Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

ALLOCATION OF SIMULATION RUNS FOR SIMULATION OPTIMIZATION

Alireza Kabirian
Sigurdur Olafsson

Department of Industrial and Manufacturing Systems Engineering
Iowa State University
2019 Black Engineering, Ames, IA 50011

ABSTRACT

Simulation optimization (SO) is the process of finding the optimum design of a system whose performance measure(s) are estimated via simulation. We propose some ideas to improve overall efficiency of the available SO methods and develop a new approach that primarily deals with continuous two dimensional problems with bounded feasible region. Our search based method, called Adaptive Partitioning Search (APS), uses a neural network as meta-model and combines various exploitation strategies to locate the optimum. Our numerical results show that in terms of the number of evaluations (simulation runs) needed, the APS algorithm converges much faster to the optimum design than two well established methods used as benchmark.

1 INTRODUCTION

Numerical optimization refers to a class of optimization methods which only use the numerical values of objective function. Such cases may arise when simulation is applied to evaluate performance measure(s) of selected design points of a stochastic system. In these cases, it is practically impossible or computationally expensive to obtain the closed form objective function based on decision variables. Hence, simulation optimization (SO) methods try to find the best designs of a system through numerical estimations of the stochastic performance measure(s) of the underlying system obtained via simulation.

Many approaches have been proposed in the literature for SO problems. These approaches and the range of the problems they address could be classified with various criteria (Henderson and Nelson 2006). Depending on the possible range of values of decision variables, the methods are classified into continuous, discrete or mixed approaches. When the number of design options is limited, statistical selection methods are usually appropriate (Kim and Nelson 2006). The methods could also be classified based on the qualitative or quantitative (or mixed) nature of decision variables and type of objective function (single criterion or multi criteria) (Azadivar 1999). For detailed review of the available methods, see Andradóttir (1998), Ólafsson and Kim (2002), Gosavi (2003) and for more recent surveys, see Andradóttir (2006), Ólafsson (2006) and Fu et. al. (2005).

Most of the practical SO methods in the literature have a core iterative search based strategy which evaluates available information of past searches (if any) in each iteration, propose new candidate solution(s), and simulate these candidate(s). Therefore, the total computational time in each iteration can be divided into two parts:

1. The time required to propose new candidate solution(s) , and
2. the simulation run time for the proposed candidates.

In the first part the methods use some intelligent strategy to propose new candidate solution(s), and in the second part the objective function(s) of these candidate solution(s) are evaluated. For example, in the first part Genetic Algorithm (GA) applies various operators such as selection, crossover and mutation to generate a new population of candidate solutions in each iteration, and then in the second part the objective functions of the new population are evaluated and returned to the core optimizer of the GA.

In the case of deterministic optimization problems where a closed-form objective function is available, the time required for the second part of the optimization of objective function evaluation is usually negligible as opposed to the time of the first part of proposing new solution(s). In contrast, when a simulation run is used to estimate the objective function, the time of the first part could be much less than the second part because running a simulation model for real systems is usually computationally time-consuming. It is therefore reasonable to measure the efficiency of a SO algorithm in terms of the number of simulation runs required and design the algorithm such that as few as possible simulation runs are needed.

Most of the current methods for SO have originally developed for problems with available closed-form objective function. Therefore, since in the deterministic context the time of evaluating the closed-form objective function could be considered negligible, these methods may not carefully allocate the simulation runs when applied directly to simulation optimization problems. In other words, they are not efficient in proposing new solutions, that is, allocating simulation runs, and hence require a large amount of time to simulate each of these solutions. This fact motivates the development of methods that make better use of the past information of simulated points in a search based strategy.

The underlying idea of this paper is that the following principles could be used to guide the allocation of simulation runs, and hence characterize a simulation optimization method that does makes good use of simulation run time:

1. It should not evaluate the objective function of a single solution point more than once.
2. It should not evaluate the objective function of a solution point that is so close to some other previously simulated points that there is no significant difference between the designs corresponding to these points. This is primarily germane to continuous problems.
3. It should control the amount of search in the neighborhood of already found good solution points and avoid excessively searching the neighborhood of local optima.
4. When searching new local optima in unvisited areas of the feasible region the search strategy should directly take into account the size of those unvisited areas.

Designing a simulation optimization method that addresses each of these four desired characteristics requires more computational memory to store past information and control over past searches. This translates into more computational burden for what we referred to above as the first part of search based methods. On other hand, we content that this increased “thinking” and higher computational overhead could be justified by significantly less cost in the second part if more appropriate candidate solutions are suggested in the first part and hence fewer simulation runs are required.

We also note that these four desired characteristics or principles are not the only desired characteristics of simulation optimization algorithms, and others may be more appropriate in certain applications. However, as we will see in the algorithm proposed below, following these principles appears to result in an effective search algorithm.

The need for more evaluation of candidate solutions before actually conducting simulation to obtain performance measures usually justifies more complex methods which are more conservative in proposing new candidates. If the simulation run time is large enough, we believe that the best methods are those which try to extract and use the valuable information obtained right after each simulation run before proposing a new candidate or at least their population of proposed candidates in each iteration is as little as possible (with a conservative perspective). This strategy of better utilization of past information usually leads to less overall time for SO methods.

The remainder of the paper is organized as follows. Guided by the four principles above, we propose a new iterative heuristic search based method for continuous 2-dimensional problems in section 2. The main ideas of the method are discussed in section 2.1 and more mathematical details are provided in section 2.2. Section 3 summarizes some experimental results for this method. Finally, section 4 concludes the paper and discusses our future research directions.

2 OPTIMIZATION METHOD FOR CONTINUOUS 2-DIMENSIONAL PROBLEMS

To illustrate the effectiveness of the four principles stated in section 1 above, we now propose a method for simulation optimization of problems with two bounded continuous decision variables. For simplicity and without loss of generality, we assume that the upper and lower bound of both decision variables are 0 and 100, respectively, which are the only constraints of the underlying optimization problem. The problem goal is to minimize an expected objective function which is a closed form function of simulation performance measure(s).

2.1 Overview

The core optimizer of the proposed optimization method views simulation as a black box. In fact, the only role of simulation in this method is to transform the new vectors of decision variables introduced by optimizer into an estimated objective function through sampling from simulation performance measure(s).

Throughout the algorithm, the square-shape feasible region is divided into smaller pieces called *partitions*. Each partition with a unique code is always a triangle with 3 corner points that have already been simulated. The space within each partition is unvisited, that is, contains no simulated point. The algorithm starts out with creating initial partitions by simulating 4 corner points of the square of feasible region and the middle point (see Figure 1). Using these 5 simulated points, the feasible region is divided into 4 same-shape partitions.

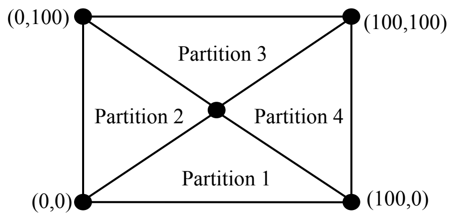


Figure 1: Initial 5 simulated points and 4 partitions

The overall procedure of the method is to select one of the available partitions called *promising partition* and simulate a point inside this partition. Then the promising partition is further divided into 3 new partitions with respect to the location of recently simulated point inside the partition (see Figure 2 with respect to promising partition 2 in Figure 1). As new simulation points are added, so do the number of partitions (in fact two more partitions per simulation are added to the partitions). The natural questions that we address here and in the next section after some definitions are how the promising partition and the new point inside it are selected, that is, *how to allocate the next simulation runs*.

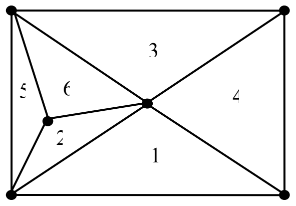


Figure 2: Promising Partition 2 in Figure 1 is further divided into new partitions 2, 5, 6

In order to determine the partition where the next simulation runs will be allocated, we use four sets of partition codes called *Available List* (AL), *Candidate List* (CL), *Blocked List* (BL), and *Taboo List* (TL). Each partition is always a member of exactly one of the three partition sets of AL, BL, or TL. CL is always a subset of AL. A partition will be selected from either AL or CL.

AL is the set of all partitions that promising partition may be selected among them. This list includes all partitions that haven't been blocked or tabooed. BL includes all partitions that have an angle which is close enough to 180 degree. For example if a threshold of 170 degree is defined, each partition with an angle of more than 170 degree is blocked. Blocked partitions are temporarily ignored for promising partition selection. Our experience shows that if these blocked partitions are allowed to be selected as promising partition, flatter partitions are obtained and simulation points are wasted along a line (principles 1 & 2). There is a process in the algorithm called *repartitioning* which merges blocked partitions with a neighboring partition and then splits the merged area into two new partitions. Using this method, the area of blocked partitions may find a new chance to be selected as the promising partition. TL includes all partitions that have 2 conditions: the area of the tabooed partitions is less than a threshold and their all 3 angles are close enough to 60 degree. Except a rare event, tabooed partitions are permanently ignored for promising partition selection. The rare event is the condition that a tabooed partition becomes a neighboring partition of a blocked partition, whereas repartitioning may let the area of a tabooed partition available for further search. Tabooed partitions are expected to be very small partitions that the user assumes no significant difference between the design points inside the partition because these points are very close to each other (see principles 1 and 2 above). Our experience also shows that the corner points of the tabooed partitions are local optima of feasible region, so we taboo them to avoid excessive search surrounding high quality points (principle 3). In some iterations, the algorithm let all available partitions to participate in promising partition selection but in some others, a subset of available list, CL, is selected and only members of this list are allowed to be the next promising partition.

Now suppose that we are in a particular iteration of the algorithm and have determined whether a candidate list or available list is used for promising partition selection. The promising partition is selected randomly among the partitions of candidate or available list, but the corresponding probability distribution should not be uniform. Instead a probability is assigned to each member of CL or AL and the promising partition is selected randomly using these probabilities. In order to achieve the characteristics of a good simulation optimization algorithm outlined in section, we propose three criteria which affect the probability of selecting a particular partition. These are called *space score*, *meta-model score*, and *quality score*. These three criteria are combined into a single indicator called *total score* for each partition and the total scores for the members of AL or CL determines the probability of selecting each partition.

The space score is motivated by the fourth principle from section 1 and is a relative area indicator for each partition. The higher the area of a partition in opposed to other partitions, the higher this criterion would be. This score helps to explore unvisited areas of the feasible region.

Again motivated by the observation that running simulations is computationally expensive relative to other activities, our method induces a meta-model to the input-output (vector of decision variables-estimated objective function) of simulation module. Some sample points are drawn from the area of each partition and the objective function of these samples are approximated with the trained meta-model, which here is a neural network. The meta-model score is based on these approximated objective functions. The better the approximated objective function of the samples of a partition, the higher this criterion would be. In fact, this criterion monitors the whole feasible region quality.

Finally, the quality score for a partition is the representation of the quality of the estimated objective function of simulated points surrounding the partition. The better the quality of these points, the higher the quality score. This criterion exploits the information of high quality simulated points with the hope of discovering local (or maybe global) optima.

To construct a total score, we use a criterion coefficient for each of the three criteria, which are changed between each iteration based on the effectiveness of using each criterion.

The meta-model is updated (restructured and retrained) periodically, for example the user of the algorithm may specify that after each five simulations, the meta-model is updated. The algorithm uses available list in an iteration if the neural network is updated in that iteration, otherwise it uses a candidate list. The members of the candidate list are selected from the set of available list based on the score of the partitions in the available list.

After selecting the promising partition, a point is selected inside the promising partition and simulated. Selection of this point is based on the most effective criteria of the promising partition. Simulating this point, an iteration ends and stopping criteria are checked to see if more iterations could be performed.

2.2 Detailed Algorithm

In this section we present more details for the algorithm motivated above. Since the key component of the algorithm is the partitions that are adapted to the information obtain from the simulation runs already made, we call it *Adaptive Partitioning Search* (APS). The APS algorithm has three phases, which can be summarized as follows.

	Major Steps
Phase 1	• Simulate five fixed points • Divide the feasible region into four partitions
Phase 2	• Select one of the four partitions (which is called promising partition) based only on metamodel criterion • Select and simulate a point inside the promising partition
Phase 3	• Stop the algorithm if any of the terminating conditions are true. • Divide the last promising partition into three new partitions • Update the information of partitions and algorithm • Select a partition called promising partition • Select and simulate a point inside the promising partition • Return to the first step of the phase

In the next three subsections, each of these steps will be elaborated upon.

2.2.1 Phase One

In this short Phase, five fixed points are always simulated which are (0,0), (100,0), (0,100), (100,100), (50,50) as shown in Figure 1. Then the feasible region is divided into four partitions using the diagonals of the square of the feasible region. The primary purpose of simulating these fixed points is to set up initial partitions. But from practical perspective, we also believe that extreme values of decision variables are generally more likely to be the optimum in real world systems.

2.2.2 Phase Two

A meta-model is induced using the first five simulated points. We use an artificial feed forward neural network as meta model that is trained by Levenberg - Merquardt algorithm. The neural network has one input layer with 2 neurons (the number of decision variables), an output layer (since our test problems have a single performance measure) and at least one hidden layer with an upper bound on the number of neurons in each hidden layer. We use a combination of constructive learning algorithms and pruning methods for finding the suitable structure of hidden layers with least mean squared error objective (see Ma and Khorasani, 2003).

Of the three criteria of space, meta-model (neural network in our case), and quality, only meta-model is used for selecting the promising partition in this phase. First, some sample points are selected from each partition. Our sample points for a partition are located on the three medians of the triangle of the partition and have a minimum distance with each other and corner points of the triangle. Using this sampling strategy, the samples represent almost all regions of each partition and are not too close to simulated corner points (principles 1 and 2). Then, the objective function of each sample of each partition is predicted by the structured and trained meta model. The predicted objective function of the j th sample of i th partition denoted by φ_{ij} is scaled such that it has a positive quantity. The Neural Network Score (N_i) is defined by comparing the predicted objective function of the best sample in each partition with that among all partitions:

$$N_i = \frac{\min\limits_{k=1,2,3,4}\left\{\min\limits_{j}\varphi_{kj}\right\}}{\min\limits_{j}\varphi_{ij}} \qquad \forall \quad i=1,2,3,4$$

The total score ψ_i of partition i and finally the probability π_i of selecting this partition are defined by:

$$\psi_i = n \cdot N_i, \forall i$$

$$\pi_i = \frac{\psi_i}{\sum_{k=1}^{4}\psi_k}, \forall i$$

where n is the neural network criterion coefficient that is equal to 1/3 for this phase.

After selecting the promising partition, the point inside the promising partition that has had the best predicted objective function based on neural network model is selected and simulated.

2.2.3 Phase Three

Before commencing a new iteration in this phase, the stopping criteria defined by user are checked and if any one of them is true, the algorithm terminates. As examples of terminating conditions, maximum number of iterations or stalled searches may be used here.

Now suppose that a new iteration must be executed. The overall procedure of this phase is the same as phase 2; a new promising partition is selected and a point inside this partition is simulated. If it is the first iteration in phase 3, the four partitions obtained in phase 1 are added to empty set of AL. Other partition sets (TL and BL) are initialized empty as well. As the first step, the old promising partition is divided into 3 new partitions (Figure 2). Now the old promising partition is removed from AL and the 3 new partitions acquired by partitioning the old promising partition are put in one of the sets of AL, BL or TL. First the conditions of membership to BL and TL (as briefly described in previous section) are evaluated and if a partition doesn't meet the requirements of membership to BL nor TL, it is put into AL. After this step, Repartitioning process of the algorithm merges the area of blocked partitions with one of immediate neighboring partitions and split the resulted tetragon into two new areas, provided that the obtained areas are actually triangle-shape partitions again (Figure 3). These new partitions are then put into TL or AL.

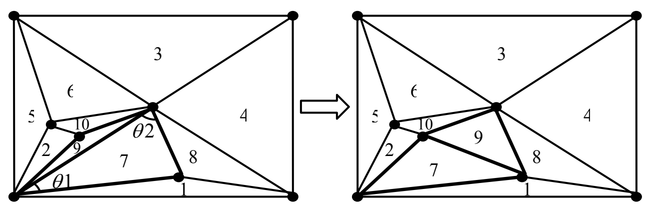


Figure 3: Repartitioning Process merges blocked partition 9 (left figure) with its neighbor (partition 7) and forms new partitions

As the algorithm goes on, the number of partitions rapidly increases and it is computationally cumbersome to let all Available partitions to have a chance to be selected as the next promising partitions; so in some iterations in order to improve the computational efficiency of the algorithm, a subset of Available partitions based on the last assigned total score (excluding the effect of criterion coefficients) are randomly selected and put into CL. The new partitions obtained by splitting the last promising partition and those obtained in Repartitioning Process are also put into CL. Then the promising partition is selected among CL. But in some other iterations, the algorithm let all Available partitions to have a chance to be further searched (based on a user defined plan). Whenever AL is used, the meta model is also updated (restructured and retrained) with all available simulated points.

Now a partition must be selected among the members of either AL or CL. For selecting this partition, a space score S_i, a meta-model Score N_i, and a quality score Q_i are assigned to partition i which is a member of AL or CL. These criteria form a total score for each partition ψ_i and the probability π_i of selecting each partition is defined by:

$$\psi_i = \max\{s \times S_i \quad , \quad n \times N_i \quad , \quad q \times Q_i\}$$
$$\pi_i = \frac{\psi_i}{\sum\limits_{k \in CL \quad or \quad AL} \psi_k} \qquad \forall \quad i \in CL \quad or \quad AL \tag{1}$$

where s, n and q are the criterion coefficient of space, meta-model, and quality respectively. These coefficients are always between zero and one and add up to one, they can vary between iterations of this phase. All three criterion scores are so defined that vary between zero and one and the best partition(s) based on a specific criterion has a score of one for that criterion.

The space score is defined by comparing the area of each partition with the area of the biggest partition in AL (no matter if AL or CL are used). If area of partition i is denoted by A_i, we have:

$$S_i = \frac{A_i\{A_k\}}{\max\limits_{k \in AL}\{A_k\}} \qquad \forall \quad i \in CL \quad or \quad AL$$

If AL is used, the meta-model is updated; hence, the meta-model score of all available partitions are also updated with the same idea as phase two (comparing local best objective function approximation in partition with global best). Otherwise, CL is used and only the meta-model score of new partitions are assigned after sampling some points from each partition.

Each side of a triangle-shape partition could be the border of two immediate neighbor partitions (e.g. partition 6 in Figure 2 is in the neighborhood of partitions 3, 2 and 5). Any two neighbor partitions have 2 simulated points in common and one non-common point. So, each partition has 3 corner points, at most 3 neighbor partitions and consequently at most 3 neighbor point (which are non-common points of neighbor partitions).

The quality score is more complex than the other two scores. The method considers a quality effect for each corner points and each neighbor point of a partition. The quality score of i th partition which is a member of AL or CL is:

$$Q_i = \max\{\zeta_{i1}\ ,\ \zeta_{i2}\ ,\ \zeta_{i3}\ ,\ \eta_{i1}\ ,\ \eta_{i2}\ ,\ \eta_{i3}\}$$
$$\forall \quad i \in CL \quad or \quad AL$$

Here ζ_{ij} and η_{ij} are the quality effect of j th corner (c_{ij}) and neighbor (n_{ij}) points of i th partition respectively. The values of quality effect of neighbors and corners are between 0 and 1. If j th neighbor doesn't actually exists (there are AT MOST 3 neighbors), $\eta_{i1} = 0$. The quality effect of j th corner of i th partition is defined by:

$$\zeta_{ij} = \frac{M(c_{ij}) \times B(c_{ij})}{f(c_{ij}) \times \rho(c_{ij})} \quad \forall \quad i \in AL \quad or \quad CL$$

and the quality effect of j th neighbor of i th partition when this neighbor point exists and is active is defined by:

$$\eta_{ij} = \frac{M(n_{ij}) \times B(n_{ij})}{f(n_{ij}) \times \rho(n_{ij})} \quad \forall \quad i \in AL \quad or \quad CL$$

In formulas of ζ_{ij} and η_{ij}, f is a scaled measure of estimated objective function of the corresponding corner or neighbor point (obtained by simulation). ρ is "saturation indicator", this indicator equals to 1 if the total number of simulated points within the circle-shape neighborhood centered by the corresponding corner or neighbor point is less than an allowed upper bound and is infinite, otherwise. This upper bound depends on the quality of the estimated objective function of the corresponding corner or neighbor point. Generally, the better the quality, the higher the upper bound would be and more searches are allowed in that neighborhood (principle 3). Any simulated point is the corner point of more than one partition simultaneously; M as the "Membership Indicator" is the fuzzy membership of the corresponding point to each partition that has this point as a corner. B as the "Belongingness Indicator" compares the belongingness of each point to the partition that has this point as a corner point and the corresponding immediate neighbor partition. More details of these indicators are beyond the limited space of this paper. Interested readers are referred to Kabirian (2006).

After selecting a new promising partition, a point inside this partition must be simulated. Selection of this point depends on the "winner" criteria of selecting promising partition. The winner criteria is the criterion that its score for the new promising partition multiplied by its coefficient is equal to the total score of the new promising partition (see equation (1)). This criterion has had the most effect in selecting the new promising partition.

If the winner criterion is space, the intersection point of medians of the new promising partition is simulated. If the winner criterion is meta-model, the best sampled point inside the promising partition based on the responses of neural network is simulated (the same idea as phase 2). Finally, if the winner criterion is quality, then the new point to be simulated is selected in the neighborhood of the corner or neighbor point of the promising partition with highest quality effect (see equation 2). However, the procedure of selecting new simulation point in any of the above cases guarantees that the point is always within the promising partition and preferably far enough away from previously simulated points (principles 1 & 2).

Depending on the success or failure of the winning criterion in finding "good" points in last iteration, the coefficients of all criteria are adapted. As an example, if the winning criterion has been meta-model and last iteration has simulated a point better than the best found point until then, the coefficient of meta-model criterion is increased and those of the other two are reduced.

Phase 3 is repeated as long as a stopping rule terminates the algorithm.

3 NUMERICAL RESULTS

Generally, there are 2 types of test beds available for evaluation the performance of SO algorithms. The first type includes actual simulation in which a number of artificial or real stochastic systems with some decision variables and performance measures are used. The second type includes closed form objective functions which mimic the role of simulation by providing numerical values of objective function given vectors of decision variables. The advantage of first type is that test beds are more realistic. But the experimenter has usually no intuition about the combination of response surfaces used. On the other hand, the advantage of the second type is that one can control the variety of the combination of response surfaces. Both types of test beds have been used in the literature. Pasupathy and Henderson (2006) have recently proposed some standards and test beds for comparing SO methods. Since the method proposed in this paper is primarily for two-dimensional problems, we design specific test beds here.

In our full numerical experiments we have used 20 closed form objective functions as test problems, and here we show a sampling of those results from four representative test functions. The feasible region of these test problems is a 100 by 100 square, the same as the problem definition for the proposed algorithm. The value of the objective function of the global optimum of all these test problems is 1 and the worst solutions have an objective function of 100. So, the objective function values fall in the range of 1-100. These objective functions have been designed such that the location of the global optimum can be changed manually. The top part of Figure 18 shows the original version of the first objective function and the lower part of the Figure show two versions of this objective function whose global optimum locations have been changed randomly. (We refer to these two as two observations of the original objective function.)

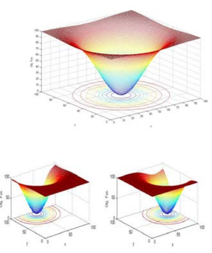
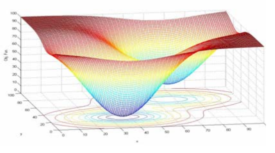
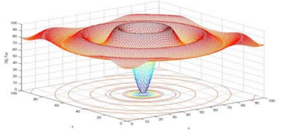


Figure 4: Objective Function 1-Monomodal

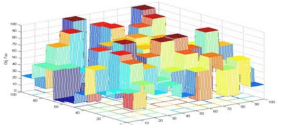
Figure 5 shows the four selected objective functions out of all 20 in their original format. The combination of our test bed of 20 objective function includes a wide range of response surfaces ranging from simple smooth single peak like that of Figure 4 to multiple peak, qualitative decision variables like objective function 16 in Figure 5 and very irregular functions like objective function 17 in Figure 5.



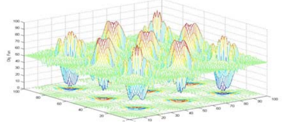
Number 5: Major with Minor



Number 9: Tsunami



Number 16: Manhattan



Number 17: Mirrored Peaks

Figure 5: Four selected closed form objective functions used as test beds

For experiments of this paper, we only let the competing methods to use the numerical values of these objective functions. We add a random variable with standard normal distribution to the value of objective functions to mimic the random nature of simulation outputs. As mentioned earlier, the minimum of objective function is 1. So, we define the range of 1-1.5 as the optimum neighborhood and for each objective function, any competing method which simulates a point whose objective function falls within this optimum neighborhood is stopped because this method is believed that has found the optimum. But we also set a maximum number of simulation runs for the competing methods so that if a method fails to find a point within optimum neighborhood in the first 1000 simulation runs, it is stopped.

We use Stochastic Approximation (see Kim 2006 for overall procedure of this method) and Genetic Algorithm (Holland 1992) to compete with the proposed algorithm in optimizing the test problems. We used MATLAB software for coding the three methods. Firstly, we defined a number of parameter sets for Stochastic Approximation and Genetic Algorithm and using the test problems, we found the best parameter set for these 2 methods. Also, Educated guesses were used for the parameters of the proposed algorithm.

For the experiments, we first draw 5 observation of each objective function. Each one of 3 competing method were tested 5 times for each observation of each objective function and the best found objective function after each simulation run for the trials of the observations of each objective function was averaged.

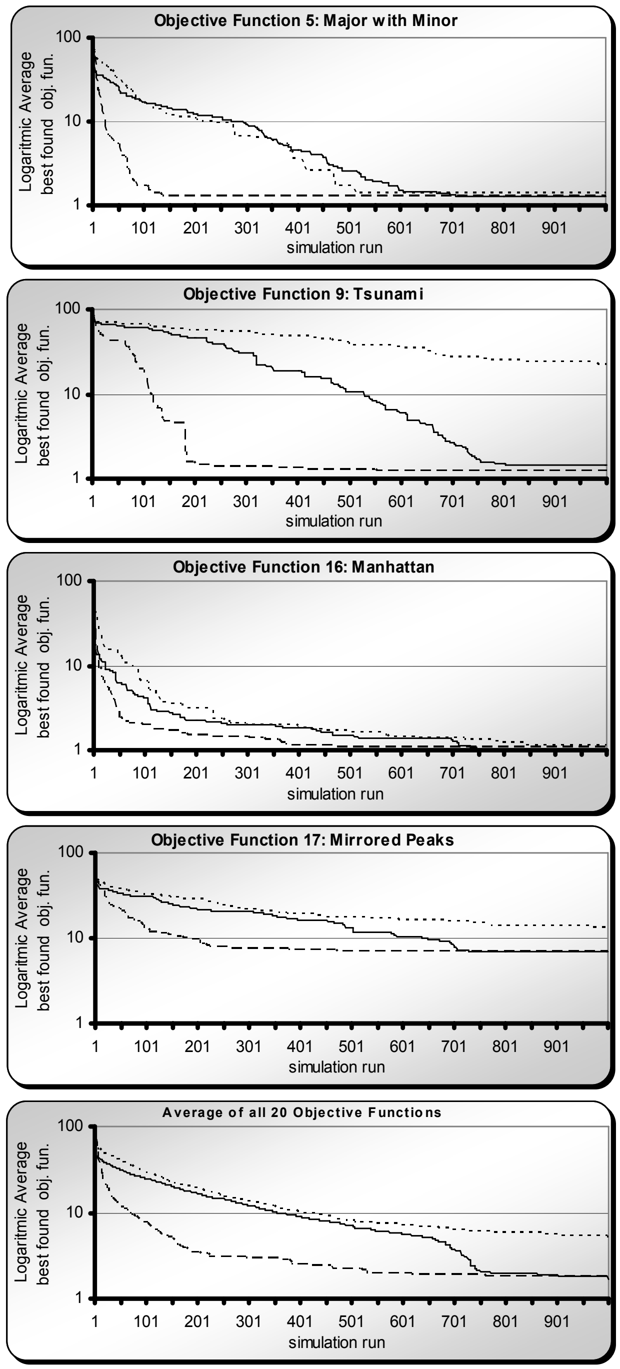
The computational time of the proposed algorithm was much more than those of the other two methods; but this time does not include simulation run time since we used closed form objective functions. In worst cases, the computation of the APS algorithm took less than a few minutes with 1000 evaluation of objective function. Hence, assuming the simulation run time is large enough, we neglected the computational time of the core optimizers and use the number of simulation runs as the performance metric to compare the competing methods.

The first 4 charts of Figure 6 shows the graphs of the average values of the objective function of the best found point after each simulation run for the 5 trials of 5 observations of objective functions of Figure 5 versus the number of simulation runs. The y-axes show logarithmic values of these averages. It must be noted that the curves of each method is not very accurate for the optimum neighborhood because when a trial of an observation of an objective function reaches this optimum neighborhood, that trial is stopped. The last chart of Figure 6 shows the average over all 20 objective functions.

The results shown in Figure 6 and those obtained for other individual objective functions clearly show that the APS algorithm converges to the optimal neighborhood much faster than its competitors. There were some objective functions in our test bed that none of the competing methods could reach the optimum neighborhood within 1000 simulation runs (like objective function 17), but the general pattern is that for almost all of the objective functions, the curve of the APS algorithm were well below those of the other two methods.

4 CONCLUSION

In this paper, a new simulation optimization method was proposed based on some ideas to improve the efficiency of simulation optimization methods. Our primary method could deal with 2 dimensional continuous optimization problems. The method uses a neural network as a meta model and combines different search strategies to explore the bounded feasible region. Using 20 objective functions with various shapes, we showed that the APS algorithm could have better convergence properties than Genetic Algorithm and Stochastic Approximations.



Legend:
Proposed Algorithm: -
Genetic Algorithm: ————————————————
Stochastic Approximation: ·······································

Figure 6: Logaritmic average best found Objective functions of Figure 4 response surfaces and the average over all 20 objective functions used as test bed.

The authors believe that the method could be extended to higher dimensions. One challenge in this extension is the time complexity of the method which is expected to grow very rapidly as dimension increases. One solution to this problem is to use Candidate list more frequently in the 3rd phase of the algorithm. Other solutions are to simplify the various indicators used in the 3 criteria of phase 3 specially Quality Score. We are studying these solutions in order to extend the algorithm to higher dimensions and our future work (Kabirian and Olafsson 2007) generalizes the method presented here.

We also believe that the method could be adapted for discrete decision variables. Our results for applying the 2-dimensional case also showed promising results for qualitative decision variables. We are also working on using the ideas of APS method for multicriteria problems.

Another issue that must be addressed is accounting for the randomness involved in simulation responses. The experimental results presented in this paper were obtained under noisy responses of closed form objective functions used as test bed (we added a standard normal to the objective function values). The nice experimental convergence properties of the APS algorithm under the noisy test bed are signals that the APS algorithm could somehow handle the randomness of responses.

Due to limited space in this paper, we weren't able to discuss about a statistical procedure that determines the number of simulation replications required for each simulation run in the algorithm (and consequently we didn't apply it in the experiments). In our next work (Kabirian and Olafsson 2007), we will present this procedure and prove asymptotic convergence under the condition that the mean response is continuous in a neighborhood of the global optimum. However, the disadvantage is that mathematical convergence guarantee usually requires simulation of design points that have already been simulated (Olafsson 2006) which is inefficient. Since the final goal of developing SO methods is practical appeal, we believe the efficiency should not be sacrificed for having mathematical convergence proofs in the tradeoff between the two.

ACKNOWLEDGMENT

The first author would like to thank Professor Hashem Mahlooji for supervising his M.Sc. dissertation in the Department of Industrial Engineering at Sharif University of Technology.

REFERENCES

Andradóttir, S. 1998. A review of simulation optimization techniques, in D.J. Medeiros, E.F.Watson, J.S. Carson, and M.S. Manivannan (eds.), In *Proceedings of the 1998 Winter Simulation Conference*, 151-158.

Andradóttir, S. 2006. An overview of simulation optimization via random search. In Henderson and Nelson (eds.), *Handbooks in operations research and management science*, 13, 617 – 632, Elsevier.

April, J. M., M. Better, F. Glover, J. Kelly, M. Laguna 2006. Enhancing business process management with simulation optimization. In *Proceedings of the 2006 Winter Simulation Conference*, 642-649

Azadivar, F. 1999. Simulation optimization methodologies. In *Proceedings of the 1999 Winter Simulation Conference*. 93-100.

Fu, M. C., F. W. Glover, J. April. 2005. Simulation optimization: A review, new developments, and applications. In *Proceedings of the 2005 Winter Simulation Conference*. 83-95

Gosavi A. 2003. *Simulation-based optimization: parametric optimization techniques and reinforcement learning*. Kluwer Academic Publishers

Holland, J. H., 1992. Genetic algorithms. *Scientific American*, July 1992, 66-72.

Kabirian, A. 2006. *A review of current methodologies and developing a new heuristic method of simulation optimization*. Master of Science Dissertation (In Farsi), Sharif University of Technology, Tehran, Iran

Kabirian, A., S. Olafsson 2007. *Simulation based optimization via golden partition search.* Working paper, IMSE Department, Iowa State University

Kim, S. 2006. Gradient-based simulation optimization. In *Proceedings of the 2006 Winter Simulation Conference*, 159-167

Kim, S, B. L. Nelson. 2006. Selecting the best system. In Henderson and Nelson (eds.), *Handbooks in operations research and management science*, 13, 501-534, Elsevier.

Ma, L., K. Khorasani. 2003. A new strategy for adaptively constructing multilayer feedforward neural networks. *Neurocomputing*, 51, 361 – 385.

Olafsson, S., 2006. Metaheuristics. In Henderson and Nelson (eds.), *Handbook in operations research and management science*, 13, 633 –654, Elsevier.

Olafsson, S., J. Kim. 2002. Simulation optimization. In *Proceedings of the 2002 Winter Simulation Conference*, 1931-1936

Pasupathy R., S. G. Henderson 2006. A testbed of simulation-optimization problems. In *Proceedings of the 2006 Winter Simulation Conference*, 255-263.

AUTHOR BIOGRAPHIES

ALIREZA KABIRIAN is a Ph.D. student in the Department of Industrial and Manufacturing Systems Engineering at Iowa State University since 2006. He received a M.Sc. in Industrial Engineering from Sharif (Aryamehr) University of Technology, Tehran, Iran in 2006 and a B.Sc. in Industrial Engineering from University of Najafabad, Esfahan, Iran. After money and home town, his interests include simulation, optimization and stochastic modeling. His e-mail address is <a_kabirian@yahoo.com>.

SIGURDUR OLAFSSON is an associate professor in the Department of Industrial and Manufacturing Systems Engineering at Iowa State University. He received a M.S. and Ph.D. from the Department of Industrial Engineering at the University of Wisconsin – Madison, and B.S. in mathematics from the University of Iceland. His research interests focus on discrete optimization, especially metaheuristics, and their uses in simulation optimization and data mining. His web page can be found at <www.public.iastate.edu/~olafsson>.