Proceedings of the 31st Conference
on Decision and Control
Tucson, Arizona • December 1992

Hybrid Optimization - An Experimental Study

I.Garai, Y.C. Ho, R.S. Sreenivas
Division of Applied Sciences
Harvard University
Cambridge, Massachusetts 02138

September 3, 1992

Abstract

This paper compares the performance of a hybrid optimization method to that of pure gradient based methods. Our hybrid optimization method comprises of an initial adaptive ordinal search phase followed by a gradient ascent (descent) phase. The adaptive ordinal search phase consists of fixing the size of the design population and ranking the members of the population using an estimated value of the performance. Members of the design population for the next stage are picked using the top designs of the previous population. This process is achieved via a variation on the standard genetic algorithm[1]. Recently it was shown that ranks of populations are relatively insensitive to simulation noise[2], as the experimental data show, this fact is useful in using *short simulation runs* to improve the search efficiency before the onset of the final gradient ascent (descent) phase.

1 Introduction and Rationale

The problem of stochastic optimization arises in several areas of discrete event dynamic systems (DEDS). A very general class of such problems can be represented in the following form:

$$J(\theta) = E[L(x(t), \theta, \xi)]$$

where θ is a multidimensional design parameter, L is some performance functional of the sample path of a DEDS $x(t)$, and ξ represents all the random effects of the problem. This problem is difficult because of the following reasons:(i) Analytical evaluation of $J(\theta)$ 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 (ii) Even though techniques such as perturbation analysis and/or standard clock simulation[8,pp.26,230] has reduced simulation time for some of these problems, accurate estimate of the performance measures and its sensitivity to different design parameters is still very time consuming and may easily exhaust one's computational budget. (iii) Most common approaches to the problem involve iterated methodologies based on gradients, feasible directions and stochastic approximations (for reviews see [3,7,8,9]). These techniques suffer from the following disadvantages[2]: (a) The (n+1)-th iterate depends only on the information near the n-th iterate ; thereby ignoring information collected earlier at the 1,2,... (n-1)-th points of the performance surface (b) 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 (c) Since they are driven by local information, 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, in order to understand the tradeoff of different decisions.

In [2] an approach using *ordinal optimization* was proposed as a complement to the solution of this problem. Ordinal optimization is concerned with (i) whether a given design is better than another (ii) finding, by means of comparison, good designs that belong to the top α-percentile of all designs considered. The usefulness of ordinal optimization is based on the observation that the order of designs are relatively immune to effects of simulation noise and it is possible to select good designs with high probability in the presence of significant simulation noise. It is certainly true that a design that is in the top α-percentile of all designs may still be significantly inferior to the true optimum. But from a practical point of view in most cases it is a top α-percentile design that is sought for. In special cases when achieving the true optimum is important, the final tuning can always be done by traditional methods. In other words, ordinal optimization can be considered as a pre processing step that can be supplemented by other well known multivariable optimization techniques. Before going into further details, we will present a simple situation that will show why gradient method alone is not adequate under constraints on the computation budget and why it was necessary to think of the proposed hybrid optimization technique.

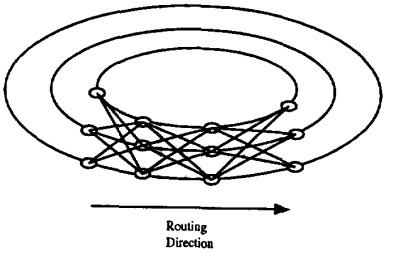

Figure 1: A 60-station closed queue network.

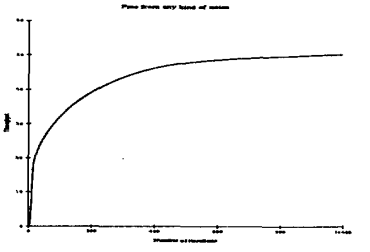

Figure 2: Gradient Ascent for 60-station example using exact gradient values.

Consider the closed queueing network of sixty stations in figure 1. The stations are arranged concentrically on three rings. Each station, independently, behaves like an $M \mid M \mid 1 \mid \infty$ queue. There are sixty customers circulating in the network. Routing is unidirectional and symmetrical, i.e. after completing service at one station, the customer proceeds to one of the adjacent stations with equal probability $\frac{1}{3}$ in the direction shown. There is a constraint on the total service effort available for all the stations. If we denote service rate of station i by μ_i then $\sum_{i=1}^{60} \mu_i = 100$. The objective is to allocate this service effort to sixty stations in a way to maximize the throughput of the network.

Using mean value analysis [10,pp.541-546], the best throughput is calculated to be 50.42. A gradient method achieves this optimum throughput value very easily using the throughput and its sensitivity with respect to μ_i's computed exactly at each step [8,pp.41] using mean value analysis (figure 2).

In practice, we usually estimate the throughput of a network and

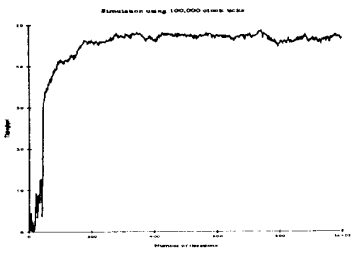

Figure 3: Gradient Ascent for 60-station example using estimated gradient values (100,000 clock ticks).

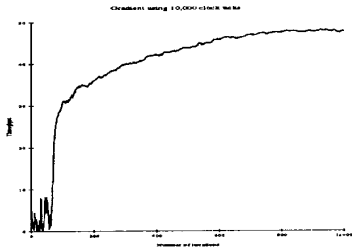

Figure 4: Gradient Ascent for 60-station example using estimated gradient values (10,000 clock ticks).

its sensitivity via simulation since many systems are too complex to model from a queueing theory view point. The accuracy of these quantities will depend on the length of simulation, i.e., total number of customers served during simulation. The result of hill climbing via gradient method, where the appropriate quantities have been estimated using a standard clock simulation and an IPA algorithm [8,pp.39] with 100,000 clock ticks at each iteration (with constant step size), is shown in figure 3.

If we reduce the number of clock ticks during each simulation, the quantities cannot be estimated that accurately. The result of hill climbing using 10,000 clock ticks at each step and reducing the step size according to a harmonic series, as dictated by stochastic approximation, is shown in figure 4.

Because of simulation noise, the gradient method failed to converge to the true best in this case in one thousand steps; it converged to only 47.2. We clarify at this point, that the graphs only show the true throughput and not the throughput obtained via coarse simulation. Once the next step (or search point) has been determined, using a coarse simulation, we calculate the throughput for that search point accurately.

Figure 5 shows the hill climbing method using 1000 clock ticks at each step (with step size reduced according to a harmonic series) of simulation. Because of the severity of simulation noise, the gradient method fails miserably in this case. This failure is primarily due to two factors: (i) Extreme sensitivity of gradient ascent method to simulation noise (ii) Improvement process being dependent on localized information. In hill climbing, gradient method completely ignores the performance of other previously searched points and focuses only in the vicinity of the point where the search is currently in progress. In order to overcome these two difficulties the following method of *hybrid optimization* is suggested: (i) *Step 1:* use *adaptive ordinal optimization* with a noisy estimate of some performance measure and its sensitivity with respect to different system parameters (ii) *Step 2:* once the percentage improvement in two consecutive steps of ordinal optimization becomes insignificant, use a gradient method with finer estimation of the above mentioned quantities to reach the best point. In this paper we claim that improvement due to this randomized yet adaptive technique of search is immune to severe simulation noise. Another byproduct of this research effort is the finding that this ordinal optimization technique tends to preserve the ranks of designs

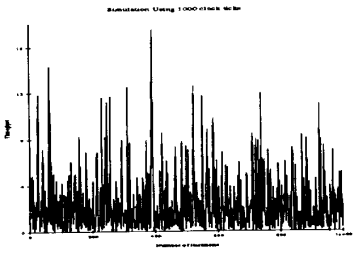

Figure 5: Gradient Ascent for 60-station example using estimated gradient values (1000 clock ticks).

under modeling error. This fact will also be demonstrated through a suitable example.

This paper is divided into the following sections. Section 2 describes in brief the mechanism of this search process. Section 3 describes the experiments done so far in suggesting the claims made in this paper. Section 4 presents the experimental evidence to support these claims. Section 5 concludes the paper with pros and cons of the experiments done and recommendations for future work. Notations used in this paper will be explained in appropriate contexts for the readers' convenience.

2 Hybrid Optimization

The following claims are made in favor of hybrid optimization: (i) Often due to restriction on computing budget and time, it is not possible to estimate performance measures and their sensitivities accurately via long simulation. In such an environment a pure gradient method is inefficient or may even be infeasible. But a hybrid method of optimization works very well because adaptive ordinal optimization is immune to noise due to short simulation. The savings incurred in the first stage of hybrid optimization can be effectively exploited during the second stage. (ii) In several cases of hybrid optimization adaptive ordinal optimization alone is good enough to achieve a "very good" level of performance (iii) Even in presence of severe errors due to modeling approximation the ranks of the designs are well preserved in some cases.

The basic philosophy of hybrid optimization is that when we have a fixed and relatively low computation budget, we can utilize this budget more efficiently through hybrid optimization. In the Adaptive Ordinal Optimization stage, we use short simulation to order the current designs and generate the next set of designs. After a chosen number of iterations of ordinal optimization we switch over to gradient method where longer simulation is used to proceed from one step to another. For example, in the case of the 60-station problem a fair basis for comparing performance of hybrid optimization and pure gradient method will be to compare the number of clock ticks used in both cases to reach interim performance values. Our belief is that since Adaptive Ordinal Optimization is insensitive to simulation noise, we will save quite a bit in terms of computational budget in the initial stage of hybrid optimization. As a result, even though we are using longer simulation to estimate performance measures and their sensitivities at the second stage of hybrid optimization (i.e., hill climbing), overall we still expect to perform better than pure gradient method in terms of computation budget.

2.1 Adaptive Ordinal Optimization

Ordinal Optimization deals with ranking designs (a design is a set of parameters that are to be chosen under certain constraints, e.g. service rates of different stations in case of the 60-station example) according to their performance values (for more details we refer the reader to [2]). It has been shown that ranks of designs are more or less immune to simulation noise. This means that while estimating some performance values corresponding to the different designs

through simulation we can shorten the length of simulation runs (i.e., serve fewer customers) and still be sure that designs will be ranked according to their true performance values with a good deal of accuracy. The term *adaptive* comes from what method is employed to choose the next set of search points from the current set of search points. The algorithm that has been used here for the purpose of adaptation is the well known genetic algorithm[1] with a few modifications. Genetic algorithm consists of few major operators : (i) Reproduction (ii) Crossover and Replacement (iii) Mutation. The operators used in this experiment will be illustrated in the context of the 60-station example. A design in the context of 60-station example is a 60-dimensional vector, where the components represent service rates of different stations. We will represent one such design, say the i-th design, by superscript i, i.e., $\mu^i = (\mu^i_1, \mu^i_2, ..., \mu^i_{60})$ where $\sum_{j=1}^{60} \mu^i_j = 100.0$. For this example, we will keep the number of designs during search limited to one hundred.

As a first step of the genetic algorithm, the initial pool of one hundred designs are chosen randomly, satisfying the constraint. The throughput corresponding to each of these designs is evaluated (these throughputs are of course a rough estimate of the actual throughputs). Let this set of designs be denoted by O. The next step towards creating a new generation of designs is to reproduce the members of the current design pool in proportion to their fitness values. In order to do so, we sort the designs in the descending order of their estimated fitness values. Let the estimated throughput of of the i-th design of the sorted design pool S be denoted by $T_{[i]}$ ($T_{[1]}$ being the best throughput and $T_{[100]}$ being the worst throughput). In this particular example, we will use only top 20 designs for the purpose of reproduction, i.e., each of the reproduced designs will always be from the top 20 designs and the remaining 80 designs temporarily will not be used. In this reproduced pool (let's denote this set by R) the number of a particular design reproduced from the set S will be proportional to its fitness value, i.e., we use the following probability law for reproduction: $P(X = j) = \frac{T_{[j]}}{\sum_{i=1}^{20} T_{[i]}}$ where X denotes the reproduced design and $P(X = j)$ denotes the probability that the reproduced design is j. Replication of one hundred such designs will complete the process of reproduction.

The next step of the Genetic Algorithm is Crossover. We choose two designs from R at random for the purpose of Crossover. Let these two designs be denoted by μ^i and μ^j. Next, we choose a p between 0 and 1 and create a new design μ^c in the following manner : $\mu^c_m = p \times \mu^i_m + (1-p) \times \mu^j_m$ (for m = 1,..60). Naturally, the new design satisfies the constraint: $\sum_{m=1}^{60} \mu^c_m = p \times 100 + (1-p) \times 100 = 100$. We repeat this process one hundred times in order to create one hundred crossed designs. Let this set of designs be denoted by C.

For each of the one hundred new crossed designs in C we choose one design from S in the following manner: $P[Y = j] = \frac{T_{[1]} - T_{[j]}}{\sum_{i=1}^{100} T_{[1]} - T_{[i]}}$ where Y is the design chosen for the purpose of replacement and $P[Y = j]$ is the probability of the chosen design being j-th design in the sorted design pool. If the older design performs worse than the new design we replace the older one by the new one. Necessary care is taken to avoid selecting a design, that has been newly introduced, for replacement.

2.2 Hill Climbing using Gradient Method

Gradient method, in case of this example, uses this very simple algorithm to choose the next step of our search: $\mu^{n+1}_i = \mu^n_i + stepsize \times (\frac{dThroughput}{d\mu^n_i} - scale)$ where μ^n_i and μ^{n+1}_i denote the i-th component of the design vector at the n-th and $(n+1)$-th steps of iteration, *stepsize* is a constant (or a harmonic series if we are using stochastic approximation) and *scale* is a constant to satisfy the constraint on the total service rate. Clearly, in this case $scale = \frac{\sum_{i=1}^{60} \frac{dThroughput}{d\mu^n_i}}{60}$. Since we are very close to optimum, very long simulation is used to estimate throughput and its sensitivities accurately. Figure 6 shows a block diagram of the hybrid optimization algorithm.

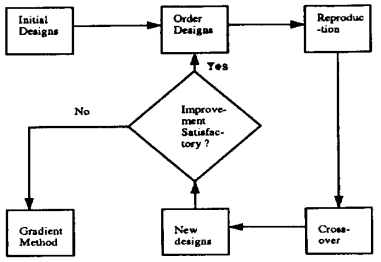

Figure 6: Block Diagram for Hybrid Optimization.

3 Experiments

In section 1 and 2, the experiment of the 60-station closed queueing network was described in detail. The results of this experiment will be presented in the next section. This section describes the other experiments that were done in order to substantiate our claims.

3.1 Experiment 2: An Open Jackson Network

Consider an open Jackson network of one hundred stations with Markovian routing. Each queue independently behaves like an $M \mid M \mid 1 \mid \infty$ queue. Cost of service associated with a unit service rate of station i is c_i. There is a constraint of the following form : $\sum_{i=1}^{100} c_i \mu_i = B$, where μ_i is the service rate of the i-th station and B is the total budget. The objective is to allocate the service rate in a fashion that would minimize the total average number of customers present in the network.

In open Jackson network, the total number of expected customers in the network is given by $\sum_{i=1}^{100} \frac{\rho_i}{1-\rho_i}$ where ρ_i is the average traffic intensity of station i [5,pp.63]. For a given routing matrix P, external arrival rate λ, service budget B and service costs c_i's, we can solve this constrained optimization problem using the Lagrange Multipliers method [6,pp.48]. This optimal solution can be compared with the outcome of our proposed optimization method. The process of hybrid optimization in this case is very similar to that in the last example. But there are a few differences with respect to implementation of the adaptive ordinal optimization method and those differences are described below. In this example, a design is a hundred dimensional vector (since there are one hundred stations). A pool of initial one hundred designs are chosen satisfying the constraint on the service rate.[1] Mean number of customers present in the network corresponding to each of these designs are evaluated and the designs are ordered in the ascending order of these numbers. Since the objective is to minimize total number of customers in the network the designs giving rise to higher number of customers in the system are ranked as bad designs. The probability rule that is used in this case for reproduction is: $P[X = j] = \frac{e^{-\beta j}}{\sum_{i=1}^{100} e^{-\beta i}}$ where β is a constant suitably chosen to control the probability intervals. The exponential probability rule guarantees that (i) better designs are selected with higher probability (ii) the entire genepool is used for reproduction and it supplies variety against narrowing down the genepool prematurely.

Crossover is carried out exactly the same way as in the case of 60-station example. Each crossed design replace the designs in the previous ordered design pool according to the following probability rule: $P[Y = j] = \frac{e^{-\beta(100-j)}}{\sum_{i=1}^{100} e^{-\beta(100-i)}}$ where Y is the design chosen for replacement from the ordered design pool and $P[Y = j]$ is the probability of the chosen design being the j-th ranked design.

In the 60-station example, simulation was used to evaluate throughput and its sensitivity. But in this example and the others, analytical formulae were used to evaluate mean number of customers in the network and its sensitivities. In order to create a noisy environment similar to that created by short simulation, we had to inject noise

[1] Note that this choice of population size being one hundred has nothing to do with number of stations in the network.

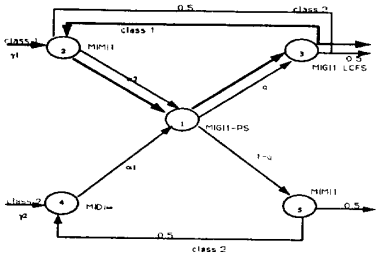

Figure 7: Network of five $M \mid G \mid 1$ queues.

into the quantities calculated using analytical formulae. It was realized during the simulation of the 60-station example that noise on gradients due to short simulation is much more severe than that on the throughputs. Also as we approached the true optimum, the noise was continuously reduced because of the increase in simulation length. For that reason it was decided to use the following type of *multiplicative noise* : $estimated\_number = true\_number(1 + noise)$ where *noise* is uniformly distributed in $[-a, a]$ Since gradients are more sensitive to noise in general, we decided to corrupt the gradients in the following manner: $estimated\_gradient = true\_gradient(1 + noise)$ where *noise* is uniform in $[-2a, 2a]$. Since noise is modulated by the number of customers in the system, in the beginning of the optimization process when the number is too high, the noise level is also too high. This corresponds to the noise generated by short simulation. When number gets smaller, the noise also gets smaller. This resembles reduction of noise level through longer simulation as we get close to the true optimum. The choice of a will determine the intensity of noise and resemble the length of simulation.

3.2 Example 3: A Network of Different Types of Queues

The network in figure 7 consists of 5 different queues as follows: (i) Queue 1 is an $M \mid G \mid$ 1-Processor Sharing queue with mean service rate μ_1 (ii) Queue 2 is an $M \mid M \mid 1$ queue with mean service rate μ_2 (iii) Queue 3 is an $M \mid G \mid$ 1-LCFS queue with mean service rate μ_3 (iv) Queue 4 is an $M \mid D \mid \infty$ queue with mean service rate μ_4 (v) Queue 5 is an $M \mid M \mid 1$ queue with mean service rate μ_5. There are two classes of customers in the network. Class one customers arrive at the queue 2 from outside at an external arrival rate γ_1 and the class two customers arrive at queue 4 from outside at an external arrival rate γ_2. Class one customers are fed back exactly three times to queue 2 when they leave queue 3. After being served at queue 1, class two customers are routed with probability q to queue 3 and with probability $1 - q$ to queue 5. After being served at queues 3 and 5, class two customers are fed back to queues 2 and 4 respectively with a probability 0.5 and they leave the network with probability 0.5. There is a constraint on the total service effort available on these stations, i.e., $\sum_{i=1}^{5} \mu_i = 24$ The objective is to minimize the total number of customers present in the network by choosing the service rates of different queues and the routing probability q.

This is also a problem for which an analytical solution exists and we can compare the result of our proposed hybrid optimization technique against the theoretical optimum. Expected number of customers in the network is given by $E(N) = \sum_{i=1,2,3,5} \frac{\rho_i}{1-\rho_i} + \rho_4$ where ρ_i's can obtained from the traffic equation as $\rho_1 = \mu_1^{-1}(4\gamma_1 + 2\gamma_2)$ (because the class one customers go through queue 1 exactly 4 times), $\rho_2 = \mu_2^{-1}(4\gamma_1 + p\gamma_2)$, $\rho_3 = \mu_3^{-1}(4\gamma_1 + 2p\gamma_2)$, $\rho_4 = \mu_4^{-1}\gamma_2(2 - p)$ and $\rho_5 = \mu_5^{-1}2\gamma_2(1 - p)$ [4]. The optimum solution can be obtained using the Lagrange Multiplier method [6,pp.48].

Hybrid optimization, is carried out the same way as in the last two examples, i.e., we do a few generations of adaptive ordinal optimization to get within a reasonably good performance and then do the final tuning using a gradient method. Implementation of adaptive ordinal optimization is similar to that of example 2 except for the technique of Crossover. In example 1 and 2, a p between 0 and 1 was used to cross two designs and create a new one. This p was different from one Crossover to another but once two designs were chosen for Crossover, their corresponding components were crossed with the same p. This had to be done to maintain the linear constraint on the service rates. The Crossover technique that is used in this problem is very similar to that in case of Crossover on binary string structures [1]. This technique is described below. Our belief is that since the individual components of a design have more freedom to be crossed "in different proportions". this technique introduces more variety in the genepool. Let two designs be denoted by μ^i and μ^j, where $\mu^i = (\mu_1^i, \mu_2^i, \mu_3^i, \mu_4^i, \mu_5^i, q^i)$ and $\mu^j = (\mu_1^j, \mu_2^j, \mu_3^j, \mu_4^j, \mu_5^j, q^j)$. We randomly choose one position for Crossover. Let us say this position is 3. The components left to the position 3 for both the designs remain unchanged and rest of the components undergo Crossover in the following manner. Let us denote the crossed designs by $\mu^{i'}$ and $\mu^{j'}$. Then $\mu^{i'} = (\mu_1^i, \mu_2^i, \mu_3^{j'}, \mu_4^{j'}, \mu_5^{i'})$ and $\mu^{j'} = (\mu_1^j, \mu_2^j, \mu_3^{j'}, \mu_4^{j'}, \mu_5^{j'})$ where

$$\mu_k^{j'} = \frac{\mu_k^j \times (\mu_3^i + \mu_4^i + \mu_5^i)}{\mu_3^j + \mu_4^j + \mu_5^j}, \mu_k^{j'} = \frac{\mu_k^i \times (\mu_3^j + \mu_4^j + \mu_5^j)}{\mu_3^i + \mu_4^i + \mu_5^i}$$

for (k = 3,4,5). Clearly, $\sum_{k=1}^{5} \mu_k^{i'} = \sum_{k=1}^{5} \mu_k^i$ and $\sum_{k=1}^{5} \mu_k^{j'} = \sum_{k=1}^{5} \mu_k^j$ i.e., except for the scale factor we simply interchange the components of the μ^i and μ^j vectors beyond the position two.

In this example q is also a component of the design vector, but there is no constraint on q except that it has to lie between 0 and 1. These q^i and q^j can be crossed by choosing a p between 0 and 1 in the following manner: $q^{'} = pq^i + (1 - p)q^j$.

In this problem too, the number of customers and its sensitivity during hybrid optimization were calculated using analytical formulae. So in order to create a noisy simulation like environment, a zero mean uniform noise was used. The process of using noise is exactly similar to that in example 2.

3.3 Example 4: Approximating an open Jackson network in light traffic

This example tests the performance of ordinal optimization under noise that is generated by modeling approximations. Under a heavy traffic condition, it is possible to model queue lengths of an open network of queues with generalized arrival and service rates, Markovian routing (but no self loop) by a multidimensional Reflected Brownian Motion (RBM) [4,pp.193-203]. The open Jackson network model with exponential arrival and service rates and with Markovian routing satisfies all the required conditions, and as a result its queue lengths can be approximated in heavy traffic with very good accuracy, by a multidimensional RBM. For a light traffic condition, this will be a very rough model of the network. We can use this model to test the performance of ordinal optimization under modeling noise. Our objective is to choose some designs and rank them in the ascending order of number of customers in the system using a Reflected Brownian Motion model in low traffic and to see if these ranks are preserved when the true analytical model is used.

4 Results and Analysis

4.1 Experiment 1

The example of 60-station closed queue network has a theoretical maximum of 50.42. Table 1 shows different optimization methods used, their corresponding best throughputs and the number of steps required to reach that throughput. *Gradient using long* stands for pure gradient ascent, using 100,000 clock ticks for each simulation. *Gradient using short* stands for pure gradient ascent with step size reduction, using 10,000 clock ticks for each simulation. *Pure Adaptive Ordinal* stands for adaptive ordinal optimization, using a population size of one hundred and 10,000 clock ticks for simulation corresponding to each member in the population. Each generation of this method corresponds to one hundred steps of gradient ascent (since the population

Experiment	Throughput	Steps	Total Clock Ticks
Gradient using long	50.42	800	8×10^7
Gradient using short	47.00	1000	10^8
Pure Adaptive Ordinal	39.00	400	4×10^7
Hybrid	50.42	600	6×10^7

Table 1: Results of 60-station Example

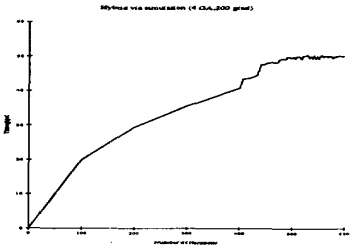

Figure 8: Hybrid Optimization for 60-station example

size is one hundred and we have to do one simulation for each member in the population). Four generations of adaptive ordinal optimization were carried out, which is equivalent to four hundred steps of gradient ascent. *Hybrid* stands for hybrid optimization using four generations of adaptive ordinal optimization, using 10,000 clock ticks for each simulation and two hundred steps of gradient ascent using 100,000 clock ticks for each simulation [figure 8].

When pure gradient ascent was carried out using 10,000 clock ticks at each step, the process converged to a sub optimal point (47.00 approximately). This was primarily due to the noise that was generated during short simulation. Even though step size was reduced continuously, the search process did not converge to the true optimum within a finite number of steps. When 100,000 clock ticks were used the process converged to the true optimum because the estimates were accurate. But in order to reach a performance level of 41.00, the number of clock ticks required was 2×10^7.

While doing adaptive ordinal optimization using 10,000 clock ticks for each simulation with one hundred designs, each generation was equivalent to $100 \times 10,000 = 1 \times 10^6$ clock ticks. In four generations of adaptive ordinal optimization the throughput reached was 39, and in an additional 50 steps of gradient steps with 100,000 clock ticks at each step, it was possible to reach a throughput level of 41.2 (figure 8). So the total number of clock ticks required in the case of hybrid optimization was $4 \times 1 \times 10^6 + 50 \times 100,000 = 9 \times 10^6$. Clearly, hybrid optimization performs better under the constraint of a limited budget. Similarly, if we compare the number of clock ticks required to reach the true best, we will see that the hybrid method still performs better. Besides, since we save in terms of computing budget during the first stage of adaptive ordinal optimization, we can exploit this to obtain finer estimates during the stage of gradient ascent. It is also clear that adaptive ordinal optimization alone is good enough to reach a performance range as close as 80 percent of the true best. These verify our first and second claims.

4.2 Experiment 2

In case of the open Jackson network for one hundred stations with a given routing matrix, external arrival rates, unit service costs and total budget, the theoretical minimum number of customers present in the system was calculated to be approximately 1836. Table 2 summarizes the results of the experiment. *Gradient using n.f 2* stands for pure gradient descent in the presence of multiplicative noise with a noise factor of two and step size reduction according to a harmonic series. *Gradient using n.f. 1* stands for pure gradient descent in the presence of multiplicative noise with a noise factor of one and similar step size reduction. *Adaptive ordinal n.f 2* stands for adaptive ordinal optimization using a population size of one hundred in the presence of multiplicative noise with a noise factor of two. *Hybrid* stands for

Method	#Customers	Steps
gradient using n.f. 2	3764	1600
gradient using n.f. 1	2300	980
adaptive ordinal n.f. 2	2579	400
hybrid	2305	500

Table 2: Results of Open Jackson Network Experiment.

Method	#Customers	Steps
gradient using n.f 2	26.67	1000
gradient using n.f 1	11.00	850
adaptive ordinal n.f 2	12.65	400
hybrid	11.02	500

Table 3: Results of Network of Mixed Queues.

hybrid optimization with four generations of adaptive ordinal optimization, using a population size of one hundred in the presence of multiplicative noise with a noise factor of two followed by two hundred steps of gradient descent in the presence of multiplicative noise with a noise factor of one and step size reduction.

We see that the pure gradient ascent does not perform very well when the noise factor is two. This is primarily because at such a high noise level the gradient estimates are very bad. This situation can be compared with a simulation, using 10,000 clock ticks at each step in case of the 60-station example. However, gradient method with a noise factor of one (equivalent to 100,000 clock ticks at each step) took about 980 steps to reach a performance level of 2300 (which amounts to $980 \times 100,000 = 98 \times 10^6$ clock ticks hypothetically). The hybrid optimization took about 500 steps to reach the same performance level (i.e., 4 generations of adaptive ordinal optimization which is equivalent to 400 steps and 100 steps of gradient ascent) and this can be compared with $100 \times 10,000 \times 4 + 100 \times 100,000 = 14 \times 10^6$ clock ticks. It may be argued that the noise factors two and one do not represent simulations using 10,000 and 100,000 clock ticks respectively in different examples. Nevertheless, if we just compare the equivalent number of steps taken by the pure gradient ascent and the hybrid optimization techniques, it is clear that the hybrid method outperforms the pure gradient method.

4.3 Experiment 3

The network of five different queues with complicated service discipline and two classes of customers had a theoretical minimum number in the system 8.65 for a given budget of 24. The optimal routing (theoretical) was 0.68. The following table summarizes the result of this experiment. The optimal routing probability was found to be 0.8 as a result of hybrid optimization.

In case of this experiment, pure gradient ascent with a noise factor 1.0 takes about 850 steps to reach 11.00. This is equivalent to 85×10^6 clock ticks. The hybrid method took a total of 14×10^6 clock ticks (i.e., four generations of adaptive ordinal optimization followed by one hundred steps of gradient descent). It is clear that the hybrid method outperforms the pure gradient method in this case too.

4.4 Experiment 4

In the example of approximating an open Jackson network of twenty stations by a RBM model at low traffic we tried to verify our claim that ranks of the design are well preserved under modeling noise. The minimum and maximum value of percentage error in estimating the number of customers in the system were approximately 22 and 45 respectively. Under such severe noise it was found that 23 of the estimated top 25 designs actually belonged to the top 25 designs. Also it was observed that the number of customers in the system estimated through the RBM model was always higher than that estimated via the true model.

In this example, we see that even though the error due to modeling approximation is very high, the order of the designs is still well

preserved. Of course, we acknowledge the fact that the reflected Brownian motion model always gives a conservative estimate of the mean number of customers in the system. Also there may be other correlations among the error and the estimated performance measure. Another interesting fact observed during this experiment was that if we increase the number of stations in the network, the maximum error in the estimation of the mean number in the system goes down. The reason for that is not clear yet, but our intuition is that the randomization introduced with an increased number of stations is beneficial in some way.

5 Conclusion and Recommendations for Future Work

Experiment 1 was carried out using simulation, while experiments 2,3 were carried out using analytical formulae. So even though we can claim that, in the case of experiment 1, we have been fair to both the methods of optimization under consideration, we cannot claim the same in the case of experiments 2 and 3. Simulation of experiment 3 is very complicated and it is very difficult to establish an IPA algorithm for this problem. However, it was noticed during simulation of the first experiment that the noise in throughput due to short simulation was more or less uniform and of the same order as the throughput, and noise in gradients due to short simulation was much more severe. In that sense, choosing a noise factor of 1.0 on the performance measure and a noise factor of 2.0 on its gradients is meaningful. Experiment 1 clearly proves that, under limited computing budget, hybrid optimization is always better than a pure gradient method. If we accept the way noise was injected in the case of experiments 2 and 3, then it is also clear that the hybrid method of optimization performs better. However, some may argue that comparing the number of clock ticks to achieve a particular performance level may not be a good way to compare the two methods of optimization. The reason for this is that while doing adaptive ordinal optimization, we incur some extra computation (e.g., sorting, random number generator etc). To counteract this argument, we compared the computing time required to reach 95 percentile of the best throughput in experiment 1. Hybrid optimization method took approximately 301 minutes i.e., 147 minutes less than than the pure gradient method which took 448 minutes. This is a substantial time difference. In addition, our second claim is verified if we define a good performance range to be within 80 percentile of the best throughput. It may be possible to achieve a higher throughput with adaptive ordinal optimization alone by increasing the population size.

The main reason why adaptive ordinal optimization may work better than pure gradient method is its insensitivity to simulation noise. Also, since the search process is global, it is very robust. The examples in this paper have performance functions that are single peaked. However, in the case of a performance function that has local optima where the gradient method may fail, the hybrid method will still perform well because it uses global information. The fact that ordinal optimization is insensitive to simulation noise gives us the privilege of identifying good designs at a very early stage. Some amount of analysis has been done to explain the underlying mathematical foundation of ordinal optimization [2]. There is still a need for further analysis in this area. As choice of step size is important in cardinal optimization, likewise in adaptive ordinal optimization the choice of population size is very important to avoid premature convergence. Perhaps it is also possible to find an optimal stopping rule in the case of adaptive ordinal optimization. Nevertheless, this experimental study has given us some clues as to the behavior of adaptive ordinal optimization. A next step towards establishing these hypotheses will require further mathematical analysis.

References

[1] Goldberg, David E., *Genetic Algorithm in search, optimization & machine learning*, Addison-Wesley Publishing Company, Inc., 1989.

[2] Ho, Y. C., Sreenivas, R.S., Vakili,P., "Ordinal Optimization of DEDS ", *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 2, No. 1, July, 1992.

[3] Wilde, Douglass, J., *Optimum Seeking Methods", Prentice Hall*, Inc., 1964.

[4] Walrand, J.,*An Introduction to Queueing Networks*, Prentice Hall Inc.,1988.

[5] Kelly, F.P.,*Reversibility and Stochastic Networks*, John Wiley & Sons Ltd.,1979.

[6] Mital, K.V., *Optimization Methods in Operation Research & Systems Analysis*,Wiley Eastern Ltd.,1976.

[7] Glynn,P.W., "Optimization of Stochastic Systems", *Proceedings of the 1986 Winter Simulation Conference*, 356-365,1986.

[8] Ho, Y.C.,Cao, X.R., *Perturbation Analysis of Discrete Event Dynamic Systems*,Kluwer Academic Publishers,1991.

[9] Jacobson,S.H., Schruben ,L.W., "Techniques for simulation response optimization",*Oper. Res. Letters*, Vol. 8, No. 1, 1989.

[10] Garcia, A.,L., *Probability & Random Processes for Electrical Engineering*, Addison-Wesley Publishing Company, 1989.