CentER

Discussion Paper

No. 2006–75

OPTIMIZATION OF SIMULATED INVENTORY SYSTEMS: OPTQUEST AND ALTERNATIVES

By Jack P.C. Kleijnen, Jie Wan

August 2006

ISSN 0924-7815

TILBURG UNIVERSITY

Optimization of simulated inventory systems: OptQuest and alternatives

Jack P.C. Kleijnen 1)

Jie Wan 2) 3)

1) Department of Information Systems and Management, Tilburg University, Postbox 90153, 5000 LE Tilburg, Netherlands
2) Operations Research and Logistics Group, Wageningen University and Research Centre, Hollandseweg 1, 6706 KN Wageningen, Netherlands
3) Department of Industrial Engineering, School of Management, Hebei University of Technology, 300130, Tianjin, China

Abstract

This article illustrates simulation optimization through an (s, S) inventory management system. In this system, the goal function to be minimized is the expected value of specific inventory costs. Moreover, specific constraints must be satisfied for some random simulation responses, namely the service or fill rate, and for some deterministic simulation inputs, namely the constraint $s < S$. Results are reported for three optimization methods, including the popular OptQuest method. The optimality of the resulting solutions is tested through the so-called Karesh-Kuhn-Tucker (KKT) conditions.

Key words: OptQuest software, Response Surface Methodology, Karesh-Kuhn-Tucker conditions, service constrained inventory system

JEL: C0, C1, C9, C15, C44

1 Introduction

In this article, we apply the methodology called *simulation optimization* to the area of *inventory simulation*. Simulation optimization is an important problem in simulation methodology, because optimization is often desired in the design of systems. Inventory simulation is an important application area in management, economics, and industry. Inventory simulation is also an important topic in the simulation literature; see, for example, the best-selling textbook

by Law and Kelton [12]. A well-known building block for inventory Discrete Event Dynamic Systems (DEDS) simulation is the following model.

An *(s, S) model* (with $s < S$) is a model of an inventory management (or control) system in which the inventory (say) I is replenished whenever it decreases to a value smaller than or equal to the *reorder level* s; the *order quantity* Q is such that the inventory is raised to the *order-up-to level* S:

$$Q = \begin{cases} S - I \text{ if } I \leq s \\ 0 \qquad \text{if } I > s. \end{cases} \tag{1}$$

There are several *variations* on this model. For example, review of the inventory (I in Eq. 1) may be either continuous (in real time) or periodic. The lead time of the order may be either a nonnegative constant or a nonnegative random variable. Random demand (say) D that exceeds the inventory at hand ($D > I$) may be either lost or backlogged. Costs may consist of inventory, ordering, and out-of-stock costs. These cost components are specific mathematical functions; for example, inventory carrying (or holding) cost may be a constant per item unit, per time unit. In practice, however, out-of-stock costs are hard to quantify so a *service* (or fill rate) constraint may be specified instead; for example, the expected fraction of total demand satisfied from stock on hand should be at least 90%. We focus on the following variant that is also studied by Bashyam and Fu in their 1998 article [5]; we shall abbreviate Bashyam and Fu to B & F.

B & F try to optimize an (s, S) inventory systems with random demand, random lead times, and a service level constraint. The randomness of these lead times implies that orders may cross in time; i.e., orders are not necessarily received in the order in which they are placed—which complicates the mathematical analysis so simulation is used. Estimating the *optimal* control limits s and S turns out to be very difficult, as the vast literature on inventory management shows.

Programming this type of inventory systems is relatively hard; for a thorough discussion we refer to simulation textbooks; see again [12], pp. 60-61. In this article, we shall use the popular *Arena* software; see [10].

Estimating the *optimal* s and S may use *brute force* methods; i.e., a grid for s and S may be used to search for this optimum. Indeed, several authors have used such a method; see [2], [5], and [16]. Unfortunately, they report different (s, S) values as being optimal! Moreover, brute force cannot be applied to realistic problems; for example, in practice the inventory systems control thousands of Stock Keeping Units (SKUs), so the optimal control levels must be estimated for all these SKUs: the curse of dimensionality. Furthermore,

the inventory system may be only a subsystem of a multi-echelon inventory system (including central warehouses), a production-inventory system (some SKUs are not purchased but are manufactured by the same company), a supply chain, etc. Therefore our results are meant to illustrate how in practice optimization of realistic inventory systems may be done.

To estimate the optimal control levels, we apply OptQuest (provided by OptTek System Inc.); we compare our results with the results in Angün et al. ([2]) and B & F ([5]). The former publication ([2]) uses a modified Response Surface Methodology (RSM); the latter publication ([5]) uses Perturbation Analysis (PA) combined with the Feasible Directions (FD) method (from mathematical programming). Our comparison checks wether the estimated optimal control levels are close to the true optimal values (s^*, S^*) estimated through brute force. Besides the effectiveness, we discuss the efficiency of OptQuest.

Note that input variables are also called control or decision variables or factors. A combination of specific input values is often called a scenario.

Our main results are as follows (details are given in subsequent sections). OptQuest gives the best estimate of the true optimum input (s^*, S^*). B & F's estimate is also close to the true optimum, but not as close as OptQuest's estimate. Angün et al.'s solution is relatively far away from the true optimum. Furthermore, the statistical performance of the KKT test with large sample sizes is better than that with small sample sizes—as is to be expected.

The remainder of this article is organized as follows. Section 2 details the (s, S) inventory simulation with a service constraint, programmed in Arena. Section 3 summarizes the off-the-shelf software product called OptQuest, using classic simulation and statistics terminology (instead of commercial terminology). Section 4 summarizes the KKT conditions for random simulation with multiple responses (multivariate output). Section 5 presents a set of experiments by three research teams (including our own team), including tests for the KKT conditions. Section 6 gives conclusions. A list with 16 references enables the readers to further study the problem addressed in this article.

2 The (s, S) inventory simulation with a service constraint

We assume real-valued Independently and Identically Distributed (IID) demand, integer-valued IID lead time, full backlogging, and continuous review. System performance is measured through steady-state (non-terminating) expected values. Details on this simulation model and its Arena code are given in [16]; also see [5].

To verify the correctness of our code, we perform several *what if* experiments. For example, if we increase s and S respectively, then the costs and the service rate increase too. Furthermore, we use Arena's *trace* facility to check the history of events during a specific simulation run. We also use this trace to check that order crossing indeed occurs in some runs. Moreover, we manually check some computer results (using Arena's 'Output Analyzer').

3 OptQuest

All practical simulation optimization methods are iterative heuristics. OptQuest treats the simulation model as a *black box*; i.e., it observes only the Input/Output (I/O) of the simulation model. In the *(s, S)* simulation model the input consists of the values of s and S; the output consists of the inventory costs excluding out-of-stock costs (the sum of holding and ordering costs is to be minimized) and the service percentage (which should satisfy a minimum value, such as 90%).

Note that PA treats the simulation model as a 'white box'; i.e., it uses the explicit formulas that are inside the simulation model; for example, it uses (1).

OptQuest combines the *metaheuristics* of Tabu Search, Neural Networks, and Scatter Search into a single search heuristic; also see the recent publications [1], [4], [8], and [14]. OptQuest is provided (free of charge) with the student version of the Arena software; see [10]. Unfortunately, the exact heuristic is unknown; i.e., OptQuest is a black box itself.

More precisely, if in OptQuest a candidate solution does not fit the constraints, then that solution is eliminated and OptQuest explores candidates that are more likely to be better. OptQuest allows the simulation analysts to explicitly define integer and linear *constraints* on the deterministic simulation *inputs*. We specify the single constraint $S - s \geq 0$. OptQuest also allows the specification of boundaries on the random simulation *outputs*. We require that the expected service percentage should exceed 0.90; also see below.

OptQuest requires the specification of lower, suggested, and upper values for the variables that are to be optimized. The *suggested* values determine the starting point (combination); this choice affects the efficiency and effectiveness of the search. In practice, the simulation analysts may base their choice on (for example) the current solution if the inventory system has already been used in practice.

Moreover, OptQuest requires the selection of the (random) simulation *output* that is the goal or objective variable to be *minimized,* and the outputs that

should satisfy given *requirements* (conditions). In our problem, we select the average total relevant costs as objective, and disservice γ (complement of service percentage) as requirement; i.e., we select $\gamma = 0.10$ (following B & F in [5]).

Finally, OptQuest enables the users to control the search as follows.

- OptQuest allows different *precision criteria* for both the objective and the constrained simulation outputs:

  (i) The simplest option is to specify a *fixed* number of replicates (or 'replications'), say, m; for example, $m = 10$.

  (ii) A more advanced option selects the number of replicates between fixed lower and upper bounds, stopping the replication if any inferior solution is found. More precisely, OptQuest uses the classic Student statistic to test the null-hypothesis that the current solution is worse than the best solution found so far (this test is also explained in classic simulation textbooks such as [12]).We specify $10 \leq m \leq 100$.
- OptQuest also allows to select a *relative precision*; i.e., OptQuest selects the number of replicates such that the halfwidth of the 95% confidence interval for the average output is within a user-selected percentage of the true mean. We select 5%. (This approach is also explained in [12]; it is used in [15].)
- OptQuest allows different *stopping criteria*; for example, we specify that the search stops either after 300 minutes (five hours) or after 500 'nonimproving solutions'. (We shall present an alternative stopping rule in the next section).

4 Karesh-Kuhn-Tucker (KKT) conditions

An alternative *stopping criterion* uses the KKT conditions (instead of rather arbitrary criteria such as the ones that OptQuest uses; see the last bullet in the preceding section). The KKT conditions use gradients (as we shall see). By definition, the *gradient* (say) $\nabla(w)$ of a function $w(x_1, \ldots, x_k)$ is the vector with the first-order partial derivatives: $\nabla(w) = (\partial w/x_1, \ldots, \partial w/\partial x_k)'$. In a service-constrained $(s,\ S)$ inventory system, the following two gradients are relevant:

$$\nabla(c) = (\frac{\partial c}{\partial s}, \frac{\partial c}{\partial S})' \tag{2}$$

and

$$\nabla(f) = (\frac{\partial f}{\partial s}, \frac{\partial f}{\partial S})' \tag{3}$$

where c denotes the expected value of the total relevant costs, and f denotes the expected disservice rate (remember that we also defined the required—not the expected—disservice rate γ).

PA estimates gradients (from a single simulation run) and RSM also estimates

gradients (from a local simulation experiment, using classic Design Of Experiments, DOE; see below).OptQuest, however, does not estimate gradients. Nevertheless, when the analysts apply OptQuest, they may still perform a local experiment to estimate the gradients at the OptQuest estimated optimum and use these gradients as a stopping criterion!

The true optimum control levels s and S are unknown (as we saw above). The brute force method and the search heuristics applied by different authors give different estimates (as we shall see). Therefore we test the KKT first-order optimality conditions for these different solutions. These conditions were originally derived in deterministic nonlinear mathematical programming (see, for example, [7]):

$$\boldsymbol{\beta}_{0;-0} = \mathbf{B}_{J;-0}\boldsymbol{\lambda} \tag{4}$$

where

$\boldsymbol{\beta}_{0;-0}$ denotes the k-dimensional vector with the (deterministic) gradient of the goal function (in our inventory problem, $\boldsymbol{\beta}_{0;-0} = (\partial c/s, \partial c/\partial S)'$; see equation 2; the subscript -0 will be explained in equation 8);

$\mathbf{B}_{J;-0}$ denotes the $k \times J$ matrix with the gradients $\boldsymbol{\beta}_{h;-0}$ $(h = 1, \ldots J)$ of the J binding constraints (in our problem, $\mathbf{B}_{J;-0} = (\partial f/s, \partial f/\partial S)'$ in equation 3 so this matrix has only one column, which consists of the two components of the gradient of the disservice f—provided the service constraint is binding for a specific solution (s, S));

$\boldsymbol{\lambda}$ denotes the J-dimensional vector with the non-negative Lagrange multipliers for the binding constraints.

In other words, (4) implies that at the optimum combination (s^*, S^*) the gradients are vectors that point in the same direction (but may have different length).

In 2004, [9] tested the KKT conditions for a single unconstrained random simulation output. Only in 2006 , [3] and [6] derived statistical tests for the KKT conditions for random simulation models with constraints on the deterministic inputs and the random outputs—as is the case in our inventory problem. The former publication ([3]) assumes a large number of simulation replicates ($m = 20$ in Section 5), whereas the latter publication ([6]) allows a minimum number of replicates ($m = 3$ in that same section).

Note that simulation models implicitly define nonlinear transformations (functions) of the deterministic (simulation) inputs into the (mean simulation) outputs.

Testing the KKT conditions in random simulation implies testing the following three null-hypotheses, denoted by the superscripts (1) through (3) in the following three equations:

(1) The current *slack* is zero; i.e., the current solution is feasible and at least one constraint is binding. In our problem this implies

$$H_0^{(1)} : E(\widehat{f}) = 0.10 \tag{5}$$

where $\widehat{f}$ is an unbiased estimator of f; this f was defined below (3).

(2) The expected value of the estimated local gradient equals the expected value of the product of the estimated gradients of the simulation outputs in the binding constraints and the Lagrange multipliers; see (4):

$$H_0^{(2)} : E(\widehat{\boldsymbol{\beta}}_{0;-0}) = E(\widehat{\mathbf{B}}_{J;-0}\widehat{\boldsymbol{\lambda}}). \tag{6}$$

(3) The Lagrange multipliers in (6) are non-negative:

$$H_0^{(3)} : E(\widehat{\boldsymbol{\lambda}}) \geq \mathbf{0}. \tag{7}$$

Because these hypotheses require multiple tests, the well-known *Bonferroni* inequality is applied. Moreover, these three hypotheses are tested sequentially (stagewise), as follows.

(1) To test whether the constraints are binding (see equation 5), the classic Student t test may be used. If the estimate $\widehat{f}$ is significantly big, then the combination tested is not feasible. If the estimate $\widehat{f}$ is significantly small, then the optimum has not yet been reached—assuming the optimum does not lie in the interior of the feasible area (otherwise, classic RSM would apply). The mean $E(\widehat{f}) = f$ is estimated at the center of the local area being simulated; the standard deviation $\sigma(\widehat{f})$ is estimated from the m replicates at that center; see the next step.

(2) To estimate the goal gradient (see equation 6), the following second-order polynomial is fitted locally:

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 s + \widehat{\beta}_2 S + \widehat{\beta}_3 sS + \widehat{\beta}_4 s^2 + \widehat{\beta}_5 S^2 \tag{8}$$

so the gradient is estimated through

$$\widehat{\nabla(c)} = (\widehat{\beta}_1 + \widehat{\beta}_3 S + 2\widehat{\beta}_4 s, \widehat{\beta}_2 + \widehat{\beta}_3 s + 2\widehat{\beta}_5 S)' \tag{9}$$

Note that $\widehat{\beta}_0$ in (8) is irrelevant; the gradient of the goal function is therefore denoted by $\widehat{\boldsymbol{\beta}}_{0;-0}$. Another second-order polynomial is fitted locally for the fill rate observations, to get $\widehat{\nabla(f)}$ analogous to (9).

To compute these gradient estimates, we use the Ordinary Least Squares (OLS) criterion. These estimates require that 'enough' simulation observations be obtained locally. A so-called *Central Composite Design* (CCD)

is popular in RSM; in our problem this design is the 2^2 design augmented with the central point and four axial points obtained by increasing and decreasing each input one at a time (so this CCD implies 9 points, which suffices to estimate the 6 parameters in equation 8). This central point is replicated m times. To avoid singular covariance matrices, m must be greater than the number of simulation responses; in our problem $m > 2$. To test whether the resulting polynomial fits well, the classic F lack-of-fit test is used; see classic RSM textbooks such as [13] and [11].

(3) Finally, these estimated gradients are used to test the two hypotheses in (6) and (7). Because these tests are rather complicated, we refer to [3] and [6] for details. We use the computer codes that are based on these two publications; see the Acknowledgment.

5 Experiments

The following assumptions are used in the set of experiments conducted by several research teams; these assumptions were originally selected by B & F (see [5]).

- Demands are exponentially distributed with mean 100.
- Lead times are Poisson distributed with mean 6 (so the probability of order crossing is relatively high).
- The maximum disservice level γ is 0.10.
- The holding cost is 1, the variable ordering cost is 1, and the fixed ordering cost is 36.

5.1 Searching for the optimum: results

We now summarize the results of the brute-force grid search and the three search heuristics; also see Table 1.

Bashyam and Fu ([5]) 's brute-force search initially uses a grid with s and S incremented in steps of size 5. Next, in the neighborhood of the optimum resulting from this coarse grid, they decrease the step size to 1 (instead of 5). They estimate that the true optimum has minimum costs of 703 and disservice rate of 0.11. Unfortunately, they do not report the corresponding optimal input combination, (s^*, S^*). Note that they simulate each grid point (combination) for $30,000$ periods, to reach steady state, and take 10 replicates (each of $30,000$ periods).

Next they estimate the optimum through PA combined with the FD search.

They again use 10 replicates, each of 30,000 periods. Now they estimate that the true optimum has minimum costs of 708 (so these costs are less than 1% higher than the brute force solution) and a disservice rate of 0.11. In private communication, they report the corresponding (s^*, S^*) as (1040, 1065) (also see [6]).

Angün et al. (see [2]) also use brute force to estimate the true optimum. They, however, find better results than B & F, namely the minimum costs are 647 with a standard error of 8.6 and a disservice rate of 0.11 with a standard error of 0.01. They report that the optimal combination is $(s^*, S^*) = (1160, 1212)$.

Next they use their modified RSM to estimate the optimum. They estimate that the minimum costs are 671 and that the disservice rate does not differ significantly from the target of 0.10. They report $(s^*, S^*) = (1185, 1231)$.

Finally, *we ourselves* search for the optimum. First, we use brute force. Initially, we use a grid with stepsize 100, in the search area defined by $0 \leq s \leq 3000$ and $0 \leq S \leq 3000$, and $s \leq S$; i.e., we simulate 325 combinations. We find that the truly optimum combination lies inside the subarea defined by $900 \leq s \leq 1250$ and $1050 \leq S \leq 1250$. Next we decrease the stepsize to 10 (instead of 100), while $s \leq S$; i.e. we simulate 546 combinations. We find that the true optimum lies inside the subarea $1010 \leq s \leq 1030$ and $1070 \leq S \leq 1090$. In that area, we next simulate each integer combination; i.e., we simulate 441 $(= 21 \times 21)$ combinations; we also increase the number of replicates from 10 to 50. Our conclusion is that $(s^*, S^*) = (1020, 1075)$ with minimum costs of 624 (and standard error 2.4) and disservice rate 0.10 (with standard error 0.004). However, many more combinations close to this combination give costs and service rates that do not differ much; see [16].

Next we use OptQuest. Initially, we specify $0 \leq s \leq 3000$ and a suggested value of 1000, and $0 \leq S \leq 3000$ and a suggested value of 1000. Furthermore, we specify 10 replicates. OptQuest finds $(s^*, S^*) = (996, 1116)$, after three hours of computer time on our PC (Intel Pentium 4, 3.2 GHz, 0.99 GB of RAM). Next, we restart OptQuest with the result of the initial search (996, 1116) as the suggested values and the more restricted search area $800 \leq s \leq 1200$ and $800 \leq S \leq 1200$. OptQuest now finds $(s^*, S^*) = (1021, 1077)$, after two and a half hours of computer time. The corresponding costs are 625 (standard error 3.8) and disservice rate 0.10 (standard error 0.005). So our OptQuest results are very close to our brute-force results.

In Table 1, we summarize the results of both the brute-force grid search and the search algorithms reported by the three research teams. Obviously, Angün et al.'s solution differs much from the solutions reported by the other two teams. The symbols A through D are explained in the next subsection.

Team	Method	Symbol	s^*, S^*	costs (SE)	disservice (SE)
Kleijnen & Wan	Brute force	A	1020, 1075	624 (2.4).	0.10 (0.004)
	OptQuest	B	1021, 1077	625 (3.8)	0.10 (0.005)
Bashyam & Fu	Brute force		N/A	703 (N/A)	0.11 (N/A)
	PA & FD	C	1040,1065	708 (N/A)	0.11 (N/A)
Angün et al.	Brute force	D	1160, 1212	647 (8.6)	0.11 (0.010)
	Modified RSM		1185, 1231	671 (N/A)	N/A (N/A)

Table 1
Optima estimated through three different research teams (SE: standard error; N/A: not available)

5.2 KKT tests: results

We test whether the KKT conditions hold at the four points that are denoted by the symbols A through D in Table 1. Moreover, we add a point—denoted by E—that is obviously not optimal, namely $(s^*, S^*) = (985, 1188)$; we found this point E during our OptQuest search for the true optimum.

To estimate the local gradients, we experiment with three *local area sizes* in which we simulate the CCD combinations that determine (s, S) (as we explained in the discussion of equation 8; for details see [16]):

- a 'small' local area of 4×4 units
- an 'intermediate' local area of 10×10
- a 'large' local area of 20×20.

Moreover, we experiment with two *noise levels*:

- relatively small noise resulting from simulation runs of 30,000 periods (as above)
- relatively big noise resulting from simulation runs of only 3,000 periods.

We use a type-I error rate of $\alpha = 0.10$ per test (we cannot control the overall, 'experimentwise' error rate in our sequential procedure; neither can any of the search procedures discussed above). To reduce the effect of the Pseudo-Random Numbers (PRN) when estimating the performance of the KKT test procedure, we use 500 *macro-replicates*; i.e., we repeat our experiments with 500 different non-overlapping PRN streams.

In Table 2, we present results only for the small local area and the small noise level (1 of the $3 \times 2 = 6$ combinations; [16] gives all results). We give the following comments on this table:

Point	Constraint	Lack of fit	KKT: linear	KKT: Lagrange
A	$37/500 = 0.07$	$33/463 = 0.07$	$11/430 = 0.03$	$44/430 = 0.10$
B	$40/500 = 0.08$	$29/460 = 0.06$	$6/431 = 0.01$	$47/431 = 0.11$
C	$33/500 = 0.07$	$35/457 = 0.08$	$21/432 = 0.05$	$48/432 = 0.11$
D	$500/500 = 1.00$	N/A	N/A	N/A
E	$92/500 = 0.18$	$20/408 = 0.05$	$3/388 = 0.01$	$68/388 = 0.18$

Table 2
KKT test of estimated optimal solutions A through D, and suboptimal solution E

- The symbols A through D in the first column refer back to Table 1.
- 'Constraint' means that the slack $(\widehat{f} - \gamma)$ is significantly different from zero. For example, at point A only 37 of the 500 macro-replicates are significant.
- 'Lack of fit' means that the second-order polynomial in (8) is rejected by the F test. For example, point A results in significant lack-of-fit for 33 macro-replicates of the 463 replicates remaining after the preceding slack test (namely, 500 - 37).
- 'KKT: linear' means that the goal gradient is not adequately expressed as a linear function (first-order polynomial) of the constraint gradient; see (6). This fit is tested through the bootstrap test for small number of replicates in [6]. For example, for the A combination 11 of the $463-33 = 430$ polynomials that were accepted after the F test now give significant lack-of-fit.
- 'KKT: Lagrange' means that the linear KKT model in (6) fits well, but at least one Lagrange multiplier is negative; see (7). For example, 44 of the same 430 estimated gradients give significantly negative Lagrange multipliers.
- Points B and C give results that are similar to A.
- Point D gives significant slacks for all 500 macro-replicates.
- Point E results in 18% (= 68/388) of the Lagrange multipliers being negative; i.e., the two estimated gradients point in different directions, so this point is suboptimal.

Results for all six combinations of local area size and noise level are presented in [16]. These results are similar to the results that we present in Table 2. Results for the asymptotic test derived in [3] are presented in [16], using $m = 20$ (instead of $m = 3$) replicates, a first-order polynomial (instead of the second-order polynomial in equation 8), and only 100 (instead of 500) macro-replicates. Now the probability of rejecting the points B through E (not A) is much higher: increasing the number of replicates increases the power of the test!

6 Conclusions

We simulated a well-known inventory management system, in Arena. Next we searched for its optimal control levels, using OptQuest. The resulting optimum was compared with optima estimated by two other research teams. We tested the optimality of all these solutions—and a clearly suboptimal point—through a KKT test procedure. Some solutions passed the KKT test; some did not. The KKT test assuming larger samples performed better, as expected.

The *efficiency* of OptQuest depends on both the size of the search area and the choice of the 'suggested' solution at the start of the search. In our simple problem the search still lasted several hours.

Acknowledgements

The authors acknowledge the financial support of the China Scholarship Council. They also thank Jack van der Vorst (Wageningen University, Netherlands) for his comments and support. And they are grateful to Enrique del Castillo (Penn State University, USA) and Ebru Angün (Galatasaray University, Turkey) for their computer programs for the KKT tests.

References

[1] Adenso-Diaz, B. and M. Laguna (2006), Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54, no. 1, pp. 99-114

[2] Angün, E., D. den Hertog, G. Gürkan, and J.P.C. Kleijnen (2006), Response surface methodology with stochastic constraints for expensive simulation. Working Paper, Tilburg University, Tilburg, Netherlands

[3] Angün, E. and J.P.C. Kleijnen (2006), An asymptotic test of optimality conditions in multiresponse simulation-based optimization. Working Paper, Tilburg University, Tilburg, Netherlands

[4] Bartz-Beielstein, T. (2006), *Experimental research in evolutionary computation; the new experimentalism*. Springer, Berlin

[5] Bashyam, S. and M. C. Fu (1998), Optimization of (s, S) inventory systems with random lead times and a service level constraint. *Management Science*, 44, pp. 243-256

[6] Bettonvil, B., E. del Castillo, and J.P.C. Kleijnen (2006), Statistical testing of optimality conditions in multiresponse simulation-based optimization. Working Paper, Tilburg University, Tilburg, Netherlands

[7] Gill, P. E., W. Murray, and M. H. Wright (2000). *Practical optimization, 12th edition*. Academic Press, London

[8] Hong, L.J. and B.L. Nelson (2006), Discrete optimization via simulation using COMPASS. *Operations Research*, 54, no. 1, pp. 115-129

[9] Karaesman, I and G. van Ryzin (2004), Overbooking with substitutable inventory classes, *Operations Research*, 52, no. 1, pp. 83-104

[10] Kelton, W.D., R.P. Sadowski, and D.T. Sturrock (2004), *Simulation with Arena; third edition*. McGraw-Hill, Boston

[11] Khuri, A. I. and J. A. Cornell (1996), *Response surfaces: design and analysis, second edition*. Marcel Dekker, New York

[12] Law, A.M. and W.D. Kelton (2000), *Simulation modeling and analysis; third edition*. McGraw-Hill, Boston

[13] Myers, R.H. and D.C. Montgomery (2002), *Response surface methodology: process and product optimization using designed experiments; second edition.* Wiley, New York

[14] Rajagopalan, H.K., F.E. Vergara, C. Saydam, and J. Xiao (2006), Developing effective meta-heuristics for a probabilistic location model via experimental design. *European Journal of Operational Research*, in press

[15] Van Beers, W.C.M. and J.P.C. Kleijnen (2006), Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. Working Paper, Tilburg University, Tilburg, Netherlands

[16] Wan, J. and J.P.C. Kleijnen (2006), Simulation for the optimization of (s, S) inventory system with random lead times and a service level constraint by using Arena and OptQuest. Working Paper, Hebei University of Technology