Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

EFFICIENT SIMULATION-BASED DISCRETE OPTIMIZATION

Seth D. Guikema
Rachel A. Davidson
Zehra Çağnan

School of Civil & Environmental Engineering
Hollister Hall
Cornell University
Ithaca, NY 14850, U.S.A.

ABSTRACT

In many practical applications of simulation it is desirable to optimize the levels of integer or binary variables that are inputs for the simulation model. In these cases, the objective function must often be estimated through an expensive simulation process, and the optimization problem is NP-hard, leading to a computationally difficult problem. We investigate efficient solution methods for this problem, and we propose an approach that reduces the number of runs of the simulation by using ridge regression to approximate some of the simulation calls. This approach is shown to significantly decrease the computational cost but at a cost of slightly worse solution values.

1 INTRODUCTION

Simulation-based optimization of discrete decision variables is difficult both because the optimization problem itself is NP-hard and because using a simulation model to evaluate each trial solution can be computationally demanding. At the same time, simulation-based discrete optimization problems are of great practical significance. For example, Çağnan and Davidson (2004) and Davidson and Çağnan (2004) developed a discrete event simulation model that estimates the amount of time needed to restore electric power to utility customers after an earthquake *given* a set number of available crews. Optimizing the number of crews at each possible location is an important way to reduce the duration of electric power outages after earthquakes given a fixed budget. However, the response of the electric power system and similar infrastructure systems is complex, making analytical solutions or expert guesses about the best crew allocation unreliable in these situations. A numerical, simulation-based optimization routine is needed. Lee and Azadivar (1985) and Nair, Keane, and Shimpi (1998) provide further examples of discrete simulation-based optimization.

The computational complexity of the simulation together with the NP-hard nature of the optimization problem suggest two main methods for reducing the computational burden of solving this type of problem. The first is by reducing the number of potential solutions that must be evaluated to find a good solution, and the second is by reducing the computational time required to evaluate each potential solution that is generated. In this paper we investigate the use of genetic algorithms for solving this problem, and we present a new approach that focuses on the second of these methods – reducing the computational time required to evaluate each potential solution – by approximating the results of the full simulation run for some candidate solution evaluations.

2 BACKGROUND

A number of different approaches have been proposed for the joint simulation-based discrete optimization (SBDO) problem that focus primarily on reducing the number of potential solutions examined. In an overview, Swisher et al. (2000) discuss the use of ordinal optimization, simulated annealing, genetic algorithms, tabu search, and the Nelder-Mead algorithm. Lee and Azadivar (1985) show how a modified simplex algorithm can be used for the SBDO problem, and Shi, Chen, and Yücesan (1999) present a procedure based on nested partitions and statistically-based control of the number of simulation runs. While the simulation control approach used by Shi, Chen, and Yücesan (1999) does attempt to reduce the number of simulation runs needed to evaluate each potential solution, the main focus of the paper was on combining this with nested partitions to reduce the number of solutions that are examined. Tompkins and Azadivar (1995) and Azzaro-Pantel et al. (1998) both present genetic algorithms for solving the SBDO problem in the context of production and manufacturing. These algorithms seek to find good solutions by successively generating candidate solutions and then evaluating the objective function value for each of these solutions based on running a simulation model. If *n* potential solutions are to be examined, *n* complete runs of the simulation model are needed. If the simulation model is computationally expensive to run, this limits the number of potential solutions that can be evaluated. In the work that follows, we use a genetic algorithm as our optimization approach because genetic algorithms are widely used in simulation-based optimization. However, the general approach that we develop – using ridge regression to reduce computational burden – could be applied in conjunction with any iterative optimization procedure.

A second approach for reducing the computational burden of solving the SBDO problem is to try to reduce the amount of computational effort required to evaluate the objective function value for each candidate solution, either through reducing the number of replications needed for each simulation run or by reducing the number of simulation runs required. Noisy genetic algorithm theory (e.g., Aizawa and Wah 1993; Giguère and Goldberg 1998) aims to achieve this in the context of optimizing continuous decision variables on the basis of a simulation model by controlling the number of candidate solutions to evaluate for each iteration of the optimization algorithm and the number of simulation replications used to evaluate each candidate solution. In this approach, the number of replications of the simulation program begins small and grows through successive generations according to pre-defined rules.

An alternate approach is to base the evaluation of some or all of the candidate solutions on an approximation to the simulation model rather than on the simulation model itself. Nair, Keane, and Shimpi (1998) and Kodiyalam, Nagendra, and DeStafano (1996) both explore this approach, the former in designing a truss and the later in designing a composite structure for a spacecraft. Both of these papers use linear regression to approximate the results of a computationally expensive structural design model together with a genetic algorithm for the optimization of discrete decision variables. In both cases, the objective function values returned by the structural models were treated as the dependent variables, and the decision variable values were treated as the independent variables. With this approach, the approximation model is first fit based on a small number of runs of the computationally expensive structural model and is then used to evaluate successive candidate solutions. The approximation model can be updated occasionally based on one or more runs of the expensive structural model. This approach thus reduces the computational burden by reducing the number of times the expensive model must be run. However, as discussed by Nair, Keane, and Shimpi (1998) and discussed further below, approximation by ordinary least squares regression has significant drawbacks.

Approximating the simulation results with an ordinary least squares (OLS) regression model and then using this approximation for at least some of the iterations of an optimization algorithm for the SBDO problem is intuitively appealing, but subject to several limitations. If properly formulated and fitted, the regression model would hopefully approximate the simulation results well enough to keep the optimization routine converging towards a good solution without running the simulation model. The approximation need not be perfect, just good enough to guide the optimization algorithm. However, OLS suffers from several drawbacks in this context. First, as discussed by Nair, Keane, and Shimpi (1998), the approximation is generally only reasonable for a bounded region containing the candidate solutions used to fit the model. For problems in which the decision variables are integer-valued, this region may be small. Approximations outside this region may be significantly in error, leading the optimization routine to move in the incorrect direction on its next step.

The second and potentially more severe limitation of using OLS regression to approximate simulation results is that in many cases an OLS fit based on a population of candidate solutions can suffer from the problem of colinearity. This colinearity can arise in two ways. First, if the number of candidate solutions used to fit the model is large relative to the number of decision variables, the input data for the fit would likely be at least nearly collinear, leading to numerical instability in the estimation of the OLS parameters. Second, assuming that the optimization algorithm examines multiple candidate solutions in each iteration, one would hope that as the algorithm converges towards a good solution, the candidate solutions it generates become more similar. These late-iteration solutions would almost certainly be collinear. This colinearity can make it difficult or impossible to fit standard OLS models, and, even when these models can be fit, the resulting parameter estimates are often highly sensitive to small changes in the input data (see, for example, Montgomery and Peck 1992). In these cases, use of OLS can give poor predictions at points not in the data set used to fit the model.

In this paper we present a more robust approach for approximating the results of the simulation model based on ridge regression. We show that this approach can overcome the problems of OLS and lead to a significant reduction in the number of simulation runs required and thus in the computational burden of solving SBDO problems.

3 RIDGE REGRESSION

When OLS regression is used with nonorthogonal (collinear) data, the resulting estimates of the regression parameters ($\hat{\beta}$) typically have inflated variance and are unstable (Montgomery and Peck 1992). That is, the parameter estimates are highly sensitive to the input data, and thus may give poor predictions when used with other data sets. This problem arises because OLS produces the minimum variance estimates among all possible *unbiased* $\hat{\beta}$s. Ridge regression, originally proposed by Hoerl and Kennard (1970), is one of the primary methods used to deal with the problem of colinearity in regression modeling. It does not require that the parameter estimates be unbiased. Instead, it seeks to minimize the *variance* in the parameter estimates ($\hat{\beta}_R$) and adds a biasing parameter, *k*, to the regression equation to remove the problem of colinearity. The standard OLS and ridge regression equations for parameter estimation are shown in equations (1) and (2) respectively, where *X* is a matrix of the independent variables, *y* is a vector of the dependent variables, *I* is the identity matrix, and *k* is the biasing parameter (see Montgomery and Peck 1992, pp. 329-331).

$$\text{OLS: } \hat{\beta} = \left(X'X\right)^{-1} X'y \tag{1}$$

$$\text{Ridge: } \hat{\beta}_R = \left(X'X - kI\right)^{-1} X'y \tag{2}$$

As the biasing parameter (*k*>0) increases, the bias in the parameter estimates increases, but the variance decreases. High bias would tend to systematically skew the estimates produced by a regression model while high variance would make the regression estimates very sensitive to the particular set of input data used. Thus, there is a trade-off between bias and variance that needs to be made in selecting a *k* to use in an analysis. One method for selecting a value for *k* uses a simple graphical technique in which the parameter estimates are plotted versus *k*, and a value of *k* is selected such that the parameter estimates are relatively stable (e.g., Hoerl and Kennard 1970). Other more quantitative methods are based on various measures of the improvement in the fit over OLS, with the goal being the maximization of the improvement (e.g., Mallows 1973, Wahba, Golub, and Health 1979). We use the graphical approach.

As with the OLS approach of Nair, Keane, and Shimpi (1998) and Kodiyalam, Nagendra, and DeStafano (1996), ridge regression can be integrated into SBDO solution procedures as a replacement for some or all of the simulation runs. The general idea is to first randomly generate a set of initial solutions and evaluate their objective function values using the simulation model, fit the regression model based on these initial solutions, and then use this fitted model in place of the simulation model in estimating the objective function values for at least some of the future candidate solutions generated by the optimization routine. The regression fit may be updated in some iterations based on new simulation runs. This approach can be used with any optimization algorithm that iteratively generates candidate solutions, the objective function values of which must then be estimated with a simulation run. Before presenting a particular implementation utilizing a genetic algorithm, we first present the pseudo-code for using ridge regression below.

```
1. Randomly generate initial set of candidate solutions, evaluating the objective function for each
   by running the simulation model.
2. Set resample = R, iter = 1.
3. Fit the ridge regression model based on the feasible solutions in the cumulative set of candidate
   solutions.
4. If iter $\neq$ resample
   a. Run optimization routine to generate a new set of candidate solutions with objective function
      evaluation based on the current ridge regression model.
   b. iter = iter +1.
   Else
   c. Run optimization routine to generate a new set of candidate solutions with objective function
      evaluation based on simulation model runs.
   d. iter = iter + 1, resample = resample + R
   e. Add all newly generated feasible candidates to the database used to fit the ridge model.
   f. Re-fit the ridge model based on the cumulative database.
5. If the stopping criteria is not met, goto (4), else stop. The optimal solution is the best solution
   in the final set of candidate solutions.
```

Note that if R, the resampling interval, is set equal to one, the ridge regression will never be used to estimate objective function values. Conversely, if R is set equal to or greater than the maximum number of iterations, the ridge regression model will be used as the basis for all objective function estimates after the initialization. In the next section we describe a particular implementation of ridge regression approximation using a genetic algorithm. This case serves as the test problem that we will use to compare our proposed approach with a genetic algorithm that does not use ridge regression.

4 TEST PROBLEM

In order to demonstrate the use of ridge regression for the SBDO problem and compare it with other approaches, we use, as an illustration, a discrete event simulation model of the process of restoring electric power after the occurrence of an earthquake. The problem is a simplified version of the problem addressed by Çağnan and Davidson (2004) and Davidson and Çağnan (2004). This simulation model seeks to estimate the time required for a utility company to restore electric power to their customers after the occurrence of an earthquake, given a fixed number of each of four different types of crews.

As discussed by Çağnan and Davidson (2004) and Davidson and Çağnan (2004), the electric power system is modeled as a set of substations, generation plants, and district yards as the nodes of the system. In order for power to be restored to a given area, that area's substation must be working, and it must be connected to a working generation plant. In the process of repairing the system after an earthquake, each of the substations and generation plants must be inspected to determine its damage state before it can be repaired. In the simplified model used in this paper, the inspections can be carried out by any of three crew types (on-duty operators, off-duty operators, and inspection teams), each of which requires a different amount of (stochastic) travel time to begin its inspection. Each substation and generation plant can have one or more on-duty or off-duty operators associated with it. If a node is inspected and found to be damaged, a repair team is sent from a district yard to restore that node to service. Inspection and repair teams can only be located at the district yards. Inter-node travel times and repair times are stochastic, while inspection times are assumed to be deterministic. In the simplified model used in this paper, there are three substations, one generation plant, and one district yard. The objective of the optimization is to minimize the time needed to restore electric power to all utility company customers.

The decision variables are the numbers of on-duty operators (I_j), off-duty operators (O_j), inspection teams (IS_j), and repair teams (R_j) to locate at each node *j*. Each type of crew (indexed in *k*) has an associated cost (e.g., training costs and salary) if located at node *j* given by $\alpha^{(k)}{}_j$, and there is a budget (*B*) available for crew training that cannot be exceeded. The objective is to minimize the expected amount of time (*T*) needed to restore power to all substations subject to a budget constraint, and the restoration time is estimated by the discrete event simulation model using the values of the decision variables as input parameters. There can be at most one of each type of operator associated with each node, and there can be no more than five inspection teams and no more than five repair teams located at the district yard. The set of all substations is noted as *S*, the set of all yards is noted as *Y*, and the set of all generation stations is noted as *G*. The mathematical formulation of this problem is given below in equations (3) – (8).

$$\underset{\mathrm{I}_j, O_j, IS_j, IR_j}{Min} \; E[T] \tag{3}$$

such that:

$$\sum_j \left(\alpha_j^{(1)} I_j + \alpha_j^{(2)} O_j + \alpha_j^{(3)} IS_j + \alpha_j^{(4)} R_j\right) \leq B \tag{4}$$

$$I_j, O_j, IS_j, R_j \geq 0 \tag{5}$$

$$I_j, O_j, IS_j, R_j \;\; \text{integer} \tag{6}$$

$$IS_j, R_j = 0 \;\; \text{for } j \notin Y \tag{7}$$

$$I_j, O_j = 0 \;\; \text{for } j \notin S \tag{8}$$

With four types of crews to allocate at only five different nodes, there are twenty integer decision variables (some of which are constrained to be zero), yielding $1\text{x}10^{21}$ possible solutions, many of which are infeasible. Due to the computational burden of the simulation model, an efficient optimization algorithm combined with as few simulation runs as possible are needed to solve this problem.

In the test problem, earthquakes occur randomly over the planning horizon, leading to random occurrence of damage states. The state of each substation and generation station is assumed to be binary – damaged or undamaged. All costs are given based on the same planning horizon (e.g., 20 years) as the damage probabilities. Table 1 gives the set of damage probabilities used in the test problem.

Table 1: Test Problem

Node	Damage Probability
Substation One	0.1
Substation Two	0.1
Substation Three	0.1
Generating Plant	0.01

The set of probabilities used in the example problem is meant to be realistic in terms of an actual infrastructure system that is subject to rare natural hazards such as earthquakes, and to test the ability of the methods to handle the types of unlikely system states likely to arise in practice. With only 2^4 possible system damage states, the restoration simulation is run a pre-determined number of times (to be discussed below) for each possible system state to estimate the expected restoration time conditional on the damage state. Then these conditional expected values are converted into an unconditional expected restoration time for a given set of decision variable values using the calculated probabilities of the 16 different system states. For a larger system, importance sampling would be needed to generate the system damage states.

5 OPTIMIZATION APPROACHES TESTED

In order to examine the impacts of using the ridge regression approximation in place of some of the simulation runs, we compared several optimization approaches. In all of these, a genetic algorithm was our underlying optimization algorithm. The particular genetic algorithm that we used was based on the Genetic Algorithm Toolbox for Matlab described by Chipperfiled et al. (1994). This toolbox provides the basic building blocks from which our algorithm was built. Our algorithm uses a non-linear fitness ranking approach with a selective pressure of two, stochastic universal sampling for the selection of parents to combine to form new children, a cross-over rate of 0.7, a high mutation rate of 20%, and fitness-based reinsertion of the children solutions with 20% of the parents replaced with the highest valued children in each generation. The high mutation rate (20%) and replacement rate (20%) mean that our genetic algorithm will maintain a relatively high degree of randomness in creating new candidate solutions while at the same time converging towards good solutions. The mutation rate and replacement rates were selected based on initial exploratory runs varying the mutation rate between 0.2% and 50% and the replacement rate from <5% to 80%. In these runs, the consistency of the solutions over multiple runs with fixed parameter values was compared with the amount of time required to find a solution, and mutation and replacement rates were selected that balanced these.

We used a population size of 5 individuals with 10 generations (again, selected based on exploratory runs). For larger problems, more individuals and a larger number of generations would be needed as discussed below. A penalty method assigning a value of 99 (plus a random number between 0 and 1 to avoid ties in the fitness ranking) was used to impose the feasibility constraints, and the simulation model was not run for infeasible solutions. In all approaches tested except random generation of candidate solutions, the number of replications of the restoration process run for each damage state was determined by the duration sizing approach of Aizawa and Wah (1993) with a base case growth parameter (γ) of 0.005 and an initial replication size of three. This yields an increasing number of replications over the course of the optimization, focusing the simulation effort on the later generations where more accurate objective function estimates are more important. We also report the results of varying γ. The optimization approaches that we tested were: (1) the genetic algorithm with the full simulation run for each candidate solution evaluation, (2) the genetic algorithm with some of the simulation runs replaced by the ridge regression approximation, and (3) pure random search for comparison. As will be discussed below, the control parameters for both genetic algorithm approaches were varied. In order to account for the variability inherent in the solutions produced by genetic algorithms, we ran each approach 30 times with identical control parameters in order to allow fair statistical comparison of the results.

5.1 Optimization with Complete Simulation Runs

The baseline against which we compared the approach using ridge regression was the genetic algorithm for which the evaluation of all candidate solutions was based on a full run of the simulation ("GA-Full"). That is, each time a candidate solution was to be evaluated, the simulation was run r times for each of the sixteen damage states, requiring $16r$ replications. As discussed above, the r used in each generation was determined by the adaptive duration sizing approach of Aizawa and Wah (1993). We ran the Full GA approach for 10 generations. For larger, more realistic problems more generations would be needed.

5.2 Optimization with Ridge Approximation

In implementing the ridge regression approximation, we used the algorithm outlined in the pseudo-code in Section 3. We will term this approach the "GA-Ridge" approach. In testing GA-Ridge, the simulation was run every R generations. That is, for generations that were an integer multiple of R, the genetic algorithm calculations were done on the basis of the full simulation, but for all other generations the objective function evaluations were based on the ridge regression model. A log was kept of all feasible candidate solutions for which a simulation-based estimate of the objective function value was available. Each R generations, the ridge regression model was re-fit based on the newly updated log of simulated solutions. Thus, while the ridge fits in the early generations may be poor, they become increasingly better over the course of successive generations. We used the Ridge GA approach with R equal to 2 and 5. The number of simulation replications was determined by the same approach as for GA-Full.

5.3 Optimization with Random Search

In the random search approach, a series of 10 sets of 5 candidate solutions were generated, with the lowest-valued (best) solution from each of the 10 sets saved. The algorithm was constrained to generate only solutions that did not violate the constraints given by equations (5) to (8). In evaluating the candidate solutions, the simulation model was run for only 1 restoration time replication for each damage state in order to decide if a randomly generated candidate solution should be kept as a good solution. This low number of replications was selected to make the CPU time required for the random search algorithm comparable to the CPU time required for the other approaches which also used small numbers of replications for early generations. The procedure used for generating the candidate solutions was the same algorithm used to generate the initial set of candidate solutions for the genetic algorithm approaches. This random solution generator, which is built in to the GA Toolbox, is relatively expensive computationally. This led to the use of a small number of evaluative simulation replications with the random search to keep the CPU time per run comparable to the other methods.

6 RESULTS AND DISCUSSION

Figures 1 and 2 summarize the results of the optimization runs for a number of different budget levels. Note the log scale on the x-axis in these figures. Figure 1 gives the mean and 95% confidence intervals of the *feasible* solutions found in the 30 runs of the four optimization approaches (GA-full, GA-Ridge (R=5), GA-Ridge (R=2), and Random Search). Figure 2 gives the same information for *all* solutions found, with a penalty of 99 plus a uniform (0,1) random number assigned to infeasible solutions. In some of the cases shown in Figure 2, a significant number of the optimization runs did not find a feasible final solution, and these infeasible solutions have been dropped in Figure 1. The percentage given by each of the confidence intervals in Figure 1 is the percentage of the 30 final solutions that were infeasible.

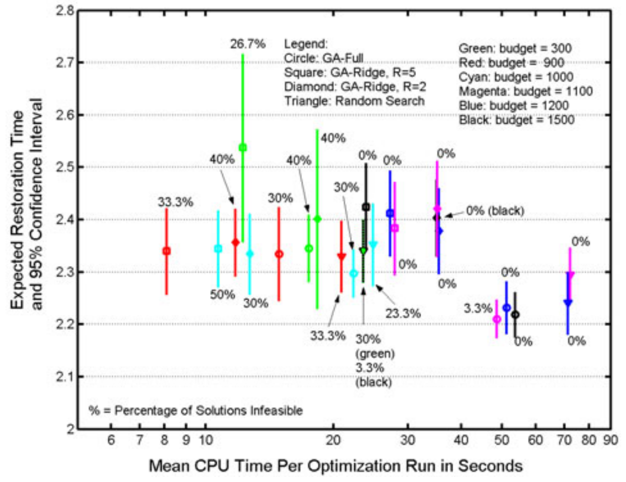

Figure 1: Mean and 95% Confidence Intervals of the *Feasible* Optimal Results from 30 Optimization Runs for Each Approach for Various Budget Levels

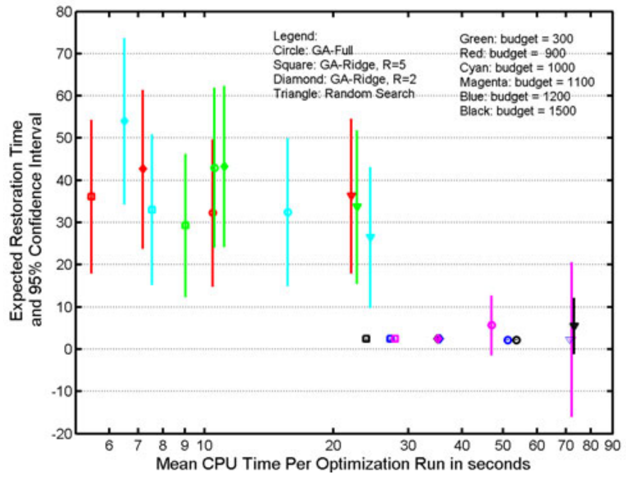

Figure 2: Mean and 95% Confidence Intervals of *all* Optimal Results from 30 Optimization Runs for Each Approach for Various Budget Levels

The results in Figures 1 and 2 show that for the three lowest budgets (300, 900, and 1000), the problem is difficult enough that none of the algorithms do particularly well or clearly outperform the other algorithms in terms of the quality of the solutions found. The two GA-Ridge approaches are faster on average than the other two approaches. All four approaches yield infeasible solutions in 30% - 40% of the optimization runs. These low budgets represent hard optimization problems in the sense that all four approaches have a difficult time finding feasible solutions. With the budget of 300, the feasible solutions found by the GA-Ridge approaches have higher (worse) mean restoration times than those found by random search and GA-Full. However, these differences are not statistically significant due to the high variability in these results. With budgets of 900 and 1000, the means and 95% confidence intervals for the feasible solutions found by all four methods are very similar.

For the three higher budgets shown in Figures 1 and 2 (1100, 1200, and 1500), the GA-Full and Random Search approaches yield slightly better objective function values than the two GA-Ridge approaches, and these differences are statistically significant. The optimal mean restoration times found by the GA-Ridge approaches were approximately 8-10% higher (worse) that those found by the GA-Full approach. These results thus suggest that based only on the feasible solutions found, the ridge-based approaches are not as accurate as the GA-Full or even Random Search (except for the budget of 1500), but the GA-Ridge approaches do save a considerable amount of computational effort. It should also be noted that even though the Random Search had the advantage of generating only solutions that did not violate equations (5) – (8), it did not always yield a feasible solution. The genetic algorithm approaches did generate solutions that violated the constraints given by equations (5) – (8).

Figure 3 summarizes the percentage of solutions found by each approach that were infeasible for each budget. This figure shows that for low budgets, none of the tested approaches consistently find feasible solutions. However, for high budgets (1200 and 1500) all approaches do well in finding feasible solutions. The percentage of feasible solutions monotonically increases with budget for GA-Full and GA-Ridge(2), while this is not the case for the GA-Ridge(5) and Random Search algorithms, suggesting that the GA-Full and GA-Ridge(2) behave in a less random manner in terms of their ability to find feasible solutions.

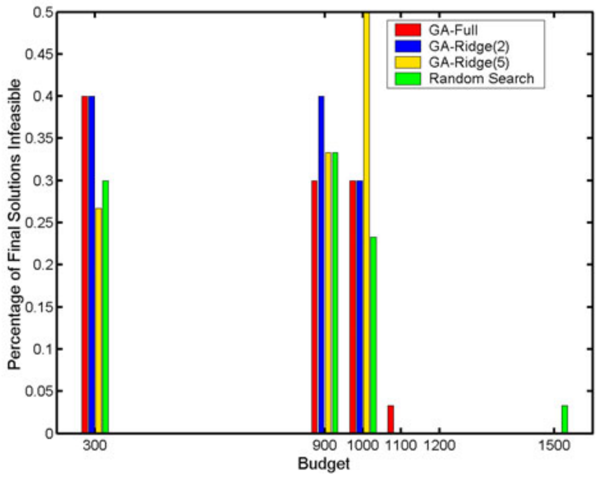

Figure 3: Percentage of Final Solutions that Were Infeasible for Each Approach for Various Budgets

The different approaches require different amounts of CPU time to find solutions, as summarized in Figure 4.

These results show that for low budgets (i.e., difficult problems), the GA-Full and GA-Ridge approaches require similar amounts of computational time. This is due to the fact that many of the solutions found at these low budgets are infeasible, reducing the number of runs of the simulation model and thus the opportunities for the GA-Ridge approaches to gain an advantage in CPU time required. However, as the budget is increased, the difference in computation time between the approaches becomes statistically significant, with GA-Ridge(5) requiring less time than GA-Ridge(2) which in turn requires less time than GA-Full. In all cases, the Random Search procedure required more computational time. This was primarily due to the computational burden of randomly generating a constrained value for each of the 20 decision variables at each iteration using the algorithm in the Genetic Algorithm Toolbox. While we may have been able to more efficiently generate random solutions with a custom algorithm, instead we used the same algorithm used to generate initial populations for the GA approaches in order to provide a fair comparison of the methods. We have not shown the time CPU time required by the random search approach in Figure 4 because it is dependent on the particular random generator that we used.

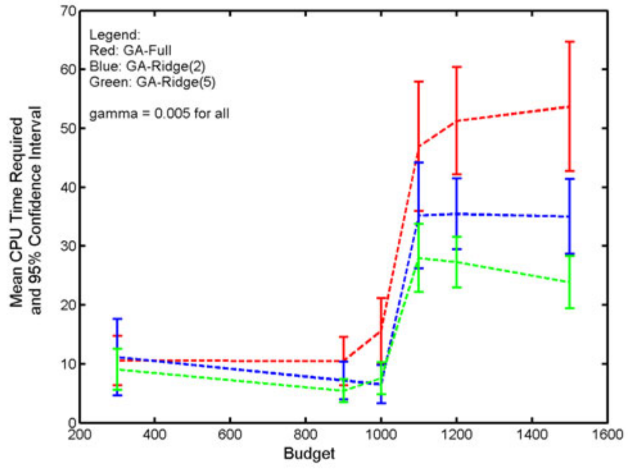

Figure 4: Comparison of CPU Time Required (in Seconds)

Figure 5 shows the means of the 30 optimal values of each of the 20 decision variables for each of the four optimization approaches together with the standard deviations of these optimal decision variable values. The results are based on a budget of 900 and a gamma value of 0.005. These results show that all four approaches return similar values for the first 10 decision variables – the number of on-duty and off-duty operators at each station. However, the mean values for these binary variables are all in the vicinity of 0.4 to 0.6, suggesting that the algorithms are returning highly variable values for these binary decision variables. The 11th through 20th decision variables correspond to the number of inspection teams and repair teams located at each station. Variables 11-14 and 16-19 give the number of each of these types of teams at the substations and generation stations – locations where these teams cannot be located and still contribute to restoration. Thus, we would expect the algorithms to return values of 0 for these decision variables. The random search does do this because it was constrained to do so. The GA-Full and GA-Ridge approaches were not so constrained, and they consequently do yield positive values of these variables in some runs. Note that although all four approaches yield similar answers on average for variables 15 and 20, the number of inspection and repair teams to have at the district yard, the two GA-Ridge approaches tend to yield lower numbers for these variables. Thus, the four approaches all give similar decision variable values for this case.

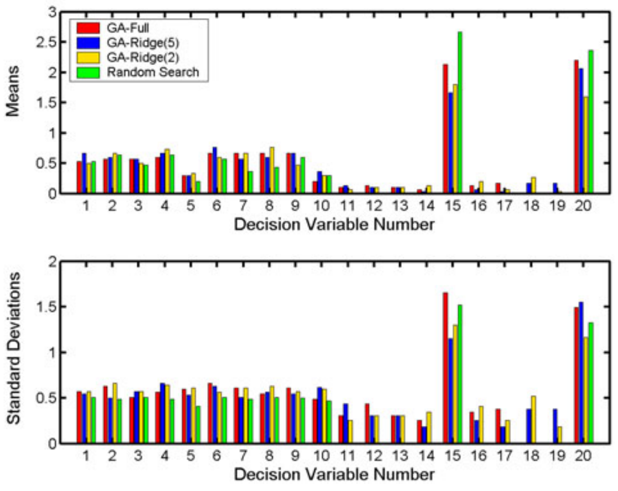

Figure 5: Means and Standard Deviations of the Optimal Decision Variable Values for 30 Runs of Each Algorithm with a Budget of 900 and gamma = 0.005

Finally, we also varied the value of γ used in the GA-Ridge and GA-Full approaches over a range from $\gamma = 0.001$ to $\gamma = 0.05$. Table 2 gives the percentage of solutions found by each approach that were infeasible for each γ used.

Table 2: Effect of γ on Percentage of Infeasible Solutions

γ	Percent Infeasible		
	GA-Full	GA-Ridge(5)	GA-Ridge(2)
0.001	30%	30%	50%
0.005	30%	33.3%	40%
0.007	26.7%	33.3%	36.7%
0.01	33.3%	26.7%	43.3%
0.05	16.7%	36.7%	40%

Table 2 shows that the percentage of infeasible solutions found by each of the approaches is similar over the entire range of γ values used, suggesting that the ability of the methods to return feasible solutions is not highly dependent on the value of γ used. The results in Table 2 also show that the GA-Full approach returned a lower percentage of infeasible solutions than either of the GA-Ridge algorithms in all but one case. Thus, the GA-Full approach does appear to be more efficient at finding feasible solutions. While not shown here, the optimal mean restoration times found by the approaches were similar for the different values of γ.

We also applied the GA-Full and GA-Ridge approaches to a larger test problem with 18 substations and 2 generation stations in order to examine how well our results generalize to larger, more difficult problems. We ran the GA-Full approach with 50 individuals/population for 30, 50, 75, and 100 generations, and we ran the GA-Ridge approach with 50 individuals for: (a) 300 generations with simulation every 50 generations, (b) 300 generations with simulation every 10 generations, and (c) 100 generations with simulation every 3 generations. The summary of the results of these runs are:

- The GA-Ridge approach reduces the computational time by an order of magnitude in the larger test problem, from approximately 55 hours to approximately 5 hour per run on a Pentium III 733 MHz computer.
- The solutions found by the GA-Ridge approaches are 10%-20% worse than those found by the GA-Full approach run for 100 generations. This result held for all GA-Ridge approaches tested for the larger problem.
- The solutions found by the GA-Full approach are worse than those found by the GA-Ridge approaches when GA-Full is run for only 30 or 50 generations. The results for 75 generations are highly variable between runs, with some runs yielding better solutions than GA-Ridge and some worse. When run for only 30 generations, GA-Full failed to find feasible solutions in most attempts.

While preliminary, these results do suggest that the conclusions based on the small test problem do generalize to larger problems. The GA-Ridge approach saves a significant amount of computation time at the cost of a small reduction in the quality of the solution found.

7 CONCLUSIONS

This paper has presented the results of testing approaches for efficiently solving simulation-based discrete optimization problems. Overall, the results presented in this paper suggest the following.

1. For problems in which there are either few feasible or many good solutions, the choice of solution approach among those tested here does not matter much. For hard problems with few feasible solutions, all of the tested approaches have difficulties finding feasible solutions consistently. For easy problems with many good solutions, all approaches yield good solutions.
2. The ridge-based optimization approaches save a significant amount of computational time in solving SBDO problems.
3. The ridge-based approaches yield slightly worse solutions than the full genetic algorithm approach.

There is clearly a trade-off between solution accuracy and computational burden in using ridge-based genetic algorithms for solving simulation-based optimization problems. As the degree of approximation increases (i.e., as R increases), the GA-Ridge approaches require much less computational time but return less accurate solutions. This trade-off must be managed on a problem-specific basis. For relatively easy problems in which it is computationally feasible to solve the problem with the GA-Full approach, this approach is likely best. However, for problems in which the simulation model is complex enough that solving the problem with the GA-Full approach is computationally prohibitive, the GA-Ridge approach is a good alternative. In many cases, it can provide good solutions with much less computational burden.

ACKNOWLEDGMENTS

This work was supported in part by the Earthquake Engineering Research Centers Program of the National Science Foundation under NSF Award Number EEC-9701471, and in part by NSF Award CMS-0196003. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

REFERENCES

Aizawa, A.N. and B.W. Wah. 1993. Dynamic Control of Genetic Algorithms in a Noisy Environment. In *Proceedings of the 5th International Conference on Genetic Algorithms*, 48-55, Urbana-Champaign, IL.

Azzaro-Pantel, C., L. Bernal-Haro, P. Baudet, S. Domenech, and L. Pibouleau. 1998. A Two-Stage Methodology for Short-Term Batch Plant Scheduling: Discrete Event Simulation and Genetic Algorithm. *Computers and Chemical Engineering* 22 (10): 1461-1481.

Çağnan, Z. and R.A. Davidson. 2004. Post-earthquake Restoration Modeling of Electric Power Systems. In *Proceedings of* the *13th World Conference on Earthquake Engineering*, paper no. 109, Vancouver, B.C., Canada.

Chipperfield, A., P. Fleming, H. Pohlheim, and C. Fonsedca. 1994. *Genetic Algorithm Toolbox For Use With Matlab, Users* Guide*, Version 1.2*. University of Scheffield, U.K.

Davidson, R.A. and A. Çağnan. 2004. Restoration Modeling of Lifeline Systems. *Research Accomplishments 2003-2004*, Multidisciplinary Center for Earthquake Engineering, Buffalo, NY.

Giguère, P. and D.E. Goldberg. 1998. Population Sizing for Optimum Sampling with Genetic Algorithms: A Case Study of the Onemax Problem. In *Proceedings of the Third Annual Genetic Programming Conference*, Madison, 1 – 9, WI.

Hoerl, A.E. and R.W. Kennard. 1970. Ridge Regression: Baised Estimation for Nonorthogonal Problems. *Technometrics* 12: 55-67.

Kodiyalam, S., S. Nagendra, and J. DeStafano. 1996. Composite Sandwich Structure Optimization with Application to Satellite Components, *AIAA Journal* 34 (3): 614-621.

Lee, Y.H. and F. Azadivar. 1985. An Application of Optimization-by-Simulation To Discrete Variable Systems. In *Proceedings of the 1985 Winter Simulation Conference*, Ed. D. Gantz, G. Blais, S. Solomon, 173-177. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Mallows, C.L. 1973. Some Comments on C_p, *Technometrics* 15: 661-675.

Montgomery, D.C. and E.A. Peck. 1992. *Introduction to Linear Regression*, New York, NY: Wiley.

Nair, P. B., A.J. Keane, and R.P. Shimpi. 1998. Combining Approximation Concepts With Genetic Algorithm-Based Structural Optimization Procedures. In *Proceedings of the 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, AIAA-98-1912.

Shi, L., C. Chen, and E. Yücesan. 1999. Simultaneous Simulation Experiments and Nested Partition for Discrete Resource Allocation in Supply Chain Management. In *Proceedings of the 1999 Winter Simulation Conference*, Ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, 395-401. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Swisher, J.R., Jacobson, S.H., Hyden, P.D., and Schruben, L.W. 2000. A Survey of Simulation Optimization Techniques and Procedures. In *Proceedings of the 2000 Winter Simulation Conference*, Ed. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 119-128. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Tompkins, G. and F. Azadivar. 1995. Genetic Algorithms in Optimizing Simulated Systems. In *Proceedings of the 1995 Winter Simulation Conference*, Ed. C. Alexopoulos, K. Kang, W.R. Lilegdon, and D. Goldsman, 757-762. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Wahba, G., G.H. Golub, and C.G. Health. 1979. Generalized Cross-validation as a Method for Choosing a Good Ridge Parameter. *Technometrics* 21: 215-223.

AUTHOR BIOGRAPHIES

SETH D. GUIKEMA, Ph.D. is a Postdoctoral Researcher in the School of Civil and Environmental Engineering at Cornell University. He has a B.S. in Civil and Environmental Engineering from Cornell University (1997), a M.E. in Civil Engineering from the University of Canterbury (1999), a M.S. in Civil and Environmental Engineering from Stanford University (1999) and a Ph.D. in Management Science and Engineering from Stanford University (2003). His research interests include risk and decision analysis, Bayesian probability, game theory, and discrete optimization and their application in assessing and managing the risk to infrastructure systems from natural and human-induced hazards. His e-mail address is `<sdg4@cornell.edu>`.

RACHEL A. DAVIDSON, Ph. D. is an Assistant Professor in the School of Civil and Environmental Engineering at Cornell University. She has a B.S.E. from Princeton University (1993), and an M.S. and Ph.D. from Stanford University (1994, 1997), all in civil engineering. Her research focuses on natural disaster risk assessment and management and infrastructure systems modeling. Her e-mail address is `<rad24@cornell.edu>`.

ZEHRA ÇAĞNAN is a Ph.D. Candidate in the School of Civil and Environmental Engineering at Cornell University. She has a B.Eng. in Civil and Environmental Engineering from University College London, University of London (1999) and a M.S. in Civil Engineering from Middle East Technical University (2001). Her research interests include modeling post-earthquake restoration processes of electric power and water supply systems and discrete event simulation. Her e-mail address is `<zc32@cornell.edu >`.