USING A NEURAL NETWORK AS A FUNCTION EVALUATOR DURING GA SEARCH FOR RELIABILITY OPTIMIZATION

DAVID W. COIT AND ALICE E. SMITH[1]
Department of Industrial Engineering
University of Pittsburgh, Pittsburgh, PA 15261

ABSTRACT:
This paper demonstrates the use of a neural network as a reliability estimator for calculation of the objective function value during genetic algorithm (GA) search. The GA searches for the lowest cost system design by selecting the appropriate components and levels of redundancy. Using a neural network approximation for system reliability is computationally efficient for optimization problems where calculation of the objective function is impractical.

INTRODUCTION

This paper presents an optimization approach using a genetic algorithm (GA) to identify the preferred choice of components and the optimal levels of redundancy for a reliability design problem. The problem is a combinatorial optimization problem where reliability goals are achieved by discrete choices made from available parts. For complicated design problems, determination of the reliability of a given solution (i.e., system configuration) can require considerable effort or even Monte Carlo simulations. Since GA require numerous objective function evaluations to calculate fitness, a problem where the evaluation of the objective function is computationally time consuming may seem ill-fitted to GA. Our approach to this barrier is to develop a neural network approximation of system reliability.

Design of a hardware system involves numerous discrete choices among available components based on cost, reliability, weight, etc. If the objective is to minimize cost for a certain reliability requirement, then a strategy is required to identify the optimal combination of components. This is an NP-hard problem (Chern, 1992). When there are many functionally similar components to choose from, it becomes increasingly difficult to find the optimal solution, particularly when redundancy is considered as a strategy to enhance reliability. Redundancy is the use of functionally similar (but not necessarily identical) components such that if one fails, the redundant component will be available to perform the required function.

Figure 1 shows a typical series-parallel system. k-out-of-n redundancy is defined as a series of n parallel components where any k are required to be operating for the system to avoid a failure. The total number of components in parallel, n_i, for each function is a variable which is an integer value greater than or equal to k_i (for the i^{th} subsystem). k_i, the required number of components for a given subsystem, is specified while n_i remains a variable to be determined through the search.

[1] Corresponding author.

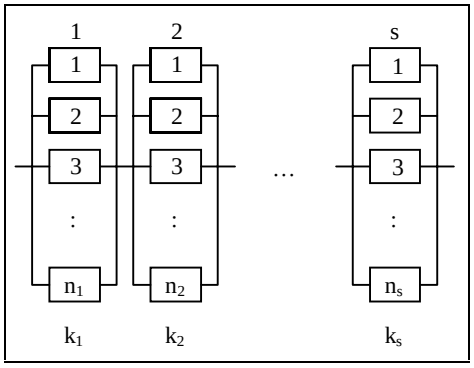

Figure 1. Typical Series-Parallel System.

The problem for a series-parallel system can be stated formally as:

$$\min \quad \sum_{i=1}^{s} C_i(\mathbf{x}_i)$$

$$\text{subject to} \quad \prod_{i=1}^{s} R_i\left(\mathbf{x}_i \middle| k_i\right) \geq R$$

$$\sum_{j=1}^{m_i} x_{ij} \geq k_i \;\; \forall \; i$$

C_i and $R_i(\mathbf{x}_i|k_i)$ are the cost and reliability of i^{th} subsystem, R is the system reliability constraint and $\mathbf{x_i}$ is the solution vector composed of x_{ij} values, the index of the j^{th} available component used in subsystem i. For most problems, it is practical to establish an upper bound on n (n_{max}). Then, the total number of possibilities can be computed by treating the selection of parts for each function as an occupancy problem (Feller, 1968). The number of available components for each subsystem i is defined as m_i. For a small problem with $s = 6$, $m_i = 10$ ($\forall$ i) and $n_{max} = 8$, there are 6.9×10^{27} possible design configurations. Clearly, a non-enumerative optimization strategy is required to identify the optimal solution.

Previous reliability optimization studies (Nakagawa, 1981; Bulfin, 1985; Misra, 1991; Gen, 1993) use integer or dynamic programming. When using dynamic or integer programming, it is necessary to restrict the search space to solutions of a particular form (Coit and Smith, 1995). As a result, the global optimum solution may not be found. GAs have also been applied to this problem (Coit and Smith, 1995; Painton and Campbell, 1994; Ida, et al, 1994). They treated the system reliability determination directly during GA search by either algorithms or by simulation. This is inefficient but the GA approach has no restrictions on the form of the solutions and has produced superior results.

For each design configuration under consideration, an estimate of system reliability is required. For complex designs (e.g., all-terminal networks, systems with multiple *k*-out-of-*n* redundancies and non-identical components), determination of exact solutions is computationally complex. System reliability is usually determined using probability theory or estimated via simulation. Use of probability theory is clearly preferred, and it is recommended for small and moderate sized problems. However, when design problems include significant usage of redundancy,

determination of an analytical solution can become very cumbersome, particularly when using a GA which explicitly and continually varies the amount of redundancy in search of the best solution. The below equation calculates system reliability.

$$R_s(\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_s}|\mathbf{k}) = \prod_{i=1}^{s} R_i\left(\mathbf{x_i}|k_i\right) = \prod_{i=1}^{s}\left(1 - \sum_{l=0}^{k_i-1} \sum_{S_l} \prod_{j=1}^{m_i} \binom{x_{ij}}{t_j} r_{ij}^{\,t_j}\left(1-r_{ij}\right)^{x_{ij}-t_j}\right) \quad (1)$$

where, $S_l = \left\{ \mathbf{t} \in \Re^{m_i} \left| \sum_{j=1}^{m_i} t_j = l \right. \right\}$

$\mathbf{t} = (t_1, t_2, \ldots, t_{m_i})$

$t_j =$ quantity of the j^{th} component functioning properly (unfailed) $(0 \le t_j \le x_{ij})$

$r_{ij} =$ reliability of the j^{th} component available for the i^{th} subsystem

Once a design configuration has been established, Monte Carlo simulation is often used, in lieu of analytical methods, to provide an estimate of system reliability. While simulation is an excellent tool to estimate reliability of a static design configuration, it is not efficient for use in a GA because of the need to recompute system reliability for each solution encountered during the search.

For large problems where determination of analytical solutions is difficult and simulation is inefficient, an alternative is the use of neural networks. Neural networks are a nonlinear robust modeling technique which are developed, or trained, based on either analytical or simulated results for a subset of possible solutions. The resulting model is then developed to estimate system reliability as a function of the component reliabilities and the design configuration. Values of k and n and individual component reliability values are directly input into the neural network. In this way, multiple estimates of system reliability are available without solving a new probability model for each new candidate solution, or performing the multiple iterations required with simulation. A disadvantage of using neural networks as a reliability evaluator is that the reliability prediction is only an estimate which may be subject to bias or variance depending on the adequacy of the neural network.

GENETIC ALGORITHM FORMULATION

A particular solution is characterized by a vector of dimension $s \times n_{max}$. For each subsystem, the total number of selected components (n_i) are ordered from most reliable to least reliable and followed by (n_{max} - n_i) of the m_i + 1 index corresponding to no further use of redundancy. For example, consider a system with s = 3, m_i = 6 ($\forall$ i) and n_{max} = 5 for each subsystem. The vector, **v** = (1 1 6 7 7 3 7 7 7 7 2 2 7 7 7) represents a solution where two of the most reliable and one of the 6th most reliable parts are in parallel in the first subsystem, the 3rd most reliable part (for that particular function) provides the second subsystem and two of the second most reliable part provides the third subsystem.

We examined population sizes of 50 and 100 based on our problem size and complexity. The GA search is started by randomly selecting an initial population.

For each subsystem a random number, say n', between k_i and n_{max} is selected, and then n' of the m_i different alternatives (with replacement) are randomly selected. This is repeated for each of the s subsystems, and then for each member of the population. Finally all parts within any subsystem are ordered from most to least reliable and indices are added for potential redundancies which are not filled. The objective function is the sum of the total cost for the selected parts plus a quadratic penalty function similar to that devised in Smith and Tate (1993). The penalty function is applied when the reliability estimation does not meet the requirement.

Parents were selected in the manner of Tate and Smith (1995) and the breeding operation was uniform crossover. The mutation operation involved uniform selection and took place after breeding and culling so that each mutated solution remained in the population for at least one generation. We used probabilities of a solution be selected for mutation from 25% to 75%. The mutation rate was set at 0.10, so that with a 10% probability, each element in the solution vector was exposed to mutation. If selected to be mutated, there was a 50% chance that the element is assigned an index of $m_i + 1$, corresponding to no component, and a 50% chance that a random component index from the m_i alternatives was selected.

After new solutions were generated, the associated reliability was estimated using the neural network model. Only the best among the parents and children were kept for the next generation, i.e., inferior solutions were culled to maintain constant population size. We terminated our GA after 500 generations for populations of 100 and after 1000 generations for populations of 50, resulting in 5×10^4 (non-unique) solutions generated.

NEURAL RELIABILITY ESTIMATION

For this research, a backpropagation neural network was developed to estimate the reliability of single k-out-of n subsystems based on k, n and n independent part choices and their associated reliability levels. The data set used to train and test the neural network was chosen using a full-factorial design of the critical parameters k, n and three underlying distributions (uniform, skewed-left and skewed-right) where reliability ranged from 0 to 1. The skewed distributions were necessary to expose the neural network to test cases with relatively high and low system reliability.

Using 8 as an upper bound for k and n, there are 192 different combinations ($8 \times 8 \times 3$). Analytical calculations of k-out-of n reliability (using Equation 1) were made for 50 randomly chosen instances for each cell in the factorial design, resulting in a total of 9,600 data vectors. 90% of these data vectors were used for neural network training while the remainder were used for validation.

When considering performance and efficiency, the best neural network was found to consist of inputs for k and each individual part reliability (up to n_{max}), one hidden layer with 15 hidden neurons and a single output (the estimation of subsystem reliability). The resulting network had a mean absolute error of 0.00484 and an RMS error of 0.00816 for estimating reliability over the validation set. This was an excellent fit to the analytical data and indicates that the neural network model can serve as a very acceptable function evaluator for the GA.

Although the neural network provided an excellent fit, there is still always some error associated with each prediction and some conditions must be introduced on the use of the neural network as a function evaluator. This approach can best be

applied if the constraint is somewhat "soft" and small deviations can be allowed, or alternatively, a pseudo-constraint can be determined by making a small conservative adjustment to the constraint to assure that the original constraint is met even if there is a small violation of the pseudo-constraint.

EXAMPLES AND RESULTS

We first considered a system with a reliability requirement of 0.80. For 6 subsystems, there were $m = 10$ different part choices and $n_{max} = 8$. Table 1 presents the reliability values and costs associated with each component alternative. There are greater than 6.9×10^{27} different possible design configurations, while our GA search examined $< 5 \times 10^4$ solutions. The system has a significant need for redundancy, as indicated by the values of *k* in the table, and is a complex combinatorial problem.

By varying the size of the population and the relative percentage of breeding and mutation, we obtained different results (Table 2). For each algorithm, 8 different random number seeds were used, and the minimum cost, mean cost and standard deviation are shown. It is evident that the GA was robust across the parameter alterations tested. Because of the size of the problem, the optimum solution could not be enumerated, however when generating 1 million random solutions, the best solution of 1308 found by the GA was 4.5 standard deviations from the mean feasible random solution (p = 0.0001%).

We wanted to more stringently evaluate the results of our GA search using the neural network in function evaluation, so we selected the best combination of parameters and developed a smaller test problem, consisting of the first two subsystems listed in Table 1. All other problem specifics remained the same. The search space of this smaller problem was 1.9×10^9 and we enumerated to find the optimal solution. We ran our GA with 10 different seeds, and all converged to the optimal design configuration within 1000 generations.

Table 1. Reliability Values and Costs for Each Component for Test Problems.

		Part Alternatives - Unit Reliability									
subsystem	k	1	2	3	4	5	6	7	8	9	10
1	4	0.98	0.93	0.73	0.72	0.71	0.70	0.66	0.62	0.60	0.35
2	2	0.93	0.92	0.89	0.86	0.84	0.81	0.61	0.43	0.39	0.34
3	1	0.94	0.88	0.85	0.76	0.73	0.62	0.60	0.59	0.34	0.31
4	1	0.93	0.67	0.63	0.62	0.62	0.48	0.41	0.41	0.39	0.32
5	2	0.95	0.95	0.90	0.86	0.67	0.66	0.64	0.54	0.38	0.38
6	3	0.96	0.85	0.84	0.76	0.75	0.66	0.65	0.61	0.50	0.48
		Part Alternatives - Unit Cost									
subsystem	k	1	2	3	4	5	6	7	8	9	10
1	4	95	86	80	75	61	45	40	36	31	26
2	2	137	132	127	122	100	59	54	41	36	30
3	1	118	113	108	59	54	49	45	35	30	25
4	1	149	84	74	69	64	58	38	31	26	21
5	2	131	120	103	93	60	43	36	31	26	21
6	3	149	104	96	79	45	40	35	30	25	20

Table 2. Variations of Genetic Algorithm Parameters Used on Larger Test Problem.

# Gen.	Popul-ation	Breed %/ Mut. %	Min. Cost	Avg. Cost	Std. Dev.	Coef. Variation
500	100	50/50	1318	1348.63	25.46	0.0189
1000	50	50/50	1308	1337.50	21.76	0.0163
1000	50	75/25	1308	1375.75	40.72	0.0296
1000	50	25/75	1318	1374.34	32.22	0.0234

CONCLUSIONS AND FUTURE WORK

We have formulated and tested a unique approach to the optimization of complex reliability design problems using a GA and a neural network approximation of the system reliability. Clearly more research is needed on the effectiveness and relative efficiency of this approach. We are working on a more difficult "all terminal network" reliability problem where the system reliability cannot be calculated directly and must be simulated. We are also pursuing a hybrid objective function evaluation approach where the neural network estimation is used early in the search, and a more exact method is used directly in later phases of the search.

REFERENCES

Bulfin, R. L., Liu, C. Y., (1985). Optimal allocation of redundant components for large systems, *IEEE Trans. Reliability*, vol R-34, Aug, pp 241-247.

Chern, M. S., (1992). On the computational complexity of reliability redundancy allocation in a series system, *Operations Research Letters*, vol 11, Jun, pp 309-315.

Coit, D. W., Smith, A. E., (1995). Optimization approaches to the redundancy allocation problem for series-parallel systems, *Proc. of the 4th Industrial Engineering Research Conference*.

Feller, W., (1968). *An Introduction to Probability Theory*, John Wiley & Sons.

Gen, M., Ida, K., Tsujimura, Y., Kim, C., (1993). Large-scale 0-1 fuzzy goal programming and its application to reliability optimization, *Computers and Industrial Engineering*, vol 24, pp 539-549.

Ida, K., Gen, M., Yokota, T., (1994). System reliability optimization with several failure modes by genetic algorithm, *Proc. of 16th International Conference on Computers and Industrial Engineering*, Mar, pp 349-352.

Misra, K. B., Sharma, U., (1991). An efficient algorithm to solve integer programming problems arising in system reliability design, *IEEE Trans. Reliability*, vol 40, Apr, pp 81-91.

Nakagawa, Y., Miyazaki, S., (1981). Surrogate constraints algorithm for reliability optimization problems with two constraints", *IEEE Trans. Reliability*, vol R-30, Jun, pp 175-180.

Painton, L., Campbell, J., (1994). Identification of components to optimize improvements in system reliability, *Proc. of the SRA PSAM-II Conf.*, Mar, pp 10-15 - 10-20.

Smith, A. E., Tate, D. M., (1993). Genetic optimization using a penalty function, *Proc. of the 5th International Conf. on Genetic Algorithms*, pp 499-505.

Tate, D. M., Smith, A. E., (1995). A genetic approach to the quadratic assignment problem, *Computers and Operations Research*, vol 22, pp. 73-83.