Discrete Event Dynamic Systems: Theory and Applications 2, (1992): 61–88

Ordinal Optimization of DEDS*

Y.C. HO AND R.S. SREENIVAS
Harvard University, Division of Applied Sciences, Pierce Hall, Cambridge, MA 02138

P. VAKILI
Boston University, Boston, MA

Received August 23, 1991; Revised February 26, 1992

Abstract. In this paper we argue that *ordinal* rather than *cardinal* optimization, i.e., concentrating on finding good, better, or best designs rather than on estimating accurately the performance value of these designs, offers a new, efficient, and complementary approach to the performance optimization of systems. Some experimental and analytical evidence is offered to substantiate this claim. The main purpose of the paper is to call attention to a novel and promising approach to system optimization.

Key Words: optimization, parallel computation, queueing theory.

1. Introduction and Rationale

The problem of stochastic optimization of a multivariable function

$$J(\theta) \equiv E[L(x(t;\ \theta,\ \xi)], \tag{1}$$

where θ is the design parameter, L is some performance functional of the sample path of a DEDS $x(t)$, and ξ is all random effects of the problem, has a long history. It is a difficult problem because

1. Analytical or numerical evaluation of $J(\theta)$ or its gradient is only available for a limited class of DEDS and a limited range of performance measures. *This leaves simulation as the only general tool for evaluating the performance and its sensitivity.*
2. *Simulation is costly and time consuming* even with large computer budgets, one frequently faces reasonable size problems that will exhaust the budget.

Two recent developments have held the promise of removing some of the difficulties at this level. On the one hand perturbation analysis and likelihood ratio have offered highly efficient algorithms for performance sensitivity estimation in simulation [Ho and Cao 1991]; on the other hand the advent of parallel and distributed computing systems has provided

*This work is supported by NSF grants CDR-88-03012, DDM-89-14277, ONR contracts N00014-90-J-1093, N00014-89-J-1023, and army contracts DAAL-03-83-K-0171, DAAL-91-G-0194.

considerably more powerful computing environments for simulation experimentation. Let us now turn to the optimization techniques that use simulation-based performance and performance sensitivity estimates as inputs.

Most common approaches to the problem involve iterated methodologies based on gradients, feasible directions, and stochastic approximations (for excellent reviews see Glynn [1986]; Jacobson and Schruben [1989]). The techniques suffer from the following disadvantages:

3. The $(n + 1)$th iterate depends only on the information collected near the nth iterate, thus ignoring information collected earlier at the 1, 2, ..., $(n - 1)$th points of the performance surface.
4. They are sequential algorithms which are difficult to carry out in a distributed fashion due to the problem of synchronization and thus cannot easily take advantage of the availability of parallel and distributed computers.
5. Their performance is sensitive to the initial choice of the design parameters. However, they do not offer guidelines for appropriate selection of these initial points.
6. Since they are driven by local information (performance sensitivity), they do not provide a global view of the performance response surface. For some practical problems this global view may be very informative and, in some cases, essential to understand the basic tradeoffs of different decisions.

In this paper we propose a simple, general, practical, and complementary approach toward solving this problem. Our proposal is simple since it is easy to state and implement; it is general because the solution is applicable to a very large class of problems; practical because it fulfills the engineering requirements for a useful and satisficing[1] solution without any overzealous claims on the degree of optimality of the solution; and, finally, complementary in the sense that the proposed approach does not replace but supplements existing techniques.

Consider a set on N simulations of $L(x(t; \theta, \xi))$ at $\theta = \theta^1, \ldots, \theta^N$ which are randomly chosen. Assuming we are minimizing, we order the sample performance of these N experiments as $\underline{J}_{[1]} < \ldots < \underline{J}_{[N]}$. Note that $\underline{J}_{[i]}$ is only an estimate of $J_{[i]}$, which is $J(\theta)$ at $\theta = \theta^k$ for some k. In fact, let us write

$$\underline{J}_{[i]} = J_{[i]} + w_{[i]}, \tag{2}$$

where w_i is the estimation error of $J_{[i]}$. Now focusing our attention on one of the first (best) r observed $\underline{J}_{[i]}$'s, we ask the question;

> What is the probability that $\underline{J}_{[i]} < \underline{J}_{[j]}$, $j > N - r$ but $J_{[i]} > J_{[j]}$? In other words, to what extent does $\underline{J}_{[i]}$ rank high because of w_i or is the design corresponding to $\underline{J}_{[i]}$ intrinsically good? (Q1)

It is interesting that significant conclusions can be drawn despite the presence of very large estimation noise.[2] Our main claim and observation are that the order of designs are relatively immune to effects of estimation noise and it is possible to select good designs with high probability in the presence of significant estimation noise.[3]

Based on the above observation which will be supported by analysis and experimental results in the paper we advocate a new approach to the stochastic optimization problem (1) which departs from the commonly used approaches in two significant ways:

(a) Our aim is to find good, better, or best designs by attempting to find designs that belong to the α percentile of all designs.
(b) In our solution technique we are more interested in whether a given design is better than another rather than accurately estimating quantities such as gradients and performance values; that is, we primarily deal with *ordinal* rather than *cardinal* optimization.

There is an obvious criticism that can be raised against the objective of the ordinal optimization as stated in (a): a design that is in the top α percentile of all designs ($\alpha = 20$ say) may still be significantly inferior to the true optimum. The discussions below offer responses to such a concern:

1. From a practical point of view of limited computational budget and time, a top α percentile design is in fact what is sought.
2. In those special cases where the true optimum belongs in the top α percentile where α is very small, the final fine tuning to reach the true optimum can be accomplished via adaptive search or other traditional methods. As mentioned earlier, our approach is meant to complement rather than replace existing methods.

In other words, we suggest that for every problem there is always a value of α that corresponds to a satisficing solution to the optimization problem. For certain problems the value of α yields a satisficing solution can be reasonably close to zero, which translates to a formidable computational burden. We hope that for a large number of interesting problems the satisficing notion of α is either not small or there are adaptive techniques that reduce the computational burden while at the same time utilizing the observations made in this paper to aid the adaptation process. Some preliminary experiments suggest that genetic algorithms (Goldberg [1989]) are a viable option in this regard. The primary purpose of this paper is to introduce the notion of ordinal optimization and its immunity to estimation errors. Further refinement and adaptation of the approach will be left to future papers.

The techniques presented in this paper can also be supplemented by other well-known multivariable optimization techniques. In this sense the method proposed here can be considered as a *preprocessing step* that precedes any other optimization technique. Alternately, if the reader is willing to accept the notion of optimality to be a probabalistic statement on the population percentiles, in other words that a satisficing solution is one that is almost certainly better than some predetermined percentile (say 95th percentile) of the population, then the technique presented in this section is capable of providing satisficing solutions efficiently to the multivariable optimization problem.

Typical iterative schemes involve two phases: a *search phase* and an *iteration phase*. Kung [1976] notes that most results in analytic computational complexity assume that good initial approximations are available and deal with the iteration phase only. He observes that if enough time is spent in the initial search phase we can reduce the time needed in the iteration phase. If our approach is used as a *preprocessing* step, we have a simple,

effective, and efficient procedure for dealing with the search phase. Traub [1976] states that the literature contains countless papers giving conditions for the convergence of infinite processes and that a process has to be more than convergent in order to be computationally interesting. He argues that it is essential that we are able to bound the cost of computation. Here again, we observe that, in the absence of any additional information on the performance surface, any robust search for a global optimum is essentially a variant of a random search.

In this sense, each iteration of the ordinal optimization can be viewed as a variant of the *nonadaptive random search* algorithm. These algorithms have been well studied in the context of *cardinal optimization*. Rubinstein [1986] (cf. Section 5.2) considers the problem of estimating the global minimum g^* of a multivariable function $g(x)$ over domain D via the following algorithm:

1. Generate a sequence of N random vectors $X^1, X^2, \ldots, X^N$ uniformly distributed over D.
2. Compute $Y^k = g(X^k)$, $k = 1, 2, \ldots, N$.
3. Estimate g^* by

$$M^N = \min(Y^1, Y^2, \ldots, Y^N).$$

Rubinstein and Weissman [1977] show that under very mild conditions

$$\lim_{N\to\infty} M^N = g^* \quad \text{a.s.}$$

DeHaan [1981] provides expressions for the confidence intervals for g^* under some mild restrictions on the limiting distributions of M^N. It should be emphasized that the primary motivation in this and other variants of random search is *cardinal optimization*. That is, computational effort is primarily spent in obtaining better estimates of the *value* of the function to be globally optimized.

We believe that numerous advantages can be gained with ordinal optimization. Note that once the burden of accurate estimation of performance measures and/or its gradient is lifted, considerable simulation work reduction can be gained: estimation and comparisons can be done based on very short observations or very approximate models. The generality of the approach is also a major plus. The ordinal optimization approach is inherently a global optimization method: while the optimization procedure is in progress, important information about the global shape of the performance surface can be obtained.

Adaptive techniques like genetic algorithms (Goldberg [1989]) can benefit from the fact that the better designs are apparent at the early stages of simulation and any occasional misjudgement on the quality of the solutions due to the influence of simulation noise can be eventually corrected as the process of adaptation proceeds. In other words, the usage of approximate ordering of designs cannot hurt the objective of optimization. In this sense, one can draw a parallel between the approach advocated in this paper and the notion of *confident search* introduced by Rosenbaum [1991].

A recent review of simulation optimization methods (Jacobson and Schruben [1989]) concludes by stating "The advent of high-speed parallel/distributed simulations along with new

ideas in computer graphics, projection, and factor screening should also lead to successful interactive simulation model optimization approaches." Ordinal optimization can be easily parallelized[4] and coupled with the new techniques for parallel/distributed simulation. This, we believe, is an important step toward realizing the prediction offered above.

A generic example considered in Section 2 illustrates the fact that (estimation) noise does not greatly influence the ranks of the designs. Section 3 concerns the theory of order statistics and its use in ordinal optimization. Section 4 is on parallel implementation of the ordinal optimization algorithm. Section 5 introduces three examples that establish the viability of our approach. We conclude with directions for future research in Section 6.

2. Generic Experiments and Worst-Case Analysis

Assume for practical purposes, we plan to explore the performance of a set of N designs corresponding to $J(\theta_i)$ for $i = 1, 2, \ldots, N$. Furthermore, suppose we can bound the performances between $J_{\min}$ and $J_{\max}$. Then in the two-dimensional plot of performance value versus design, we can generally expect to see a jagged performance "curve" with many peaks and valleys. However, if we plot the ordered performance curve $J_{[i]}$ versus i, then the figure becomes far more regular. This comparison is illustrated in Figure 1a, b.

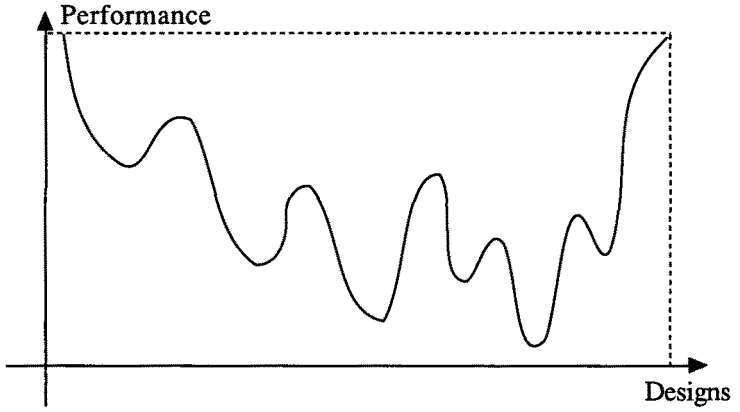
Since we are primarily interested in the behavior of the ordered performance curve near the minimum, we can restrict our investigation of question (Q1) to the three generic cases illustrated in Figure 1(b), namely, steep, linear (or neutral), and flat. In fact, we can use a set of simple generic experiments to study (Q1). For example, let the

- designs be numbered 1 through N (=200).
- performance J be 1 through N (=200); that is, $J_{[i]} = i$ (neutrally ordered performance curves).[5]
- observed value of J be $\underline{J}$.
- estimation noise of $\underline{J} \equiv J + w$, w be $U(0, W)$; i.e., estimation errors are i.i.d. uniformly distributed between 0 and W(=100).

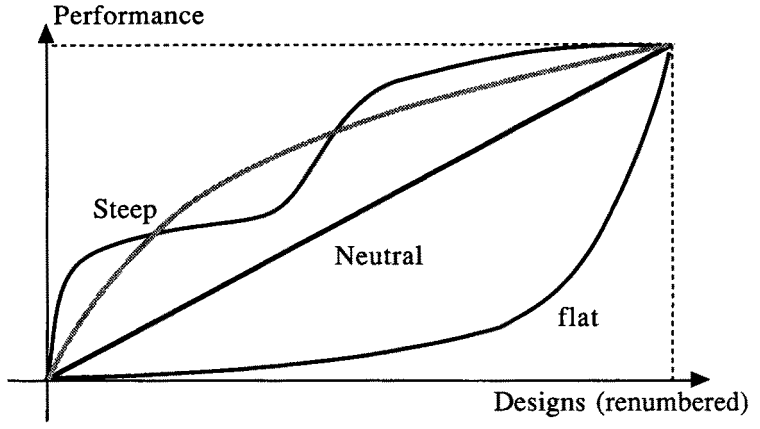
This can be easily implemented in spreadsheet form as shown in Table 1.

We sort the values of column four in ascending order, the observed performance of the N (=200) designs, and examine the distribution of the values in the top r rows of column one. This will give us an indication of how badly noise can distort the true order; and thus, provides some answers to question (Q1). Such a set of generic experiments were carried out for the three types of ordered performance curves discussed above for 10 replications each. The results and conclusions are displayed in Figures 2, 3, and 4.

Further experiments involving varying the parameters N, r, and W do not appear to affect the results and conclusions significantly. These generic experiments are completely general and strongly suggest that performance order is highly immune to a very large additive noise. As a further piece of evidence, let us consider a worst-case analysis under the assumption of i.i.d. noise. Suppose the noise amplitude $W \to \infty$. This is essentially equivalent to the case of picking designs out of N designs blindfolded without the benefit of any observation.



(a)

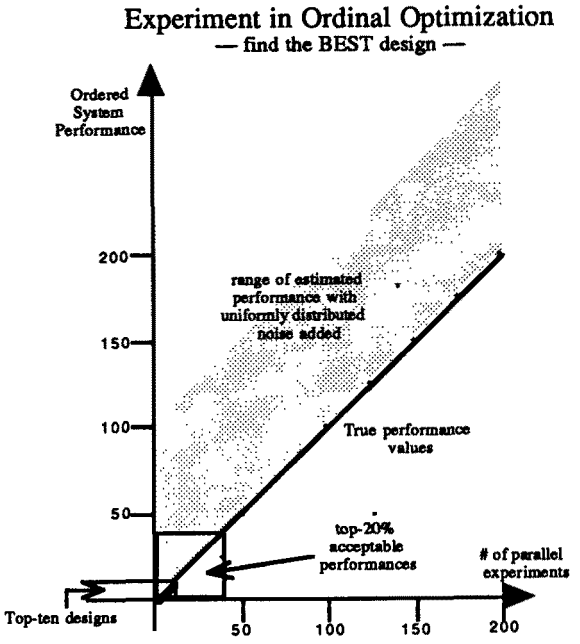


(b)

Figure 1. (a) Unordered performance curve; (b) Ordered performance curve.

Table 1. Generic experiments for ordinal optimization.

Design # = θ	Performance, $J(\theta)$	Noise, $w \in U(0, 100)$	Observed $\underline{J} = J(\theta) + w$
1	1	71.368349423	72.368349423
.	.	.	.
.	.	.	.
.	.	.	.
200	200	11.58390628	211.58390628



- On the average, 4.4 of the top-ten designs remain in top ten despite the additive noise.
- No design worse than the top-20% became identified as the top-ten due to the added noise.

Figure 2. The linear case.

What is the probability that the observed top-r designs actually contain at least k ($\leq r$) of the actual top-r designs?

In this case we have two complementary subsets of a set of size N: the subset of top-r designs (of size r) and the subset of bottom-$(N - r)$ designs (of size $(N - r)$). Question (Q2) asks if we randomly choose r designs what is the probability that k ($\leq r$) of them belong to the subset of size r. The answer to this question, which follows from the probability mass function of a hypergeomtric random variable [Feller 1968, Chapter II], is

Prob[exactly k of the observed top-r design actually belong in top r]

$$= \frac{\binom{r}{k}\binom{N-r}{r-k}}{\binom{N}{r}} = P[k, r, N] \tag{3}$$

Prob[at least k of the observed top-r design actually belong in top r]

$$= \sum_{i=k}^{r} P(i, r, N) \tag{4}$$

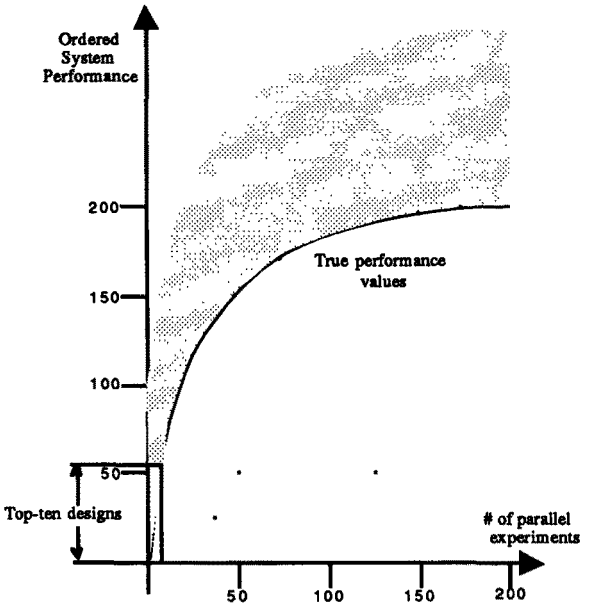


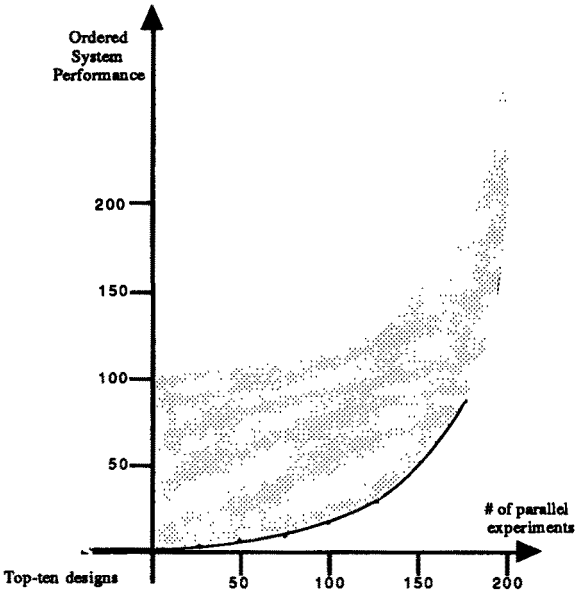
Figure 3. The steep case.

We illustrate (4) for the simple case $k = 1$, $N = 200$ for different values of r in Figure 5.

One interesting observation from Figure 5 is that, if we estimate the performance of 200 designs, no matter how bad the estimation noise, one can be practically sure that one of the top 35 designs will be a true top-35 design. Alternatively, if we blindly pick out 12 designs, then with probability 1/2 there exists a true top-12 design in the set.[6] It is clear from Figure 5 that in practice we can very quickly narrow the scope of the search for the optimum even in the presence of large noise. Various adaptive search tactics based on the observed ordered performance values suggest themselves. A general approach would be as follows:

1. Start with uniformly distributed designs in the design space.
2. As simulation proceeds and the good designs begin to reveal themselves, discard old designs and create new designs according to some "survival of the fittest" rule. In other words change the distribution for design selection from uniform to one more concentrated about successful designs. This idea has strong parallels in the theory of genetic algorithms [Goldberg 1989].

Returning to the generic spreadsheet experiments of Table 1, we can briefly consider the case of correlated noise.[7] In the extreme case of positive correlation, that is, when $w_i = w_j$ for i, j. It is clear that the observed order of performance will coincide with the



- On the average, 1 of the top-ten designs remain in the top-ten.
- However, average actual score of the observed top-ten designs = 5.3 which is within top 3% of the performance range.

Figure 4. The flat case.

actual order. This also implies that constant bias in the estimation error does not matter. Similarly, in the case of extreme negative correlation between adjacent performances, the effect at worse is to remove half of the performance samples from consideration. The remaining half will be positively correlated. From this crude analysis, we can conclude that correlated noise cannot significantly affect our conclusions here. Experimental evidence obtained on the spreadsheet experiments with correlated noise and investigation into the correlation noise model for discrete event simulation confirm the statements made above. They will be reported separately in a future paper [Ho, Deng, and Hu 1992].

3. Order Statistics and Ordinal Optimization

For the present consider the situation when the estimation noise described earlier is absent. That is, we can estimate the *exact* performance for each design. A natural follow-up is the computation of the value of N that guarantees with high confidence that the best design is in the top α percentile of the design population. This question falls into the purview of *order statistics*. In the remainder we review some results from the aforementioned theory.

In order statistics we are concerned with N independent variates $J_1, J_2, \ldots, J_N$ each with the cumulative distribution (cdf) $P(x)$. We order the samples $J_1, J_2, \ldots, J_N$ to obtain *order statistics* $J_{[1]} < J_{[2]} < \ldots < J_{[N]}$. for a detailed treatment of this subject we refer the reader to David's book [David 1982].

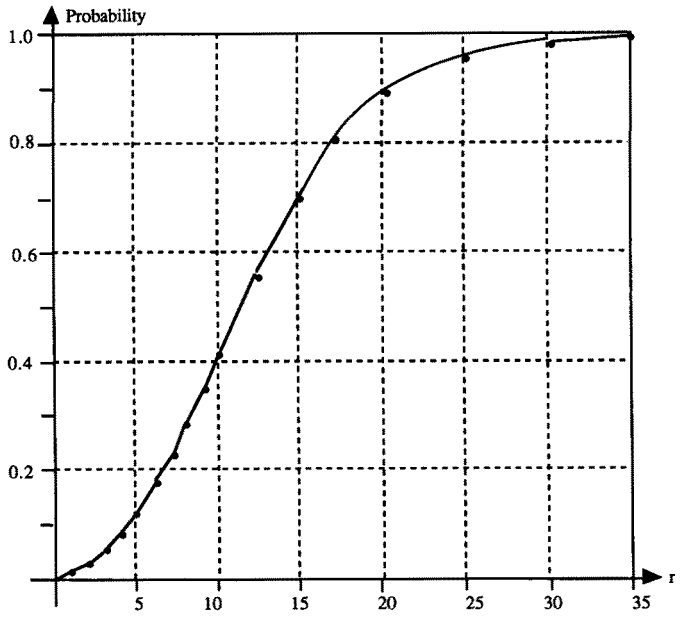


Figure 5. Probability that at least one of the top-r designs is contained in the observed top-r designs with infinite noise.

It is possible to derive *distribution-free* confidence intervals for population percentiles using the aforementioned theory. Suppose J is a continuous variate with a strictly increasing cdf $P(x)$; then the equation $P(x) = p$ where $0 < p < 1$, has a unique solution. Let the solution be ξ_p, which is the *population percentile of order p.* For example, $\xi_{0.5}$ is the *median* of the distribution. We are interested in computing the value of N that guarantees

$$\text{Prob}\{J_{[N]} > \xi_p\} > q.$$

But

$$\text{Prob}\{J_{[N]} > \xi_p\} = 1 - \text{Prob}\{J_{[N]} < \xi_p\} = 1 - p^n.$$

This results in the following inequality

$$1 - p^N > q \Rightarrow N \geq \left\lceil \frac{\log(1 - q)}{\log(p)} \right\rceil. \tag{5}$$

For example, for $q = 0.95$ and $p = 0.9$, the value of N obtained from the above expression is 29. That is, if we took 29 independent samples from any population, we can be 95% confident that 90% of the population would be below the largest order statistic $J_{[29]}$. Another quantity frequently encountered in the study of order statistics is the expected value $E[J_{[r]}]$ of $J_{[r]}$. David [1982] shows that for the transform $u_i \equiv P(J_{[i]})$, $P(J_{[1]}) < P(J_{[2]}) < \ldots < P(J_{[N]})$, and the variates $u_i \equiv P(J_{[i]})$ are uniformly distributed in (0, 1) and

$$E[P(J_{[r]})] = \frac{r}{N+1}. \tag{6}$$

That is, the above result implies that the order statistics divide the area under the curve $y = p(x)$ into $N + 1$ parts, each with expected area of $1/(N + 1)$. That is, the expected values of an order statistic can be approximated by the appropriate population percentile, especially for large values of N. Loosely, this implies that a representative of any arbitrary population percentile can be obtained by a suitable choice of the number of independent samples. In a sense, the expressions derived above can be interpreted as a characterization of the initial behavior of the nonadaptive random search algorithm defined in the introduction, where the initial behavior is stated in terms of the probability of M^N, the estimate of the optimum, exceeding different population percentiles as a function of N.

The connection between order statistics and the problem of stochastic optimization in this paper can be viewed as follows. If we randomly choose (sample) a set of designs in the design space according to some distribution, then the resulting set of performances of the system can be thought of as the sample values of the $J_1, J_2, \ldots, J_N$ in the above discussion with one important proviso. Since the values of $J(\theta)$ can only be estimated via simulation, we really cannot observe the sample values $J_1, J_2, \ldots, J_n$. Instead, what we observe are noisy versions of $J_1, J_2, \ldots, J_n$, i.e., $\underline{J}_i \equiv J_i + w_i$, $i = 1, 2, \ldots, N$. The question now is to what extent must we modify the results of (5)–(6). In our view we need a theory of noisy order statistics. We suggest that (4) of Section 3 is a first step in this direction where we assume that w_i is uniformly distributed in $[0, W)$ and $W \to \infty$.

4. Ordinal Optimization and Massively Parallel Implementation

If we recognize at the outset that the ultimate purpose of simulation is optimization or improvement of system designs, then it is clear that repeated experiments must be carried out. These experiments may be carried out sequentially or in parallel. The question we pose in this section is which of these two approaches (i.e., sequential versus parallel) can best take advantage of the new parallel and distributed computing systems.

Traditionally, the hill-climbing type of successive approximation approach to the optimum is used, where iterative or sequential evaluations of the performance and/or performance gradient is utilized. This approach appears logical when memory is limited and a centralized CPU is used for calculation. With the advent of parallel and distributed computers, attempts have been made to distribute such sequential computation schemes to gain a speed advantage. The problem here is that of *synchronization.* Simulation algorithms of single trajectories of DEDS—which is required for performance and/or performance gradient estimation—are inherently difficult to parallelize. One must take great care so that one part of the distributed computation does not get too far ahead of another [Fujimoto 1990; Righter and Walrand 1989]. The resultant overhead often negates much of the speed advantage. In short, two problems stand in the way of parallel implementation of traditional approaches:

1. The optimization scheme is inherently sequential.
2. The simulation of a single trajectory of a DEDS is hard to parallelize.

We submit that by parallelizing the experiments instead of the sequential algorithms we eliminate a major component of the synchronization problem and can take maximal advantage of the modern parallel computing systems. Ordinal optimization requires a set of parametrically different-structurally similar (PDSS) experiments which corresponds to the evaluation of different designs. The ordinal optimization scheme is inherently a parallel scheme since one does not need to know the result of one experiment in order to perform another. A recently developed "standard clock" (SC) scheme for parallel simulation [Vakili 1991, 1992; Ho, Cassandras, and Makhlouf 1991] seems made to order parallelizing such PDSS experiments. In particular this technique can be executed on parallel and massively parallel computers very efficiently (see Section 5 where some of the experiments are performed on the massively parallel SIMD connection machine). In short, the combination of an ordinal optimization scheme and the standard clock scheme makes a powerful tool that can be executed on massively parallel computing systems to gain orders of magnitude of speedup.

For completeness, we briefly describe below the basic idea of the standard clock technique via simple queueing examples and refer the reader for more details to Vakili [1991, 1992], Ho, Cassandras, and Makhlouf [1991], and Ho and Cao [1991]. Consider several variants of a single server queue (corresponding to the PDSS systems); assume that the only difference between the variants is their service discipline. Note first that to simulate the trajectories of all the variants simultaneously it is sufficient to consider only one arrival process (the arrival process common to all the variants); we assume further that the departure processes at different variants can be subordinated to a single Poisson process. Then one single clock mechanism can be constructed that simulates the common arrival process and the dominating Poisson process. The key element of this construction is that now the clock mechanism is completely decoupled from the individual variants and can be simulated with no information about the state of the variants. At each tick of this common clock the type and time of the event—in this case an arrival or a potential departure—is reported to each variant; each variant then responds to the clock signal according to its operating policy. Therefore all trajectories are synchronized by the single clock such that the same event takes place at the same time at all variants. This feature is the basis of massively parallel execution of the simulation algorithms. There is considerable work in the literature on stochastic coupling of trajectories of parametric families of stochastic processes. Our parallel simulation can be viewed as the simulation of the coupled trajectories. In the particular case when all event lifetimes (e.g., service times and interarrival times) are exponentially distributed, it is based on the well-known uniformization procedure. The exponential assumption is not essential, the approach can be extended to phase-type and bounded hazard rate distributions, and for those events that remain active all through the simulation (for example, the common arrival process to all the variants above) no distributional assumption is necessary. For more details we refer the reader to Vakili [1991].

More specifically, consider a simple queueing network that consists of one arrival stream at rate λ, and two tandem servers which can serve at rates μ_1 and μ_2 and have finite buffers preceding them. Assume that the difference between designs is the number of buffers at the servers. The standard clock determines the time of occurrences and type of events. Let τ_n denote the time instance of the nth tick of the standard clock and ϵ_n the type of the nth occurred event; i.e., the event occurred at τ_n. The standard clock ticks at rate

$\Lambda = \lambda + \mu_1 + \mu_2$; i.e., $(\tau_{n+1} - \tau_n)$ is exponentially distributed with rate Λ. The type of an event on the standard clock is determined separately according to the ratio of all the event rates; that is $P(\epsilon_n = \text{arrival}) = \lambda/\Lambda$, $P(\epsilon_n = \text{departure from server } i) = \mu_i/\Lambda$. Once the time and type of the nth event are determined by the standard clock this information is sent to all designs. Each design determines if the occurred event is feasible; if so, it updates the state of the system accordingly; otherwise it simply ignores the event, and the state of the system remains unchanged. For details see Section 7.1 of Ho and Cao [1991].

5. Examples of Applications

In this section we present experimental investigations into the applicability of our approach. The following scheme is utilized in all the experiments of this section.

- The design space Ω is specified or inferred from the problem description.
- N designs are uniformly chosen from the design space.
- These designs are simulated for a "short" duration and ranked based on sample performances.

The experiments depart from the conventional approach in two important ways. The comparison is done based on (i) transient sample performances and (ii) only one sample path of the system.

We call the reader's attention to the following unifying theme in all the experiments in this section: the ranking "stabilizes" very "early" in the simulation, and therefore can be effectively used to "solve" the ordinal optimization problem.

Subsection 5.1 concerns a two-station queueing network with feedback previously reported in Lirov and Melamed [1990]. Section 5.2 concerns massively parallel implementation of the ordinal optimization algorithm. As expected, we observe a significant speedup in this context. Finally, we concern ourselves with an example of a large closed queue network that consists of 60 stations.

5.1. A Two-Node Jackson Queueing Network

We first consider a two-node Jackson network reported in Lirov and Melamed [1990]. The structure of this network is shown in Figure 6. Customers arrive at a rate $\lambda = 0.5$. Upon completion of service at station 1, customers loop back for another service with probability 0.5 or continue to station 2. The objective here is to minimize the mean sojourn time subject to the constraint that the sum of the service rates at station 1 and station 2 equals 4.

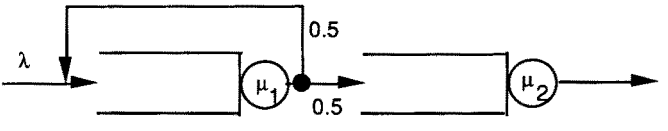


Figure 6. A two-node Jackson queueing network.

The mean sojourn time as a function of μ_1 and μ_2 is given by the expression [Lirov and Melamed 1990]

$$J(\mu_1) = \frac{2}{\mu_1 - 1} + \frac{1}{3.5 - \mu_1}.$$

It can be shown that the mean sojourn time is the smallest for $\mu_1^* = 2.46$, where $J(\mu_1^*) = 2.34$.

The design space for this example is the line segment defined by $\mu_1 + \mu_2 = 4$, where $\mu_1, \mu_2 > 0$. Since $\lambda = 0.5$ it follows from stability consideration that μ_2 should be at least 0.5. Looking at the first queue, for stability we have the inequality

$$\mu_1 > \lambda + 0.5\mu_1 \Rightarrow \mu_1 > 2\lambda = 1.$$

Therefore we restrict the design space to the subspace defined by $\mu_1 + \mu_2 = 4$, and $\mu_1 > 1$, $\mu_2 > 0.5$. Therefore the design space Ω is the interval $\mu_1 \in (1.0, 3.5)$.

We choose an upper bound, M, on the number of customers that leave the system ($M = 100$). At every tenth fraction of M (i.e., at 10, 20, ..., 100 customers) we compute a sample-path estimate of the mean sojourn time and rank the N designs. The network is initialized with empty queues.

The ranks of the N designs can be viewed as an N-dimensional, vector-valued, discrete, random function of the number of customers served. In principle we can obtain a sample realization of this discrete, random function by computing the ranks every time a customer leaves the network. For practical reasons, we choose to sample the above mentioned realization of the vector-valued function at uniform intervals.

It is experimentally observed that there is a "continuity" in the ranks from one fraction to another. For example, if we compute the set of the top 35 ranks after serving 20 customers and the similar set after serving 30 customers, then the intersection of these sets is typically not empty. We can visualize the rank dynamics by using a rank dynamics plot. This is a plot of the relative ranking of the designs at various sampling points. If a particular design is present in adjacent sampling points then a line is drawn from the previous position to the current position in the relative ranking.

Since in this example we are interested in minimizing the mean sojourn time, the best design is given a rank of unity. Figure 7 shows the rank dynamics plot for the case when $N = 100$ for the top 10 designs. It can be seen that the design that was ranked fourth at the 10 customer point is ranked in the top 10 at all the sample points. The final rank of this design after serving 100 customers is nine. The path taken by this design is highlighted in Figure 7. However, the design that is in the sixth position at the 50 customer point seems to have stabilized at the first position. Figure 8 shows the top 30 ranks of the same experiment where the path taken by the best design is highlighted. We observe that the design that is ranked sixth at the 50 customer point is actually ranked 11th at the 10 customer point and is in the top 30 ranks at all sample points. It is natural to suggest that this design is the best design from the sample of 100 chosen for this experiment. From the closed-form expression for the mean sojourn time we observe that this design is offset by 6.64% from the minimal mean sojourn time.[8] Table 2 represents the percentage offset of each of the

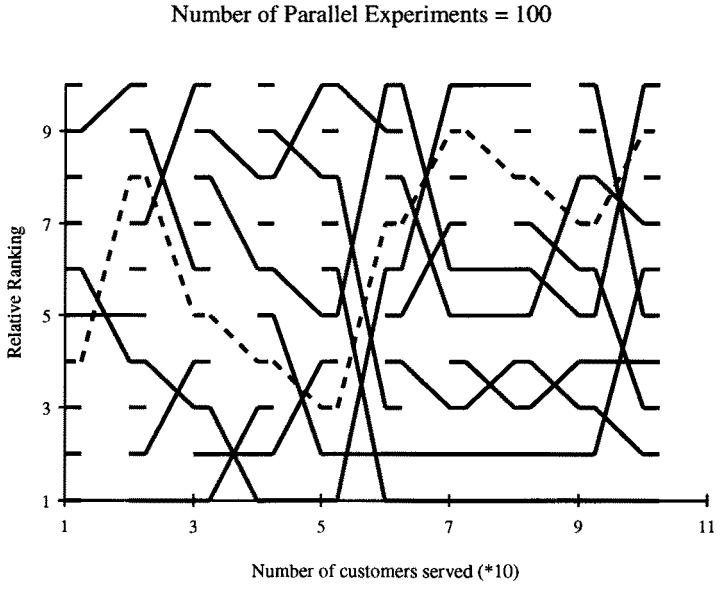


Figure 7. The rank dynamics plot for the top 10 ranks for 100 designs.

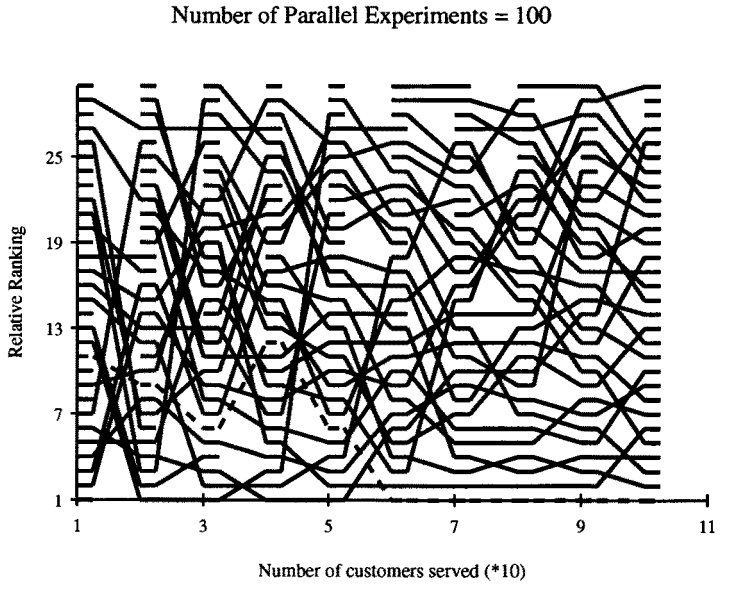


Figure 8. The rank dynamics plot for the top 30 ranks for 100 designs.

Table 2. Percentage offset of the top 10 ranks with respect to the true minimum mean sojourn time for $N = 100$.

	Customers									
Ranks	10	20	30	40	50	60	70	80	90	100
1	267.186	17.950	17.950	9.926	9.926	6.645	6.645	6.645	6.645	6.645
2	17.267	42.221	15.510	15.510	0.171	0.171	0.171	0.171	0.171	17.341
3	6.918	82.210	9.925	17.950	4.441	0.914	17.341	6.600	17.341	6.243
4	4.441	9.930	42.220	4.441	15.510	17.341	6.591	17.341	6.591	6.591
5	15.349	15.349	4.441	0.171	1.151	31.779	26.297	26.297	1.151	0.076
6	9.926	57.403	6.645	1.151	6.645	9.926	1.151	1.151	6.243	0.171
7	0.076	12.856	6.918	46.720	0.309	4.441	31.779	6.243	4.441	26.297
8	39.641	4.441	1.152	17.267	0.914	26.297	14.996	4.441	26.297	14.751
9	113.806	6.645	17.267	0.914	14.751	17.267	4.441	6.918	31.779	4.441
10	121.099	113.806	12.856	15.349	17.267	1.151	9.926	9.926	0.076	1.152

top 10 ranks at the various sample points with respect to the true minimum mean sojourn time. The rank dynamics plot of Figure 7 can be constructed from this table.

Figures 9 and 10 show the rank dynamics plot for the top 10 and top 30 designs for the case when $N = 200$ where the path of the best design is highlighted. From Figure 9 we note that the design that is ranked sixth at the 30 customer point stabilizes at the best design by the 50 customer point. From Figure 10 we observe that this design was originally ranked at the 17th position at the 10 customer point. As before, we also note that this design is offset by 1.95% from the optimal mean sojourn time. Table 3 shows the percentage offset

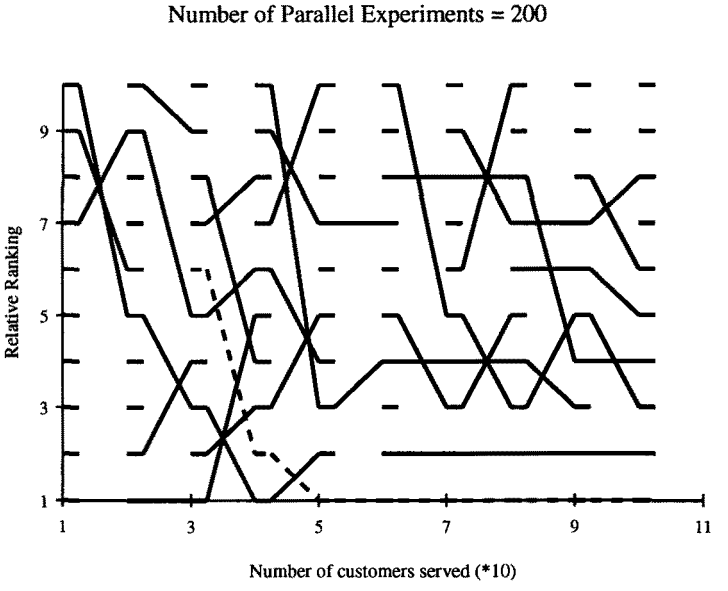


Figure 9. The rank dynamics plot for the top 10 ranks for 200 designs.

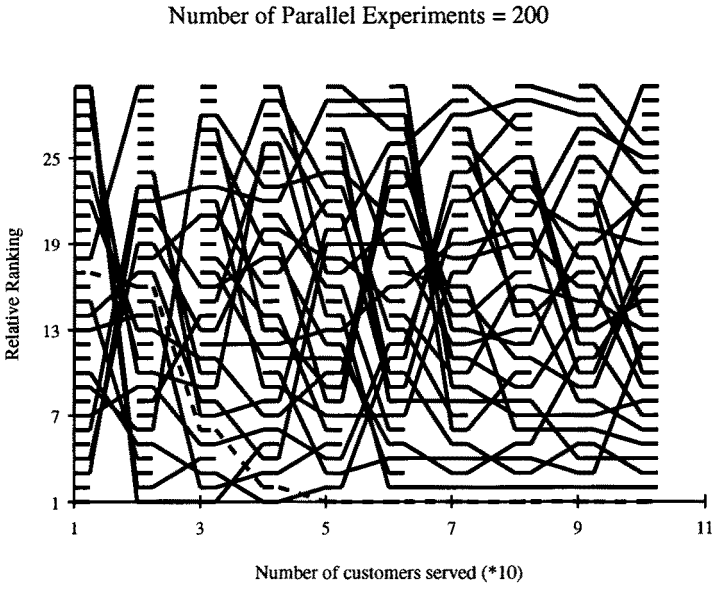


Figure 10. The rank dynamics plot for the top 30 ranks for 200 designs.

Table 3. Percentage offset of the top 10 ranks with respect to the true minimum mean sojourn time for $N = 200$.

	Customers									
Ranks	10	20	30	40	50	60	70	80	90	100
1	267.187	17.949	9.926	1.948	1.948	1.948	1.948	1.948	1.948	1.948
2	47.467	42.221	15.509	1.948	9.926	6.645	6.645	6.645	6.645	6.645
3	0.059	82.209	9.926	15.509	0.171	0.129	3.631	1.489	0.171	1.489
4	17.267	2.877	42.221	0.106	4.441	0.171	0.171	0.171	17.341	17.341
5	30.636	9.926	4.441	17.950	15.510	3.631	1.490	3.631	1.489	1.181
6	6.918	15.349	1.948	4.441	47.467	0.914	0.272	1.181	1.181	4.569
7	4.441	57.403	0.221	0.0002	21.202	21.202	30.636	6.591	6.591	6.243
8	3.916	12.856	0.106	0.221	25.140	17.341	17.341	17.341	4.569	6.591
9	15.349	4.441	6.645	21.202	2.745	31.779	6.591	25.140	0.0002	0.106
10	9.926	6.645	6.918	0.171	0.0002	1.489	26.297	0.272	1.151	0.076

of the top 10 ranks with respect to the true minimum mean sojourn time for the case when $N = 200$. As before, the rank dynamics plot of Figure 9 can be inferred from this table.

For the case when $N = 200$, we know from Figure 5 that there is a high probability that one of the top 35 designs obtained from the ranks computed after serving a very large (theoretically infinite) number of customers should be in the top 35 designs obtained from the ranks computed after serving any fraction of this large number of customers. Various heuristics on using the "rank dynamics" to obtain an estimate of the best design suggest themselves.

For example, we suggest that if there is a nonempty set of designs present in the top 35 at all sample points, then the best design among this class of designs has a high probability of surviving in the top 35 for longer simulation runs (i.e., serving more customers). It is quite possible that under a different sampling scheme there is no single design that is present in the top 35 at *all* the sample points, in this case alternate heuristics suggest themselves. For the case when $N = 100$, the designs that are ranked in the 2nd, 4th, 5th, 6th, and 11th positions are present in the top 30 positions at all the sample points. The best design at the 100 customer point is the one that was ranked in the 11th position at the 10 customer point. This design is offset by 6.64% from the minimum mean sojourn time. For the case when $N = 200$, the designs that are ranked in the 7th, 14th, 17th, 22nd, and 28th positions are present in the top 30 positions at all the sample points. The best design at the 100 customer point is the one that is ranked 17th at the 10 customer point, this design is offset by 1.94% from the minimum mean sojourn time.

Another heuristic that is worth considering is the *maximum/minimum average rank* heuristic. In this heuristic the ranks of the each design at various sampling points is cumulated and then divided by the number of sampling points. Depending on the objective, the design with the minimum or maximum average rank is identified as a solution. The sampling instants can be suitably skewed depending on the availability of a priori information. Using this heuristic on the in the previous example yields the same conclusion as above. That is, for the case when $N = 100$ we obtain a design that is offset by 6.64% from the minimum mean sojourn time and the case when $N = 200$ we obtain a design that is offset by 1.94% from the minimum mean sojourn time. The design of these and other heuristics and their statistical analysis are left as a future research direction.

Given the fact that we have a purportedly good design we can proceed in two directions. If we are willing to accept a design that is in the top $35/200 \sim 17.5$ percentile of the population as a satisficing solution to the optimization problem, then we can stop the computation with the suggestion that the design at hand is the satisficing solution. If the above-mentioned justification is unacceptable we can use the design at hand as an initial condition for other known optimization techniques and improve on the optimality of the solution.

5.2 Examples using Massively Parallel Computers[9]

Experiments reported in this section are performed on the massively parallel connection machine (CM). Connection machine is a single-instruction, multiple-data (SIMD) computer that has 64K (in our platform) processors each with its associated memory. A serial computer called the front end acts as the central control mechanism that directs all the processors as to what instruction to perform next. The standard clock mechanism is implemented at the front-end computer and each processor simulates one of the designs. (It is possible to request access to 8K, 16K, 32K, or 64K of the processors of the CM.)

A Buffer allocation problem for a transfer line. We consider a well known buffer allocation problem for a transfer line of reliable machines depicted in Figure 11.

There are 11 reliable machines (machine i is denoted by M_i) and 10 buffers (we use B_i to denote the ith buffer). It is assumed that there is an infinite number of unfinished parts

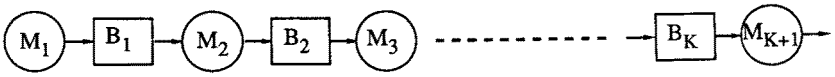


Figure 11. A finite buffer tandem queueing network.

available at M_1 and infinite space for finished parts after M_{11}. The service times at M_i are assumed to be i.i.d. Erlang(μ_i, r_i). We consider the following blocking scheme: M_i begins processing a part only if the immediate down stream buffer; i.e., B_i, is not full. The buffer allocation problem is defined as follows: there are $C = 20$ buffers to be allocated among the 10 buffer locations; at least one buffer needs to be allocated to each location. We seek the buffer allocation that maximizes the steady state throughput in the system.

We randomly choose $N(=8000)$ designs from the design space Ω, the space of all admissible buffer allocations. Since at least one buffer needs to be allocated to each location, there remain 10 buffers to be allocated to 10 locations; hence $|\Omega|$, the size of the design space, is

$$|\Omega| = \binom{19}{9}.$$

The designs are identified by numbers from 1 to 8000. Note that this numbering has nothing to do with the design's performance, but only with when the design was generated. For example, design 2 does not suggest that performance of this design is ranked second among all designs. The standard clock approach is used to simulate all designs in parallel. The designs are ranked based on their cumulative production (denoted by NPP, number of parts produced) at different time instances $t = T_1, T_2, T_3, \ldots$. The ranking is in descending order of NPPs.

The following parameters were used in this experiment:

$$\mu_i = 0.2, \qquad (r_1, \ldots, r_{11}) = (3, 3, 6, 5, 5, 3, 3, 4, 3, 6, 3).$$

The time instances T_1, T_2, and T_3 are defined as follows: $T_1 = \tau_{5000}$, $T_2 = \tau_{10000}$, $T_3 = \tau_{15000}$. (τ_{5000} is the time instant of the 5000th click of that standard clock).

The whole experiment, including random generation of 8000 designs took about 20 s. The CM was busy for 8 s during this time.

Table 4 shows the top 10 designs and the number of parts produced at this designs at T_i (denoted by NPP). If two designs produced the same number of parts, the one with the lower design number was ranked higher.)

Note that only one of the best 10 designs at T_1 survives at T_2 and T_3, while 7 of the best 10 designs at T_2 survive at T_3. The same experiment was repeated with the same 8000 designs, however in this case the designs were ranked based on their NPPs at $T_2 = \tau_{10000}$ and $T_4 = \tau_{50000}$. The results are shown in Table 5. In this case 3 designs that were ranked among the top 10 at T_2 were also ranked among the top 10 at T_4. If the ranking at T_4 is taken as the true ranking of the designs then this experiment strongly suggests that the problem of choosing the optimal buffer allocation among the 8000 designs is essentially

Table 4. Top 10 designs at T_1, T_2, and T_3.[10]

	T_1		T_2		T_3	
	Design	NPP	Design	NPP	Design	NPP
1	603	65	1027	141	1027	211
2	837	65	4951	141	4951	211
3	1027	65	5532	141	5532	211
4	1271	65	5183	140	1517	210
5	2173	65	273	139	2335	210
6	2413	65	1571	139	3280	210
7	2698	65	2335	139	4998	210
8	2783	65	2940	139	5183	210
9	3010	65	3280	139	6349	210
10	3041	65	4998	139	64	209

Table 5. Top 10 designs at T_1 T_4.

	T_1		T_4	
	Design	NPP	Design	NPP
1	1027	141	7011	7207
2	4951	141	7382	7207
3	5532	141	1746	7205
4	5183	140	6548	7204
5	273	139	3971	7203
6	1571	139	3280	7189
7	2335	139	1027	7188
8	2940	139	4951	7188
9	3280	139	193	7185
10	4998	139	1566	7185

"solved" at $T_2 = \tau_{10000}$, i.e., after producing only 140 pieces from the line, a 50-fold reduction in simulation time. One of the "solutions," for example design number 1027 corresponds to the buffer allocation (1, 2, 3, 2, 2, 1, 2, 2, 3, 2).

When replications of the above experiments were performed with the same designs, the same phenomenon was observed; i.e., the above designs were consistently ranked high.

A Cyclic Server Problem. We now consider the problem of finding the optimal service policy for a cyclic server from a specified set of policies. Consider a single cyclic server serving 10 buffers in a round-robin fashion (Figure 12).

There are 10 streams of arrivals, forming Poisson processes with rates $\lambda_1, \ldots, \lambda_{10}$, respectively. The server serves m_i customers at buffer i or until i is emptied, whichever comes first. There is a changeover time of length δ_i for going between buffer i and buffer $i + 1$. It is assumed that for each unit of delay at buffer i, C_i units of cost is incurred by the system. The objective is to find a service policy that minimizes the average cost per unit time in the system. We impose a limit on the admissible policies and assume that $0 \leq m_i \leq 10$ for all i; that is, no more than 10 customers will be served at any buffer. Therefore the design space

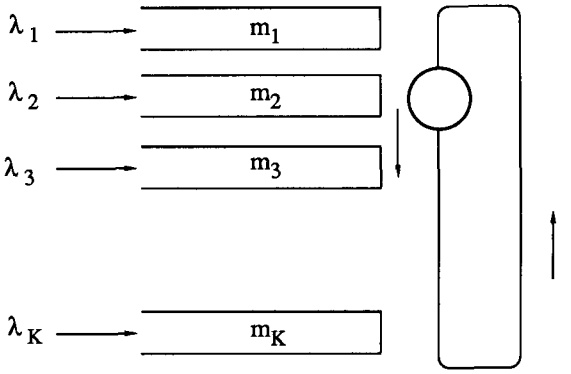


Figure 12. A cyclic server serving K stream stream of arrivals.

$$\Omega = \{\mathbf{m} = (m_1, \ldots, m_{10}); \quad 0 \leq m_i \leq 10\}.$$

N (=4000) random designs are simulated in parallel, and the results are shown below; Table 6 shows the top 10 designs, and the average cost per unit time incurred for each of them at $T_1 = \tau_{5000}$, $T_2 = \tau_{10000}$, and $T_3 = \tau_{15000}$.

Only one of the best 10 designs at T_1 (design 3570) survives at T_2 and T_3, while 5 of the best 10 designs at T_2 survive at T_3. The same experiment was repeated with the same 4000 designs. However in this case the designs were ranked based on their average cost per unit time estimates at $T_2 = \tau_{10000}$ and $T_4 = \tau_{50000}$. The results are shown in Table 7. In this case one design that was ranked among the top 10 at T_2 was also ranked among the top 10 at T_4.

It is illustrative to also look at the bottom 10 designs. It turns out that for these designs the system becomes unstable and the cost increases indefinitely with time. However, the order between designs show similar behavior as those of the top 10 designs. These results are shown in Tables 8 and 9. Three of the worst 10 designs at T_1 survive at T_2 and T_3, while 7 of the worst 10 designs at T_2 survive at T_3, and 6 of the worst 10 designs survive at T_4.

Table 6. Top 10 designs at T_1, T_2, and T_3.

	T_1		T_2		T_3	
	Design	Avg C	Design	Avg C	Design	Avg C
1	3889	2149.4	2887	2170.0	20	2211.1
2	1328	2172.9	3570	2177.6	2081	2218.5
3	3570	2173.3	2245	2182.2	1558	2231.5
4	1245	2177.1	2914	2182.2	2592	2237.9
5	2887	2177.7	3547	2182.2	3570	2240.6
6	1058	2181.8	1305	2185.1	3311	2241.3
7	1816	2181.8	2592	2190.1	2245	2242.4
8	1869	2181.8	9	2190.2	2914	2242.4
9	1936	2181.8	2627	2190.7	3547	2242.4
10	1479	2185.5	1778	2191.1	910	2251.0

Table 7. Top 10 designs at T_2 T_4.

	T_2		T_4	
	Design	Avg C	Design	Avg C
1	2887	2170.0	9	2194.9
2	3570	2177.6	1245	2198.3
3	2245	2182.2	1864	2211.2
4	2914	2182.2	2397	2214.5
5	3547	2182.2	1869	2216.7
6	1305	2185.1	544	2216.9
7	2592	2190.1	1558	2218.0
8	9	2190.2	394	2218.4
9	2627	2190.7	3889	2218.5
10	1778	2191.1	1058	2218.5

Table 8. Bottom 10 designs at T_1, T_2, and T_3.

	T_1		T_2		T_3	
	Design	Avg C	Design	Avg C	Design	Avg C
1	1257	31802.0	499	58089.8	499	64185.9
2	3300	31751.6	3402	57241.2	3059	63189.2
3	499	31658.9	1257	57100.3	3802	63175.5
4	3403	31597.3	3059	57044.7	2391	63147.2
5	772	31504.1	3300	57025.0	3783	63132.5
6	3402	31350.8	3802	57023.2	2301	63130.3
7	2299	31286.9	2391	56965.1	320	63125.4
8	1810	31057.0	3783	56963.0	150	63060.3
9	3474	31046.3	2301	56944.4	2700	63040.5
10	3059	30978.1	320	56940.5	3402	63039.4

Table 9. Top Bottom designs at T_2 T_4.

	T_2		T_4	
	Design	Avg C	Design	Avg C
1	499	58089.8	3802	205100.1
2	3402	57241.2	2391	205097.4
3	1257	57100.3	3059	205089.6
4	3059	57044.7	2301	205079.5
5	3300	57025.0	3783	205074.9
6	3802	57023.2	150	204893.7
7	2391	56965.1	320	203368.6
8	3783	56963.0	3268	203335.7
9	2301	56944.4	1308	202985.8
10	320	56940.5	298	202966.4

5.3. 60-Station Closed Queue Network

We now consider a 60-station closed queue network with 60 customers. The network consists of three concentric rings of servers and each ring contains 20 servers. To illustrate the routing of customers in this network let us refer to the servers by two integers (i, j), where $i(i = 0, 1, 2)$ denotes the ring number and $j(j = 0, 1, 2, \ldots, 19)$ denotes the position of the server under consideration in the ith ring. After service completion at server (i, j) the customer has an equal probability of being routed to either $(i, (j + 1)_{\text{mod}20})$, $((i + 1)_{\text{mod}3}, (j + 1)_{\text{mod}20})$, or $((i - 1)_{\text{mod}3}, (j + 1)_{\text{mod}20})$; that is, we have a completely symmetric network. This is illustrated in Figure 13. The network in initialized with 60 customers randomly allocated to the servers. For the remainder we assume the servers are indexed by numbers from 1 to 60. It is of interest to compute the service rate μ_i $(i = 1, 2, \ldots, 60)$ of the 60 servers that maximize the throughput, subject to the restriction that the sum of the service rates equals 100. That is,

$$\sum_{i=1}^{60} \mu_i = 100.$$

Since the network is symmetric, the optimal solution to this problem is $\mu_i = 100/60$. We measure the throughput of the closed queue network as the cumulative rate of completions at all stations in the network.

The design space for this problem is the 60-dimensional hyperplane represented by the constraint on the μ_i's. We simulate the network using the standard-clock procedure described earlier. We decide on an upper bound on the number of clock ticks, and at every tenth fraction of this upper bound we compute the ranks of the N designs based on a sample-path estimate of the throughput. We compute the throughput for any choice of the vector

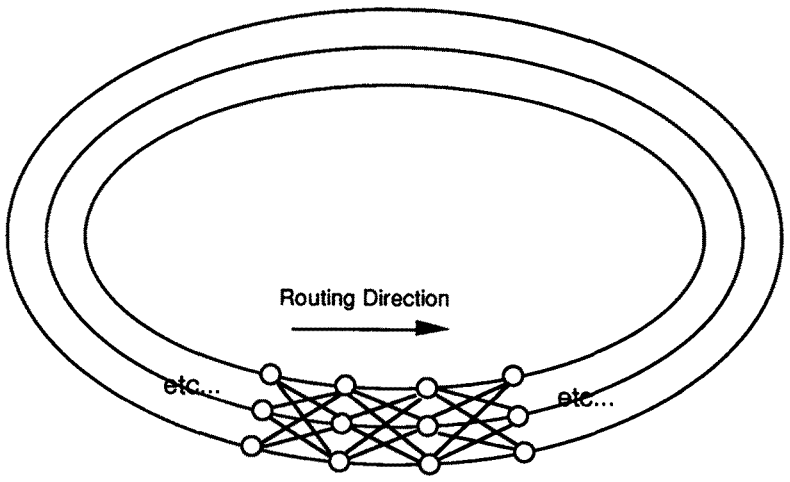


Figure 13. The 60-station closed queue network.

$(\mu_1, \mu_2, \ldots, \mu_{60})$ using mean value analysis [Reiser and Lavenberg 1980]. This enables us to compute the percentage offset of the throughput of the design at hand from the optimal value. The throughput of the optimal design, where $\mu_i = 100/60$ for $i = 1, 2, \ldots, 60$, is 50.42.

Before we present the details of the simulation, we present some observations on the population profiles over the design space. We took 50,000 uniformly distributed samples over the design space and computed the throughput using mean value analysis. the percentage offset of the throughput of these designs with respect to the optimum was computed. The number of designs that fell within every 10th percent from the optimum is presented in Table 10. This table suggests that a large percentage of the population in the design space are poor designs. In fact, (5) suggests that we can almost certainly assume that 99.9% of the population is worse than the best design obtained from the 50,000 designs and we are 99.32% confident that 99.99% of the population is worse than the best design in Table 10, thus implying that a statement on population percentiles over the present design space is not a satisficing notion of optimality. This suggests that designs picked by our scheme should be improved by using other cardinal optimization techniques.

We consider two cases of $N = 100$ and $N = 200$ and a maximum of 100,000 clock ticks. The rank dynamics plot of the top 11 designs for $N = 100$ and $N = 200$ are shown in Figures 14 and 15, respectively, where the paths taken by the best design are highlighted. Since we are interested in maximizing the throughput, the best design has a rank of N (=100 or 200). As before, we observe that the ranks stabilize very early in the simulation. For the case when $N = 100$, Figure 14 illustrates the fact that the design in the 96th position at the 10,000th clock tick stabilizes at the top position by the 30,000th clock tick. Using mean value analysis, we note that this design offers a throughput that is offset by -85.74% from the optimum, confirming the observation on the population profiles illustrated in Table 10.

Figure 15 shows the top ranks for the case when N=200. Here we observe that the design in the 200th position at the 10,000th clock tick remains in the 200th position through out the simulation. Using mean value analysis we find that this design is offset by -77.05% from the optimum. As mentioned earlier, when the performance surface tends to be "peaky" the exact value of the optimal performance can be much higher than the value of the performance of a representative of a reasonably large population percentile. Tables 11 and 12 show the percentage offsets of the top 11 positions at the various sample points for the

Table 10. Histogram of the population profile for 50,000 uniformly distributed samples.

Percentage Intervals	No. of Designs
−0 to −10	0
−11 to −20	0
−21 to −30	0
−31 to −40	0
−41 to −50	0
−51 to −60	0
−61 to −70	1
−71 to −80	110
−81 to −90	2,462
−91 to −100	47,427

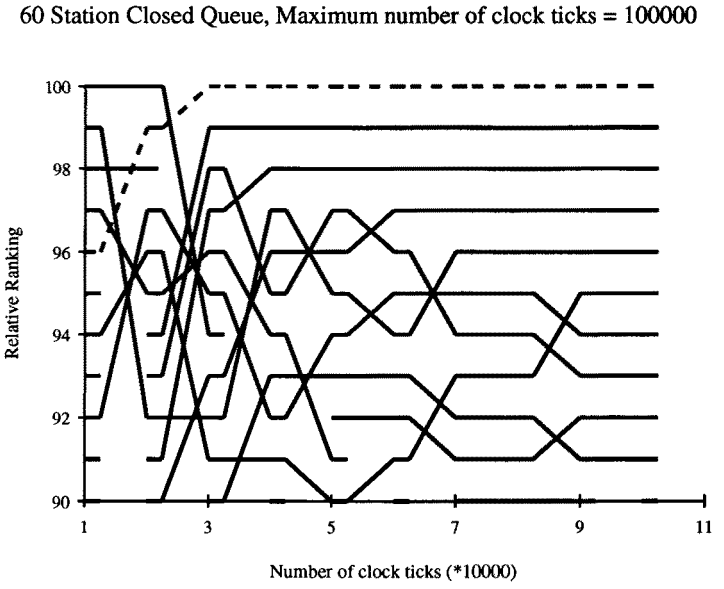


Figure 14. The rank dynamics plot of top 11 designs for the 60 station example for $N = 100$.

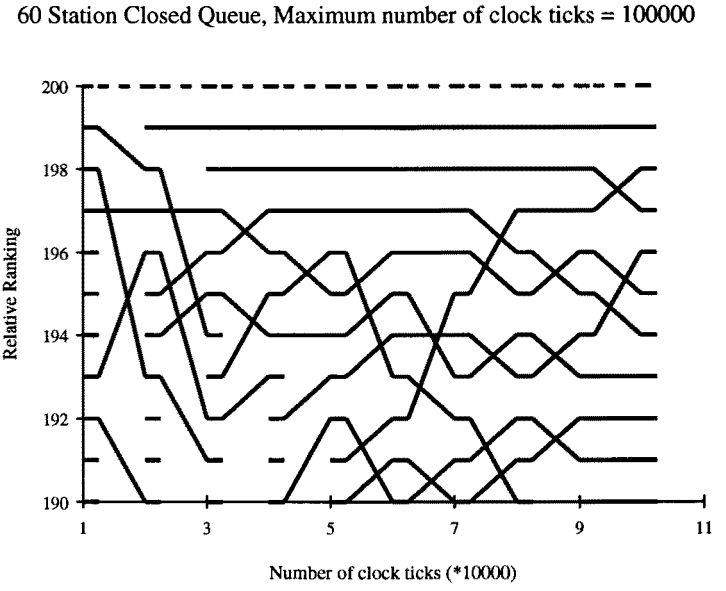


Figure 15. The rank dynamics plot of the top 11 designs for the 60 station example for $N = 200$.

Table 11. Percentage offsets of the top 11 positions for $N = 100$ with respect to true optimum.

	Ticks									
Rank	1×10^4	2×10^4	3×10^4	4×10^4	5×10^4	6×10^4	7×10^4	8×10^4	9×10^4	1×10^5
100	−99.06	−99.06	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74
99	−91.25	−85.74	−87.70	−87.70	−87.70	−87.70	−87.70	−87.70	−87.70	−87.70
98	−97.73	−97.73	−87.94	−84.26	−84.26	−84.26	−84.26	−84.26	−84.26	−84.26
97	−94.47	−90.48	−84.26	−91.25	−87.94	−88.09	−88.09	−88.09	−88.09	−88.09
96	−85.74	−87.31	−94.47	−88.09	−88.09	−87.94	−91.25	−91.25	−91.25	−91.25
95	−96.45	−94.47	−90.48	−87.94	−91.25	−90.48	−90.48	−90.48	−87.31	−87.31
94	−87.31	−87.70	−99.06	−94.47	−90.48	−91.25	−87.94	−87.94	−90.48	−90.48
93	−97.49	−87.94	−88.09	−90.91	−90.91	−90.91	−87.31	−87.31	−87.94	−87.94
92	−90.48	−91.25	−91.25	−90.48	−87.63	−87.63	−90.91	−90.91	−87.63	−87.63
91	−96.18	−84.26	−87.31	−87.31	−94.47	−87.31	−87.63	−87.63	−90.91	−90.91
90	−94.42	−88.09	−90.91	−97.73	−87.31	−93.48	−92.79	−87.63	−87.63	−90.24

Table 12. Percentage offsets of the top 11 positions for $N = 200$ with respect to true optimum.

	Ticks									
Rank	1×10^4	2×10^4	3×10^4	4×10^4	5×10^4	6×10^4	7×10^4	8×10^4	9×10^4	1×10^5
200	−77.05	−77.05	−77.05	−77.05	−77.05	−77.05	−77.05	−77.05	−77.05	−77.05
199	−91.97	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74	−85.74
198	−91.25	−91.97	−86.21	−86.21	−86.21	−86.21	−86.21	−86.21	−86.21	−87.70
197	−84.26	−84.26	−84.26	−87.31	−87.31	−87.31	−87.31	−87.70	−87.70	−86.21
196	−97.49	−92.91	−87.31	−84.26	−90.91	−84.26	−84.26	−87.31	−84.26	−88.09
195	−93.50	−87.31	−87.94	−90.91	−84.26	−87.94	−87.70	−84.26	−87.31	−84.26
194	−96.07	−87.94	−91.97	−87.94	−87.94	−88.09	−88.09	−87.94	−88.09	−87.31
193	−92.91	−91.25	−90.91	−92.91	−88.09	−90.91	−87.94	−88.09	−87.94	−87.94
192	−90.19	−88.72	−92.91	−88.09	−88.72	−87.70	−90.91	−88.72	−88.72	−88.72
191	−99.06	−94.47	−91.25	−90.17	−87.70	−88.72	−88.72	−88.72	−88.72	−88.72
190	−95.07	−90.19	−98.38	−88.72	−88.72	−88.72	−88.72	−90.91	−90.25	−90.25

cases $N = 100$ and $N = 200$ respectively. As before, the rank dynamics plots of Figures 14 and 15 can be inferred from Tables 10 and 11, respectively.

6. Conclusion

We believe ordinal optimization opens a new chapter in the design optimization of systems, stochastic or deterministic. Much is possible and remains to be done. Some speculative conclusions are offered below:

1. Traditional optimization literature focuses on the metric properties of, and the relationships between, the multidimensional θ-space and the one dimensional $J(\theta)$-space. For example, via concavity (convexity), we can translate neighborhood properties in the θ-space to that of maximum (minimum) property in the J-space. However, in the "real world" J is often multipeaked and even discontinuous. By introducing an order (or gross

metric) on the J-space directly in the form of the ordered performance curve, we abandon the usual metric property of the θ-space. However, this is not necessarily a loss, as the next point shows.

2. The concepts of genetic algorithm where "neighborhood" in the θ-space is defined via notions such as "natural selection,' 'crossbreeding," and "mutation" offer way of adaptively searching through the θ-space. This may be the analog of the gradient or other the successive approximation method in the ordered J-space.
3. Hybrid techniques—If the system performance is highly peaked, then ordinal optimization alone may be inefficient in finding the optimum. In such cases, usual hill-climbing techniques can be employed for the end-game aspects of the optimization problem.
4. The rank dynamics introduced in the examples of Section 5 naturally suggests pattern recognition and other time-domain signal processing techniques (e.g., matched filters) for determining optimal designs in discrete-event simulation.
5. The relative noise immunity of ordinal optimization implies computational savings either in terms of shorter simulations or simpler but more approximate simulation models.
6. The statistical analysis of ordinal simulation outputs (or a noisy theory of order statistics) is a relatively underexplored area that deserves attention.
7. Computer architectural issues—Ordinal optimization of discrete-event systems is particularly adapted for use on parallel computers and the standard-clock technique of simulation. As the subject gains importance, it will have an impact on computer design, and vice versa.

Acknowledgment

The authors wish to thank four anonymous referees for their detailed comments on an earlier draft of this paper.

Notes

1. Satisficing is the process of developing a satisfactory but not necessarily optimum solution to a problem (cf. pp. 4149 Vol. 6, *Systems and Control Encyclopedia: Theory, Technology, Applications*, Ed. Madan G. Singh).
2. The issues here are different from those typically addressed in ranking and selection literature in simulation (Law and Kelton [1990]).
3. For example, the important sample path monotonicity results of Glasserman and Yao [1989] for some DEDS's show that this order remains unchanged at all times during simulation.
4. In the parlance of computer science, the scalability of our approach is high (Kumar and Gupta [1991]).
5. For the steep and flat cases shown in Figures 3 and 4 the explicit function form of $J_{[i]}$ can be obtained by interpolation using the following data: Steep (1, 45), [20, 90], [40, 135], [80,172], [120, 185], [160, 192], [200, 200]; flat [1, 0.15], [40, 6], [80, 12], [120, 23], [160, 65], [200, 200] in case the readers are interested in replicating the experiments.
6. This is analogous to the birthday paradox.
7. In discrete event simulation when a parallel set of experiments are being carried out, it is reasonable to use common random numbers and leading possibly to correlated sample performance among different experiments.
8. This is not bad even for cardinal optimization considering the small number of customers served and transient effects.
9. The third author would like acknowledge Tak Wing Edward Lau for his help with the connection machine experiments.

10. The 8000 designs for Table 4 were chosen randomly, but for ranking purposes if two designs provided the same NPP, the design with the lower number was chosen. That is, say the NPP of the design 3141 is the same as the design 6973. Then design 3141 is ranked higher.

References

Bertsekas, D.P. and Tsitsiklis, J.N. 1989. *Parallel and Distributed Computation: Numerical Methods.* Englewood Cliffs, NJ: Prentice Hall.

David, H.A. 1982. *Order Statistics*, 2nd Ed. New York: Wiley.

DeHaan, L. 1981. Estimation of the minimum of function using order statistics. *JASA*, 76: 467–469.

Feller, W. 1968. *An Introduction to Probability Theory and Its Applications*, 3rd Ed. New York: Wiley.

Fujimoto, R.M. 1990. Parallel discrete event simulation. *Comm. ACM*, 33: 31–53.

Glasserman, P. and Yao, D.D. 1989. Monotonicity in generalized semi-Markov processes. submitted.

Glynn, P.W. 1986. Optimization of stochastic systems. Proc. 1986 Winter Simulation Conf. 356–365.

Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Ho, Y.C. and Cao, Xiren. 1991. *Perturbation Analysis of Discrete Event Dynamic Systems*. Kluwer.

Ho, Y.C., Cassandras, C., and Makhlouf, M. 1991. Parallel simulation of real time systems via the standard clock approach. Submitted.

Ho, Y.C., Deng, M., and Hu, J.Q. 1992. Correlated estimation noise in discrete event simulation and ordinal optimization. Manuscript in preparation.

Jacobsen, S.H. and Schruben, L.W. 1989 Techniques for simulation response optimization. *Oper. Res. Lett.* 8:

Kumar, V. and Gupta, A. 1991. Analyzing scalability of parallel algorithms and architectures. *Proc. 1991 Int. Conf. on Supercomputing*, Cologne, Germany.

Kung, H.T. 1976. The complexity of obtaining starting points for solving operator equations by Newton's method. *Analytic Computational Complexity*, J. F. Traub (ed)., Academic Press.

Law, A.M. and Kelton, W.D. 1990. *Simulation Modeling and Analysis.* New York: McGraw-Hill.

Lirov, Y., Melamed, B. 1990. Expert design systems for telecommunications. To appear.

Reiser, M. and Lavenberg, S.S. 1980. Mean value analysis of closed multichain queueing networks. *J.A.C.M.*, 27: 313–322.

Righter, R. and Walrand, J.C. 1989. Distributed simulation of discrete event systems. *Proc. IEEE*, 77: 99–113.

Rosenbaum, P.R. 1991 Confident search. Unpublished Manuscript, Department of Statistics, University of Pennsylvania, Philadelphia, PA.

Rubinstein, R. 1986. *Monte Carlo Optimization, Simulation and Sensitivity of Queueing Networks.* New York: Wiley.

Rubinstein, R. and Weissman, I. 1977. The Monte Carlo method for global optimization. *Cah. Cen Etud. Rech. Oper.* 21: 143–149.

Traub, J.F. 1976. The introduction to *Analytic Computational Complexity.* J.F. Traub (ed.), Academic Press.

Vakili, P. 1991. Massively parallel and distributed simulation of a class of discrete event system: a different perspective. Submitted.

Vakili, P. 1992. A standard clock technique for efficient simulation. To appear.