SIMULATION OPTIMIZATION BASED ON A HETEROGENEOUS COMPUTING ENVIRONMENT

BY

KRUNG SINAPIROMSARAN
MATHEMATICS DEPARTMENT,
FACULTY OF SCIENCES,
CHULALONGKORN UNIVERSITY
EMAIL: KRUNG@MATH.SC.CHULA.AC.TH
TELEPHONE: 218-5158, FAX: 255-2287

MICHAEL C. FERRIS
COMPUTER SCIENCES DEPARTMENT,
UNIVERSITY OF WISCONSIN-MADISON
MADISON, WISCONSIN 53706
EMAIL: FERRIS@CS.WISC.EDU
TELEPHONE: (608)262-4281, FAX: (608)262-9777

The work of the second author is partially supported by Microsoft Corporation, the National Science Foundation and the Air Force Office of Scientific Research.

ABSTRACT

We solve a simulation optimization using a deterministic nonlinear solver based on the sample-path concept. The method used a quadratic model built from a collection of surrounding simulation points. The scheme does not require the modification of the original simulation source code and is carried out automatically. Due to the large number of simulation runs, the high-throughput computing environment, Condor, is used. Simulation computations are distributed over a network of heterogeneous machines and execute on entirely different computer architectures. Additionally, a resource failure is automatically handled using the checkpoint and migration feature of the Condor environment. To reduce the scheduling overhead, we use a Master-Worker Condor PVM implementation that improves overall execution times.

Keywords: Simulation optimization, sample-path concept, nonlinear program, high-throughput computing, Master-Worker implementation.

INTRODUCTION

Many practical engineering problems give rise to mathematical models that are intractable for current algorithmic schemes. The complex nature of the system or the real-world process motivates researchers to employ simulation modeling as an alternative method to draw conclusions about the system behavior. A simulation model is used to describe and analyze both the existing and the conceptual system, sometimes helping in the design of improvements for the real system. Good summaries on simulation components and the advantages and disadvantages of using simulation models can be found in ‘Shannon [20]’ and ‘Banks [3]’.

The popularity of simulation models comes from their use as decision aids for determining possible outputs under various assumptions. A common use of simulation models is in goal-driven analyses that determine values of the decision variables to allow the model to achieve specific goals. This goal-driven simulation is used in many fields of economics, business and engineering. Moreover, the decision variables are typically constrained by other relationships, for example, budgetary or feasibility restrictions. These constraints give rise to an optimization model using the goal as an objective function.

There are two classes of goal-driven simulation optimization. The first class is the level crossing which tries to identify the values of the decision variables of a simulation that produces a simulation output with the specified value, d. We can cast an optimization of this class of problems as $min||S_r(x) - d||_p$ where $S_r(x)$ is the simulation function, d is the target value, $\| \cdot \|_p$ is the p-norm measure function, usually using the Euclidean norm (p=2), and x is the decision variable. The second class is a straightforward optimization that tries to determine the best possible values of the decision variable of the simulation. This problem can easily be cast as a mathematical programming problem.

From the optimization perspective, the simulation model is simply a function that takes input values and derives one or more output values. In reality, most optimization algorithms also rely on the first order or second order derivative of the simulation function. Due to random effects and the stochastic nature of the simulation, the exactness of the gradient of the simulation function may misguide the optimization algorithm to search along incorrect direction. More reliable methods to estimate derivatives must be used.

In addition to the noise of the simulation, the long running time of a simulation model is also an unfavorable situation. Even though the use of the simulation model is cheaper than building the real system, it is very time-consuming to run and analyze. One simulation run may consume hours of execution time on a single computer. Furthermore, multiple simulation runs are required in order to reliably estimate a steady state simulation function or estimate its gradient. Many methods have been used to reduce the execution time of this problem, such as infinitesimal perturbation analysis which approximates the gradient evaluation using one simulation run, see 'Glasserman [7]', 'Ho and Cao [13]', 'Plambeck et. al. [16]' or the score function method (also called Likelihood Ratio method), see 'Rubinstein and Shapiro [19]'.

The paper is organized as followed. In the next section, a simulation optimization model is defined and illustrated. In the third section, a serial version of the quadratic simulation optimization is explained, while the following section describes a parallel implementation of the quadratic simulation optimizer that uses Condor and a Master-Worker server. After detailing some computational results, the last section provides conclusions and outlines future research.

SIMULATION OPTIMIZATION

The optimization of simulations is a challenging problem that researchers have attacked for many years from as long ago as 1950 until the present day. In this paper, we apply a deterministic optimization method based on the sample-path optimization from 'Robinson [18]' and 'Plambeck et al [16]'. However we apply a very different gradient evaluation procedure. The simulation is assumed to be cast as a mathematical model $y = S_r(x)$ where y is a vector of a system response, x is a vector of decision variables and S_r is the simulation model.

There are other popular methods such as the stochastic optimization from 'Robbins and Monro [17]', 'Glynn [9]', 'Spall [22]', or heuristic optimization approaches such as genetic algorithms from 'Azadivar and Tompkins [2]' and 'Hill [12]', simulated annealing from 'Haddock and Mittenthal [11]', nested partitions from 'Shi [21]', or the tabu search method from 'Glover et al [8]'. Even though these methods converge to the solution in the presence of noise, they are usually slower than deterministic mathematical programming methods.

The aim of our work is to use pre-existing algorithms available within a modeling language to determine an optimal solution of a sample path simulation model. We regard the simulation function, S_r: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ as a mapping from a decision variable, $x \in \mathbb{R}^n$, to a simulation response, $y \in \mathbb{R}^m$. The goal-driven simulation problem can be cast as

$$\begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & y = S_r(x) \quad ,(x,y) \in B \end{aligned} \qquad \text{.....[1]}$$

where f:$\mathbb{R}^{n+m} \rightarrow \mathbb{R}$ is an objective function, y is a variable that holds the simulation output and B specifies additional constraints imposed on the variables.

Based on the sample-path optimization, the simulation program must use the same stream of random numbers for a fixed simulation length. Since most simulation implementations use pseudo random number generators from a fixed seed, the simulation function with the same random seed will generate the same output values. Hence, the simulation is deterministic and we can apply a deterministic optimization algorithm to find the minimizer of this function which is close to some minimizer of the steady-state function as described in 'Ferris et al [6]'.

OVERVIEW OF THE QUADRATIC SIMULATION OPTIMIZATION ALGORITHM

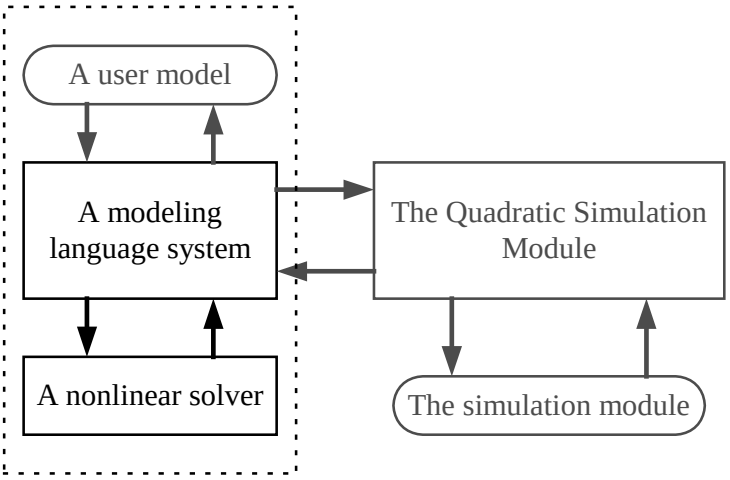

Figure 1: Overview of the simulation optimization via a modeling language system and the external simulation computations.

A practitioner supplies a nonlinear model written in the GAMS modeling language 'Brooke et al [4]' and a simulation module written in traditional programming language such as FORTRAN or C as an external routine, see figure 1. The simulation optimization problem is solved by executing the simulation within the modeling language system. The modeling system first constructs an internal nonlinear model and passes the information to the nonlinear solver. Except for the function and gradient of the external equation (in our case the simulation function), all function and derivative evaluations are supplied by the modeling language using its internal evaluation and automatic differentiation as appropriate. For the evaluation of the external function call, the modeling language system gives control to the quadratic simulation module that calls the simulation module for function computation. However, the derivative computation is different. The module constructs the quadratic model using surrounding simulation points computed by the simulation module. It refines the model using a statistical test. If appropriate termination criteria are met, the nonlinear solver returns the result to the modeling language system. The modeling language system then reports the final result to the modeler. This approach is modular because it can easily apply a new state-of-the-art nonlinear programming solver without recoding additional information about the simulation. This is accomplished by a solver link and dynamic linked library routines within the modeling language system. To deal with a stochastic function such as a simulation, we use a model building technique that relies on surrounding points that are randomly and independently generated. For a linear model of n independent variables, this technique requires at least $n+1$ sampling points, the same requirement as the finite difference method. For the quadratic model of n independent variables, it requires at least $n(n+1)/2+n+1$ for a symmetric model and n^2+n+1 for a non-symmetric model. For higher order polynomial models of n independent variables, the sampling point requirement grows in the order of the degree of the polynomial. We use a quadratic model because of its nonlinear nature and its smaller sampling point requirement. In addition, a higher order polynomial model can cause an over-fitting problem that incorporates simulation noise in the model.

After the simulation runs are complete, we build a small-scale quadratic model using least squares approximation. The aim is to find the best quadratic fitting model of the simulation points that exhibits a similar derivative to the gradient of the simulation function. The least

squares approximation of finding the quadratic fitting model is equivalent to performing a linear regression on the coefficients of the quadratic model using the simulation points as the regression data. Analysis of regression can then be used to determine the best quadratic fitting model.

We use the coefficient of determination, R^2, to test the fitness of our quadratic model to the simulation data. R^2 can be interpreted as the ratio of the regression variation and the total variation in the data points, where the regression variation is the difference between the total variation and the error variation. If R^2 is close to 1, then the regression variation is approximately equal to the total variation. In another words, the error variation is close to zero. We can use the quadratic model to predict all simulation points so that the derivative of this quadratic model is close to the gradient of the simulation function within that neighborhood. If R^2 is close to 0, then the total variation is close to the error variation. That means the quadratic model could not be used to predict any of the data points within that neighborhood.

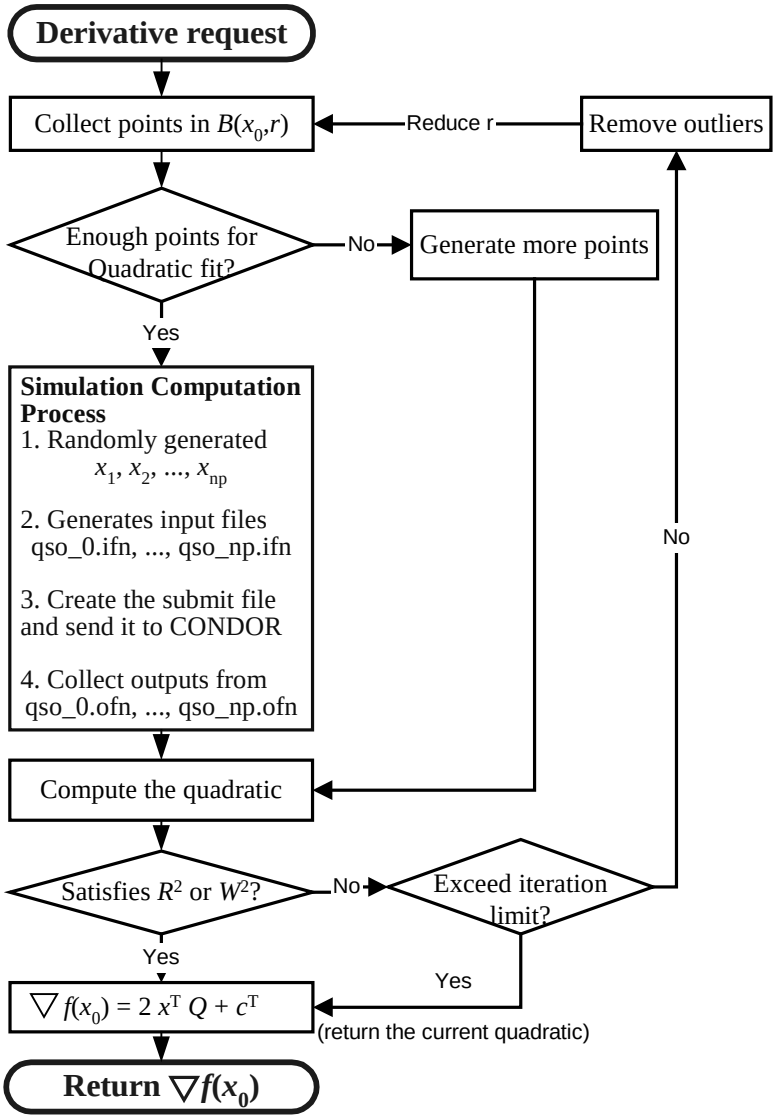

Figure 2: Derivative computation using the QSO algorithm.

Because of the stochastic behavior of the simulation, the quadratic model always has error variation even though it perfectly fits the trend of the simulation function. Instead of rejecting the model, we derive an additional statistical test to be able to accept this type of quadratic model. Figure 2 shows our QSO algorithm in more detail.

In this paper, we propose that the acceptable quadratic model in the neighborhood of the point can be (1) a quadratic model that shows acceptable fitness, i.e. the R^2 is close to 1, or (2) a quadratic model with white noise error. The white noise error is defined as having an error distribution that is normal with unknown mean and unknown variance. This can be determined using goodness-of-fit test statistics.

Goodness-of-fit test statistics from 'Stephens [23]' relate to the problem of determining whether the samples x_1, x_2, ..., x_n are extracted from a population with known distribution $F(x)$. Generally, the Chi-square test is used because it can be easily adapted to any distribution, both discrete and continuous. Because we are interested in the goodness-of-fit test statistics with the normal distribution $F(x)$, a more powerful test such as the Cramér-von Mises statistic in 'Stephens [24]' (herein termed the W^2 statistic), can be used. The W^2 statistic is designed to test the null hypothesis that the (error) distribution of data fits a general distribution such as a normal or an exponential. When the null hypothesis is rejected, then with high probability the error distribution reveals that some simulation points do not fit with the current quadratic model.

In this case, the quadratic model is not acceptable. In particular, for noisy functions, a quadratic model using least squares approximation may not always fit the function. The error distribution of the fit may exhibit a non-symmetric distribution that is caused by some extreme values. In order to fit the quadratic model with this noisy function, we want to exclude these points from our quadratic model building.

To determine extreme points or outliers from the simulation data, we use a skewness measure of the distribution shape. If there are outliers or extreme points in the sample, then the shape of the distribution is not symmetric with respect to the mean or the distribution skews. A distribution is said to be skewed to the right if the mean is larger than the median and it is said to be skewed to the left if the mean is smaller than the median. The coefficient of skewness in 'Allen [1]' can be used to determine the lack of the symmetry in data. It is computed as $\sum_{i=1}^{n}(x_i - \bar{x})^3/(n-1)s_e^3$, the ratio of the expectation of the third power of the sample from its means to the third power of the standard deviation. If the data is distributed symmetrically from the left and the right of the center point, then the skewness is zero. For example, the normal and uniform distribution have zero skewness. If the distribution is skewed to the left, the skewness is negative. The positive skewness indicates that the distribution is skewed to the right.

The skewness can be used to determine the group of extreme points or outliers that we need to exclude from our model building. By reducing the radius, we can exclude some undesirable point from our model. The strategy is to obtain the largest radius so the model building can use as many points as possible.

We sort all points according to their values and group them in a fixed number of blocks. If the coefficient of skewness is negative or the distribution is skewed to the left, the highest block is discarded because it contains extreme points. However, if the derivative point is in this block, we rebuild the quadratic model within this block. Otherwise, we discard the block by computing the largest radius that all points in that block are excluded.

PARALLEL COMPUTATION OF QUADRATIC MODEL

An increasing need for large amount of computational power to solve real-life problems causes researchers to search for a robust computational tool that supplies massive computational power with ease of expandability. The use of one centralized supercomputer machine is limited and becoming less popular because of the cost of buying and maintaining the machine. Condor 'Epema et al [5]' is an environment that is developed to exploit the computational resources of idle heterogeneous machines within a network. It is an efficient and reliable high-throughput computer system that manages the dynamic and heterogeneous resources in a computer network environment. Condor uses a notion of ClassAd's to match a resource request from a client (which defines the resource requirements needed to run a job) to a resource offer from a machine in the network (which advertises its resources such as an available memory, computer type, etc.) To preserve the computational work that has been performed on each machine, Condor offers a checkpointing and migration feature, see 'Litzkow et al [14]'. Condor periodically checkpoints a job during the execution of the program to protect the computational results in the event of an owner of the machine returning or a system failure, e.g. a machine crashing or a power outrage.

To take advantage of the heterogeneous machines within the Condor pool, we compile our simulation module on different operating systems on different machines. The simulation code is re-linked to the Condor library to be able to utilize the checkpoint and migration facilities. Internally, this maintains a trace of the execution of the task so that another machine can restart the execution of the program when the task has been interrupted. Thus, the simulation runs can be performed in parallel on separate machines, with Condor guaranteeing that each simulation task sent out will be eventually completed.

We solve the tandem production line problems appearing in 'Plambeck et al [15]' but with longer simulation runs. Each simulation collects data from a 100,000 products with a warm-up of 1,000 products using the same random seed. This example was generated so that the time for each simulation is significant.

Case	*Serial QSO*		*Condor QSO*		
	Total runs	Usage time	Total runs	Master time	Condor time
1a	239	515	239	118	2527
1b	256	581	256	106	2315
2a	176	51	176	13	2742
2b	149	43	281	30	3910
3a	801	4251	749	1552	2618
3b	955	5168	910	1009	13061
4a	3648	46824	6528	28555	38612
5a	4355	4312	4092	1137	29150
5b	11408	15514	6397	3496	32943

Table 1: Total simulation runs between the serial and the parallel program using Condor.

Unfortunately, the computational results of Table 1 are disappointing since each time a derivative evaluation is requested, the Condor matching algorithm requests new machines that may not be received for a long time. To alleviate this problem, we use a Master-Worker (MW) server paradigm, adopted from 'Goux et al [10]'. The key idea in this approach is to maintain hold over a cluster of machines throughout the complete optimization process, rather than requesting new computational resources 'on-the-fly'.

Typical use of the MW framework generates either an initial list of tasks to perform, or generates new tasks based on the solution of previous tasks (e.g. branch and bound procedures). Once the task list is empty, the program terminates. In contrast, our server is assumed to be persistent, only terminating when a particular flag is set. In order to effect this, we have implemented a "idlewait" task that just spins for a certain amount of time. Thus whenever the server has no work (simulations) to perform, it just idles. Whenever new work is generated, the server already has enough computational resources to perform many tasks in parallel.

The MW server runs as a separate process along with the optimization solver and maintains control of the collection of computational resources throughout the process in the manner described above. The master and worker processes use the file system to exchange information. This significantly reduces the overhead of waiting and rescheduling time that occurs within the Condor environment. We will explain how our gradient approximation based on a quadratic model fits in this framework. This quadratic model requires at least $n(n+1)/2+n+1$ sampling simulation runs. These work loads can take an enormously long time to compute serially. Because these points are randomly and independently generated, it is a perfect fit for the MW paradigm of parallel computation. In this paradigm, the master generates the work for each worker (in this case simulation runs at different point) and waits for the results from each worker. The master will wait for a preset amount of time. If some workers do not finish the jobs within this time period, then the master will cancel the current running jobs and resubmit the incomplete job to the workers again. After all workers have completed their tasks, the master then reports the simulation results and stops.

The MW server runs as a separate program that monitors a request of computations from the file system. A practitioner executes a model in GAMS that passes the control to the nonlinear solver. If the nonlinear solver requests a function or derivative evaluation of the external equation, then it passes a request to QSO module. The QSO module writes an input file that contains all simulation points and updates the status file to trigger the MW server. After the MW server receives a request, it reads the input file and assigns all points to the Tasks To-Do List. It takes care of requesting workers from the Condor pool, assigning tasks to workers, and reassigning tasks if workers fail. Upon completing the last task in the Tasks To-Do List, the MW server writes the output to a file and then updates the status file to inform the QSO module, which has been waiting for the output simulations. The QSO module passes the result back to the nonlinear solver.

RESULTS

The following table demonstrates the effectiveness of our approach. The results are again reported for the tandem production line simulations.

Case		*Serial QSO*			*MW QSO*		
	Runs	*Grad.*	*Time*	*Runs*	*Grad.*	*Time*	*Saving*
1a	432	20	589	432	20	314	275
1b	424	20	584	424	20	275	309
2a	248	16	36	248	16	20	16
2b	316	23	48	316	23	24	24
3a	796	45	2983	862	55	2626	357
3b	1490	238	4963	1504	45	3583	1380
4a	4016	204	32075	5772	402	19885	12190

Case	*Serial QSO*			*MW QSO*			
	Runs	*Grad.*	*Time*	*Runs*	*Grad.*	*Time*	*Saving*
4b	4840	214	40641	6650	309	17856	22785
5a	10712	435	5630	12904	209	3888	1742
5b	8170	429	4500	7824	188	2875	1625

Table 2: Total simulation runs/times for the serial and the MW-server QSO.

We achieve savings in overall time using the MW server for two reasons. First, simulation computations can be performed concurrently, and second, we may be fortunate to run of some very fast machines in the heterogeneous network environment. A chart detailing the computational time savings is shown in Figure 3.

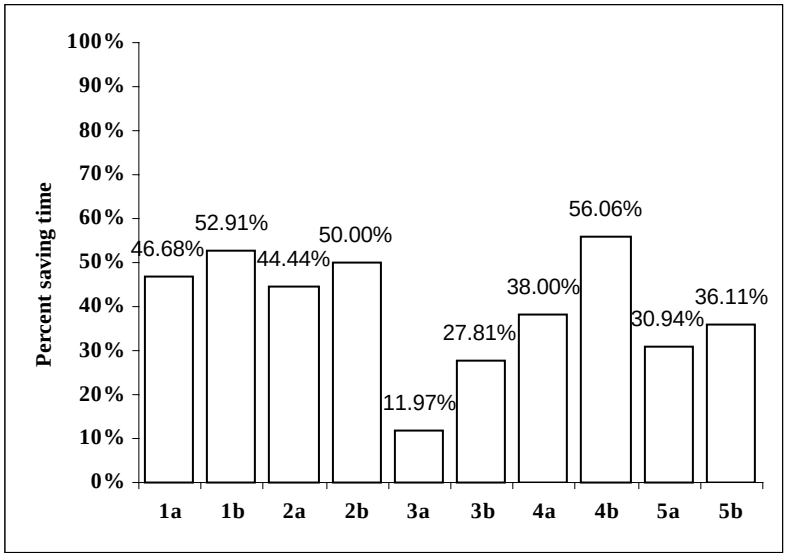

Figure 3: Percentage of saving time comparison between the MW server and its serial version.

Figure 3 shows that we gain savings on all problems. In problems 3a, 4a and 4b, the MW-server QSO uses more derivative computations than the serial QSO. However, it achieves a better running time due to the parallel computation. In the remaining problems, the MW-server QSO uses less derivative computations so that we would expect a faster running time.

In general, the parallel computation does not effect the savings in time for the MW-server scheme as much as we would expect. In part, this is due to the fact that the nonlinear programming solver we are using, CONOPT, performs relatively few gradient evaluations as compared to function evaluations. Only gradient evaluations give rise to concurrent parallel execution of simulations. A side effect of our MW-server implementation is that even for this nonlinear programming solver (that does not require many derivative computations) we achieve a better running time due to the availability of faster machines within the Condor pool.

CONCLUSIONS AND FUTURE RESEARCH

We have exhibited a method for simulation optimization based on the sample-path technique using a gradient based optimization solver. A key feature of our technique is to build a model of the simulation that can then be used to estimate appropriate derivative information.

Our method is applicable for solving engineering simulation design problems that incorporate constraints within a modeling language such as GAMS, commonly used in many areas of economics, business and engineering. Our work also exhibits how modular implementations of different components such as the optimization solver, the external simulation module, and the gradient approximation module can be replaced by new technologies without affecting other components of the solution process. This shows the potential of applying the new state-of-the-art optimization solvers to deal with even more difficult problems. In our simulation optimization method, we treat the simulation module as a black-box that is implemented outside the modeling language system. Its implementation can be written in a standard programming language such as C, FORTRAN or JAVA or in a shell script that extracts data directly from the system. This opens the opportunity for researchers to make use of optimization algorithm without re-implementing the optimization module. They can concentrate on the validity and verification of the simulation model, the analysis of the optimization results and its interpretation. They can also add or modify the constraints to help the optimization algorithm to find suitable solutions for their real problems.

In using Condor, the algorithm makes use of all available computing resources within a network to solve large computational simulation optimizations. The implementation is transparent in that a practitioner does not have to understand anything about Condor. In fact, a practitioner can specify to use Condor via an option file from our solver implementation. To use a MW server, a practitioner needs to start a server as a separate process before executing the GAMS model. The MW server terminates only when the practitioner changes the status file to contain the termination code. The major advantage of using the MW server over using a standard Condor submit file is the reduction in the time spent waiting for an idle machine to become available. We believe that the mechanism to hold onto these resources within the MW framework has significant ramifications in other applications. The overall mechanism gives the illusion of a large scale parallel machine, but at a substantially reduced cost. In many practical cases, this may be an effective way to solve difficult problems.

There are still many research directions that can enhance the capability of the simulation optimization solver to solve large stochastic optimizations, such as the applicability of this method to problems involving a mixture of discrete and continuous decision variables. A method for solving simulation optimization (with discrete design variables) could use gradient based optimization together with search algorithms such as the branch and bound method, genetic algorithms, nested partitions, tabu search or other search techniques. In this paper, we only consider single-stage deterministic simulation optimization problems. Multi-stage, nondeterministic and nonparametric simulation optimization is the subject of further research.

REFERENCES

[1] Allen, A. O. PROBABILITY, STATISTICS AND QUEUEING THEORY, second ed. Academic Press, London, 1990.

[2] Azadivar, F., and Tompkins, G. Genetic algorithms in optimizing simulated systems. *WSC95 1995 Winter Simulation Conference Proceedings* (1995), 757-762

[3] Banks, J. Introduction to simulation. *WSC99 1999 Winter Simulation Conference Proceedings* (1999), 7-13

[4] Brooke, A., Kendrick, D., and Meeraus, A. GAMS: A USER'S GUIDE. The Scientific Press, South San Francisco, CA, 1988.

[5] Epema, D. H. J., Livny, M., Van Dantzig, R., Evers, X., and Pruyne, J. A worldwide flock of Condors: Load sharing among workstation clusters. *Journal on Future Generations of Computer Systems 12* (1996), 53-65

[6] Ferris, M. C., Munson, T. S. and Sinapiromsaran K. A pactical approach to sample-path simulation optimization. In *Proceedings of the Winter Simulation Conference* (2000), J. Joines, R. Barton, K. Kang and P. Fishwick (eds), Omnipress, Orlando, Florida, 795-804.

[7] Glasserman, P. GRADIENT ESTIMATION VIA PERTURBATION ANALYSIS. Kluwer, Norwell, MA, 1991.

[8] Glover, F., Kelly, J. P., and Laguna, M. New advances for wedding optimization and simulation. *WSC99 1999 Winter Simulation Conference Proceedings* (1999), 255-260.

[9] Glynn, P. W. Optimization of stochastic systems. *WSC86 1986 Winter Simulation Conference Proceedings* (1986), 52-59.

[10] Goux, J.-P., Linderoth, J., and Yoder, M. Metacomputing and the master-worker paradigm. Tech. Rep., Argonne National Laboratory, 1999.

[11] Haddock, J., and Mittenthal, J. Simulation optimization using simulated annealing. *Comput. Ind. Eng. 22* (1992), 387-395.

[12] Hill, R. R. A monte carlo study of genetic algorithm initial population generation methods. *WSC99 Winter Simulation Conference Proceedings* (1999), 543-547.

[13] Ho, Y. C., and Cao, X. R. PERTURBATION ANALYSIS OF DISCRETE EVENT DYNAMIC SYSTEMS. Kluwer, 1991.

[14] Litzkow, M., Tannenbaum, T., Basney, J. and Livny, M. Checkpoint and migration of unix processes in the Condor distributed processing system. Tech. Rep. 1347, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1997. http://www.cs.wisc.edu/Condor/doc/ckpt97.ps

[15] Plambeck, E. L., Fu, B. R., Robinson, S., and Suri, R. Throughput optimization in tandem production lines via nonsmooth programming. In *Proceedings of the 1993 Summer Computer Simulation Conference* (1993), Society for Computer Simulation, 70-75.

[16] Plambeck, E. L., Fu, B. R., Robinson, S., and Suri, R. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming 75* (1996), 137-176.

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

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

[19] Rubinstein, R. Y., and Shapiro, A. Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation 32* (1990), 373-392.

[20] Shannon, R. E. Introduction to the art and science of simulation. *WSC98 1998 Winter Simulation Conference proceedings 1* (1998), 7-14.

[21] Shi, L. An integrated framework for deterministic and stochastic optimization. *WSC97 1997 Winter Simulation Conference Proceedings* (1997), 358-365.

[22] Spall, J. C. Stochastic optimization and the simultaneous perturbation method. *WSC99 1999 Winter Simulation Conference Proceedings* (1999), 101-109.

[23] Stephens, M. A. EDG statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association 69* (1974), 730-737.

[24] Stephens, M. A. Asymptotic results for goodness-of-fit statistics with unknown parameters. *Annals of Statistics 4* (1976), 357-369.