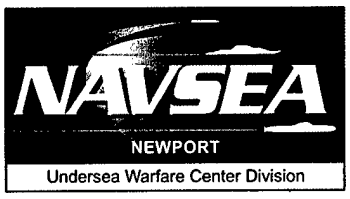
NUWC-NPT Technical Report 11,437
1 July 2003

Simulation Optimization by Genetic Search: A Comprehensive Study with Applications to Production Management

James M. Yunker
NUWC Division Newport

Jeffrey D. Tew
General Motors Global R&D Center



Naval Undersea Warfare Center Division
Newport, Rhode Island

Approved for public release; distribution is unlimited.

20040317 196

PREFACE

This report was prepared under Code 22 internal funding.

The technical reviewer for this report was David J. Ferkinhoff (Code 2213).

Reviewed and Approved: 1 July 2003

Philip A. La Brecque
Head, Combat Systems Department

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 July 2003	

4. TITLE AND SUBTITLE

Simulation Optimization by Genetic Search: A Comprehensive Study with Applications to Production Management

5. FUNDING NUMBERS

6. AUTHOR(S)

James M. Junker
Jeffrey D. Tew

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Undersea Warfare Center Division
1176 Howell Street
Newport, RI 02841-1708

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 11,437

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In this report, a relatively new simulation optimization technique, the genetic search, is compared to two more established simulation techniques—the pattern search and the response surface methodology search. The pattern search uses the Hooke-Jeeves algorithm, and the response surface methodology search uses the computer code of Dennis Smith. The three algorithms are compared for both accuracy and stability. Accuracy is evaluated in terms of how close each algorithm comes to the optimum, the optimum having been previously determined from exhaustive testing. Stability is evaluated using the variance of the response function determined from sample searches—the lower the variance, the more stable the response. The examples tested are an inventory system with integer decision variables, a university time-sharing computer system with two real decision variables, and a job-shop with five decision variables (the number of machines located at each station). The response of interest for each system is the cost of operating the system. The genetic algorithm is shown to be a superior optimization method compared to the two other search techniques.

14. SUBJECT TERMS

Simulation
Genetic Algorithms
Simulation Optimization
Artificial Intelligence

15. NUMBER OF PAGES
40

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure **Page**

LIST OF TABLES

Table **Page**

SIMULATION OPTIMIZATION BY GENETIC SEARCH: A COMPREHENSIVE STUDY WITH APPLICATIONS TO PRODUCTION MANAGEMENT

1. INTRODUCTION

The field of mathematical modeling is divided into two areas: (1) generative techniques such as linear programming; and (2) evaluative techniques such as simulation.[1] A generative technique shows the optimum given specific parameters and constraints. However, its application requires certain (sometimes unrealistic) assumptions. The solutions derived from generative techniques like linear programming are a set of recommended operating parameters for a system that is now optimized. Generative techniques remove the analyst from the decision-making process. In addition, with such techniques it is difficult to get a feel for the system, and the methods suffer from being a "black box".[1] Evaluative techniques like simulation are quite different. They only determine the outcome of an operation given certain variables (or factors) that are put into the model. Simulation models are often more reflective of the actual systems they are modeling than are generative techniques. However, very few optimization algorithms allow for effective application to simulation models, and little is known of their relative performance. Thus, a lot of simulation optimization in practice is done by undirected, trial-and-error comparisons, leading to suboptimal results.

The objective of the research described in this report was to develop meaningful comparisons of leading candidate simulation optimization algorithms through careful evaluation and optimization of simulations using genetic algorithms (GAs). Specifically, this study compares the search techniques of the relatively new and promising GAs with those of pattern search and response surface methodologies, the traditional search techniques in simulation optimization.

For each example considered in this report, the criterion for comparison is proximity to a known optimum. The three search techniques were benchmarked on three example problems common in simulation, with each optimum previously determined through exhaustive trial searches. It is important to emphasize that the uniqueness of this study is the linking of GAs to simulation optimization. The result is a very practical method of simulation model optimization.

Since the cost of central processing unit (CPU) time is of little concern in today's personal computer (PC) environment, CPU time was not a test criterion. The attainment of sample points in a simulation run is usually a time-consuming process, and a genetic search is no exception. Typically, a genetic search will consume more CPU time than a pattern or a response surface search. However, a genetic search attains something for its additional simulation time, a more desirable optimum.

This report discusses three search techniques, starting with the non-derivative-based search, followed by the derivative-based search, and the genetic search. The three examples of the methodologies are then discussed with an emphasis on how each search technique's algorithm was adapted to that example. Following each example is a discussion of the results using the three search methods.

2. NON-DERIVATIVE-BASED SEARCH

Historically, there have been two basic approaches to the development techniques of simulation optimization. The first has been to employ search techniques that do not need mathematical information, such as the derivative of the gradient of the response surface information. Examples of techniques that do not require such information are the Hooke-Jeeves pattern search,[2] Rosenbrock's[3] method of rotating coordinates, and the simplex method of Nelder and Mead.[4] Although these methods are unique, they share the traits of a derivative-free search.

2.1 HOOKE-JEEVES ALGORITHM

The Hooke-Jeeves[2] pattern search method was selected as the non-derivative method for the comparison test. The main reason for its selection is the extensive literature available on previous attempts to optimize using this method.[5] The Hooke-Jeeves method is also a good representative of a non-derivative algorithm for comparison to the genetic search algorithm. The Hooke-Jeeves algorithm has been combined with GAs to form a hybrid algorithm designed to improve the search over the use of either algorithm alone.[6]

Wilde and Beightler[7] provide the following analysis of the method. Consider a univariate function with independent variables x_i ($i = 1, 2, ..., N$) to be maximized. The pattern is established during the execution of the search itself since the algorithm consists of a series of conditional statements that determine future search directions and magnitudes. This is done by stepping δ_i for variable x_i and checking whether or not the function has improved. This new step is designated $\mathbf{t}_{ii}$ ($t = 1, 2, ..., N$), which is a temporary location that is tested to determine if the optimization move was successful. To begin the search, choose a base point $\mathbf{b}_1$ and a step size δ_i. Let δ be a vector with its ith component being δ_i. All other components are zero. All criteria are measured at b_1 (i.e., in simulation optimization, the simulation is run with $\boldsymbol{b}_1$'s decision variables input and then again at $\mathbf{b}_1 + \delta_1$, with decision variables $\mathbf{b}_1 + \delta_1$ input). If at this new point the function is greater than at $\mathbf{b}_1$, $\mathbf{b}_1 + \delta_1$ is the temporary head $\mathbf{t}_{11}$ and there was a successful step. If at $\mathbf{b}_1 + \delta_1$ the function is less than at $\mathbf{b}_1$, try $\mathbf{b}_1 - \delta_1$, and if the function is greater than at $\mathbf{b}_1$, then make $\mathbf{b}_1 - \delta_1$ the temporary head $\mathbf{t}_{11}$ and consider this a successful step. If neither point gives a greater function value than $\mathbf{b}_1$, then the optimum is kept for that independent variable at $\mathbf{b}_1$ and there are no successful steps. This analysis can be stated formally as

$$\mathbf{t}_{11} = \begin{cases} \mathbf{b}_1 + \boldsymbol{\delta}_1 & \text{if } y(\mathbf{b}_1 + \boldsymbol{\delta}_1) > y(\mathbf{b}_1) \\ \mathbf{b}_1 - \boldsymbol{\delta}_1 & \text{if } y(\mathbf{b}_1 - \boldsymbol{\delta}_1) > y(\mathbf{b}_1) > y(\mathbf{b}_1 + \boldsymbol{\delta}_1) \\ \mathbf{b}_1 & \text{if } y(\mathbf{b}_1) > \max[y(\mathbf{b}_1 + \boldsymbol{\delta}_1), y(\mathbf{b}_1 - \boldsymbol{\delta}_1)] \end{cases}, \tag{1}$$

where $\mathbf{t}_{11}$, with its double subscript, denotes the first pattern and the first variable x_1 is being perturbed. Note that $\mathbf{t}_{10}$ is $\mathbf{b}_1$. Adopting the convention, then $\mathbf{t}_{10} = \mathbf{b}_1$. This is shown in figure 1.

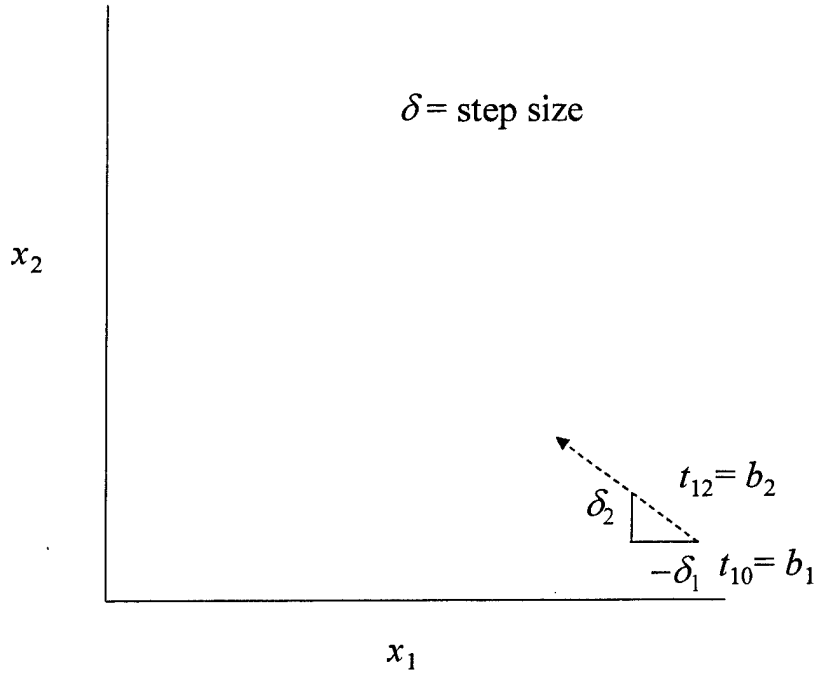


Figure 1. Hooke-Jeeves Algorithm Example 1

The next perturbation is for x_2 and is the same as that shown in equation (1) except that the perturbation begins around $\mathbf{t}_{11}$, not $\mathbf{b}_1$. Putting these instructions into a formula,

$$\mathbf{t}_{11} = \begin{cases} \mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1 & \text{if } y(\mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1) > y(\mathbf{t}_{1,j-1}) \\ \mathbf{t}_{1,j-1} - \boldsymbol{\delta}_1 & \text{if } y(\mathbf{t}_{1,j-1} - \boldsymbol{\delta}_1) > y(\mathbf{t}_{1,j-1}) > y(\mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1) \\ \mathbf{t}_{1,j-1} & \text{if } y(\mathbf{t}_{1,j-1}) > \max[y(\mathbf{t}_{1,j-1} + \boldsymbol{\delta}_1), y(\mathbf{t}_{1,j-1} - \boldsymbol{\delta}_1)] \end{cases}, \tag{2}$$

which explains how the j^{th} temporary head is obtained from the preceding one $\mathbf{t}_{1,j-1}$. When all the variables have been perturbed (and there has been at least one successful step), then the last temporary head point $\mathbf{t}_{1N}$ is the second base point $\mathbf{b}_2$, or formally stated $\mathbf{t}_{1N} = \mathbf{b}_2$. This is shown in figure 2.

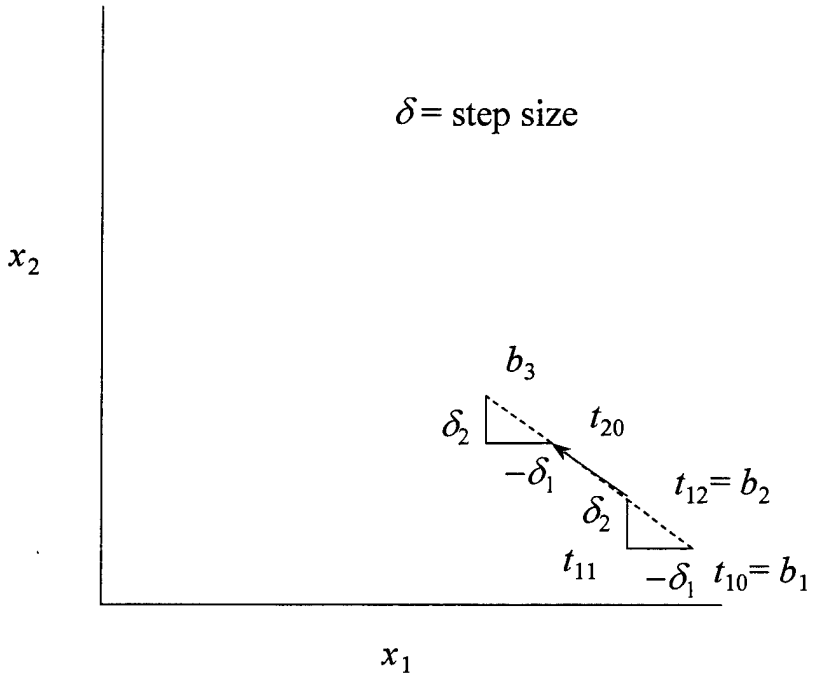


Figure 2. Hooke-Jeeves Algorithm Example 2

The original base point $\mathbf{b}_1$ and the newly determined base point $\mathbf{b}_2$ establish the first pattern move. Looking at figure 1 it might be helpful to think of a pattern as an arrow with its base at one end and its origin at the other end. If a similar path of exploration were conducted from $\mathbf{b}_2$, the results would probably be the same, so the search can be expanded by the equation

$$\mathbf{t}_{20} = \mathbf{b}_1 + \gamma * (\mathbf{b}_2 - \mathbf{b}_1),$$

where the expansion factor is γ, and $\mathbf{t}_{20}$ shows that while the second pattern has begun it has yet to perturb the variables. If all the points are moved to $\gamma * (\mathbf{b}_2 - \mathbf{b}_1)$, and each variable is perturbed as before, the step sizes are larger after being multiplied by the expansion factor γ (i.e., if the expansion factor $\gamma = 2$, then the new steps in the pattern move are twice as large as before). When all of the steps are completed in the sequence and at least one step improves the function, then

$$\mathbf{t}_{30} = \mathbf{b}_1 + \gamma * (\mathbf{b}_3 - \mathbf{b}_1).$$

This sequence continues as long as at least one perturbation out of the N independent variable perturbations improves the function under consideration (see figure 3).

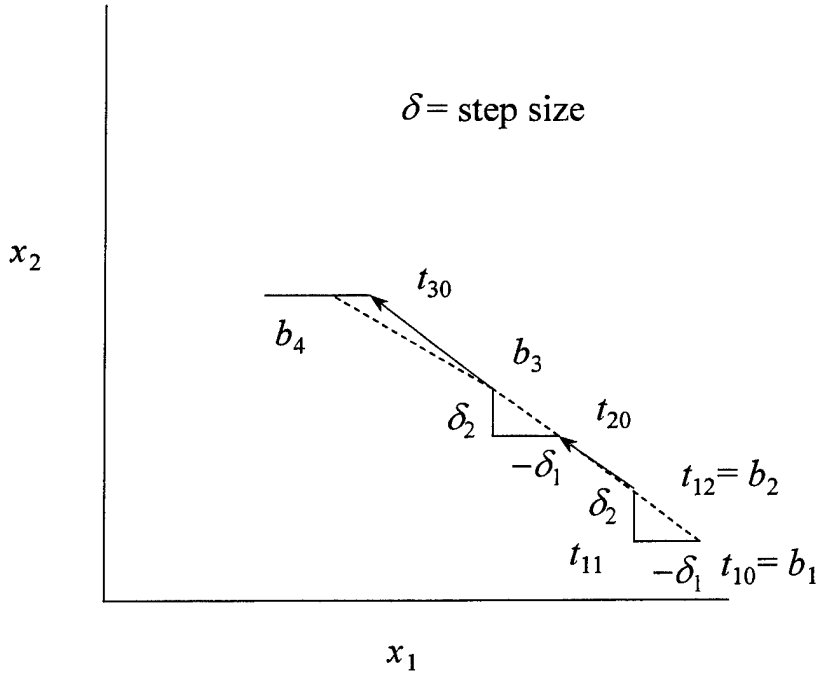


Figure 3. Hooke-Jeeves Algorithm Example 3

When there are no successful moves out of the N moves of the independent variables, the step size δ is reduced by dividing the current steps by the expansion factor. The search concludes when a sequence of variable perturbations is carried out and none are successful, and the algorithm is at the minimum step size. The optimum of the univariate function x_i ($I = 1, 2, \ldots, N$) is assumed to lie at the last base point attained in the algorithm. For example, if the algorithm is on perturbation $\mathbf{t}_{40}$ and no moves are successful, the base point $\mathbf{b}_4$ is then taken to be the optimum of the function.

As an illustration, consider figure 4, which shows the steps just discussed. The broken line shows the optimal path delineated, the solid line shows the test steps or the exploration state (horizontal or vertical), and the solid line with an arrowhead is the pattern move.

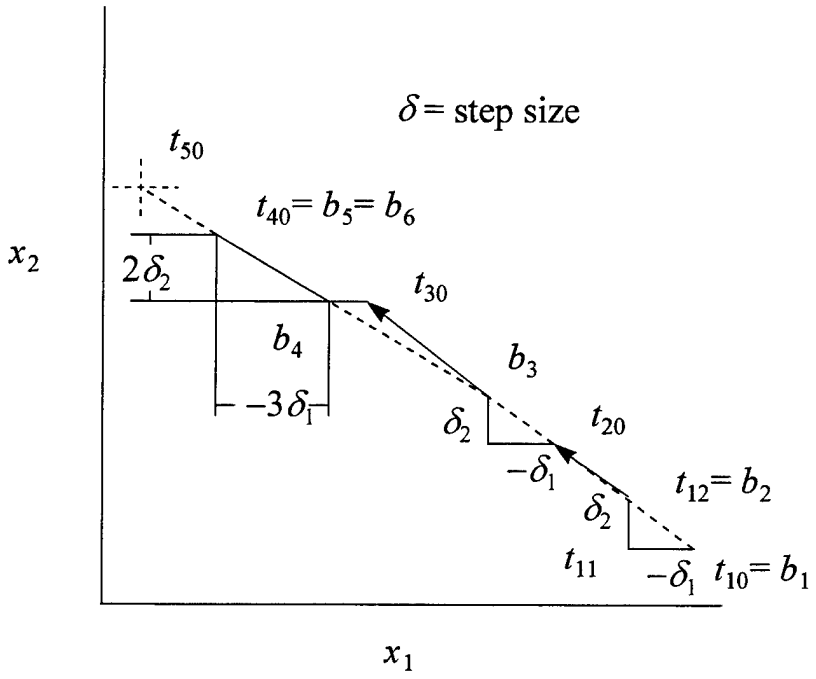


Figure 4. Hooke-Jeeves Algorithm Example 4

The first two sets of solid lines at right angles to each other in figure 4 are steps for variables x_1 and x_2, respectively (for this case $N = 2$). Equation (2) holds when calculating $\mathbf{t_{11}}$ and, similarly, when calculating $\mathbf{t_{21}}$. Also notice in the third test (attaining $\mathbf{t_{40}}$) that there is no improvement in stepping in the x_2 direction, only in stepping in the x_1 direction; hence, the move is only horizontal. (In the first two sets, the move is horizontal and vertical.) The pattern search thus veers to the left. A Hooke-Jeeves pattern search works well when it finds a ridge on the surface and follows that ridge up the response surface when maximizing the model or down the response surface when minimizing the model.

2.2 PEGDEN-GATELY ALGORITHM

This section explains the function and use of the Pegden-Gately algorithm as applied to simulation optimization.

Because of the difficulty in obtaining simulation points, Pegden and Gately have modified the algorithm named for them.[5] When a set of N pattern moves fail to find at least one improvement, the algorithm sets the step sizes to a user-defined minimum, rather than reducing the steps by a pre-selected expansion factor in graduated reductions.

Pegden and Gately use two other modifications to the Hooke-Jeeves search technique.[5] The first modification allows the algorithm to save old points and their values; then, if the pattern search returns to a previously evaluated point (a finite possibility), the algorithm uses that previous point and makes no simulation run. This creates some programming overhead, but the modification is worth incorporating into the simulation program because of the potential number

of simulation runs it saves. The method was also used by Haupt as a time saver when the cost function is expensive.[8] The second modification also incurs minimal overhead cost, but it is quite effective in simulation execution: in a sequence of successful variable perturbations, the direction of the successful search is retained and used in the subsequent search.

3. DERIVATIVE-BASED SEARCH

A derivative-based search employs more mathematical modeling information than a non-derivative search. It can find an optimum by employing traditional mathematical techniques. Each step in this type of search has two purposes: (1) to attain an improved value of the objective function and (2) to give information useful for locating future points that have more desirable values. The research in derivative optimization is extensive and thorough. For this discussion, the response surface methodology (RSM) was chosen to represent the derivative-based search technique.

In RSM, the algorithm works with a function of several variables, with the exact form of the function unknown. This can be written in more formal terms as

$$y = f(x_1, x_2, \ldots, x_N), \tag{3}$$

which is in an $N + 1$ dimensional space. The difference between the number of variables and the number of independent equations that describe the hyper plane is called the "degrees of freedom." In equation (3), there are $N + 1$ variables $(x_1, x_2, \ldots, x_N)$ linked by one equation.

In RSM, the algorithm is trying to approximate a surface. The optimum of the objective function depends on N independent variables $(x_1, x_2, \ldots, x_N)$, but that objective function is unknown. Experimentation determines the value of y for any set of values $(x_1, x_2, \ldots, x_N)$. Thus, the point in the $(x_1, x_2, \ldots, x_N)$ hyper plane corresponds to an experimental sample, and the point above it on the response surface, the value of y, represents the experimental outcome. The experimental points are confined to a bounded portion of the $(x_1, x_2, \ldots, x_N)$ hyper plane that is called the experimental region. Each experiment gives the elevation of the response surface above a new point in the experimental region. The objective is to climb as high as possible on the response surface, using historical information to guide the search toward the summit. This information is derived from the theoretical mathematics of optimization. For example, the first derivative of a function tells whether the function is increasing or decreasing and the second derivative can determine whether the search has hit a maximum, a minimum, or an inflection point. In simulation optimization, the goal is to find a proxy for the true response values, or a response surface, and use the proxy instead of the simulation runs in optimization studies. The objective function is analytically unobtainable and, hence, must be estimated from simulation data in the manner described below.

Assuming that the true response surface can be expressed as a Taylor series, it can be written as

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=1}^{p} \beta_{ij} x_i x_j + (\text{higher order terms}), \tag{4}$$

where y is the dependent variable, x_i represents the independent variables, β_i is the coefficient, and p is the number of independent variables. If first order or linear effects dominate, an estimate of the response at $(x_1, x_2, \ldots, x_N)$ is

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \,, \tag{5}$$

where b_i, obtained by least squares, is an estimate of β_i. This allows the estimation of the position of the optimum, which lies between experimental samples.

The assumptions a mathematical solution requires, such as continuity and existence of derivatives, may not hold in complex real world examples, however. Thus, a derivative-based solution may be suspect. Given these restrictions, a new methodology must be used to optimize a complicated system.

4. GENETIC ALGORITHMS

Nature is an efficient optimizer. It can find acceptable solutions to intractable optimization problems using efficient optimization mechanisms. Researchers have studied nature and applied its precepts to system optimization. They copied the mechanisms of a natural phenomenon and applied them to problems that seemed insolvable by traditional operations research methods. One concept based on this premise that has shown promise is the use of GAs.

GAs are probabilistic search-optimizing algorithms that do not require mathematical knowledge of the response surface of the system of interest. The method borrows the paradigms of genetic evolution—specifically, reproduction, crossover, and mutation—in the search for an optimum. Given a population at time t, called $P(t)$, a simple pseudo-code outline of a GA can be written as[9]

$t \leftarrow 0$;
initialize $P(t)$;
evaluate $P(t)$;
while (termination condition not satisfied) do;
begin;
$t \leftarrow t + 1$;
select $P(t)$;
recombine $P(t)$;
evaluate $P(t)$;
end.

Only binary strings (representing genes) were used in the GAs in this study. The string consists of the set $V = \{0, 1\}$.[10] Consider, a seven-bit string (a chromosome) X that can be symbolized as

$$X = x_1x_2x_3x_4x_5x_6x_7,$$

and that has one realization

$$X = 0111000.$$

Now, x_i represents a single binary feature or detector that may take on a value of 0 or 1. In keeping with the genetic analogy, x_i is called a gene.

A meaningful genetic search requires a population of strings. Consider a population of individual strings $X_j, j = 1, 2, ..., n$ contained in a population $P(t)$ at time (or generation) t. The interest is not in the strings alone, but in their similar templates or schema. A schema describes a subset of strings that have similarities at certain positions.[11] Schema are used to simplify the analysis of GAs.

Each of the three defined GA operators—reproduction, crossover, and mutation—has a respective parameter that must have an assigned value. They are: (1) population size; (2) crossover rate; and (3) mutation rate.

Various studies have determined that the following values can be appropriate:[9,12]

l) Population size — 30 to 200,

2) Crossover rate — 0.60 to 1.00, and

3). Mutation rate — 0.001 to 0.003.

The decision variables in a simulation optimization may not always be integers; they can also be real valued. In the case of real valued variables, the value must be converted from a parameter $x \in [0,2^l]$ to another interval $[U_{min},U_{max}]$. An appropriate formula (but certainly not the only one) to map to a different variable interval is[10]

$$\pi = \frac{U_{min} - U_{max}}{2^l - 1}, \tag{6}$$

where U_{max} is the specified maximum that the variable can be, U_{min} is the specified minimum that the variable can be, and l is the length of the chromosome or number of bits.

Theoretically, GAs can be very efficient search mechanisms. To reinforce the point, several examples are discussed below that test the strength of a GA. Both the decision variables and the output variables are either integer or real (following several distributions). In each of the three examples, the cost function is minimized. Finally, the basic concepts of a GA are adapted to a problem, and existing solution methodologies are used whenever possible.

5. EXAMPLE 1 — INVENTORY SYSTEM

The first example to test the three search methodologies is an inventory system.[13,14] Each search methodology (pattern search, derivative-based search, and GAs) is applied with the results displayed graphically. This approach enables direct comparison of the search capabilities of each algorithm.

A company sells a single product and wants to know how many items to have on hand in the next n months. The time between demands (when a customer places an order) are independent identically distributed, or *i.i.d.*, exponential random variables with a mean of 0.1 month. The size of the demands D is an *i.i.d.* random variable with distribution of

$$D = \begin{cases} 1 \; wp \frac{1}{6} \\ 2 \; wp \frac{1}{3} \\ 3 \; wp \frac{1}{3} \\ 4 \; wp \frac{1}{6} \end{cases}.$$

Inventory review is conducted at the start of each month. The formula for the average ordering cost (AOC) is

$$AOC = K + icZ, \tag{7}$$

where K is the setup cost ($32.00), ic is the incremental ordering cost per item ($3.00), and Z is the number of items ordered. The delivery lag is probabilistic with a uniform distribution of 0.5–1.0. The formula for the policy on ordering inventory is

$$Z = \begin{cases} S - I & \text{if } I < s \\ 0 & \text{if } I \geq s \end{cases}, \tag{8}$$

where I is the inventory level at the beginning of each month before orders are placed, S is the level of inventory to order up to, and s is the level of inventory that triggers the placement of an order. Demand is immediately satisfied if inventory is available; if inventory is not available, the excess demand is backlogged and satisfied by future deliveries. Each arrival satisfies backlogged demand first with the remainder going into inventory.

Besides the costs outlined above, there are holding costs and shortage costs. The average holding costs (AHC) can be calculated by

$$AHC = h \frac{\int_o^n I + (t) \mathrm{d}t}{n}, \tag{9}$$

where n is the number of months, $\int_o^n I+(t)dt$ is the time-average inventory, and h is the item cost per month ($1). The average shortage cost (ASC) can be calculated as

$$ASC = \pi \frac{\int_o^n I-(t)dt}{n}, \tag{10}$$

where n is the number of months, $\int_o^n I-(t)dt$ is the time-average number of items in backlog, and π is the item cost per month ($5). The total operating cost (TOC) function is

$$TOC = AOC + AHC + ASC. \tag{11}$$

The following discussion assumes that $I(0) = 60$ (initial inventory is 60 units) and there are no orders outstanding. The example is simulated for 10 years and the values of s and S, the decision variables, have been determined to attain the lowest operating costs of the inventory system.

5.1 EXAMPLE 1 RESULTS

Two decision variables, s and S, the reorder point and the order up to point, respectively, have the same range of values: a low of 10 and a high of 200, with s always being smaller than S. The decision variables are only integers, but equation (6) is used in the genetic search because of the ranges involved. A rounding subroutine was employed to round off the values to the nearest integer. The pattern search and the response surface search were initiated from a $U(12, 198)$ distribution for the 50 initial points. For all of the algorithms

$10 < s \leq 200$,

$10 < S \leq 200$, and

$s \leq S$.

So the cost function is equation (11) and the search space is s + S.

5.2 EXAMPLE 1 PATTERN SEARCH

The pattern search used an initial step size of 2, and an expansion factor of 2. The plot in figure 5 is of the initial points for an inventory system, s versus S, with the approximated optimum $s = 25$ and $S = 65$ (indicated by a square).

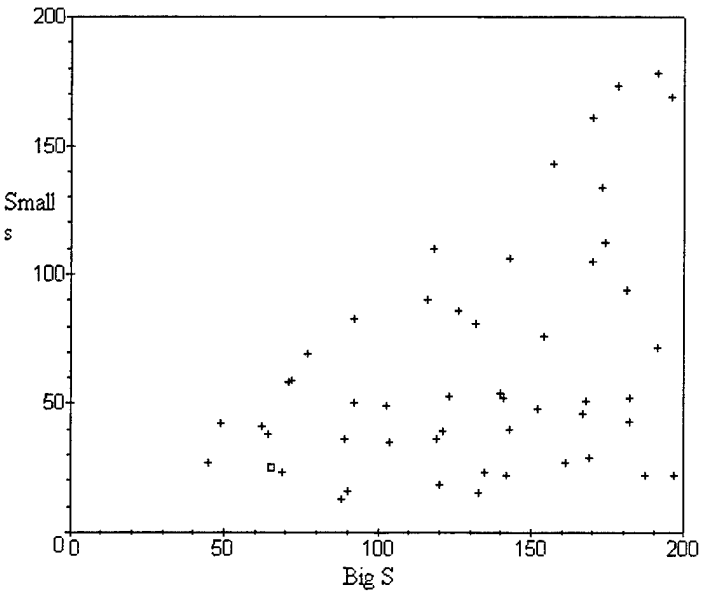


Figure 5. Initial Points for Inventory System Search (optimum designated by a square)

Figure 6 is a plot of the 50 simulation optimization results realized after the pattern search. The points are the same as the points in figure 5, except for one. The pattern search has a very high hurdle—either there is a statistically significant improvement in the function (a decline in costs) or that search path is not taken. As a result, this test was successful for only one point; it failed for all the others. The pattern search does not find an optimum quickly or easily.

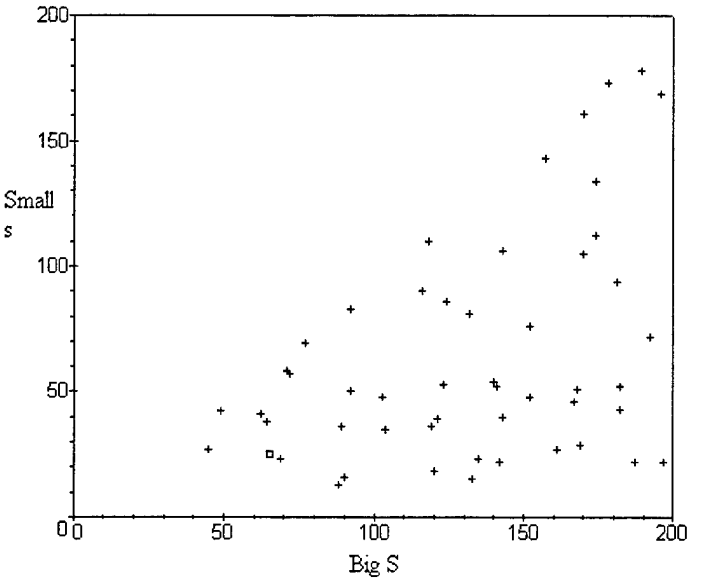


Figure 6. Final Points for Inventory System — Pattern Search (optimum designated by a square)

5.3 EXAMPLE 1 RESPONSE SURFACE SEARCH

A better search method resulted from the response surface methodology outlined by the computer code of Dennis Smith.[15] The initial points are identical to those in figure 5. Again, the step size for the response surface is 2 for both variables. The plot after the response surface search is shown in figure 7.

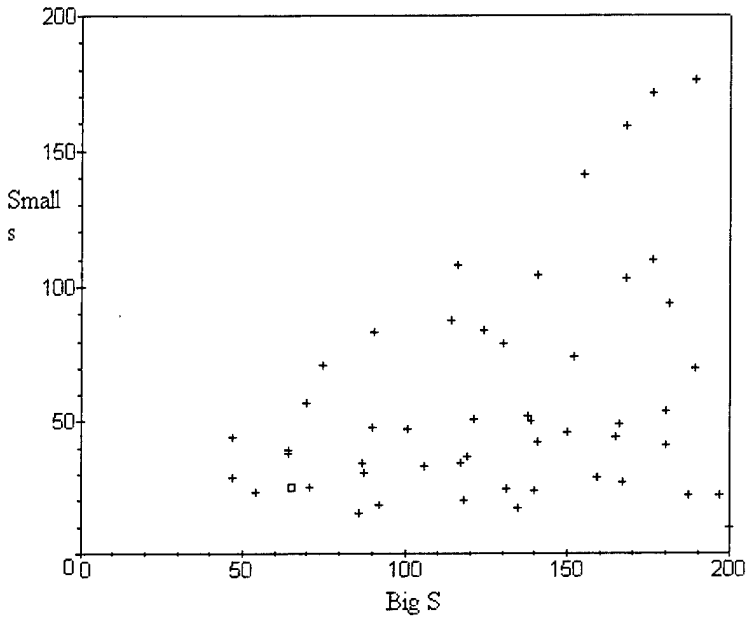


Figure 7. Final Points for Inventory System — Response Surface Search (optimum designated by a square)

5.4 EXAMPLE 1 GENETIC SEARCH

The genetic search method for example 1 used 50 searches, with each search consisting of a simple genetic search and each generation having a population of 30 individuals. Seven generations were used with each individual in a generation replicated three times to account for sampling error. Each sample point has variability that must be considered. To account for sampling error, the computer simulation was run three times with identical inputs to get a variance. The average of three samples accounts for any outlier.

The inputs for the decision variables used equation (6) with U_{min} of 10 and U_{max} of 200. An example chromosome is A_1 where

A_1 = 101110110101010101001010010101.

Each chromosome is 30 units long and each variable makes up 15 units of that string; thus l = 15 in equation (6). The plot in figure 8 shows the final points and the calculated optimum, indicated by a square, for the inventory system in figure 5. In each of the 50 runs of the genetic

search, the entire set of seven generations was scanned for an optimum, and that optimum was used. Hence, there are 50 points for the 50 runs. The data show a very tight pattern close to the calculated optimum. This is also the optimum reported by Law and Kelton,[16] and is clearly an improvement over the output for the pattern search and the response surface search.

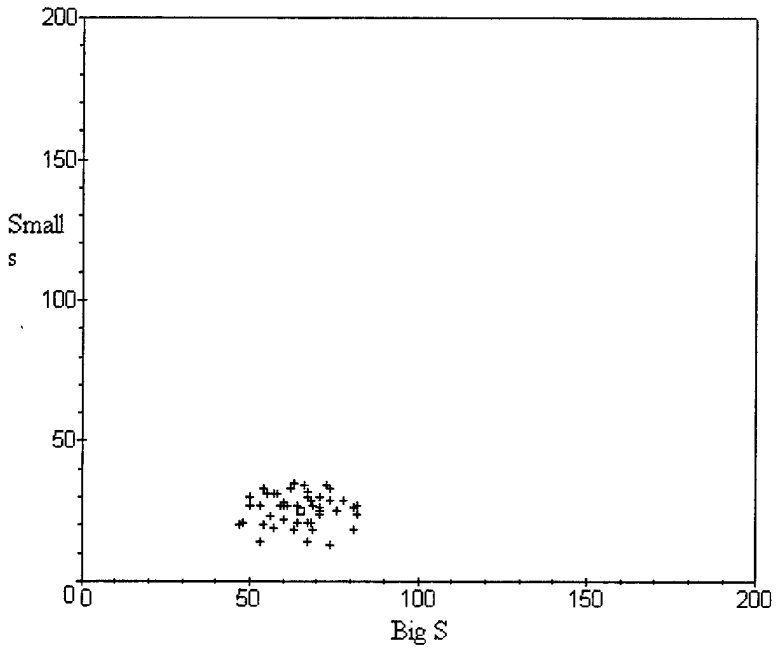


Figure 8. Final Points for Inventory System — Genetic Search (optimum designated by a square)

6. EXAMPLE 2 — UNIVERSITY TIME-SHARED COMPUTER SYSTEM

The second example is based on a university computer system.[13,17] In this section, a university committee plans a computer system consisting of 40 terminals and would like to know how to set the design parameters to minimize the average response time at a given terminal. Each student working at a terminal "thinks" for a time that is exponentially distributed with a mean of 2.5 seconds. The student then sends a command to the CPU of the mainframe that has a service time that is an exponentially distributed random variable with a mean of s seconds. The arriving jobs join a queue and are served in round-robin fashion with each job getting a maximum service quantum of q seconds. When the service time of a job s is less than or equal to q, the CPU spends time q plus an overhead τ processing the job, and then sends the job back to the terminal. If the service time is greater than q, the CPU spends time q plus τ processing the job, and then sends the uncompleted job back to the end of the queue, with its service time decreased by q seconds. This process is repeated until the service is completed.

The response time is the time that elapses from when the job is submitted to when it is finished being processed by the CPU. This is the response to be minimized. The decision variables are quantum of processing time q and overhead τ. The response function can be written as

$$cost = \overline{T}_{res} * 1000 + \overline{N}_{queue} * 1000 + \ldots$$

$$\left(\frac{(0.005 - ohd)}{0.005}\right)^2 * 1000 + \left(\frac{(0.50 - qntm)}{0.50}\right)^2 * 10{,}000 \cdot \qquad (12)$$

6.1 EXAMPLE 2 RESULTS

As noted above, the example has two decision variables: quantum and overhead. The quantum variable has a minimum value of 0.010 and a maximum value of 1.00, and the overhead has a minimum value of 0.001 and a maximum value of 0.01. The quantum variates were generated by a pseudo-random number generator with uniform distribution $U(0.012, 0.98)$, while the overhead variates by $U(0.0012, 0.098)$ to obtain the 50 initial points for the pattern search. The decision variables were real valued, so equation (6) was used when performing the genetic search. For all algorithms, the constraints $0.01 \leq quantum \leq 1.00$ and $0.001 \leq quantum \leq 0.01$ hold.

6.2 EXAMPLE 2 PATTERN SEARCH

The pattern search used an initial step size of 0.10 for quantum and 0.001 for overhead, and an expansion factor of 2 for both decision variables. The step size for the quantum variable is 0.02, which is also its minimum step size; the step size for overhead is 0.00002, which is also the minimum step size. Figure 9 is a plot of the initial 50 points, quantum versus overhead, with the optimum of quantum of 0.50 and overhead of 0.005 indicated by a square.

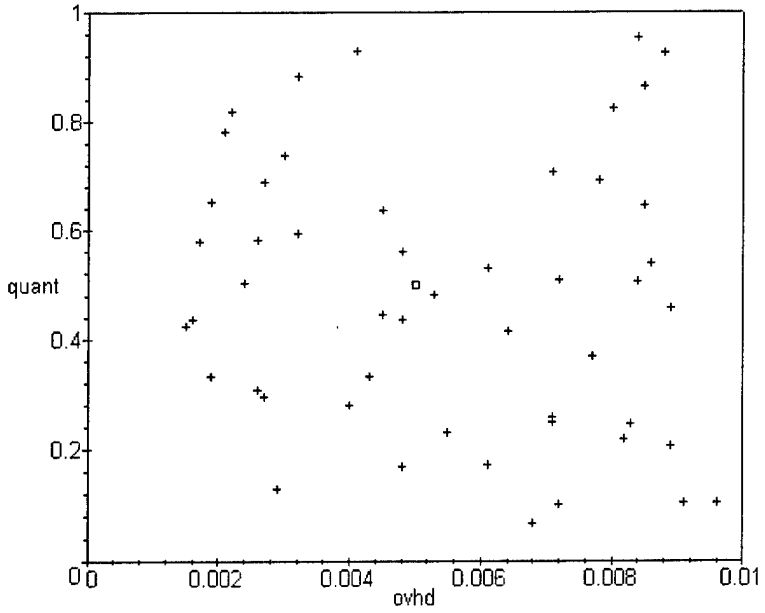


Figure 9. Initial Points for Computer System Search (optimum designated by a square)

The pattern search program was run on a SunSparc *IPX*. The plot of 50 simulation optimization results is shown in figure 10 for the pattern search with the true optimum indicated by a square. These points are the same as those in figure 9, which again indicates poor performance for this algorithm.

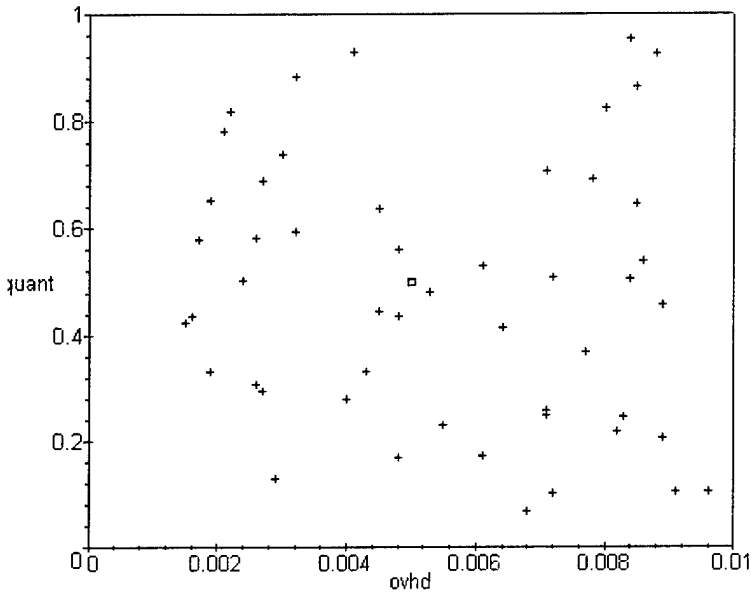


Figure 10. Final Points for Computer System — Pattern Search (optimum designated by a square)

6.3 EXAMPLE 2 RESPONSE SURFACE SEARCH

Using Smith's method,[15] the initial points for the response surface search are the same as in figure 9. The step size is 0.02 for the quantum decision variable and 0.0002 for the overhead decision variable. The response surface program, run on an IBM RS/6000, took about 2 hours. The plot of the final response surface search points is shown in figure 11. While this is an improvement compared to the Hooke-Jeeves pattern-search algorithm, the scattering is fairly loose.

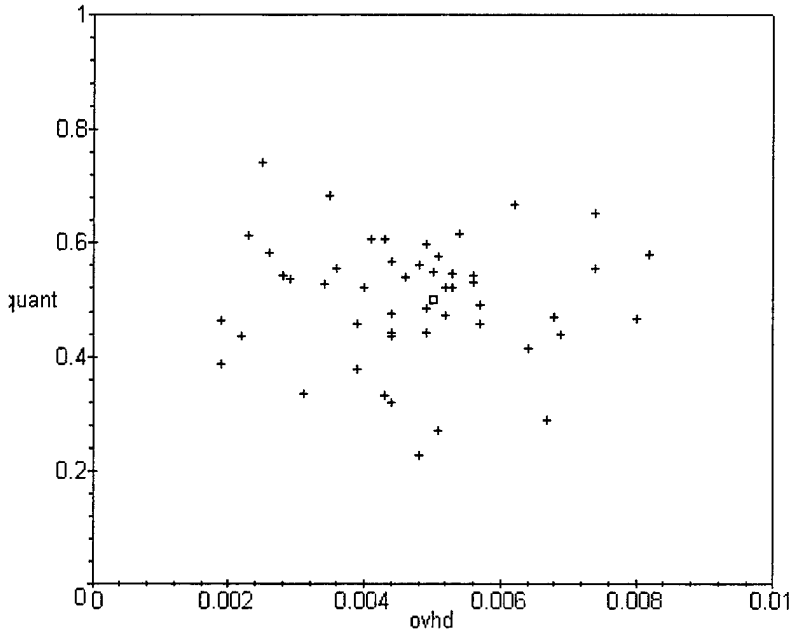


Figure 11. Final Points for Computer System — Response Surface Search (optimum designated by a square)

6.4 EXAMPLE 2 GENETIC SEARCH

The genetic search method employed 50 searches, each of which consisted of a simple genetic search with a population of 30 individuals per generation. There were seven generations with each individual in a generation replicated three times.

Equation (6) was used again since the variables were real valued. The chromosome was 30 units long with each variable taking up 15 units. The value of l was 15.

The genetic search program was run on a SunSparc *IPX*. Figure 12 is a plot of the final points. The data show a very tight pattern very close to the optimum, which is a clear improvement over the output for the pattern search and the response surface search.

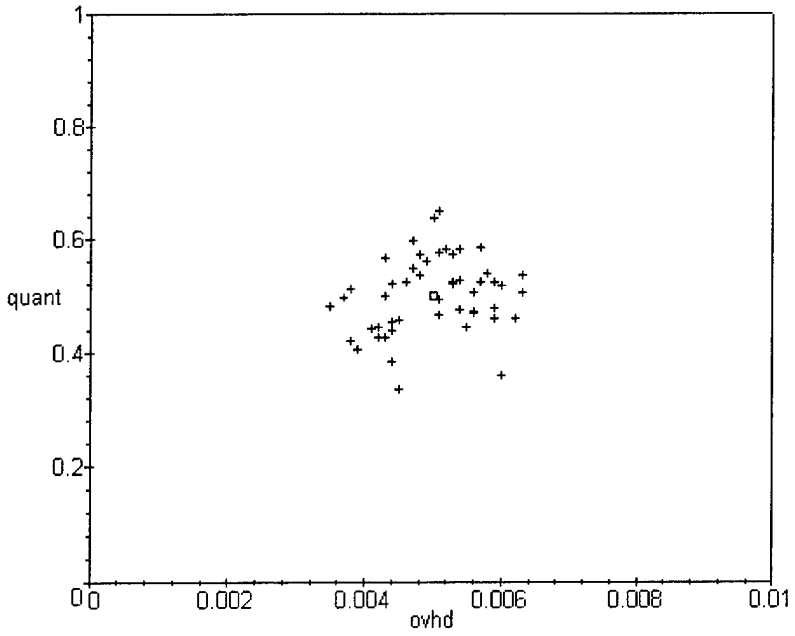


Figure 12. Final Points for Computer System — Genetic Search (optimum designated by a square)

7. EXAMPLE 3 — JOB-SHOP MODEL

The third example is a job-shop model that consists of a manufacturing facility made up of five groups of identical machines.[13,18] The jobs arrive at an *i.i.d.* exponential rate with a mean of 0.125 hour. There are three types of jobs, 1, 2, and 3, and their respective probabilities of arrival are 0.30, 0.50, and 0.20. Each job type requires specific tasks; thus, each job type has a different routing to the machine groups. A job does not have to visit all five machine groups, and most do not. The time to perform a task at a particular machine is an independent, 2-Erlang, randomly distributed variable whose mean is dependent on the job type and the group in which the machine belongs.

The routing of the job types is shown in table 1. The respective mean time at each group expansion appears in parentheses next to the machine group number.

Table 1. Routing of Job Types

Job Type	Machine Groups
1	3 (0.50), 1 (0.60), 2 (0.85), 5 (0.50)
2	4 (1.10), 1 (0.80), 3 (0.7-5)
3	2 (1.20), 5 (0.25), 1 (0.70), 4 (0.90), 3 (1.00)

It is possible for a job to arrive at a group and find all the machines busy. In that case, the job joins a first-in-first-out queue. The job-shop is simulated to run for 365 days, 8 hours per day. Each group contains the same machines; the company wants to minimize the overall average job total delay by optimally placing the machines in the different groups. The overall average delay is a weighted average of the average delays calculated in each of the five queues for each machine. The weights are the probabilities of a part going to each station. The decision variables are the number of machines in each of the five groups. The response function can be written as

$$cost = overall\ average\ delay\ in\ queue \times 1000 + total\ number\ of\ machines \times 1000. \quad (13)$$

7.1 EXAMPLE 3 RESULTS

All decision variables in the job-shop example, in this case the number of identical machines at each of the five workstations, are integers. A $U(3,13)$ distribution was used to generate the 50 initial points for the pattern search. For all algorithms $1 \le M_i \le 15$ and

$$\sum_{i=1}^{5} M_i \le 75,$$

where $i = 1, 2, 3, 4,$ and 5.

7.2 EXAMPLE 3 PATTERN SEARCH

The pattern search used an initial step size of 2, a minimum step size of 2, an expansion factor of 2, a lower boundary of 1, and an upper boundary of 15 for the five variables. The approximated optimums were determined to be $M_1 = 9$, $M_2 = 5$, $M_3 = 8$, $M_4 = 6$, and $M_5 = 3$. The data for the job-shop system are not shown because with the five decision variables it is not possible to show the data points on the same type of figures used earlier. While up to three sets of decision variables can be shown on a single figure, to show them all would require 30 figures. Thus, they are impractical to display.

7.3 EXAMPLE 3 RESPONSE SURFACE

The initial points are identical to those in the pattern search, so they are not displayed again.

7.4 EXAMPLE 3 GENETIC SEARCH

The methodology was 50 searches, with each search consisting of a simple genetic search and a population of 30 individuals. There were seven generations with each individual in a generation replicated three times. Each chromosome was 30 units long with each of the five variables 4 units long for a low of 1 and a high of 15. If by chance the string worked out to be 0, a finite possibility since this example deals with integers, the string was increased by 1.

The run itself took about 150 hours. The data show a very tight pattern close to the calculated optimum. This is an improvement over the output of both the pattern and the response surface searches.

8. DISCUSSION OF RESULTS

The results of the computer experiments were very informative, as shown in table 2. The data show that a genetic algorithm executes a superior search compared to a pattern or a response surface search, but requires more computer time. However, speed of the search was not a critical factor in the evaluation of the applications considered in this study.

Table 2. Inventory System Comparison of Pattern, Response Surface, and Genetic Search Methods

Type of Response	Average	Variance	Lower 9.5% Limit	Upper 9.5% Limit	Search Method
Optimum response	129.91	544.48	135.35	124.38	Pattern
	144.22	989.17	151.68	136.77	Response Surface
	106.73	2.40	107.10	106.37	Genetic
Distance traversed	30.46	3,005.43	NA	NA	Pattern
	9.06	837.83	NA	NA	Response Surface
	NA	NA	NA	NA	Genetic
Distance from initial points to optimum	86.09	2,479.59	NA	NA	Pattern
	86.09	2,479.59	NA	NA	Response Surface
	NA	NA	NA	NA	Genetic
Distance from final points to optimum	59.76	1,985.01	NA	NA	Pattern
	80.32	2,270.68	NA	NA	Response Surface
	9.88	18.83	NA	NA	Genetic
Runs per search	56.04	739.26	62.48	49.60	Pattern
	20.22	188.38	23.47	16.96	Response Surface
	414.00	34,053.06	4.57.74	370.26	Genetic

The average cost for the genetic search was 106.73 (variance = 2.40); for the response surface search it was 144.22 (variance = 989.17), and for the pattern search it was 129.91 (variance = 544.48). (See figure 13.) The result is that the genetic search method is superior to the other two in terms of both response and stability. The low variance for the genetic algorithm also results in a narrower confidence interval as shown in table 2. Based on these results, the genetic search is also superior in value and stability. The larger relative variance indicates a failure for the pattern search and the response surface search.

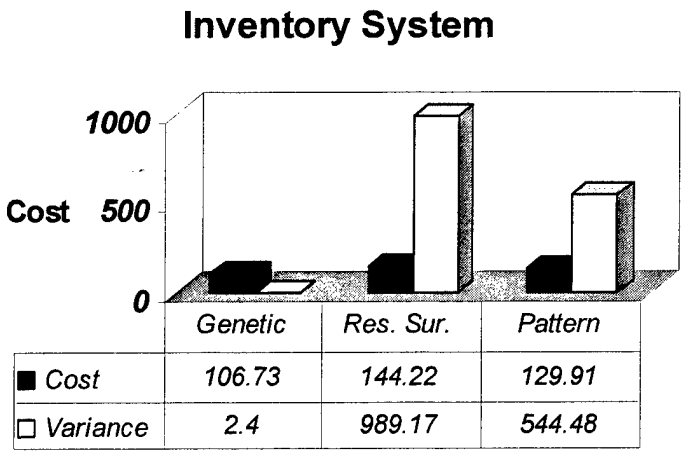


Figure 13. Average Costs for the Inventory System Searches

The average number of runs per search for the inventory system is much larger for the genetic search at 414 (variance = 34053.06) than for the other two search methods. This compares with the average number of runs for the response surface of 20.22 (variance = 188.38) and for the pattern search of 56.04 (variance = 739.26). (See figure 14.) The larger number of runs seems to be a small tradeoff for a clearly superior search.

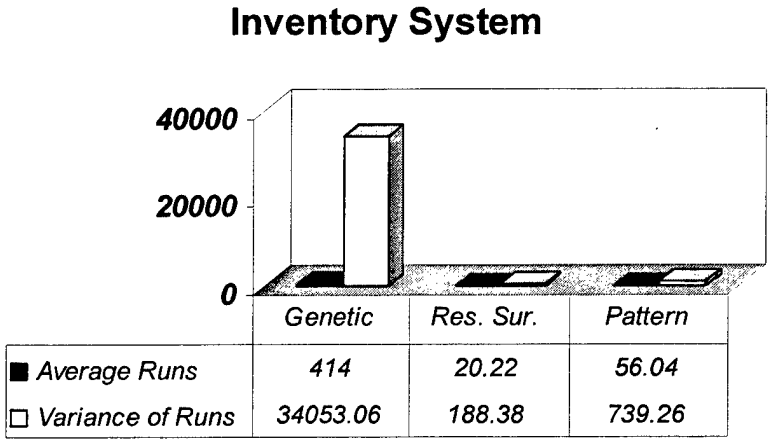


Figure 14. Average Number of Runs for the Inventory System Searches

Table 3 compares the results of the computer time-sharing model searches.

Table 3. Computer System Comparison of Pattern, Response Surface, and Genetic Search Methods

Type of Response	Average	Variance	Lower 9.5% Limit	Upper 9.5% Limit	Search Method
Optimum response	51497.53	1394843.85	51777.47	51217.60	Pattern
	17699.50	2837811.31	18098.78	17300.22	Response Surface
	16150.66	133220.98	16237.18	16064.1.5	Genetic
Distance traversed	0.16	0.02	NA	NA	Pattern
	0.16	0.03	NA	NA	Response Surface
	NA	NA	NA	NA	Genetic
Distance from initial points to optimum	0.21	0.02	NA	NA	Pattern
	0.21	0.02	NA	NA	Response Surface
	NA	NA	NA	NA	Genetic
Distance from final points to optimum	0.09	0.01	NA	NA	Pattern
	0.09	0.01	NA	NA	Response Surface
	0.0.5	0.01	NA	NA	Genetic
Runs per search	49.38	453.02	54.43	44.34	Pattern
	40.08	630.6.5	46.03	34.13	Response Surface
	365.40	39147.80	412.30	318.50	Genetic

The average cost for the computer systems example for the genetic search was 16,150.66 (variance = 133,220.98); for the response surface search it was 17,699.50 (variance = 2,837,811.31); and for the pattern search it was 51,497.53 (variance = 1,394,843.84). (See figure 15.) Therefore, the genetic search method is superior to the other two in terms of response and stability. The low variance for the genetic algorithm also results in a narrower confidence interval, as shown in table 3.

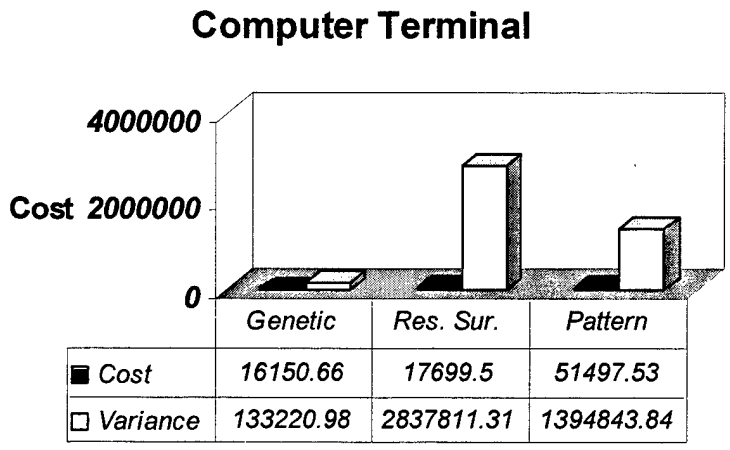


Figure 15. Average Costs for the Computer System Searches

The average number of runs per search for the computer system example is quite large for the genetic search at 365.40 (variance = 39,147.80). This compares with the average of the response surface search of 40.08 (variance = 630.65) and the pattern search of 49.38 (variance = 453.02). (See figure 16.) The larger number of runs seems to be a small tradeoff for a superior search.

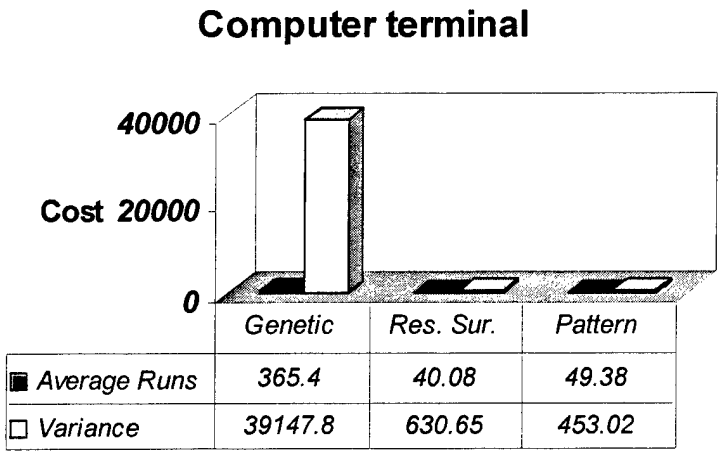


Figure 16. Average Number of Runs for the Computer System Searches

Consider the job-shop example's search output in table 4.

Table 4. Job-Shop Comparison of Pattern, Response Surface, and Genetic Search Methods

Type of Response	Average	Variance	Lower 9.5% Limit	Upper 9.5% Limit	Search Method
Optimum response	222436453	1.927×10^{16}	255335993.61	189536913.30	Pattern
	39721.31	16998523	40698.54	38744.09	Response Surface
	40082.05	8864470	40787.74	39376.36	Genetic
Distance traversed	0.67	1.82	NA	NA	Pattern
	5.23	4.00	NA	NA	Response Surface
	NA	NA	NA	NA	Genetic
Distance from initial points to optimum	9.02	5.39	NA	NA	Pattern
	9.02	5.39	NA	NA	Response Surface
	NA	NA	NA	NA	Genetic
Distance from final points to optimum	8.61	5.84	NA	NA	Pattern
	6.94	6.61	NA	NA	Response Surface
	6.57	4.82	NA	NA	Genetic
Runs per search	41.52	279.32	45.48	37.56	Pattern
	65.28	265.06	69.14	61.42	Response Surface
	385.20	38694.85	413.80	338.58	Genetic

The average cost of the job shop search for the genetic search was 40,082.05 (variance = 8,864,470), while for the response surface search it was 39,721.31 (variance = 16,998,523) and for the pattern search it was 222,436,453 (variance = 1.927×10^{16}). (See figure 17.) The result is that the genetic search method is superior compared to the pattern search in terms of both average response and stability and was equal to the response surface search in terms of average response and superior to it in terms of stability. The variance also resulted in narrower confidence intervals, as shown in table 4. The confidence intervals for the pattern search are very wide, narrow somewhat for the response surface search, and are narrower still for the genetic search.

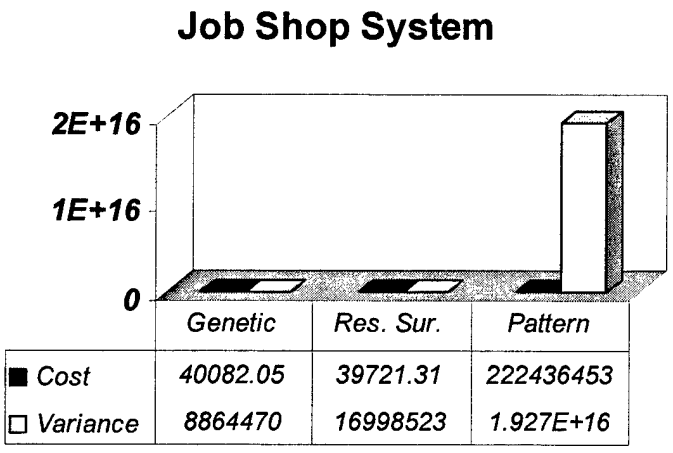


Figure 17. Average Costs for the Job-Shop System Searches

The average number of runs per search for the job-shop system is much larger for the genetic search at 385.20 (variance 38,694.85) than for the response surface search at 65.28 (variance = 265.06) and for the pattern search at 41.52 (variance = 279.32). (See figure 18.) Again, the larger number of runs seems to be a small tradeoff for a superior search.

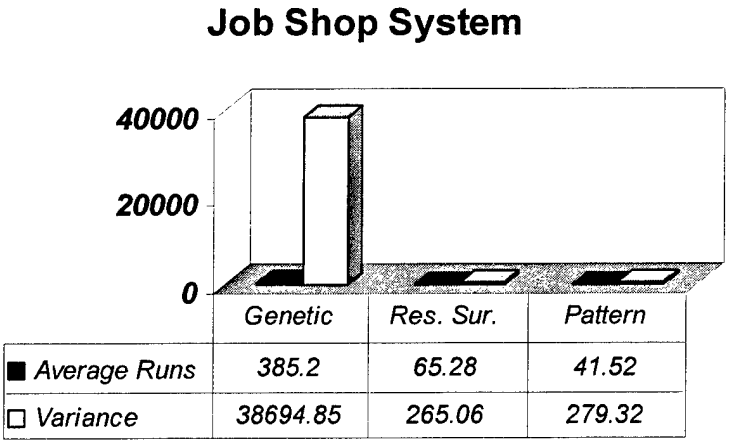


Figure 18. Average Number of Runs for the Job-Shop System Searches

9. STATISTICAL TESTS

A *t* test was used for equal but unknown variances to determine if there was a statistically significant difference between the means of either the pattern search or the response surface search. The stability or variance of the response of the genetic search for each example was compared to both the pattern search and the response surface search. As shown in table 5, the difference for the inventory system optimized using the pattern search versus the genetic search is statistically significant, as it is for the response surface search versus the genetic search. In both comparisons the *p* value, the level of significance, was essentially zero.

Table 5. Statistical Tests for the Pattern Search and the Response Surface Methods Versus the Genetic Search Method

Example	Comparison Scheme	T_1	*p* Value	F_2	*p* Value
Inventory System	Pattern versus genetic search	7.01	0.00	227.27	0.00
	Response versus genetic search	8.41	0.00	413.0	0.00
Computer Systems	Pattern versus genetic search	202.19	0.00	10.47	0.00
	Response versus genetic search	6.35	0.00	21.30	0.00
Job-Shop	Pattern versus genetic search	11.32	0.00	2.1×10^9	0.00
	Response versus genetic search	5.02	0.62	1.92	0.02

The difference for the computer system example optimized using the pattern search versus the genetic search is statistically significant, as it is for the response surface search versus the genetic search. For this example, the *p* value was employed instead of a threshold test for statistical significance. Since the *p* value was essentially zero, any level of significance chosen by the analyst would have led to the conclusion that the difference in the means was statistically significant. Chance has very little to do with the difference.

The job-shop system example shows a change in the trend. While it is true that the difference in the pattern search versus the genetic search is significant at zero, there is no significant difference at any low level between the response surface search and the genetic search.

The genetic search is superior in terms of stability in every case. The only time the *p* value is not zero is in the job-shop system example, specifically the response surface search method versus the genetic search. In that case, the *p* value is only 0.02. The genetic searches show more stability in all comparisons.

Consider the 95% confidence interval half-length about the response of each of the two traditional searches compared to the genetic search. A ratio is created with the genetic search confidence interval's half-length in the numerator (since it was the smallest) to see the comparison clearly. The results are shown in table 6.

Table 6. Ratio of the Confidence Interval Half-Lengths for the Pattern and Response Surface Search Methods to Those for the Genetic Search Method

Comparison Scheme	Ratio	Numerical Value
Inventory System:		
Genetic Search / Pattern Search	0.367 / 5.531	0.046
Genetic Search / Response Search	0.367 / 7.255	0.049
Computer System:		
Genetic Search / Pattern Search	86.51 / 279.93	0.103
Genetic Search / Response Search	86.51 / 399.28	0.217
Job-Shop:		
Genetic Search / Pattern Search	705.69 / 32899540.15	0.00
Genetic Search / Response Search	705.69 / 977.69	0.722

The genetic search produces confidence intervals in all three examples with the smallest ratios for the inventory system (genetic search/pattern search, genetic search/response surface search) and the job-shop (genetic search/pattern search) examples. The largest ratio occurs in the job-shop (genetic search/response surface search) examples. None of these ratios is greater than 1 (i.e., the genetic search always produces a smaller confidence interval ratio than the other methods).

The comparison of the genetic search method to the pattern search method for the inventory system example showed the genetic search method's confidence interval to be about 4.6% of the pattern search's confidence interval and 4.9% of the response surface search's confidence interval.

The comparison of the genetic search method to the pattern search method for the computer system example showed the genetic search confidence's interval to be about 10% of the pattern search's confidence interval and 22% of the response surface search's confidence interval.

For the job-shop system example, the genetic search's confidence interval half-length compared to the pattern search's half-length is essentially zero. However, the genetic search's confidence interval half-length is about 72% of the response surface search's confidence interval half-length.

10. SUMMARY

An examination of the response surface search method versus the GA search method shows that the response surface search actually compares quite well with the GA in terms of accuracy and speed on some occasions. Unlike the GA method, which showed itself to be robust in all circumstances, the response surface method does not always hold up.

The pattern search method does not compare favorably with either of the other two more sophisticated search methods, GA and response surface. The difference between the pattern search and the response surface search is most obvious in the graphic plots around the approximated optimum. The response surface search showed a more intense concentration around the optimum than did the pattern search.

The genetic search method gave excellent robust results, where the other two methods seemed to work well in only certain circumstances. This was, however, at the cost of more CPU time.

REFERENCES

1. R. Suri, "An Overview of Evaluative Models to Flexible Manufacturing Systems," *Annals of Operations Research,* vol. 3, 1985, pp. 13-21.

2. R. Hooke and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," *Journal of the Association for Computing Machinery*, vol. 8, no. 2, 1961, pp. 212-229.

3. H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *Computer Journal*, vol. 3, 1960, pp. 175-184.

4. J. A. Nelder and R. Mead, "A Simplex Method for Function Optimization," *Computer Journal*, no. 7, 1965, pp. 308-313.

5. C. D. Pegden and M. P. Gately, "A Decision Optimization Module for SLAM," *Simulation*, vol. 34, 1980, pp. 18-25.

6. M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*, John Wiley and Sons, New York, 2000.

7. D. J. Wilde and C. S. Beightler, *Foundations of Optimization*, Prentice-Hall, Englewood Cliffs, NJ, 1967, pp. 307-310.

8. R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, John Wiley and Sons, New York, 1998.

9. J. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 1, no. 16, 1986, pp. 122-128.

10. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, Reading, MA, 1989, p. 5.

11. J. H. Holland, *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1970.

12. K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1975.

13. A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis,* Second Edition, McGraw-Hill, New York, 1991, pp. 75-103.

14. J. Banks and J. S. Carson, *Discrete-Event System Simulation*, Prentice-Hall, Englewood Cliffs, NJ, 1984, pp. 253-251.

15. D. E. Smith, "Automated Response Surface Methodology in Digital Computer Simulation, Volume 1: Program Description and User's Guide," Desmatics Inc., State College, PA, 1978.

16. A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis,* McGraw-Hill, New York, 1982, p. 379.

17. P. Bratley, B. Fox, and L. E. Schrage, *A Guide to Simulation,* SpringerVerlag, New York, 1987, pp. 257-259.

18. R. Varela, R. V. Camino, J. Puente, A. Gomez, and A. M. Vidal, "Chapter 8, Solving Job-Shop Scheduling Problems by Means of Genetic Algorithms," in *The Practical Handbook of Genetic Algorithms*, L. Chambers ed., Chapman and Hall, Boca Raton, FL, 2001.

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
Center for Naval Analyses	1
Defense Technical Information Center	2