Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

GENETIC ALGORITHMS IN OPTIMIZING SIMULATED SYSTEMS

George Tompkins
Farhad Azadivar

Department of Industrial and Manufacturing Systems Engineering
Kansas State University
Manhattan, Kansas 66506, U.S.A.

ABSTRACT

Advances have been made in optimizing quantitative variables within a simulation model, and many methodologies now exist for this purpose. However, many of the design decisions which confront a system's users involve policy alternatives. Often, variables used to represent these alternatives are not only discrete, but qualitative.

This work seeks to develop a simulation-optimization methodology which can operate on qualitative variables. The proposed approach is to link a genetic algorithm with an object-oriented simulation model generator. The system designs recommended by the genetic algorithm are converted to simulation models and executed. The results then guide the genetic algorithm in its selection of future designs.

A simulation model generator for a class of manufacturing systems, and a genetic algorithm which can interface with the generator have been developed. The methodology has shown positive results.

1 INTRODUCTION

In general, simulation-optimization procedures require the variables in question to be quantitative; the number of machines of a certain type to employ, or the number of operators assigned to a machine cell. The values of the decision variables must be fully ordered. This must be true if there is to be any concept of direction-of-improvement. If the MACHINE_ RESOURCE decision variable has a value of 3 and the response is better than when the value was 2, it appears that increasing this variable is good. There is no question of what values of the variable are larger than 3; direction is defined. In other words, the decision space can be defined by means of an n-dimensional geometric space.

When dealing with qualitative variables, techniques that require ordered values fail. Assume there is a decision variable called QUEUEING_ DISCIPLINE in the model. This variable determines how parts waiting for machining will be ordered, and it may take one of three values; LIFO (last-in-last-out), FIFO (first-in-first-out), or SPF (shortest-processing-time-first). If the current value of QUEUEING_DISCIPLINE is FIFO, and it is desired to change it to a larger value what should the new value be? This question has no answer because the values FIFO, LIFO, SPF have no order. Thus quantitative search methods cannot deal with them.

Finally, when dealing with quantitative variables the framework of the system is left unchanged, only the levels of various resources available to the system are changed. Qualitative variables allow for changes in the physical and logical structure of the system.

A simulation-optimization technique for qualitative variables must require only the system response, must cope with stochastic responses, must deal with discrete variables, and not assume any geometric relationship between the discrete points in the search space. In addition, because of the structural changes in the system, a unique simulation model is required for each design to be tested. Thus a method for automatically creating the simulation models is required. A simulation-optimization technique would be of little practical use if the user had to hand-code a simulation model at each iteration.

1.1 Existing Methods for Optimization of Qualitative Variables

The current methods available for optimizing qualitative variables include complete enumeration, knowledge-based systems, and random search.

The simplest of these, complete enumeration, is also the most limited. As the number of variables and their levels increases the number of points in the solution space increases exponentially. This method is completely unusable for all but the smallest problems.

Knowledge-based systems seek to embody a modeler's diagnostic abilities in a rule-base (Prakash 1991) (Chong, Sen, and de Souza 1994). Given the results of a simulation run the expert system can diagnose the causes of poor performance and recommend changes in the system to eliminate them. Prakash (1991) developed a generalized rule base for diagnosis, and presented an implementation of the system which can recommend capacity changes for a system. The simulation models are constructed in SIMAN, and all changes are made to the experiment file while the structure of the model is left unchanged.

Random sampling is, perhaps, the most often used technique for this type of optimization. The procedure is quite simple, and one can even determine the sample size required for a fixed degree of certainty in the result (Wilde 1964). However, while information about the surface is generated as the sampling proceeds, the procedure makes no use of this information.

Genetic algorithms attempt to make use of the information generated by an initial random sample to guide the selection of future samples. The procedure's name stems from the fact the it is a simplified version of natural evolution. The next section provides an overview of how genetic algorithms make use of the information developed in previous samples.

2 GENETIC ALGORITHMS

Genetic algorithms (GA's) are a family of randomized search procedures developed by John Holland (1975). There are many variations, but this explanation is limited to a basic GA. The family takes its name from its analogy to biological evolution. In biology it is believed an organism's structure, and thus much of its ability to survive in its environment, are largely determined by its DNA. This ability to survive is termed an organism's "fitness". In sexual reproduction, offspring are a combination of both parents' DNA. Thus the offspring inherit traits from both parents and, because of the recombination process, may exhibit traits not evidenced by either parent. These new traits, or the combination of existing traits, may increase an offspring's fitness. Such highly fit individuals will survive more frequently than other individuals in the population, and have the opportunity to pass the traits which helped them survive to their offspring. Thus, over time, the average fitness of the population will improve. Note that evolution is not concerned with individuals, but the trend of the population as a whole.

GA's are a computer simulation of a simplified and idealized evolution. DNA is represented as a string where each position in the string may take on one of a finite set of values. In this work each position in the string represents a variable from the user's system. The fitness of the organism is determined by a fitness (or objective) function; the function decodes the string and returns a real scalar value greater than 0. The larger the value the more fit the individual is. A group of strings taken together forms a population.

2.1 GA Operators

A GA has 3 basic operators; selection, crossover, and mutation. The selection operator embodies survival of the fittest. Individuals with higher fitness evaluations are more likely, but not guaranteed, to survive. Likewise, some poorly fit individuals survive in spite of their low fitness value. In short, the selection operator uses the fitness information to adjust the survival probability of each member of the population. These probabilities are then used to randomly select survivors.

The crossover and mutation operators act on the "DNA" of the survivors. Crossover models the combining of parent's DNA to form offspring. The crossover operator selects 2 survivors to act as parents. A position in the strings is chosen randomly, the strings are broken at this position, and the fragments are recombined to create 2 new offsprings. The mutation operator simply chooses a random position in a string, and changes the value to a new randomly selected value. In general, the selection operator selects individuals from the current generation of the population who will survive. These survivors undergo crossover and mutation and their offspring form the next generation. As this process repeats the average fitness of the population increases. The selection and crossover operators serve to direct the GA's sampling toward fruitful regions of the sampling space.

2.2 Motivation for Use of GA

GA's do not attempt to determine a direction of improving response as do other simulation-optimization techniques. Because of this the GA is able to operate on qualitative variables. GA's have also demonstrated an ability to operate with noisy observations, such as will be generated by a simulation (Goldberg 1989).

The GA has been developed and implemented in the MODSIM II language environment. This afforded two advantages; all details of the implementation would be known and open to change as the research progresses, and it would allow the simulation model generator and GA to be implemented in the same language. This ensures easy data transfer between the two components. The results of all simulation runs are stored so that once a configuration is evaluated it need never be evaluated again. The number of simulation evaluations, the population size, and the selection and crossover methods to use are controlled by the user and stored in data files.

3 SIMULATION MODEL GENERATOR

The model generator is made possible because of object-oriented nature of MODSIM II. The following sections highlight the assumptions which have guided the generator's development, and its operation.

3.1 Assumptions

The model generator is able to create models for job shop systems. It assumes there are a finite number of unique operations which can be performed. It allows multiple part types, where each part type may use a different subset of the available operations. It allows alternative equipment types for each operation, alternative routes for each part type, and alternative layouts for equipment.

Any system designed for an actual user is subject to the constraints of the user's budget. Therefore, this generator first extracts from the user what potential resources are currently available to it, or what the user is willing to make available.

3.2 Model Components

Each model has 4 components: the part object, the generator object, the processor objects, and a repeater object. For each part type produced by the manufacturing system there is a corresponding part object. There also exists a generator for each part type, it creates instances of its part type at random intervals and sends them to the processors. The processor objects receive part objects, place them in a queue, and eventually delay them for a specified time representing the processing time. After processing, a part object is sent to the next processor in its route. The final object, the repeater, handles the transportation of the part objects and simplifies the exchange of messages between the generator and processor objects.

3.3 Operation of Model Generator

The model generator is a set of data bases whose information is brought together by a central object called the SystemModelObject (Figure 1). The SystemModelObject is responsible for coordinating and constructing a simulation model.

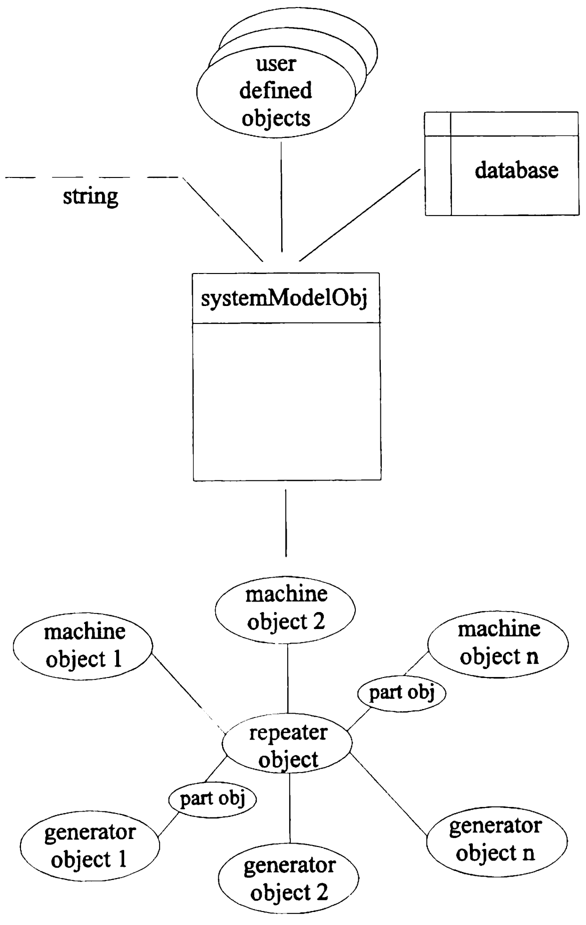

Figure 1: Model Generator

Based on the information passed to SystemModelObj from the GA the number of operations and part types in the system are determined and the required processor and generator objects are created. Each object is customized with information from the appropriate data base. For example, a processor has 3 characteristics which identify it: mean-time-to-failure, mean-time-to-repair, and queuing discipline. As the SystemModelObject creates an instance of a processor object it retrieves the appropriate values for these characteristics from the database and inserts them in the instance. This process is repeated for the generator objects. When the 4 elements (part objects, generator objects, processor objects, and repeater object) of the model are present, the SystemModelObject schedules the necessary initial events and starts the simulation. Upon completion of a run, the statistics in SystemModelObject are retrieved by the ReportObject, which calculates the response, and returns it.

4 TESTING OF PROPOSED METHODOLOGY

The preceding sections have presented the proposed methodology and the components necessary for its implementation. This section presents testing with 3 problems and compares the results with those generated by simple random sampling. The test cases consist of a small problem with 3888 alternatives, a large problem with 1 billion alternatives, and a very large problem with 13 billion alternatives.

4.1 Test Problems

The small test problem is comprised of 4 operations and 4 part types. There are 3 possible types of equipment which may be used to perform each operation. In addition, there are 2 alternative process plans, or routes, for each part type. Finally, the physical equipment may be arranged in 3 possible layouts on the shop floor. Figure 2 shows a possible configuration for the system. The goal is to find the combination of equipment types, routes, and layouts which minimize the work-in-process. The other problems are larger versions of this same structure.

The medium problem also has 4 operation and 4 part types, but there are 10 alternatives for each of these variables. There are 10 alternative layouts as well, for a total of 9 variable and 10^9 system configurations. In the large problem there are 6 operations and 6 part types with 6 alternatives each (and 6 layouts). It has a total of 6^{13} possible configurations.

The data required by the model generator was selected at random from various probability distributions. To ensure these randomly generated problems contained a known solution, the random routings and process times were modified to create a unique optimum in each problem.

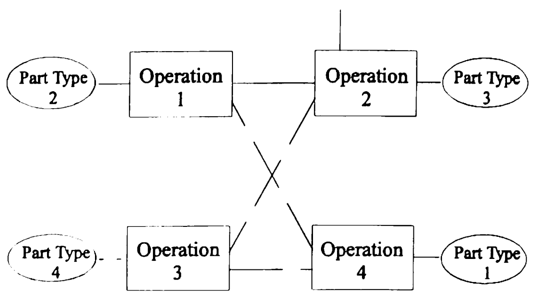

Figure 2: A Possible Configuration for Problem 1

4.2 Empirical Study

The goal of this study is to provide empirical evidence that the proposed methodology is effective in finding good solutions for this class of problems. A comparison of the results with those of random sampling is also presented. The objective function used for this study is work-in-process, i.e. the sum of the average queue lengths across all operations. The goal of the optimization is minimization.

To obtain a more reliable picture of a method's performance, seven repetitions (with unique random variate sequences in each repetition) were made. Thus the final measure of a method's performance on a problem is the average of the fitness values reported in each repetition.

The GA to be tested in the study uses roulette wheel selection, single point crossover, no mutation, and population seeding (Davis 1991). To create an initial population this GA selects a random sample of size 2N, where N is the population size. The strings are evaluated and the N best are used as the initial population.

Before proceeding, the GA was evaluated at several population sizes on each problem. This provided information on the trade off between population size and solution quality and allowed an effective population size to be selected (Table 1).

Because of the size difference between the smaller and larger problems (3888 alternatives compared to a billion or more), the larger populations were only used on the larger problems. The first number is the average performance measure and the second is the average number of alternatives evaluated. Based on

Table 1
Results for Various Population Sizes

Population Size	Problem 1	Problem 2	Problem 3
120			36.6 [2083]
90		20.3 [1193]	48.8 [1030]
70	9.4 [292]	22.9 [764]	46.8 [797]
50	14.4 [211]	26.6 [504]	51.6 [456]
30	18.4 [120]	46.5 [219]	58.1 [206]
20	25.8 [81]		

this information a population of size 70 was used for each problem. This population size strikes a balance between performance and cost.

The results of the new methodology are compared to those of a simple random sample. In random sampling a fixed number of alternatives, selected at random, are evaluated and the alternative with the best response is reported as the solution. The sample size for the procedure was the same used for the GA. This is calculated as the average number of evaluations used by the GA at the selected population sizes. Seven unique samples were drawn for each problem.

4.3 Results

It is evident from Table 2 that the GA is quite effective on these test problems. Compared to the random sampling procedure, the GA produced a 63.7, a 62.5, and a 19.7 percent improvement in the solutions for problems 1, 2, and 3 respectively.

Table 2
Comparison of GA and Random Sampling

	GA	Random Sample
Problem 1	9.4	25.9
Problem 2	20.3	54.2
Problem 3	48.8	60.8

The procedures may also be compared on the amount of improvement that takes place between an initial solution and the final solution. For the GA, the configuration in the initial population with the best objective function value will be the initial solution. In the case of random sampling the best objective function value from the first N values of the sample will be taken as the initial solution. Recall that N is the population size. The average percent difference in the objective function values of these solutions is shown in Table 3. This data indicates that given the same average number of evaluations, the GA consistently produces greater average improvement than random sampling. It also shows that this average improvement decreases as the problem size increases. This is most likely due to the fact that the population to problem-size ratio did not remain constant in this testing.

Table 3
% Change in Objective Function Value

	GA	Random Sample
Problem 1	71.4	22.9
Problem 2	67.3	15.4
Problem 3	37.7	18.1

4.4 Conclusions

This study has shown that the proposed methodology performed well on each of the 3 test problems, and outperformed random sampling by a significant margin. In addition, the data indicated that the GA was consistently capable of producing much greater improvements than random sampling could achieve.

Further, experience shows that with a given budget, in terms of simulation runs, the GA shows more advantage as the ratio of the number of affordable runs to the total number of alternative configurations increases. If only a few simulation runs can be made, random sampling might be the preferred method. But, if more runs can be used, the GA will tend to give better results than those produced by an equal number of runs using random sampling.

5 SUMMARY

The goal of this research was to develop a methodology which would allow qualitative, or policy, decisions to be optimized in a manufacturing system using simulation-optimization. Dealing with qualitative variables means none of the usual simulation-optimization methodologies based on the assumption of an n-dimensional geometric solution space can be used.

The proposed methodology uses a genetic algorithm coupled with an automatic simulation-model generator to search through the many alternative policy combinations. This approach has been implemented in the language Modsim II. Testing on 3 problems indicated the methodology was effective and it offered significant gains over random sampling.

This research represents a step toward the generalization of simulation-optimization. Simulation-optimization is not limited to quantitative variables or the optimization of a given system. It can be used to help determine the design of the system and its operational policies.

ACKNOWLEDGEMENT

This research was funded by a grant from the Advanced Manufacturing Institute of Kansas State University.

REFERENCES

Chong, C., A. Sen, and R. de Souza. 1994. ISE - An Interactive Knowledge-Based Simulation Interface for Manufacturing. *Proc. of the Conf. New Directions in Simulation for Manufacturing and Communications*, 287-293.

Davis, L. 1991. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, NY.

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

Holland, J. 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

Prakash, S. 1991. Goal Directed Simulation Environment for Discrete Part Manufacturing Systems. *Ph.D. Dissertation*, Department of Industrial Engineering, Texas A&M University, College Station, TX.

Wilde, D. 1964. *Optimum Seeking Methods*, Prentice-Hall, Englewood Cliffs.

AUTHOR BIOGRAPHIES

GEORGE TOMPKINS received his Ph.D. degree in industrial engineering from Kansas State University in 1995. He has a B.S. and an M.S. degree in industrial engineering from Kansas State University also. His research interests are in computer simulation, optimization and applications of OR and statistics to manufacturing systems.

FARHAD AZADIVAR is a professor in the Department of Industrial and Manufacturing Systems Engineering at Kansas State University. He is also the Director of Kansas State University's Advanced Manufacturing Institute. He received his Ph.D. in 1980 from Purdue University. He has a B.S. degree in mechanical engineering and an M.S. degree in systems engineering. His areas of interest are in computer simulation, simulation optimization, and modeling and optimization of manufacturing systems.