
	75^{th} quantile	0.8540	0.8893	97.30	49.55
	90^{th} quantile (worst)	0.8571	0.8904	93.03	48.14

Table 3.2: Variability of the estimated objectives over 1000 macro-replicates for the newsvendor problem: $a = 1$

		Deviations from the true optimal		*Percent deviations from the initial objective*	
		SPSA	RSM	SPSA	RSM
$\beta=0.25$	10^{th} quantile (best)	0.9095	0.9179	271.06	236.51
	25^{th} quantile	0.9099	0.9181	269.58	235.86
	50^{th} quantile	0.9102	0.9182	268.05	235.17
	75^{th} quantile	0.9106	0.9184	266.64	234.50
	90^{th} quantile (worst)	0.9109	0.9186	265.31	233.84
$\beta=0.50$	10^{th} quantile (best)	0.8831	0.8872	7.57	3.81
	25^{th} quantile	0.8834	0.8873	7.34	3.68
	50^{th} quantile	0.8836	0.8875	7.10	3.54
	75^{th} quantile	0.8838	0.8877	6.89	3.38
	90^{th} quantile (worst)	0.8841	0.8878	6.67	3.24
$\beta=0.75$	10^{th} quantile (best)	0.9095	0.9179	22.30	10.91
	25^{th} quantile	0.9099	0.9181	21.81	10.70
	50^{th} quantile	0.9102	0.9182	21.31	10.47
	75^{th} quantile	0.9106	0.9184	20.84	10.25
	90^{th} quantile (worst)	0.9109	0.9186	20.40	10.03

Matlab code of newsvendor problem application on RSM and SPSA can be found in Appendix A.

3.3. (s, S) Inventory Problem with a Service-level Constraint

We now compare SPSA and RSM on the optimization of the (*s*, *S*) inventory system with a service-level constraint. Unlike most authors on inventory models, Bashyam and Fu assume a service-level constraint instead of a penalty cost for back-orders, which is also the approach in practice.

Our goal is to find an estimate $\left(\widehat{s^*}, \widehat{S^*}\right)$ of the optimal reorder and order up to levels s^* and S^* that is, we have two input variables $d = (s, S)^T$. The objective function $E\left[F_0(d, w)\right]$ is the steady-state expected total costs, namely the sum of order setup, ordering, and holding costs. There is a single stochastic constraint $E\left[F_1(d, w)\right]$, which is the expected steady-state 'fill rate', i.e., the fraction of demand directly met from stock on hand. There is also a deterministic constraint on the inputs, namely $S \geq s$.

Like Bashyam and Fu , we assume an infinite horizon, periodic review inventory system with continuous-valued and independent, identically distributed (i.i.d) demands and full backlogging of orders [83]. The basic sequence of events in each period is as follows: orders are received at the beginning of the period, the demand for the period is subtracted, and order review is done at the end of the period. An order is placed when the inventory position (stock on hand plus outstanding suppliers' orders minus customers' back-orders) falls below the reorder level *s*; the order amount is the difference between the order up to level *S* and the current inventory position. Suppliers' orders can cross in time (which has made the analytical solution impossible, so far).

Bashyam and Fu consider various distribution types for customers' demands during a period and order lead times [83]. In particular, Bashyam and Fu calibrate their algorithm using exponentially distributed demands with mean 100 and Poisson distributed order lead times with mean 6 [83]. Furthermore, they report that the results of the calibration problem are quite representative of the large number of test cases that they consider. Therefore, in our simulation experiments, we also assume that customers' demands have an exponential distribution with mean 100 and order lead times have a Poisson distribution with mean 6. Moreover, they set per order setup cost $Z = 36$, per unit order cost $u = 2$, and per period per unit holding cost $h = 1$; we use the same cost values in our simulation experiments. Like Bashyam and Fu [83], each run is simulated for 20,000 periods, since they find that this is sufficient to reach steady-state conditions. Finally, whereas Bashyam and Fu consider various target fill rates including 0.90, in our simulation experiments, we only focus on a target fill rate of 0.90

[83]. Our simulation of the inventory system starts with the inventory position and the inventory level (stock on hand) at *S* without any outstanding suppliers' orders.

In the original Bashyam and Fu's inventory problem, there are no box constraints on *s* and *S* [83]. During the iterations, the matrix in (2.12) can become ill-conditioned. To prevent this situation, we add tentative upper and lower bounds on *s* and *S*, as follows. The lower and upper bounds on s are given by $s_{\min} = E(D)E(L)$ and $s_{\max} = E(D)E(L) + 3\sigma_D E(L)$, respectively, where $E(D)$, $E(L)$ and σ_D are the expected demand, the expected lead time, and the standard deviation of demands, respectively. This gives $600 \le s \le 2400$. To obtain lower and upper bounds on *S*, we use the economic order quantity (EOQ) formula in Bashyam and Fu [83]: $EOQ = \sqrt{2ZE(D)/h}$. We choose the lower and upper bounds on S as $s_{\min} + 0.5EOQ \le S \le s_{\max} + 2EOQ$, which approximately equals $643 \le S \le 2570$.

Bashyam and Fu does not report the optimal (s^*, S^*) for the test case with the Poisson distributed lead times with mean 6, exponentially distributed demands with mean 100, and 0.90 target fill rate [83]. Therefore, Angün et al. found the 'optimal' (s^*, S^*) through brute-force simulation over a grid [105]. They simulated for 30,000 simulation periods and average the costs and the fill rates over 10 macro-replicates over (s, S) plane, starting with $600 \le s \le 2400$ and $643 \le S \le 2570$, and a coarse grid. By refining the grid successively and reaching to 1×1, their conclusion of these brute-force simulation experiments is that *s* = 1160 and *S* = 1212 - which has the average cost of 647.15 with a standard error of 8.55 and the average fill rate of 0.8948 with a Standard error of 0.01 - is the best estimate of (s^*, S^*).

Table 3.3 shows the test results of 70 simulation runs per macro-replicate and the starting point is *s* = 700 and *S* =750.

Table 3.3: Variability of the estimated objectives over 1000 macro-replicates for the (*s*, *S*) inventory problem.

	Deviations from the true optimal		*Percent deviations from the initial objective*	
	SPSA	RSM	SPSA	RSM
10^{th} **quantile (best)**	0.615	0.658	21.5	20.51
25^{th} **quantile**	0.664	0.668	22.12	19.45
50^{th} **quantile**	0.694	0.703	19.23	18.56
75^{th} **quantile**	0.721	0.736	14.25	15.69
90^{th} **quantile (worst)**	0.804	0.797	13.49	13.57

We again comment on Table 3.3 in the final subsection.

3.4. Generic Call Center Problem

Given in Simulation with Arena book by Kelton et al., the generic call center system provides a central number in an organization that customers call for technical support, sales information, and order status [109]. This central number feeds 26 trunk lines. If all 26 lines are in use, a caller gets a busy signal; or an answered caller hears a recording describing three options: transfer to technical support (76%), sales information (16%), or order-status inquiry (and 8%). The estimated time for this activity is UNIF (0.1, 0.6) with times in minutes.

Technical support calls are routed to a second recording that asks the caller which of three product types (1, 2 or 3) he is using (with the percentages of 25%, 34% or 41% respectively. The estimated time for technical support product type choice is UNIF (0.1, 0.5).

If a qualified technical support person is available for the choosen product type, the call is automatically routed to that support person for immediate service. If no support person is available at the moment, the call is placed in an electronic queue until one is available. The customer exits the system as soon as the the call is completed. Besides,

4% of these technical support calls needs further assistance after completion of the call. The questions of these callers are forwarded to another technical group, outside the boundaries of the defined model that prepares a response. And the resulting response is sent back to the same technical support person who answered the original call. Then, this support person calls the customer back. These calls use of one of the 26 trunk lines and takes priority over incoming calls. A returned call is carried over to the next day if it could not be completed on the same day the original call was received. The estimated time for all technical support calls is given as TRIA(3, 6, 18), for the response preparation time for further investigation required technical calls is given as EXPO(60) and for customer recall time is given as TRIA(2, 4, 9).

Sales calls are automatically routed to the sales staff which consists of seven sales people. If sales person is not available, the customer is treated to space music. Again, upon completion of the call, the caller exist the system. The sales calls are estimated to be TRIA(4, 15, 45).

Order-status calls are automatically handled by the phone system. Except 26 trunk lines, there is no limit on the number handled at a time. The estimated time for order status call transactions is TRIA(2, 3, 4). 15% of callers talk to a real person after they have received their order status. Having the same priority as incoming sales calls, these customers are routed to the sales staff, and then exit the system. The estimated time for follow up order status calls is TRIA(3, 5, 10).

The call center operates from 8 a.m. to 6 p.m., and a small proportion of the staff stays until 7 p.m. No new calls are accepted after 6 p.m., but all calls that entered before 6 p.m. are answered.

The call arrival rate varies substantially over the day, and is expressed in calls per hour for each 30-minute period during which the system is open. The call-arrival rates are given in Table 3.4 [109].

The daily schedules of 7 sales people are given as (number of people @ time period in minutes): 3@90, 7@90, 6@90, 7@60, 6@120, 7@120 and 4@90.

Table 3.4: Call arrival rates (Calls per hour)

Time	*Rate*	*Time*	*Rate*	*Time*	*Rate*	*Time*	*Rate*
8.00-8.30	20	10.30-11.00	75	13.00-13.30	110	15.30-16.00	90
8.30-9.00	35	11.00-11.30	75	13.30-14.00	95	16.00-16.30	70
9.00-9.30	45	11.30-12.00	90	14.00-14.30	105	16.30-17.00	65
9.30-10.00	50	12.00-12.30	95	14.30-15.00	90	17.00-17.30	45
10.00-10.30	70	12.30-13.00	105	15.00-15.30	85	17.30-18.00	30

All technical support people work an eight-hour day with 30 minutes off for lunch (lunch is not included in the eight hours). An example of 11 technical support people's schedule can be found in Kelton et al. [109]. Some of these staff are qualified for product type 1, some are qualified for 2, some are qualified for 3, and some of them are qualified for all product types (1, 2 and 3).

In the system, the term "balking" is used for the number of customer calls that are not able to get a trunk line. Customers who hang up the phone before reaching a real person are not considered.

Based on the defined model, our objective is to find an optimal solution (x_1^*, x_2^*, x_3^*), where x_1 is all product support people (support people qualified for all product types), x_2 is the additional sales staff, x_3 is the number of trunk lines, all of which minimize the total system cost. These decision variables have the following constraints: $0 \le x_1 \le 22$, $0 \le x_2 \le 22$, $x_1 + x_2 \le 22$ and $26 \le x_3 \le 50$. The objective function is the expected total system cost of new technical people and new sales staff, and new trunk lines. Besides the constraints about decision variables, we have another constraint called "percent available signal", which says that the ratio of available lines over all trunk lines in the system shall be greater than or equal to 0.95.

We use resolution-III design, so we have $n = 8$ input combinations. We simulate $m_l = 1$ replicates at each design point and observe total system cost and percent free for each input combinations as given ın Table 3.5. Our stopping criterion - the maximum number of runs – is 25 simulation runs.

Table 3.5: Input combinations for call center model

Simulation Run	x_1	x_2	x_3	*Total system cost* ($\widehat{F}_0$)	*Percent available (%)* ($\widehat{F}_1$)
1	5	3	32	2305	0.9544
2	5	3	31	2333	0.9604
3	5	2	32	2267	0.9534
4	4	3	32	2274	0.9404
5	4	2	31	2247	0.9359
6	4	3	31	2285	0.9188
7	4	2	32	2226	0.9672
8	4	2	31	2212	0.9394

The problem explained above can be formulated as:

$$\begin{aligned} &\text{minimize } \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 + \widehat{\beta}_3 x_3 \\ &\text{subject to } \hat{\gamma}_0 + \hat{\gamma}_1 x_1 + \hat{\gamma}_2 x_2 + \hat{\gamma}_3 x_3 \geq 0.95 \\ &\qquad x_1 + x_2 \leq 22 \\ &\qquad 0 \leq x_1 \leq 22, 0 \leq x_2 \leq 22, 26 \leq x_3 \leq 50 \end{aligned} \tag{3.3}$$

where $\widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 + \widehat{\beta}_3 x_3 \hat{\beta}$ is expected total cost and $\hat{\gamma}_0 + \hat{\gamma}_1 x_1 + \hat{\gamma}_2 x_2 + \hat{\gamma}_3 x_3 \geq 0.95$ is expected percent available.

We calcule as given in (2.10), and as:

$$\hat{\gamma} = \left(X^T X \right)^{-1} X^T \widehat{F}_1 \tag{3.4}$$

where $\widehat{F}_0$ and $\widehat{F}_1$ are total system cost vector and percent available vector gathered from simulation runs.

We choose the input combination which staisfies the constraints and has the lowest cost as a starting point $\hat{d}_p$. So, we have $\hat{d}_p = (4 \quad 2 \quad 32)$.

Table 3.6 shows the best objectives reached at the end of 25 simulation runs, and percent improvements from the starting point for RSM and SPSA. Results of each simulation run for RSM and SPSA can also be found in Appendix B.

Table 3.6: Estimated objectives and percent improvements over 25 simulation runs for call center application

	RSM	*SPSA*
Initial Expected Cost	2226	2226
Best Expected Cost	2082	2136
Percent Improvement (%)	6.47	4.04

We comment on Table 3.6 in the final subsection.

3.5. Conclusions for Numerical Examples

According to our results in Tables 3.1, 3.2, and 3.3 SPSA performs better than RSM. However, SPSA is shown to be more sensitive to randomness and changes in parameters. According to Table 3.6, RSM performs beter than SPSA, and has a beter improvement based on the starting point than SPSA.

4. CONLUSION

In this study, we compared performances of black box simulation optimization methodologies response surface methodology and simultaneous perturbation stochastic approximation numerically, under the assumption that the computer budget is limited and each simulation run is time-consuming.

Firstly, we applied RSM and SPSA on the newsvendor problem. Here, we considered a single item inventory control problem with nonnegative, continuous demand over a single period, which was the simplest unconstrained version of the newsvendor problem.

Then, we compared these two methodes on the optimization of the (*s*, *S*) inventory system with a service-level constraint. In this example, we adopted Bashyam and Fu's approach which assumes a service-level constraint instead of a penalty cost for back-orders [83].

At last, we applied RSM and SPSA on generic call center model described in Kelton et al [109], which was a constrained optimization problem example.

We observed that, in the newsvendor and (*s*, *S*) inventory applications, SPSA performed better than RSM, but was more sensitive to randomness and changes in parameters. However, in the call center model application, RSM performed beter than SPSA, and had a beter improvement than SPSA based on the starting point.

We conclude that, for future research, we can compare SPSA and RSM on a badly scaled problem, where RSM can outperform SPSA. We comment that this can be

expected since with the new search directions, RSM is scale independent whereas SPSA is scale dependent.

REFERENCES:

[1] Gass, S.I., Haris, C.M., *Encyclopedia of Operations Research and Management Science*, 2nd ed. Kluwer Academic, Boston, (2000).

[2] Tekin E., Sabuncuoglu, I., "Simuation optimization: A comprehensive review on theory and applications", *IEE Transactions*, 36, 1067 – 1081, (2004).

[3] Fu, M.C., "Optimization for Simulation: Theory vs. Practice", *INFORMIS Journal on Computing,* 14 (3), 192-215, (2002).

[4] Andradóttir, S., Carson, J.S., Fu, M.C., Glover, F., Harrell, C.R., Ho, Y.C., Kelly, J.P., Robinson, S.M, "Integrating optimization and simulation: research and practice", *Proceedings of the 2000 Winter Simulation Conference*, 610-616, (2000).

[5] Azadivar, F., "Simulation optimization methodologies", *Proceedings of the 1999 Winter Simulation Conference*, (1999).

[6] Andradóttir, S., "A review of simulation optimization techniques", *Proceedings of the 1998 Winter Simulation Conference*, 151-158, (1998).

[7] http://www.wintersim.org/

[8] Azadivar, F., Truong, T.H., "Simulation based optimization for supply chain configuration design", *Proceedings of the 2003 Winter Simulation Conference*, (2003).

[9] April, J., Glover, F., Kelly, P.J., Laguna, M., "Practical introduction to simulation optimization", *Proceedings of the 2003 Winter Simulation Conference*, (2003).

[10] Martin, E., Schouwenaar, M., "Optimization of a telecommunication billing System", *Proceedings of the 2003 Winter Simulation Conference*, (2003).

[11] den Hertog, D., Stehouwer, H.P., "Optimizing color picture tubes by high-cost nonlinear programming", *European Journal of Operational Research,* 140 (2), 197-211, (2002).

[12] Ben-Gal, I., Bukchin, J., "The ergonomic design of workstations using virtual manufacturing and response surface methodology", *IIE Transactions,* 34, 375-391, (2002).

[13] Hutchison, D.W., Hill, S.D., "Simulation optimization of airline delay with constraints", *Proceedings of the 2001 Winter Simulation Conference,* (2001).

[14] Kleijnen, J.P.C., "Simulation and optimization in production planning: a case study", *Decision Support Systems*, 9, 269-280, (1993).

[15] Irizarry, M., Kuhl, M.E., Lada, E.K., Subramanian, S., Wilson, J.R., "Analyzing transformation-based simulation metamodels", *IIE Transactions*, 35, 271-283, (2003).

[16] Dean, A.M., Lewis, S.M., eds., *Screening*, Springer-Verlag, New York, (2004).

[17] Trocine, L., Malone, L.C., "Finding important independent variables through screening designs: a comparison of methods", *Proceedings of the 2000 Winter Simulation Conference,* (2000).

[18] Myers, R.H., Montgomery, D.C., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed. Wiley, New York, (2002).

[19] Kleijnen, J.P.C., *Statistical Tools for Simulation Practitioners*, Marcel Dekker, New York, (1987).

[20] Bettonvil, B., Kleijnen, J.P.C., "Searching for important factors in simulation models with many factors: sequential bifurcation", *European Journal of Operational Research*, 96, 180-194, (1996).

[21] Cheng, R.C.H., "Searching for important factors: sequential bifurcation under uncertainty", *Proceedings of the 1997 Winter Simulation Conference,* (1997).

[22] Campolongo, F., Kleijnen, J.P.C., Andres, T., "Screening methods in sensitivity analysis", *Mathematical and Statistical Methods for Sensitivity Analysis of Model Output*, (2000).

[23] Box, G.E.P., Wilson, K.B., "On the experimental attainment of optimum Conditions", *Journal Royal Statistical Society*, Series B, 13 (1), 1-38, (1951).

[24] Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P., "Design and analysis of computer experiments", *Statistical Science*, 4 (4), 409-435, (1989).

[25] Jones, D.R., Schonlau, M., Welch, W. J., "Efficient global optimization expensive black-box functions", *Journal of Global Optimization*, 13, 455-492, (1998).

[26] Simpson, T.W., Mauery, T.M., Korte, J.J., Mistree, F., "Kriging metamodels for global approximation in simulation-based multidisciplinary design optimization", *American Institute of Aeronautics and Astronautics Journal*, 39 (12), 2233-2241, (2001).

[27] Robbins, H., Monro, S., "A stochastic approximation method", *Annals of Mathematical Statistics*, 22:400-407, (1951).

[28] Kiefer, J., Wolfowitz, J., "Stochastic estimation of the max function", *Annals of Mathematical Statistics*, 23, 462-466, (1952).

[29] Spall, J.C., "A Stochastic Approximation Technique for Generating Maximum Likelihood Parameter Estimates", *Proceedings of the American Control Conference*, 1161-1167, (1987).

[30] Spall, J.C., "A Stochastic Approximation Algorithm for Large-Dimensional Systems in the Kiefer-Wolfowitz Setting", *Proceedings of the IEEE Conference on Decision and Control* ,1544-1548, (1988).

[31] Spall, J.C., "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", *IEEE Transactions on Automatic Control,* 37 (3), 332-341, (1992).

[32] Spall, J.C., *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley, Hoboken, New Jersey, (2003).

[33] Ho, Y.C., Cao, X.R., *Perturbation Analysis of Discrete Event Dynamic Systems*, Kluwer Academic, Boston, (1991).

[34] Fu, M.C., Hu, J.Q., *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, Kluwer Academic, Boston, (1997).

[35] Rubinstein, R.Y., Shapiro, A., *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, Wiley, Chichester, (1993).

[36] Kleijnen, J.P.C., "Experimental design for sensitivity analysis, optimization and validation of simulation models", *Handbook of Simulation*, (1998).

[37] Brekelmans, R., Driessen, L., Hamers, H., den Hertog, D., "Gradient estimation schemes for noisy functions", *Journal of Optimization Theory and Applications*, (2004).

[38] Spall, J.C., "Adaptive stochastic approximation by the simultaneous perturbation method", *IEEE Transactions on Automatic Control,* 45 (10), 1839-1853, (2000).

[39] Spall, J.C., "Implementation of the simultaneous perturbation algorithm for stochastic optimization", *IEEE Transactions on Aerospace and Electronic Systems,* 34 (3), 817-823, (1998).

[40] L'Ecuyer, P., "An overview of derivative estimation", *Proceedings of the 1991 Winter Simulation Conference*, (1991).

[41] Glasserman, P., *Gradient Estimation Via Perturbation Analysis*, Kluwer Academic, Boston, MA., (1991).

[42] Glynn, P.W., "Optimization of stochastic systems via simulation", *Proceedings of the 1989 Winter Simulation Conference,* (1989).

[43] Boesel, J., Bowden, R.O., Glover, F., Kelly, J.P., Westwig, E., "Future of simulation optimization", *Proceedings of the 2001 Winter Simulation Conference*, (2001).

[44] Goldsman, D., Nelson, B.L., Opicka, T., Pritsker, A.A.B. "A ranking and selection project: experiences from a university-industry collaboration", *Proceedings of the 1999 Winter Simulation Conference*, (1999).

[45] Boesel, J., Nelson, B.L., Kim, S.H., "Using ranking and selection to clean up after simulation optimization", *Operations Research,* 51 (5), 814-825, (2003).

[46] Gürkan, G., Özge, A.Y., Robinson, S.M., "Sample-path optimization in simulation", *Proceedings of the 1994 Winter Simulation Conference*, (1994).

[47] Gürkan, G., Özge, A.Y., Robinson, S.M., "Solving stochastic optimization problems with stochastic constraints: an application in network design", *Proceedings of the 1999 Winter Simulation Conference*, (1999).

[48] Spall, J.C., "Stochastic optimization and the simultaneous perturbation method", *Proceedings of the 1999 Winter Simulation Conference,* (1999).

[49] Kleinman, N.L., Spall, J.C., Naiman, D.Q., "Simulation-based optimization with stochastic approximation using common random numbers", *Management Science,* 45 (11), 1570-1578, (1999).

[50] Maryak, J.L., Chin, D.C., **"**Global random optimization by simultaneous perturbation stochastic approximation", *Proceedings of the 2001 Winter Simulation Conference*, (2001).

[51] Gerencsér, L., Hill, S.D., Vagó, Z., “Optimization over discrete sets via SPSA”, *Proceedings of the 1999 Winter Simulation Conference,* 466 - 470, (1999).

[52] Whitney, J.E., Solomon, L.I., Hill, S.D., “Constrained optimization over discrete sets via SPSA with application to non-separable resource allocation”, *Proceedings of the 2001 Winter Simulation Conference*, (2001).

[53] Michalewicz, Z., Schoenauer, M., “Evolutionary algorithms”, *Encyclopedia of Operations Research and Management Science*, 2nd ed., (2001).

[54] Fouskakis, D., Draper, D., “Stochastic optimization: a review”, *International Statistical Review,* 70 (3), 315-349, (2002).

[55] Glover, F., Laguna, M., *Tabu Search*, Kluwer, Norwell, MA., (1997) .

[56] Glover, F., “Tabu search”, *Encyclopedia of Operations Research and Management Science*, 2nd ed., (2001).

[57] Kirkpatrick, S., Gelatto, C., Vecchi, P., “Optimization by simulated annealing”, *Science*, 221, 671-680, (1993).

[58] Gutjahr, W.J., Pflug, G.Ch., “Simulated annealing for noisy cost functions”, *Journal of Global Optimization,* 8, 1-13, (1996).

[59] Alrefaei, M.H., Andradóttir, S., "A simulated annealing algorithm with constant temperature for discrete stochastic optimization." *Management Science,* 45: 748-764, (1999).

[60] Anandalingam, G., “Simulated annealing”, *Encyclopedia of Operations Research and Management Science*, 748-751, (2001).

[61] Ho, Y.C., Sreenivas, R., Vakili, P., "Ordinal Optimization of Discrete Event Dynamic Systems", *J. of Discrete Event Dynamic Systems,* 2(2), 61-88, (1992).

[62] Ho, Y.C., Cassandras, C.G., Chen, C.H., Dai, L., “Ordinal optimisation andsimulation”, *Journal of the Operational Research Society,* 51, 490-500, (2000).

[63] Dai, L., “Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems”, *Journal of Optimization Theory and Applications,* 91, 363-388, (1996).

[64] Dai, L., Chen, C., “Rate of convergence for ordinal comparison of dependent simulations in discrete event dynamic systems”, *Journal of Optimization Theory and Applications,* 94, 29-54, (1997).

[65] Goldsman, D., Nelson, B.L., "Ranking, selection and multiple comparisons in computer simulation", *Proceedings of the 1994 Winter Simulation Conference*, 192-199, (1994).

[66] Patsis, N.T., Chen, C., Larson, M.E., "SIMD parallel discrete event dynamicsystem simulation", *IEEE Transactions on Control Systems Technology*, 5, 30-41, (1997).

[67] Ho, Y.C., Larson, M.E., "Ordinal optimization approach to rare event probability problems", *Journal of Discrete Event Dynamic Systems*, 5, 281-301, (1995).

[68] Chen, C., Kumar, V., Luo, Y., "Motion planning of walking robots using ordinal optimization", *IEEE Robotics and Automation Magazine*, 5, 22-32, (1998).

[69] http://www.jhuapl.edu/SPSA/Pages/MATLAB.htm

[70] Jacobson, S.H., Schruben, L.W., "A review of techniques for simulation optimization, *Operations Research Letters*, 8, 1-9, (1989).

[71] Safizadeh, M.H., "Optimization in simulation: current issues and future Outlook", *Naval Research Logistics*, 37, 807-825, (1990).

[72] Fu, M.C., Glover, F.W., April, J., "Simulation optimization: a review, new developments, and applications", *Proceedings of the 2005 Winter Simulation Conference*, (2005).

[73] Henderson, S.G., Nelson, B.L., editors, Handbooks in Operations Research and Management Science, Volume 13, Elsevier/North Holland, (2006).

[74] Kleijnen, J.P.C., *DASE: Design and Analysis of Simulation Experiments*, Springer Science + Business Media, (2008).

[75] Robinson, S.M., "Analysis of sample-path optimization", *Mathematics of Operations Research*, 21, 513-528, (1996).

[76] Fu, M.C., Hu, J.Q., *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*. Kluwer Academic, Boston, (1997).

[77] Khuri, A.I., Cornell, J. A., *Response Surfaces: Design and Analysis*, 2nd edn. Marcel Dekker, New York, (1996).

[78] L'Ecuyer, P., Giroux, N., Glynn P.W., "Stochastic Optimization by Simulation: Numerical Experiments with the M/M/1 Queue in Steady-State", *Management Science*, 40, 1245 - 1261, (1994).

[79] Simpson, T.W., Booker, A.J., Ghosh, D., Giunta, A.A., Koch, R.J., Yang, R.J., "Approximation methods in multidisciplinary analysis and optimization: A panel discussion", *Structural and Multidisciplinary Optimization*, 27 (5), 302–313, (2004).

[80] Chin, D.C., "Comparative study of stochastic algorithms for system optimization based on gradient approximations", *IEEE Transactions on Systems, Man, and Cybernetics*, Series B, 27, pp. 244-249, (1997).

[81] Fu, M.C., Hill, S. D., "Optimization of discrete event systems via simultaneous perturbation stochastic approximation", *IIE Transactions* 29, 233-243, (1997).

[82] Kleijnen, J.P.C., Wan, J., "Optimization of simulated inventory systems : optquest and alternatives", *Discussion Paper 75*, Tilburg University, Center for Economic Research, (2006).

[83] Bashyam, S., Fu, M.C., **"**Optimization of (s, S) Inventory Systems with Random Lead Times and a Service Level Constraint", *Management Science*, 44, No. 12-Part-2, 243-256, (1998).

[84] Blum, J.R., "Multidimensional stochastic approximation methods", *Ann. Math. Statistics*, 25 737-744, (1954).

[85] Burnett, R. (2004), "Application of Stochastic Optimization to Collision Avoidance," *Proceedings of the American Control Conference*, 29, 2789-2794, (2004).

[86] Bhatnagar, S., Borkar, V.S., "Multiscale Chaotic SPSA and Smoothed Functional Algorithms for Simulation Optimization", *Simulation*, 79 (10), 568-580, (2003).

[87] Maryak, J., Spall, J., Heydon, B., "Use of the Kalman filter for inference in state-space models with unknown noise distributions", *IEEE Transactions on Automatic Control*, 49 (1), (2004).

[88] Dabbene, F., Gay, P., Sacco, N., "Optimization of Fresh-Food Supply Chains in Uncertain Environments, Part I: Background and Methodology", *BioSystems Engineering*, 99 (3), 348-359, (2008).

[89] Gosavi, A., Ozkaya, E., Kahraman, A., "Simulation Optimization for Revenue Management of Airlines With Cancellations and Overbooking.", *To appear in OR Spectrum: Special issue on revenue management*, 29, 21-38, (2007).

[90] Rezayat, F., “On the use of an SPSA -based model- free controller in quality improvement”, *Automatica*, 31, 913-915, (1995).

[91] Sadegh, P., Spall, J.C., “Optimal Random Perturbations for Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation”, *Proc. American Control Conf*., 3582–3586, (1997).

[92] Rezayat, F., “Constrained SPSA controller for operations processes”, *IEEE Transactions on Systems, Man, and Cybernetics Part A*, 29 (6), 645-649, (1999).

[93] Wang, I., Spall, J.C., "Stochastic optimisation with inequality constraints using simultaneous perturbations and penalty functions," *International Journal of Control*, 81 (8), 1232-1238, (2008).

[94] Sadegh, P., “Constrained optimization via Stochastic Approximation with a Simultaneous Perturbation Gradient Approximation”, *Elsevier Science,* Vol.33, No.5, 889-892, (1997).

[95] Maryak, J.L., Chin, D.C., “Efficient global optimization using SPSA”, *Proc. Amer. Control Conf.*, 890–894, (1999).

[96] Hill S., D., “Discrete Stochastic Approximation with Application to Resource Allocation”, *Johns Hopkins Apl technicalDigest*, 26 (1), (2005).

[97] Biles, W. E., “A gradient regression search procedure for simulation experimentation”, *Proceedings of 1974 Winter Simulation Conference*, 491-497, (1974).

[98] Barton, R. R., Meckesheimer, M., “Metamodel-based simulation optimization”, *Proceedings of the 2006 Winter Simulation Conference*, (2006).

[99] Irizarry, M., Wilson, J.R., Trevino, J., “A flexible simulation tool for manufacturing-cell design, I: model structure, operation, and case study”, *IIE Transactions,* 33*,* 827 - 836, (2001).

[100] Yang, T., Tseng, L., “Solving a multiple objective simulation model using a hybrid response surface method and lexicographical goal programming approach-a case study on IC ink marking machines”, *Journal of the Operational Research Society*, 53, 211-221, (2002).

[101] Chin, D. C., "Comparative Study of Stochastic Algorithms for System Optimization Based on Gradient Approximations", *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, 27, 244-249, (1997).

[102] Kao, C., Chen, C.Y., Tseng, M.C., "Backward Compatible Modulation Schemes for Improving Spectrum Efficiency in DAB Systems", *IEEE 2006 VTC Proc.*, Montreal, (2006).

[103] Montgomery, D. C., *Introduction to statistical quality control, 3rd ed*, John Wiley & Sons, New York, NY, (1997).

[104] Kleijnen, J.P.C., den Hertog, D., Angün, E., "Response surface methodology's steepest ascent and step size revisited", *European Journal of Operational Research*, 159, 121-131, (2004).

[105] Angün, E., Kleijnen, J.P.C., den Hertog, D., Güurkan, G., "Response surface methodology with stochastic constraints for expensive simulation", *Journal of the Operational Research Society,* 60, 735-746, (2009).

[106] Kleijnen, J.P.C., "An overview of the design and analysis of simulation experiments for sensitivity analysis", *European Journal of Operational Research*, 164(2), 287-300, (2005).

[107] Kleijnen, J.P.C., "Response surface methodology for constrained simulation optimization: an overview", *Simulation Modelling Practice and Theory,* 16, 50-64, (2008).

[108] Silver, E.A., Pyke, D.F., Peterson, R., *Inventory Management and Production Planning and Scheduling 3rd edn,* John Wiley & Sons, New York, NY, (1998).

[109] Kelton, W.D., Sadowski, R.P., Sadowski, D.A., *Simulation with Arena 2nd edn.*, McGraw-Hill, Boston, (2002).

APPENDIX A:

```
% fix seeds
rand('state',0);
randn('state',0);
s = 6;
c = 5;
w = 2;
%initializations
ratioestobjspsa=[];
ratioestobjrsm=[];
percentimpspsa=[];
percentimprsm=[];

%inputs
overagecost = input('Overage cost = ');
underagecost = input('Underage cost = ');
numberofdemands = input('Number of demands = ');
demanddistribution = input('Demand distribution type (uniform (l, u) (1), exponential (2), normal (3), lognormal (4): ');
beta = underagecost / (overagecost + underagecost);

%input parameters of the demand distribution
if demanddistribution == 1
   parameter1 = input('Demand parameter l = ');
   parameter2 = input('Demand parameter u = ');
elseif demanddistribution == 2
   parameter1 = input('Demand mean = ');
```

```
    parameter2 = 0;
elseif demanddistribution == 3
    parameter1 = input('Demand mean = ');
    parameter2 = input('Demand standard deviation = ');
elseif demanddistribution == 4
    parameter1 = input('Ln demand mean = ');
    parameter2 = input('Ln demand standard deviation = ');
end

%analytical optimum solution and optimal objective value
[optimumsolution,optimalobjvalue]=analyticaloptimum(demanddistribution,parameter1,
parameter2, underagecost,overagecost,beta);

%optimization inputs
initialorderquantity = input('Initial order quantity = ');
demands = unifrnd(parameter1,parameter2,10000,1);
initalobjectives = s * min(demands(:),initialorderquantity) + w *
max(initialorderquantity-demands(:),0) - c * initialorderquantity;
initialobjective = mean(initalobjectives);
iterationnumber = input('Number of iterations = ');
stepconstant1 = input('Step constant 1 = ');

%common constants for rsm and spsa
stepconstant2 = 0.1*iterationnumber;
stepconstant3 = 0.602;
perturbconstant2 = 0.101;
for macro = 1:1000

%initial point for both spsa and rsm
    currentiteratespsa = initialorderquantity;
    currentiteratersm = initialorderquantity;
%initialize
```

```
   steps = [];
   perturbs = [];
   spghats = [];
   iteratesspsa = [];
   objspsa=[];
   rsmghats = [];
   iteratesrsm = [];
   objrsm = [];
   iteratesspsa = [iteratesspsa;initialorderquantity];
   iteratesrsm = [iteratesrsm;initialorderquantity];

%initial demand generation through monte carlo simulation
[demands,demandsplus,demandsminus,cvdemand,stdevdemand]=montecarlo(demanddi
stribution,parameter1, parameter2,numberofdemands);

%loop for spsa and rsm
   for iteration = 1 : iterationnumber

%step and perturbation sizes
      perturbconstant1 = cvdemand;
      step = stepconstant1 /(iteration + stepconstant2)^stepconstant3;
      steps = [steps;step];
      perturb = perturbconstant1 / (iteration^perturbconstant2);
      perturbs=[perturbs;perturb];

%profit estimation at the design points for spsa
[averageprofitplusspsa,averageprofitminusspsa,averageprofitcurrentspsa,delta]=profitest
imationspsa(currentiteratespsa,perturb,numberofdemands,demands,demandsplus,deman
dsminus,underagecost,overagecost);
      objspsa = [objspsa;averageprofitcurrentspsa];

%profit estimation at the design points for rsm
```

```
[averageprofitplusrsm,averageprofitminusrsm,averageprofitcurrentrsm]=profitestimatio
nrsm(currentiteratersm,perturb,
numberofdemands,demands,demandsplus,demandsminus,underagecost,overagecost);
      objrsm = [objrsm;averageprofitcurrentrsm];

%spsa iterates
[ghatspsa,currentiteratespsa]=spsa(averageprofitplusspsa,averageprofitminusspsa,pertur
b,delta, currentiteratespsa,step);
      spghats=[spghats;ghatspsa];
      iteratesspsa = [iteratesspsa;currentiteratespsa];

%rsm iterates
[ghatrsm,currentiteratersm]=rsm(currentiteratersm,perturbconstant1,averageprofitplusrs
m, averageprofitminusrsm,step);
      rsmghats=[rsmghats;ghatrsm];
      iteratesrsm = [iteratesrsm;currentiteratersm];

%demand generation through monte carlo simulation
[demands,demandsplus,demandsminus,cvdemand,stdevdemand]=montecarlo(demanddi
stribution,parameter1, parameter2,numberofdemands);
   end
   if iteration == iterationumber
      ratioestobjspsa=[ratioestobjspsa;(optimalobjvalue-
averageprofitcurrentspsa)/optimalobjvalue];
      ratioestobjrsm=[ratioestobjrsm;(optimalobjvalue-
averageprofitcurrentrsm)/optimalobjvalue];
      percentimpspsa=[percentimpspsa;(averageprofitcurrentspsa-
initialobjective)/initialobjective];
      percentimprsm=[percentimprsm;(averageprofitcurrentrsm-
initialobjective)/initialobjective];
   end
end
```

```
pr10spsa=prctile(ratioestobjspsa,10);
pr10rsm=prctile(ratioestobjrsm,10);
pr25spsa=prctile(ratioestobjspsa,25);
pr25rsm=prctile(ratioestobjrsm,25);
pr50spsa=prctile(ratioestobjspsa,50);
pr50rsm=prctile(ratioestobjrsm,50);
pr75spsa=prctile(ratioestobjspsa,75);
pr75rsm=prctile(ratioestobjrsm,75);
pr90spsa=prctile(ratioestobjspsa,90);
pr90rsm=prctile(ratioestobjrsm,90);

primp10spsa=prctile(percentimpspsa,10);
primp10rsm=prctile(percentimprsm,10);
primp25spsa=prctile(percentimpspsa,25);
primp25rsm=prctile(percentimprsm,25);
primp50spsa=prctile(percentimpspsa,50);
primp50rsm=prctile(percentimprsm,50);
primp75spsa=prctile(percentimpspsa,75);
primp75rsm=prctile(percentimprsm,75);
primp90spsa=prctile(percentimpspsa,90);
primp90rsm=prctile(percentimprsm,90);
```

APPENDIX B:

Table B.1: Simulation run results for RSM

Simulation Run	x_1	x_2	x_3	*Total system cost* $(\widehat{F}_0)$	*Percent available (%)* $(\widehat{F}_1)$
1	5	3	32	2305	0.9544
2	5	3	31	2333	0.9604
3	5	2	32	2267	0.9534
4	4	3	32	2274	0.9404
5	5	2	31	2247	0.9359
6	4	3	31	2285	0.9188
7	4	2	32	2226	0.9672
8	4	2	31	2212	0.9394
9	3.8958	1.9559	32.0077	2209	0.9563
10	3.8958	1.9559	32.0077	2222	0.9528
11	3.9479	1.9779	32.0039	2225	0.9528
12	3.9479	1.9779	32.0039	2213	0.9563
13	3.9219	1.9669	32.0058	2223	0.9528
14	3.9219	1.9669	32.0058	2211	0.9563
15	3.9219	1.9669	31.0058	2232	0.9045
16	3.9219	0.9669	32.0058	2155	0.9469
17	2.9219	1.9669	32.0058	2181	0.9554
18	3.9219	0.9669	31.0058	2154	0.9561
19	2.9219	1.9669	31.0058	2174	0.9718
20	2.9219	0.9669	32.0058	2116	0.9520
21	2.9219	0.9669	31.0058	2102	0.9781
22	2.7843	0.7231	31.4967	2082	0.9781

23	2.7843	0.7231	31.4967	2097	0.9509
24	2.8531	0.8450	31.2512	2092	0.9781
25	2.8531	0.8450	31.2512	2107	0.9509

Table B.2: Simulation run results for SPSA:

Simulation Run	x_1	x_2	x_3	*Total system cost* $(\widehat{F}_0)$	*Percent available (%)* $(\widehat{F}_1)$
1	5	3	32	2305	0.9544
2	5	3	31	2333	0.9604
3	5	2	32	2267	0.9534
4	4	3	32	2274	0.9404
5	5	2	31	2247	0.9359
6	4	3	31	2285	0.9188
7	4	2	32	2226	0.9672
8	4	2	31	2212	0.9394
9	4	2	32	2197	0.9454
10	4.0154	2.0154	31.9846	2214	0.9394
11	3.9846	1.9846	32.0154	2227	0.9528
12	4.0452	2.0126	32.1033	2200	0.9454
13	4.0452	2.0126	32.1033	2228	0.9672
14	4.0226	2.0063	32.0156	2227	0.9672
15	4.0226	2.0063	32.0156	2199	0.9454
16	4.0339	2.0094	32.0774	2228	0.9672
17	4.0339	2.0094	32.0774	2199	0.9454
18	4.0339	2.0094	31.0774	2214	0.9394
19	4.0339	1.0094	32.0774	2175	0.9474
20	3.0339	2.0094	32.0774	2156	0.9546
21	4.0339	1.0094	31.0774	2196	0.9478
22	3.0339	2.0094	31.0774	2136	0.9576

23	3.0339	1.0094	32.0774	2206	0.9528
24	3.0339	1.0094	31.0774	2118	0.9402
25	3.0339	2.0094	31.0774	2184	0.9444

BIOGRAPHICAL SKETCH:

Makbule Rengin Burhanoğlu was born in Bursa, on March 6, 1981.

She completed the high school in Samsun Anatolian High School, in 1999.

She received his B. Sc. degree in Computer Engineering from Galatasaray University, İstanbul, in 2005. Since 2005, she is in the M. Sc. program in Industrial Engineering of Galatasaray University.

For the completion of the program she has studied on "Numerical Comparisons of Black Box Simulation Optimization Methodologies: RSM and SPSA". She wrote her first academic article in 2008 with Yrd. Doç. Dr. M. Ebru Angün.