Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

GLOBAL SEARCH STRATEGIES FOR SIMULATION OPTIMISATION

George D. Magoulas
Tillal Eldabi
Ray J. Paul

Centre for Applied Simulation Modelling (CASM)
Department of Information Systems and Computing
Brunel University, Uxbridge, Middlesex UB8 3PH, U.K.

ABSTRACT

Simulation optimization is rapidly becoming a mainstream tool for simulation practitioners, as several simulation packages include add-on optimization tools. In this paper we are concentrating on an automated optimization approach that is based on adapting model parameters in order to handle uncertainty that arises from stochastic elements of the process under study. We particularly investigate the use of global search methods in this context, as these methods allow the optimization strategy to escape from sub-optimal (i.e., local) solutions and, in that sense, they improve the efficiency of the simulation optimization process. The paper compares several global search methods and demonstrates the successful application of the Particle Swarm Optimizer to simulation modeling optimization and design of a steelworks plant, a representative example of the stochastic and unpredictable behavior of a complex discrete event simulation model.

1 INTRODUCTION

It has long been established that simulation is a powerful tool for aiding decision-making. This is due to a number of factors, such as the inherent ability to evaluate complex systems with large numbers of variables and interactions for which there is no analytical solution. Secondly, simulation can model the dynamic and stochastic aspects of systems, generating more precise results when compared with static and deterministic spreadsheet calculations. It is also considered as a tool to answer "What if" questions. Simulation itself is a solution evaluator technique, not a solution generator technique (Harrel and Tumay 1994). This scenario could be changed with the aid of optimization procedures. In this case, simulation could answer not only "What if" questions but also answers "How to" questions (Azadivar 1992), providing the best set of input variables that maximize or minimize some performance measure(s) (based on the modeling objectives).

According to Stuckman et al. (1991), simulationists can be classified into 3 categories with regards to the optimization of quantitative variables: the first category tend to use the "trial and error" method, varying the input variables in order to find which set gives the best performance. The second category tends to systematically vary the input variables, to see their effects on the output variables. The third category will apply an automated simulation optimization approach. In the paper we are concentrating on the last category. General reviews and suggestions/approaches are extensively discussed in Andradóttir (1998), Bowden and Hall (1998) and Lee et al. (1999) but this paper particularly investigates the use of global search strategies in this context.

Global search strategies include methods like simulated annealing (Corana et al. 1987, Kirkpatrick 1983), genetic and evolutionary algorithms (Michalewicz 1996), and swarm intelligence (Eberhart et al. 1996, Kennedy and Eberhart 1995). The paper is organized as follows. In the next section, three popular global search strategies are reviewed. We then proceed by describing a typical simulation model that we use as a benchmark in our study. Finally, simulation optimization results are presented and discussed, and the paper ends with concluding remarks.

2 GLOBAL SEARCH STRATEGIES

2.1 Simulated Annealing

Simulated Annealing (SA) is a method based on Monte Carlo simulation, which solves difficult combinatorial optimization problems. The name comes from the analogy to the behavior of physical systems by melting a substance and lowering its temperature slowly until it reaches freezing point (physical annealing). Simulated annealing was first used for optimization by Kirkpatrick et al. (1983). In the numerical optimization framework, SA is a procedure that has the capability to move out of regions near local minima (Corana et al. 1987). SA is based on random evaluations of the objective function, in such a way that transitions out of a local minimum are possible. It does not guarantee, of course, to find the global minimum, but if the function has many good near-optimal solutions, it should find one. In particular, SA is able to discriminate between "gross behavior" of the function and finer "wrinkles". First, it reaches an area in the function domain space where a global minimizer should be present, following the gross behavior irrespectively of small local minima found on the way. It then develops finer details, finding a good, near-optimal local minimizer, if not the global minimum itself.

2.2 Genetic Algorithms

Genetic Algorithms (GA) are simple and robust search algorithms based on the mechanics of natural selection and natural genetics. The mathematical framework of GAs was developed in the 1960s and is presented in Holland's pioneering book (Holland 1975). GAs have been used primarily in optimization and machine learning problems. The fundamental principle of GAs is that at each generation of a GA, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than their progenitors, just as in natural adaptation. A high level description of the simple GA is shown in the code below.

```
STANDARD GENETIC ALGORITHM MODEL
{
//initialize the time counter
t := 0;
//initialize the population of individuals
InitPopulation(P(t));
//evaluate fitness of all individuals
Evaluate(P(t));
//test   for   termination   criterion   (time,
//fitness, etc.)
while not done do
t := t + 1;
//select   a   sub-population   for   offspring
//production
Q(t) := SelectParents(P(t));
//recombine the "genes" of selected parents
Recombine(Q(t));
//perturb the mated population stochastically
Mutate(Q(t));
//evaluate the new fitness
Evaluate(Q(t));
//select   the   survivors   for   the   next
//generation
P(t + 1) := Survive(P(t), Q(t));
end
}
```

More specifically, a simple GA processes a finite population of fixed length binary strings called *genes*. GAs have two basic operators, namely: *crossover* of genes and *mutation* for random change of genes. The crossover operator explores different structures by exchanging genes between two strings at a crossover position and the mutation operator is primarily used to escape the local minima in the weight space by altering a bit position of the selected string; thus introducing diversity in the population. The combined action of crossover and mutation is responsible for much of the effectiveness of GA search. Another operator associated with each of these operators is the *selection* operator, which produces survival of the fittest in the GA. The parallel noise-tolerant nature of GA and their hill-climbing capability make GA eminently suitable for simulation optimization, as they seem to search the parameter space efficiently.

2.3 The Particle Swarm Optimization Method

In "*Particle Swarm Optimization*" (PSO) algorithm the population dynamics simulates a "bird flock's" behavior where social sharing of information takes place and individuals can profit from the discoveries and previous experience of all other companions during the search for food. Thus, each companion, called *particle*, in the population, which is now called *swarm*, is assumed to "fly" over the search space in order to find promising regions of the landscape. For example, in the minimization case, such regions possess lower functional values than other regions visited previously. In this context, each particle is treated as a point in an *n*-dimensional space which adjusts its own "flying" according to its flying experience as well as the flying experience of other particles (companions). There are many variants of the PSO proposed so far, after Eberhart and Kennedy introduced this technique (Eberhart et al 1996, Kennedy and Eberhart 1995). In our experiments we have used a version of this algorithm, which is derived by adding a new inertia weight to the original PSO dynamics (Eberhart and Shi 1998). This version is described in the following paragraphs.

First let us define the notation used in this PSO: the *i*-th particle of the swarm is represented by the *n*-dimensional vector $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ and the best particle in the swarm, i.e., the particle with the smallest function value, is denoted by the index *g*. The best previous position (the position giving the best function value) of the *i*-th particle is recorded and represented as $p_i = (p_{i1}, p_{i2}, \dots, p_{in})$, and the position change (velocity) of the *i*-th particle is $V_i = (v_{i1}, v_{i2}, \dots, v_{in})$.

The particles are manipulated according to the relations

$$v_{in} \leftarrow w \cdot v_{in} + c_1 \cdot r_1 (p_{in} - x_{in}) + c_2 \cdot r_2 (p_{gn} - x_{in})$$

and

$$x_{in} \leftarrow x_{in} + v_{in},$$

where n=1,2,…,N; i=1,2,…,NP and NP is the size of population; w is the inertia weight; c_1 and c_2 are two positive constants; r_1 and r_2 are two random values in the range [0, 1].

The first relation is used to calculate *i*-th particle's new velocity by taking into consideration three terms: the particle's previous velocity, the distance between the particle's best previous and current position, and, lastly, the distance between swarm's best experience (the position of the best particle in the swarm) and *i*-th particle's current position. Then, following the second equation, the *i*-th particle flies toward a new position. In general, the performance of each particle is measured according to a predefined fitness function, which is problem-dependent.

The role of the *inertia weight*, *w*, is considered very important in PSO convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. In this way, the parameter *w* regulates the trade-off between the global (wide-ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e., fine-tuning the current search area. A suitable value for the inertia weight *w* usually provides balance between global and local exploration abilities and consequently a reduction on the number of iterations required to locate the optimum solution. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions, thus a time decreasing inertia weight value is used.

From the above discussion it is obvious that PSO, to some extent, resembles GAs. However, in PSO, instead of using genetic operators, each individual (particle) updates its own position based on its own search experience and other individuals (companions) experience and discoveries. Adding the velocity term to the current position, in order to generate the next position, resembles the mutation operation in evolutionary programming. Note that in PSO, however, the "mutation" operator is guided by particle's own "flying" experience and benefits by the swarm's "flying" experience. In another words, PSO is considered as performing mutation with a "conscience", as pointed out by Eberhart and Shi (1998).

3 THE STEELWORKS SIMULATION PROBLEM

In this section, we use a typical simulation model as a benchmark for comparing optimization methods. We will first explain the system to be modeled and we will proceed by specifying the variables used for optimizing the output from the model.

The steelworks simulation model is an example of a manufacturing simulation model fully described in Paul and Balmer (1993). A brief description is provided here. There are two blast furnaces in the plant, which melt iron at certain daily volumes, which blows and fills as many torpedoes as available and are used to transport molten iron. If no torpedo is available, the molten iron is dropped on the floor and waste is produced. Each torpedo can hold a fixed quantity of molten iron. All torpedoes with molten iron travel to a pit, where crane(s)-carrying ladles are filled from torpedoes, one at a time. The ladle holds 100 tons of molten iron, which is exactly the volume of a steel furnace that is fed from the crane. There are five steel furnaces, which produce the final product of the steelworks. Figure 1 shows the schematic layout (not to scale) for the steelworks plant.

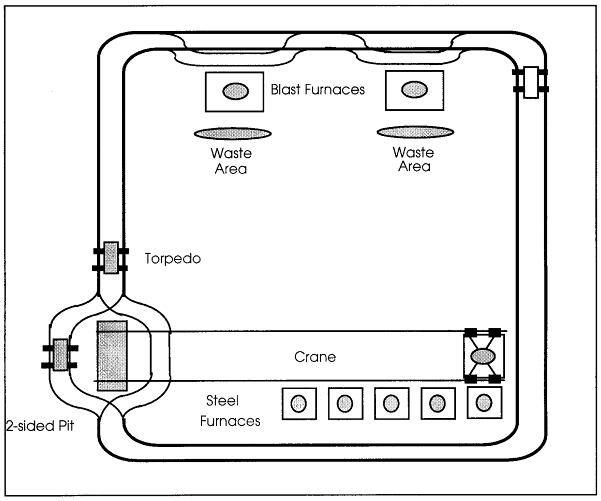

Figure 1: Layout for the Steelworks Plant

When a blast furnace has emptied its blast into the minimum number of torpedoes required (if available) all torpedoes with molten iron in go to the pit (by railway-the torpedoes run on a railway track). This includes partially full torpedoes.

The crane travels along an overhead gantry and carries a ladle (large spoon shaped vessel). The crane is filled at the pit from one torpedo at a time. The pit is 2 sided to avoid any delay if the crane requires more than one torpedo to load it (yet one at a time). In the event that two torpedoes are insufficient to load the crane, the time take to discharge the second torpedo can be considered sufficient for a third torpedo (if there is one) to take the place of the first, ready for unloading. Note that the pit's function is to catch any spillage of molten iron. It does not hold molten iron. The simulation model was built using Simul8 package running on Windows NT (see Figure 2).

Simul8 is a simulator package, which in its simplest form allows the building of models by using graphics (e.g., dragging and dropping icons), by selecting items from menus and filling in dialog boxes. Due to its two-way interface facilities with Microsoft Excel and also Visual Basic for Applications (VBA) it helps the user to build extra routines for the mode. This adds to the software's flexibility and power.

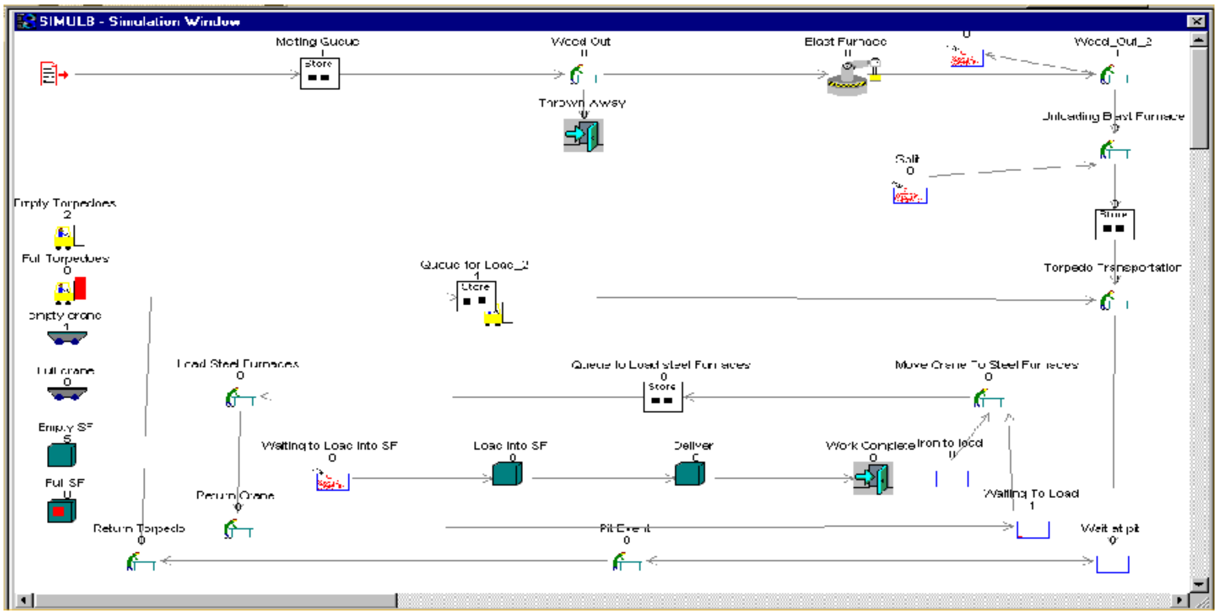

Figure 2: Simulation of the Steelworks Model

4 OPTIMISATION OF THE STEELWORKS MODEL AND RESULTS

This section shows the use of SA, binary GAs and PSO in optimizing simulation outputs using the steelworks model described in the previous section.

Figure 3 shows the interaction between Simul8 and Excel. The model has been built using Simul8 and the optimization algorithm has been implemented in Visual Basic. Excel acts as the 'middle-man' sending information back and forth between the two applications. However, at this point we must note that the end user will only be using Excel as his/her point of interaction. The other applications will run in the background.

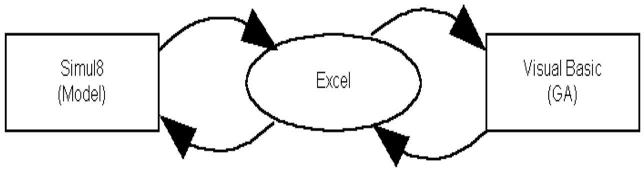

Figure 3: Interactions between the Software Components

In Simul8 we used Visual Logic (a logic based component within Simul8) to pass arguments to Excel and Excel's macro language (VBA) was used to implement the optimization algorithms (see in Figure 4 the optimization worksheet of the PSO algorithm). The data exchange between Simul8 and the optimization algorithm was performed via an Excel spreadsheet using DDE facilities. Simul8 provides an Excel/VB Library (a list of all the functions within SIMUL8 which Excel/VB logic can reference) and Excel/VB Signals (a list of the special signals which SIMUL8 sends to Excel/VB logic while the model runs), which will be employed.

The simulation model has many parameters that could be used as input variables for optimization. We have chosen to keep the number of blast furnaces constant (i.e., two) throughout the experimentation and vary the number of torpedoes, number of steel furnaces, volume that the torpedoes can hold and the number of cranes. This would keep in line with the input variables used in other publications of the same model (Paul and Chanev 1997, Paul and Chanev 1998, Barretto et al. 1999). The discrete parameters, describing the number of torpedoes, cranes or steel furnaces are part of the simulation model and determine the volume of the queues in the model. The real valued parameter, the torpedo volume, is used in the process that models the blowing of the blast furnaces and in the loading of the torpedo with molten iron. The same parameter is used in the process modeling the cranes loading from the torpedoes.

Every fitness evaluation requires a run of the simulation model of the steelworks, so it is important to keep the running time as short as possible. On the other hand, the model has its stochastic behavior, and a long enough run time is needed so that the statistics measured are reliable. The quality of a solution depends on the cost of the proposed design and consists of two factors.

The first factor is the price of the molten iron wasted, which occurs when inappropriate parameter values are chosen, measured for a time horizon equal to 30 days, i.e., 60 min×24 hours× 30 days. The second factor is the investment cost, which is calculated by the amortization of equipment (torpedoes, cranes and steel furnaces) on a monthly basis. The overall cost of the design is calculated by adding the two factors.

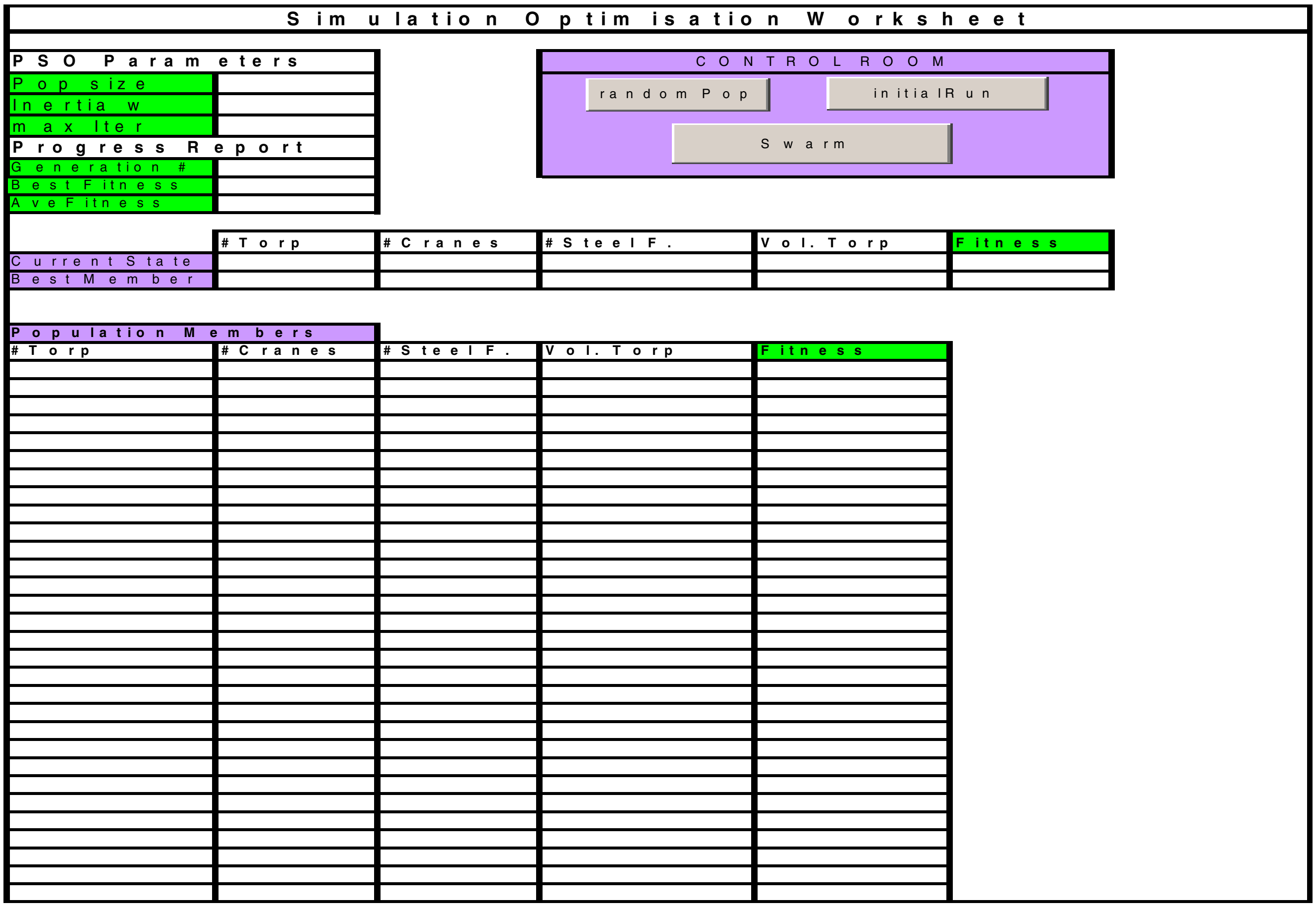

Figure 4: PSO Simulation Optimization Worksheet in Excel

Figure 5 presents the behavior of the best particle in the swarm for various sizes (the results are from 20 independent runs). A summary of the best results is shown in Table 1. Note that no special fine-tuning of PSO heuristics was done; default values suggested in the literature were used instead.

We conducted additional experiments to compare PSO with SA (Barretto et al. 1999). PSO is a population-based method while SA works with a single point. That results of course in totally different convergence behavior. However, we decided to run several experiments with the SA and record the initial points and the solutions found. We then initialized the swarm and the population of the GA (a binary GA with variable parameter encoding, roulette -wheel selection, bit-wise mutation, and one-point crossover was used) in the same neighborhoods. According to our experiments PSO offers superiority over our versions of SA and GAs. The criteria used to arrive at this conclusion are two fold.

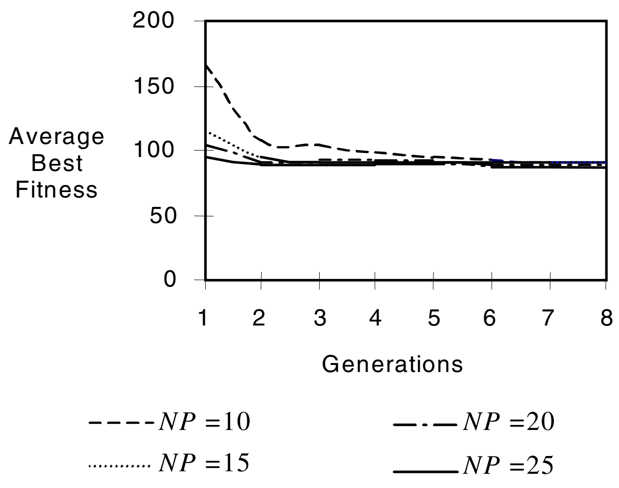

Figure 5: Average Best Fitness for Various Swarm Sizes

Table 1: Summary of the best PSO Results for each Swarm Size

NP	**#Torpedo**	**# Cranes**	**# Furnaces**	**Torpedo Volume**	**Objective Function (£K)**
10	4	2	4	350	94
15	5	2	4	350	92
20	5	1	4	240	86
25	4	1	4	236	84

Table 2: Comparative Results

Method	# Torpedoes	# Cranes	# Furnaces	Torpedo Volume	Objective Function (£K)
SA	6	2	5	260	112.70
PSO	4	2	4	250	92
GA	4	2	5	235	108.5

Firstly the number of iteration required in reaching a "global" optimum is less, and secondly the value of the fitness value reached is lower than that achieved by SA. This may be a fair comparison because the swarm was deliberately initialized in a region analogous to that of the SA test. The SA found a feasible solution after 130 function evaluations (average number) that correspond to searching the 0.3% of the search space. In the PSO test, the swarm consisted of 10 particles and a typical result is shown in Table 2. An average of 6 generations was needed in these tests, which corresponds to searching the 0.14% of the search space. The best result obtained with PSO had a value of 79; a value that is significantly better than other results obtained using SA and GAs. The GA needed 400 function evaluations using 20 individuals on average, a typical example is shown in Figure 6.

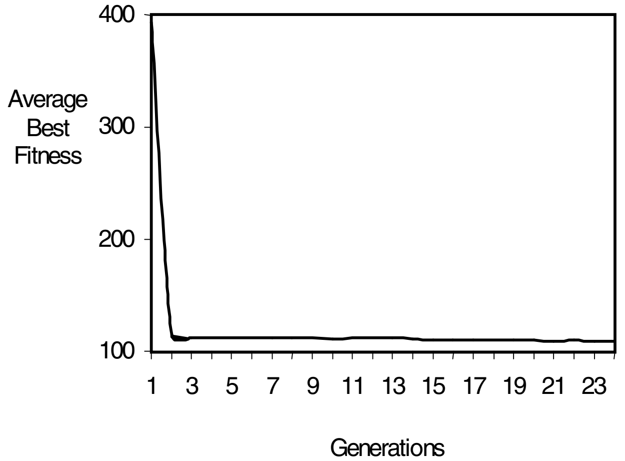

Figure 6: Average Best Fitness for a 20 Member Population

The reasons for the high quality of the PSO algorithm in our experiments may be the following: (i) the coverage of the search space is wide and random; hence an increase in population size has a greater probability of reaching an optimum at an early stage of the search (cf. with Figure 5); (*ii*) the global nature of the search offers insight into various local neighborhoods of the search space; (*iii*) particles moving fast towards the best particle of the swarm allow PSO to perform detailed search of a good region at an early stage. Obviously this can in certain cases to result in locating local minima, as discussed in (Kennedy and Eberhart 1999). To alleviate this problem a modified PSO can be used as suggested in Parsopoulos et al. (2001a). Experiments in classic optimization problems have shown that this modified method exhibits improved performance (Parsopoulos et al 2001a, Parsopoulos et al. 2001b).

5 CONCLUDING REMARKS

Advances have been made in optimizing quantitative variables within a simulation model, and many methodologies now exist for this purpose. However, there are many optimization problems for which there exists no direct or efficient method of solution. Global search methods, such as genetic algorithms and swarm intelligence are eminently suitable for optimizing discrete-event simulation models. These methods do not require derivative-related information and are characterized by good global convergence behavior.

In this paper we presented an application of global search methods to the field of simulation optimization. PSO proved it could be successfully used to optimization and design problems, even if the model is of a stochastic nature. As the environment changes, information contained within the swarm allows efficient discovery of better-adapted solutions. In our case, a very small portion of the search space is investigated even in the discrete variant, which confirms the PSO ability to search effectively huge spaces. PSO is simple to implement and the code is short to any programming language; thus it can be easily included in a simulation package as an add-on tool (Bowden and Hall 1998).

REFERENCES

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

Azadivar, F. (1992) Tutorial on Simulation Optimization. In *Proceedings of the 1992 Winter Simulation Conference*, Arlington, J. Swain, D. Glodsman, R. C. Crain, and J. R. Wilson, eds. New York: Institute of Electrical and Electronics Engineers, 198–204.

Barretto, M.R.P., L. Chwif, T. Eldabi, and R.J. Paul (1999). Simulation optimization with the linear move and exchange move optimization algorithm. In *Proceedings of the 1999 Winter Simulation Conference*, P.A. farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, eds. New York: Institute of Electrical and Electronics Engineers, 806–811.

Bowden, R. O. and J. D. Hall (1998). Simulation optimization research and development. In *Proceedings of the*

1998 Winter Simulation Conference, D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds. New York: Institute of Electrical and Electronics Engineers, 1693-1698.

Corana, A., M. Marchesi, C. Martini, and S. Ridella (1987). Minimizing multimodal functions of continuous variables with the Simulated Annealing algorithm. *ACM Transactions on Mathematical Software* 13 (3): 262–280.

Eberhart, R. C., P. K. Simpson, and R. W. Dobbins (1996). *Computational Intelligence PC Tools*. Boston, MA: Academic Press Professional.

Eberhart, R. C. and Y. H. Shi (1998). Evolving Artificial Neural Networks. In *Proceedings International Conference on Neural Networks and Brain*, PL5–PL13. Beijing, P.R. China: Publishing House of Electronics Industry.

Harrel, C. and K. Tumay (1994) *Simulation Made Easy: A Manager's Guide*, Norcross, Georgia IE: Management Press.

Holland, J. H. (1975). *Adaptation in Neural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Kennedy, J. and R. C. Eberhart (1995). Particle Swarm Optimization,. In *Proceedings IEEE International Conference on Neural Networks*, 1942–1948. Piscataway, New Jersey: IEEE Press. Available online via <`http://dsp.jpl.nasa.gov/members/payman/swarm/`> [accessed February 16, 2001].

Kennedy, J. and R. C. Eberhart (1999). Particle Swarm: Social Adaptation in Information-Processing Systems. In *New ideas in Optimization*, eds.: D. Corne, M. Dorigo, and F. Glover. London: McGrawHill, 379-387.

Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi (1983). Optimization by simulated annealing, *Science* 220: 671–680.

Lee, Y.H., K. J. Park, and T. G. Kim (1999). An approach for finding discrete variable design alternatives using a simulation optimization method. In *Proceedings of the 1999 Winter Simulation Conference*, P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds. New York: Institute of Electrical and Electronics Engineers, 678-685.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs*, New York: Springer.

Parsopoulos, K., V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis (2001a). Improving the particle swarm optimizer by function "stretching", in N. Hadjisavvas and P. Pardalos (eds.), *Advances in Convex Analysis and Global Optimization, vol. 54, Noncovex Optimization and its Applications,* Kluwer Academic Publishers, Dordrecht, The Netherlands, Chapter 28, pp.445–457.

Parsopoulos, K., V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis (2001b). Stretching technique for obtaining global minimizers through Particle Swarm Optimization. In *Proceedings of Particle Swarm Optimization Workshop (PSOW)*, ed. Y. Shi, 22–29. Indianapolis, Indiana: Purdue School of Engineering and Technology, IUPUI Press.

Paul, R. J. and D. W. Balmer (1993). *Simulation modelling*. Lund: Chartwell-Bratt.

Paul, R. J. and T. S. Chanev (1997). Optimizing a complex discrete event simulation model using a genetic algorithm. *Neural Computing and Applications* 6: 229–237.

Paul, R. J. and T. S. Chanev (1998). Simulation Optimization Using a Genetic Algorithm*, Simulation Practice and Theory* 6 (6): 601–611.

Stuckman, B., G. Evans, G., and Mollaghasemi, M. (1991) Comparison of Global Search Methods for Design Optimization using Simulation, In *Proceedings of the 1991 Winter Simulation Conference*, Phoenix, eds: B. L. Nelson, W. D. Kelton, and G. M. Clark, Institute of Electrical and Electronics Engineers, New York, 937–943.

AUTHOR BIOGRAPHIES

GEORGE MAGOULAS is a Lecturer at the Department of Information Systems and Computing, Brunel University. He worked in the industry, participated in several projects, and developed software tools for the design and simulation of fuzzy logic controllers and neuro-fuzzy controllers for real-time control of cement plants and for embedded automotive applications. His research interests include methods for learning and evolution, intelligent optimization in simulation, and real-world problem solving. Dr. Magoulas is a member of the Operational Research Society, IEEE, the Technical Chamber of Greece and the Hellenic Artificial Intelligence Society. Dr. Magoulas' email and web addresses are <`george.magoulas@brunel.ac.uk`> and <`www.brunel.ac.uk/~csstgdm`>, respectively.

TILLAL ELDABI is a Researcher and Secretary to the Centre for Applied Simulation Modelling and a part-time Lecturer at the Department of Information Systems and Computing, all at Brunel University, UK. He received a B.Sc. in Econometrics and Social Statistics from the University of Khartoum and a M.Sc. in Simulation Modelling from Brunel University. His doctoral research is in aspects of healthcare management and the intervention of simulation. His main research concentrated on the economy of healthcare delivery. He is looking to exploit the means of simulation on the wider healthcare system management to assist in problem understanding. Dr. Eldabi's email and web addresses are <`tillal.eldabi@brunel.ac.uk`> and <`www.brunel.ac.uk/~cssrtte`>, respectively.

RAY J. PAUL is a Professor of Simulation Modelling, Director of the Centre of Applied Simulation Modelling, the creator of the Centre of Health Informatics and Computing, and the Dean of the Faculty of Technology and Information Systems, all at Brunel University. Professor Paul has published widely, in books, journals and conferences, many in the area of the simulation modelling and information systems development. He has acted as a consultant for a variety of United Kingdom government departments, software companies, and commercial companies in the oil industries. Professor Paul is the co-editor-in-chief of the European Journal of Information Systems and he is an editor of the Springer-Verlag Practitioners book series. His research interests are in methods of automating the process of modelling, and the general applicability of such methods and their extensions to the wider arena of information systems. He is currently working on aspects of simulation in the social sciences, in particular health management. Professor Paul's email and web addresses are <ray.paul@brunel.ac.uk> and <www.brunel.ac.uk/~csstrjp>, respectively.