USE OF A GENETIC ALGORITHM TO OPTIMIZE A COMBINATORIAL RELIABILITY DESIGN PROBLEM

David W. Coit and Alice E. Smith[1]
Department of Industrial Engineering
1048 Benedum Hall
University of Pittsburgh
Pittsburgh, PA 15261
412-624-9830
aesmith@engrng.pitt.edu

ABSTRACT

We apply a genetic algorithm (GA) to optimize a reliability design problem using a neural network as a reliability estimator for calculation of the objective function value during the search. The GA produces the lowest cost option for a required system reliability by selecting the appropriate parts and levels of redundancy. We show that the neural network estimator is effective and computationally efficient, and that the GA optimization approach is robust for reliability design of complex systems.

INTRODUCTION

Determination of an optimal or near-optimal design configuration which achieves a stated reliability requirement at a minimum cost is of utmost importance to system designers. This paper presents an optimization approach using a genetic algorithm (GA) to efficiently identify both the preferred choice of parts and the optimal levels of redundancy (if any) to be used. Distinguishing this research from related work is that part selection is treated as a combinatorial problem where reliability goals are to be achieved based on discrete choices made from among available parts. Further distinguishing this work is the consideration and implementation of artificial neural networks as a means to evaluate reliability of complex system designs.

Background

Design of a hardware system involves numerous discrete choices among available part types based on cost, reliability, performance, lead-time, vendor reputation, etc. A successful design program involves the simultaneous selection and integration of parts which then collectively achieve reliability and performance objectives at an acceptable price. When a formal reliability program is in effect, quantitative reliability requirements are established in terms of either the reliability (i.e., probability of

operating over a stated time period without failure) or a mean time between failure (MTBF). If the design objective is to minimize cost for a certain reliability requirement, or to maximize reliability within cost constraints, then a strategy is required to identify the optimal combination of parts. When there are many functionally similar parts to choose from, it becomes increasingly difficult to identify the optimal solution, particularly when redundancy is considered as a strategy to enhance reliability.

Many previous optimization studies concerned with improving system reliability, summarized in [1], use techniques such as nonlinear programming to identify optimal reliability levels at the part or subsystem level. In general, these studies have considered system and part reliability to be continuous random variables, and have determined optimal levels of part or subsystem reliability to maximize (or minimize) an objective function based on design (e.g., size, weight) or cost constraints. The designer then uses this information to establish lower-level requirements, to "design-in" reliability to the optimal level and to assist in part selection. However, there exists another class of problems where system reliability must meet or surpass a specified requirement (or be maximized), but part level choices are limited to existing products which have specified reliability (for a given time) and cost. In these cases, the design problem becomes a combinatorial optimization problem. The designer must choose from among the parts currently available from existing vendors, which each typically offer a series of functionally similar parts at different reliability levels, and of course, costs.

The use of redundancy is one method to enhance reliability. For the purposes of this paper, redundancy is defined as the use of functionally similar (but not necessarily identical) components together in a design such that if one component fails, the redundant part will be available to perform the required function without the system experiencing a failure. k-out-of-n redundancy is defined as a series of n parallel parts where any k of the n components

---

[1]Corresponding author.

are required to be operating for the system to avoid a failure.

Problem Statement

Consider the reliability block diagram in Figure 1. The designer has determined the required subsystems and their associated performance requirements, in this case five subsystems (s = 5). Each subsystem represents a required function which must be provided by a part selected by the designer from a finite number of alternatives. In the figure, when two parts are shown in parallel with notation such as 2/6 underneath, it means that two (out of 6) of these parts are required for the subsystem to function, that is k/n. If there are no redundant parts, then k = n = 2, and both parts are required. For each required subsystem there are m alternatives or options available, and for each alternative, there is a known reliability and cost. If there are m possible alternatives for each required subsystem in Figure 1, then there are

$$m^4 \left(\frac{m!}{9} - 1 \right) m^2 \qquad (1)$$

possible design configurations, without redundancy, which can be considered. Even for moderately high values of m, the total number of possible design configurations is manageable, and it is not inconceivable to consider all possible combinations and explicitly identify the design configuration which meets the reliability requirement at the minimum cost.

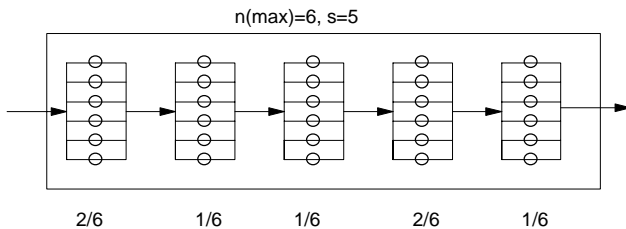n(max)=6, s=5

2/6   1/6   1/6   2/6   1/6

Figure 1. Schematic of System Under Analysis.

Expansion of the problem to consider the use of redundancy results in a much more complex problem. Considering the general case of Figure 1, the total number of parts in parallel, n_i, for each function is now a variable which can assume any integer value greater than or equal to k_i. For this case and subsequent examples, it is assumed that k_i, the required number of parts for a given subsystem has been specified, while n_i remains a variable to be determined as part of the optimization process.

If no limits or upper bounds are established for n, then the total number of possible design configurations is unbounded. For some design problems, such as the

selection of phased array modules in a phased array radar, the number of parts in parallel can take very high values, but for the great majority of design problems, it is practical to establish an upper bound on n (n_{max}). In Figure 1, n_{max} is 6 for each subsystem. Once upper limits have been established, then it becomes possible to compute the total number of possibilities by treating the selection of parts for each function as an occupancy problem [2]. If the system in Figure 1 includes s subsystems, then the total number of possible design configurations is equal to

$$\prod_{i=1}^{s} \left(\frac{m_i + n_{max}}{m_i} - 1 \right) \qquad (2)$$

For a moderately sized problem with s = 6, m_i =10 ($\forall$ i) and n_{max} = 8, there are greater than 7.0 x 10^{27} possible design configurations. Clearly, a heuristic optimization strategy is required to identify the optimal or "very good" solutions by explicitly searching a much smaller subset of the total number of solutions.

Optimization Issues

There are two primary challenges which must be resolved to solve this reliability optimization problem. First, for each design configuration under consideration, an estimate of reliability is required as a function of the part reliabilities and the reliability model corresponding to the particular configuration of serial and parallel parts. For complex system designs employing multiple k-out-of-n redundancies with non-identical part types in parallel, determination of exact analytical solutions is computationally complex. Second, since the problem being addressed here is a combinatorial design problem, search strategies depending on gradient information are not applicable, and other means are required.

Evaluation of the objective value of a given solution requires an estimation of the system reliability to establish whether the reliability requirement has been met, i.e. is feasible. System reliability is usually determined exactly using probability theory, or estimated using Monte Carlo simulation. Use of probability theory to obtain exact solutions for system reliability is clearly the preferred approach, and it is the recommended approach for all small and moderate sized problems. However, when design problems include significant amounts of redundancy, determination of the analytical solution can become very cumbersome. This is a particularly salient point when devising optimization search strategies which explicitly and continually vary the amount of redundancy in search of the best solution. For each iteration considered as part of the optimization search, a different probability model must be solved.

Once a design configuration has been established, Monte Carlo simulation provides an accurate method to compute the system reliability of very complex system designs. Most practitioners use simulation in lieu of exact analytical solutions when predicting the reliability of complex systems. While simulation is an excellent tool to estimate reliability of a static design configuration, it is not particularly efficient for use in a heuristic optimization search because of the need to recompute system reliability for each solution encountered during the search. (Each recomputation would require multiple simulation runs.)

For large problems where determination of exact analytical solutions is difficult and simulation is computationally inefficient, an alternative is the use of artificial 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. A resulting single model is then developed to predict system reliability as a function of the part reliabilities and the design configuration. Values of k and n are directly input into the model. 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.

Several assumptions were made in this analysis. In each instance, the assumption was made to contain the size of the overall problem and could be relaxed. The assumptions made were:

1. There is a specific time period of interest, such as a mission time or a warranty period, for the design problem. This assumption allows all analyses to be conducted on the basis of the reliability for that specific time period. By making this assumption, it is not required to address the underlying time to failure distribution and the time dependent nature of part hazard rate.

2. The part reliabilities are known and constant for each of the available parts.

3. Failure of individual parts are independent.

4. All redundancy is active redundancy without repair. This is the most pessimistic case and no information on failure detection accuracy, inception intervals or repair rates are required.

5. Each of the required subsystems is critical for mission success. Therefore, the overall system model has each of the subsystems in a series configuration. Within each subsystem, the parts can be in any k-out-of-n parallel configuration or a series of k-out-of-n parallel configurations.

GENETIC OPTIMIZATION MODEL

A genetic algorithm is a heuristic optimization search technique patterned after the natural selection process taking place during biological evolution. The basic GA methodology consists of (a) random selection of an initial population of solutions, (b) calculation of fitness (objective function value) for each solution, (c) breeding of existing solutions to create new solutions, (d) mutation of existing solutions to create new solutions, and (e) culling of inferior solutions. This proceeds in an iterative manner until the search converges. GAs have proven to be effective for problems characterized by non-convex search areas.

Solution Encoding

A particular system design is composed of s distinct subsystems. For each subsystem s_i, there are m_i different part alternatives unique to that subsystem. The designer may select any one of these or place two or more of these parts, in any combination, in parallel. In practice, the parts may represent any functional unit used to build the subsystem. The number of possible parts which can be placed in parallel is bounded by a practical constraint, n_{max}. For each function, the m_i alternatives are ordered from most reliable to least reliable and indexed from 1 to m_i. The option of not using an additional part in parallel receives an index of $m_i + 1$.

A particular solution is characterized by a vector of dimension $\sum_{i=1}^{s} n_{max_i}$. For each function, the total number of selected parts (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 following vector

$$\mathbf{v} = (1\ 1\ 6\ 7\ 7\ 3\ 7\ 7\ 7\ 7\ 2\ 2\ 7\ 7\ 7)$$

represents a possible 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.

Initial Population Selection

GAs use populations, i.e. a set, of solutions to search multiple directions in the search space simultaneously. We examined population sizes of 50 and 100; these were chosen arbitrarily by considering our problem size and complexity. The GA search is started by randomly selecting an initial population. For each of the subsystems, the designer has specified the required number of parts (k_i) for the system to operate. 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 the indices are added for blanks (potential redundancies which are not filled). We found that sequencing by reliability improved the efficiency of our search operators.

Objective Function

The objective function is the sum of the total cost for the selected parts plus a quadratic penalty function. The penalty function is applied when the reliability prediction does not meet the reliability requirement, and thus, does not represent a feasible solution. It was important to allow infeasible solutions into the population because good solutions are often the result of breeding between a feasible and an infeasible solution, and the optimal solution can sometimes only be approached from an infeasible region for highly constrained problems [3]. The penalty function (P) is given by

$$P = \delta \, (500 \, (R - R_s))^2 \qquad (3)$$
$$\text{where} \quad \delta = 0, \, R \geq R_s$$
$$\delta = 1, \, R < R_s$$
$$R = \text{predicted reliability}$$
$$R_s = \text{system reliability requirement}$$
$$500 = \text{empirically determined constant.}$$

Breeding

The reproductive process of breeding involves the selection of parents and application of a breeding operation. Breeding explores the search space by using information on the objective function values of previously considered solutions. Selection of parents to reproduce is based on the fitness (i.e. the objective function value) of the members of the population. As in natural evolution, those members of the population who are fitter are more likely to reproduce. Parents were selected by taking a uniform random number between 1 and the square root of the population size, and

then selecting the parent with an objective function rank closest to the square of the selected random number. Two parents were required for breeding and the algorithm forbade a parent breeding with itself.

The breeding operation retained identical portions of the parents, and then randomly selected, with equal probability, from the two parents for those parts which differ. Because the parts were ordered from most to least reliable for each subsystem, matches were fairly common. The child is then ordered from most to least reliable. For example, consider the two following parent vectors ($\mathbf{v}_1$, $\mathbf{v}_2$) and resulting child vector ($\mathbf{v}_c$),

$\mathbf{v}_1 = (1\ 1\ 6\ 7\ 7\ 3\ 7\ 7\ 7\ 7)$
$\mathbf{v}_2 = (2\ 3\ 6\ 6\ 7\ 2\ 2\ 7\ 7\ 7)$
_____

$\mathbf{v}_c = (X\ X\ 6\ X\ 7\ X\ X\ 7\ 7\ 7)$ - identical indices
$\mathbf{v}_c = (2\ 1\ 6\ 7\ 7\ 3\ 2\ 7\ 7\ 7)$ - random selection of non-matching indices
$\mathbf{v}_c = (1\ 2\ 6\ 7\ 7\ 2\ 3\ 7\ 7\ 7\ )$ - final arrangement

Mutation

Mutation perturbs solutions to expand the search space in a random manner. Our mutation operation took place after breeding and culling so that each mutated solution remained in the population for at least one generation. This was important to promote diversity among the population. The population members to be mutated were uniformly selected. The mutated member replaced the original member in the population, except if the pre-mutated solution was the best solution in the population.

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 part, and a 50% chance that a random part index from the m_i alternatives was selected.

Evolution

After new offspring solutions were generated, the associated reliability was estimated using the neural network model as described later and the objective function value was determined. 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 from generation to generation.

Genetic algorithms continue until a pre-determined stopping criteria has been met. The criteria may be based on the total number of generations or when improvement in the best solution ceases. We terminated our GA after 500 generations for populations of 100 and after 1000 generations for populations of 50.

NEURAL RELIABILITY PREDICTION

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 m independent part choices and their associated reliability levels. The system reliability was then computed as a function of the neural network output for each required subsystem.

The data set used to train and test the neural network was simulated based on a full-factorial design of the critical parameters k, n and three underlying distributions (uniform, skewed-left and skewed-right) in equal proportions for component reliabilities, where reliability ranged from 0 to 1. Using eight as an upper bound for k and n, there are 192 different combinations (8*8*3). Analytical calculations of k-out-of n reliability were made for fifty 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. Therefore, most of the analytically generated reliability data was used to construct the estimation model (neural network), and the model was validated on a different set of data than that used in its construction.

Three separate classes of network architectures were evaluated for possible implementation as a reliability function evaluator in the genetic optimization model. The three classes were (1) a series of independent networks, each pertaining to a specific value of k, with n_{max} part reliability inputs and a single output for system reliability, (2) a single network using k as an input in addition to the n_{max} part reliabilities and outputing a single system reliability, and (3) a single network using part reliability inputs and outputing n_{max} system reliabilities, each pertaining to a specific value of k.

When considering performance and efficiency, the best model was found to be of formulation 2 above, and consisted of inputs for k and each individual part reliability (up to n_{max}), one hidden layer with 15 hidden neurons and a single output. Network formulation 1 was inefficient because k neural networks had to be trained, validated and interfaced with the GA optimization. Network formulation 3 was efficient, but less accurate due to the difficulty of

learning multiple (k) estimations for a single given input vector. The resulting model 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 in the genetic optimization model.

EXAMPLE AND RESULTS

We considered a system with s = 6 and a reliability requirement of 0.80. For each subsystem there were m = 10 different part choices and n_{max} = 8. There are therefore greater than 7×10^{27} different possible design configurations, while our GA search examined about 5×10^4 solutions. Table 1 presents the reliability values and costs associated with each part alternative. The system has a significant need for k-out-of-n redundancies, as indicated by the values of k in the table, and is therefore, a fairly complex combinatorial problem.

By varying the size of the population and the relative percentage of breeding and mutation, we obtained different results (Table 2). Figure 2 shows the total cost as it varied with the number of generations for three different random number seeds. Convergence was fairly rapid, with our GA converging after about 100 generations for a population of 100 (approximately 10^4 solutions searched of the possible 7.0×10^{27}). For each algorithm, eight 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 we tested. However, since we cannot calculate an optimum solution for this test problem, we cannot compare our final solution quality in a meaningful way.
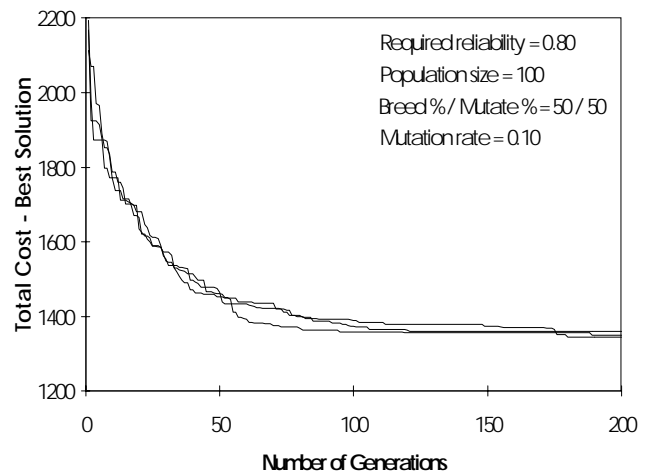

Figure 2. Total Cost for Three Seeds Over Evolution.

CONCLUSIONS

We have formulated and tested a unique approach to the optimization of complex reliability design problems using a a heuristic combinatorial optimization procedure and a neural network approximation of the system reliability calculation. We have shown the robustness of the approach to changes in the GA parameters. Clearly more research is needed on the effectiveness and relative efficiency of this approach. We would like to expand our study to examine scale-up issues, and would also like to optimize over other criteria along with cost.

REFERENCES

[1] C. Tillman, L. Hwang and W. Kuo, *Optimization of Systems Reliability*, Marcel Dekker, New York (1980).

[2] William Feller, *An Introduction to Probability Theory*, John Wiley & Sons, New York (1968).

[3] Alice E. Smith and David M. Tate, "Genetic Optimization Using a Penalty Function," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 499-505 (1993).

BIOGRAPHICAL SKETCHES

David M. Coit is a PhD candidate at the University of Pittsburgh. He holds an MSIE degree from the University of Pittsburgh, an MBA from Rennselear Polytechnic Institute and a BSME degree from Cornell University. His research areas include development of reliability test plans and optimization. Previously, he worked for IIT Research Institute where he was involved in reliability program management and the development of reliability prediction models.

Alice E. Smith is Assistant Professor of Industrial Engineering at the University of Pittsburgh. Her research interests are in neural network modeling and heuristic optimization. Her research has been funded by NSF, Lockheed Corporation and the Ben Franklin Technology Center of Western Pennsylvania. She is a Senior Member of IIE.

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

subsystem	k	Part Alternatives - Unit Reliability									
		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
subsystem	k	Part Alternatives - Unit Cost									
		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 Test Problems and Their Results.

Generations	Population	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