CONF-930116--30

BNL--48250

DE93 005709

ALGORITHMS FOR OPTIMAL REDUNDANCY ALLOCATION*

J. VANDENKIEBOOM,[a] R. YOUNGBLOOD
Department of Nuclear Energy
Brookhaven National Laboratory
Upton, New York 11973

ABSTRACT

Heuristic and exact methods for solving the redundancy allocation problem are compared to an approach based on genetic algorithms. The various methods are applied to the bridge problem, which has been used as a benchmark in earlier work on optimization methods. Comparisons are presented in terms of the best configuration found by each method, and the computational effort which was necessary in order to find it.

INTRODUCTION

The optimal redundancy allocation problem is to find the allocation of available resources over design elements which optimize system reliability (best reliability available for a given set of constraints on resources). Typical constraints are those on the cost, reliability, weight or volume of the total system or on individual components of the system. Many authors have presented algorithms for optimal reliability and redundancy allocation problems and a broad survey of past approaches can be found in several reviews.[1,2]

This paper will compare the performance of several heuristic approaches previously applied with two different approaches, one based on a genetic algorithm, the other on a Boolean procedure. In particular we shall concern ourselves with the problem of how best to allocate resources in the bridge network problem to obtain optimal reliability while satisfying a total system cost constraint. The performance of these various algorithms will be compared in terms of the optimality of the solution as well as the computational work required to arrive at the solution. The computational work will be expressed as the number of unique function evaluations necessary during the entire run. For a large (realistic) problem, in which a single function evaluation might be nontrivial, this would be a measure of the real computational effort.

BRIDGE NETWORK PROBLEM

Many authors[3-6] have used the bridge network, shown in Figure 1, as a standard test problem for optimal systems reliability algorithms. Purely series networks and purely parallel networks are easy to optimize; this problem is "complex" in that it is neither series nor parallel and is not amenable to analytical solution. Each block in this figure is to be understood as a supercomponent consisting of n identical components in parallel. The component reliability and cost of a single element vary among the five supercomponents. The unreliability of the i^{th} supercomponent is then $Q_i = q_i^n$, where q_i is the unreliability of a single component of type i. The total system unreliability is:

$$Q_s = Q_1Q_3 + Q_2Q_4 + Q_1Q_4Q_5 + Q_2Q_3Q_5 - Q_1Q_2Q_3Q_4 - Q_1Q_3Q_4Q_5 - Q_1Q_2Q_3Q_5 - Q_1Q_2Q_4Q_5 - Q_2Q_3Q_4Q_5 + 2Q_1Q_2Q_3Q_4Q_5.$$

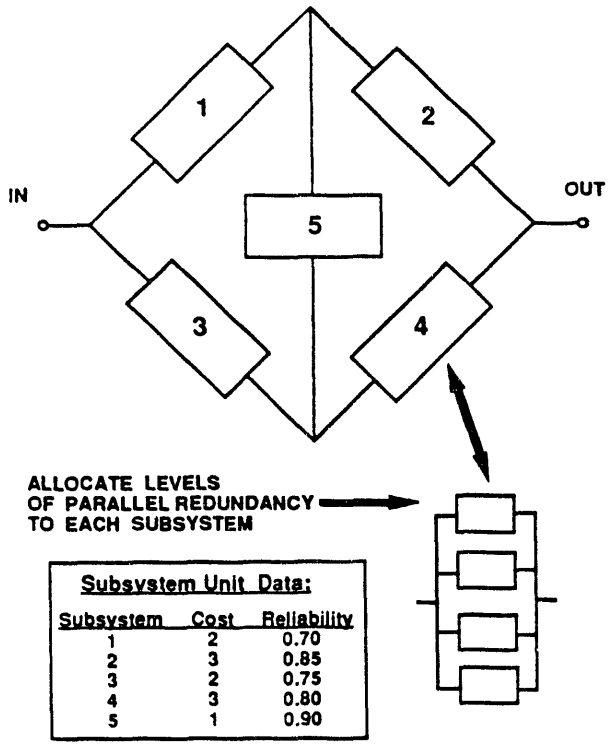
The total system cost is simply the sum of the costs of all components included in the configuration. We shall seek to minimize the system unreliability while maintaining the total system cost within the cost constraint.

---

* This work was performed under the auspices of the U. S. Department of Energy under Contract No. DE-AC02-76CH00016.

[a] Permanent address: The University of Michigan, Department of Nuclear Engineering, Ann Arbor, MI.

DISCLAIMER



Subsystem Unit Data:

Subsystem	Cost	Reliability
1	2	0.70
2	3	0.85
3	2	0.75
4	3	0.80
5	1	0.90

FIGURE 1. Bridge Network Problem

With the computational power offered by today's computers, a global optimal solution to the bridge network problem can be guaranteed quickly by examining all points in the search space. However, large problems of this type, with complex objective functions, still can present a computational challenge. The large number of possible configurations in the search space generally make exhaustive search prohibitively expensive. Alternative methods which can perform intelligent search within a large space are needed.

ALTERNATIVE METHODS

A. Lawler-Bell Algorithm

The Lawler-Bell algorithm is a partial enumeration method which can greatly reduce the size of the search space.[7] It can be applied to any problem with monotone objective functions and constraints. The algorithm takes advantage of this monotonicity and the natural lexicographic ordering of the solution vectors to skip over certain sections of the search space. This allows for a complete search of the space with far fewer objective function evaluations. The performance measure we are most interested in is the number of objective function evaluations, since for large scale problems, function evaluation may be a laborious computation.

B. Heuristic Methods

Many authors[3-6] have produced heuristic algorithms for the solution of redundancy optimization problems. These algorithms are basically hill-climbers in the sense that they begin with a trial configuration and build upon it in accordance with a rule which effectively takes the system "uphill" in reliability space. Choosing which supercomponent gets the next element is done on the basis of a quantitative measure reflecting both the cost of the added element and its importance to the existing configuration. Typically, this rule would be something like "add an element to the supercomponent which shows the greatest increase in reliability per unit cost." This procedure would be repeated until the available system resources are exhausted, that is, a system constraint is violated. These algorithms converge rather quickly with few objective function evaluations. They do not, however, guarantee the global optimal configuration.

C. Boolean Approach

The Boolean approach generates configurations having the property that no system cut set has a probability greater than a chosen cutoff.[8-10] This method could also be considered a heuristic, because it tends towards configurations which have high reliability, but without making direct use of the full system reliability function, and without guaranteeing global optimality. One begins by choosing a cut set probability cutoff which is significantly less than the hoped-for total system unreliability. Assuming independence of component failures, applying this requirement to the (1,3) cut set means choosing m,n such that $q_1^m x q_3^n <$ "cutoff." Developing such options for each cut set, and then combining them, leads to a set of minimal system-wide allocations which each have the property of keeping all individual cut sets below the cutoff. This is not the same as maximizing the system reliability; rather, it is simply an abstract way to require that a design be "balanced." Doing this for a series of progressively lower cutoffs, we obtain a collection of configurations reflecting a range of cost and unreliability. Applying a constraint at the cut set level seems artificial, but this approach performs surprisingly well on the bridge problem.

D. Genetic Algorithm

The genetic algorithm (GA) approach is based on ideas borrowed from the field of genetics and also incorporate some principles of Darwinian evolution. GAs are global search algorithms which work with a string coding of the parameter set and search from a population of points. The search uses only objective function information and is guided by stochastic operators as opposed to deterministic rules.[11] In some ways, it resembles simulated annealing. An initial set of configurations for the bridge problem is randomly generated. This initial "population" is then allowed to "breed" a new generation of configurations, by a process which is designed by analogy with sexual reproduction. First, two especially "fit" (cost-effective) configurations are selected for breeding (Darwinian survival-of-the-fittest). From the string codings of these "parent" configurations, string codings for their "offspring" are generated by two operations, crossover and mutation. These operations allow the "parent" strings to exchange their genetic information with each other. Given the right combinations of genetic material, the "offspring" produced by this exchange may be "fitter" than either "parent." Then, another two "parent" configurations are bred, and so on until the new population (i.e. next generation) is complete. This new generation is then allowed to breed, and the entire process is iterated.

Any configuration of the bridge network can be represented by a string of five integers, but not all strings of five integers correspond to useful solutions. For example, a solution of the form $(0,0,i,j,k)$, where $i,j,k>0$, is pointless, because the k elements in subsystem 5 do not contribute to success. In the present work, a coding scheme has been implemented to avoid consideration of such configurations. The approach to coding can have an enormous influence on how well the algorithm performs.

We use the following notation to represent system configurations:

$$P_1P_2P_3P_4:(n_1,n_2,n_3,n_4,n_5)$$

where: P_i = binary flag (0/1 = off/on) indicating whether path set i is active, i.e. contains at least 1 unit in each subsystem associated with that path.

n_i = number of redundant components in subsystem i.

This path set coding was used to eliminate non-consistent configurations from the population. The 4 path sets for the bridge network are listed in Table 1 below. We maintain at least one element in every subsystem that is part of an active path. If a subsystem is not in any active path then the number of elements is appropriately forced to zero. This is accomplished with the use of a template associated which each of the 15 possible combinations of active path sets. In this coding scheme, 6 of the 15 possible path combinations represent all path sets as being active, i.e. each subsystem of the bridge contains at least one element. This means that many notationally distinct path set configurations actually correspond to one: all paths being active. This feature is undesirable, and its effects are still being explored.

TABLE I. Path Sets

Path Set	Subsystems in Path Set
P_1	{1, 2}
P_2	{3, 4}
P_3	{1, 4, 5}
P_4	{2, 3, 5}

In the context of the cost-constrained problem, the "fitness" of a configuration depends both on its reliability and its cost. Initially, a configuration having a lower system unreliability (or a higher system reliability) is considered fitter. The selection of two parents from a population on the basis of better fitness functions is the first step of the genetic algorithm. These are then "mated" to produce offspring. Crossover is the main genetic operator used. We used a two-point crossover operator which involves cutting both parent strings at an two identical points within the (n_1,n_2,n_3,n_4,n_5) portion of the string. These two sections are then exchanged. Mutation is allowed to occur on any component position with a small probability (0.03). This involves increasing or decreasing, decided by a coin flip, the number of components by a single unit. After both crossover and mutation are complete, the offspring are checked for superfluity. If either offspring is superfluous - contains elements in path sets with are incomplete and therefore contribute nothing to performance - it is discarded and no objective function is calculated. This procedure effectively reduces the size of the search space. This process is repeated until the new population is filled. It is through this process

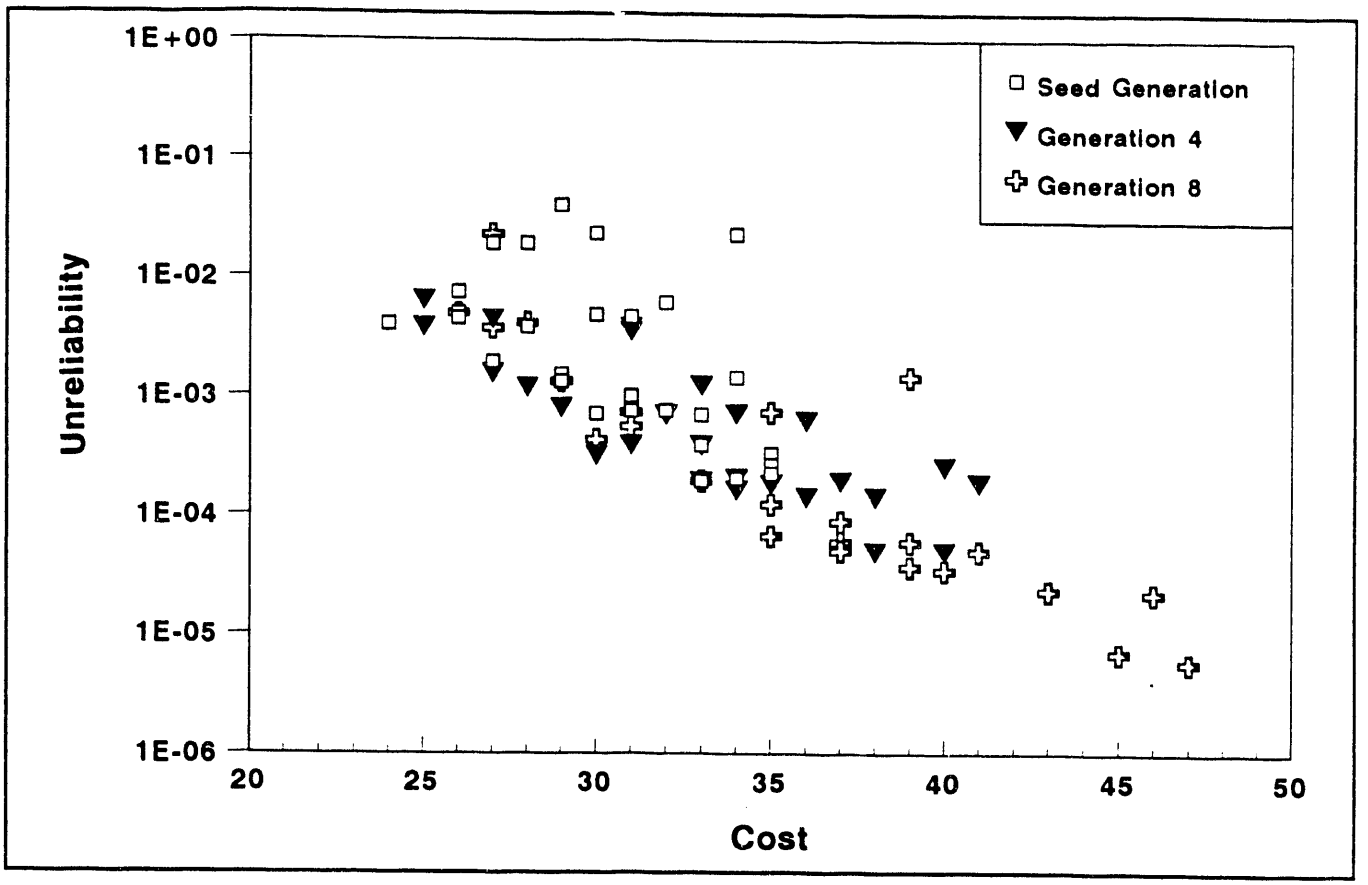


FIGURE 2. Typical Evolution of Genetic Algorithm Population

that we form new "fitter" configurations. Eventually, those elements which made one parent stand out will be combined with those different elements which made another parent stand out, so that the "offspring" contains all these outstanding elements and is fitter still.

The average fitness of the population increases in later generations. This is shown in Figure 2 for a typical run with a cost constraint of 35. The initial generation was seeded with configurations having low cost. Crossover and mutation drive the configurations of subsequent generations toward higher reliability and generally, higher costs. Without a factor which rewards cost-effectiveness, the population will evolve to a highly reliable but very expensive set of configurations.

Correspondingly, the objective function, or "fitness," used is the total system unreliability, with an additional factor accounting for the difference in a configuration's performance relative to that of other configurations of similar cost. Since an unrestrained selection process tends to drive configurations toward higher reliability and cost, a penalty function is used to maintain the population in the vicinity of the cost constraint.

RESULTS

Previous results for the bridge network problem using the specific component data given in Figure 1 with a total system cost constraint of 20 have been reported in the literature.[3-6] Some of these results are summarized in Table 2. The optimal configuration commonly found for this problem is (3,2,2,1,1). Referring back to Figure 1, this notation is to be interpreted as the configuration having three parallel elements in subsystem block 1, two parallel elements in subsystem blocks 2 and 3, and one element in subsystem blocks 4 and 5. This configuration, with a system unreliability of 6.78E-3, has been claimed as the global optimum. We have applied both the genetic algorithm and the Boolean procedure to this same problem and have found optimal configurations which are better than

TABLE II. Results

Algorithm	Cost Constraint	Best Configuration Found	Unreliability of Configuration	Global Optimum? (Yes/No)	Computational Effort for Result
Total Enumeration	20	(0,0,4,4,0)	5.50E-3	Yes	2628 func. eval.
Shi Algorithm	20	(3,2,2,1,1)	6.78E-3	No	5 iterations
Aggarwal Algorithm	20	(2,1,3,2,1)	7.90E-3	No	4 iterations
Genetic Algorithm	20	(0,0,4,4,0)	5.50E-3	Yes	0-40 generations 1-1600 func. eval.
Boolean Approach	20	(0,0,4,4,0)	5.50E-3	Yes	Not quantified
Total Enumeration	35	(4,3,4,3,1)	6.64E-5	Yes	26424 func. eval.
Shi Algorithm	35	(4,3,4,3,1)	6.64E-5	Yes	10 iterations
Aggarwal Algorithm	35	(3,2,5,4,1)	6.87E-5	No	10 iterations
Genetic Algorithm	35	(4,3,4,3,1)	6.64E-5	Yes	0-20 generations 1-1200 func. eval.

that given above. Both methods give the (0,0,4,4,0) configuration, with a system unreliability of 5.50E-3, as the best solution found, and an exhaustive search has verified this as the true global optimum. This solution cannot be found by the heuristics unless they are seeded with a configuration very close to the global optimum. Thus, for the component data given in Figure 1 and a cost constraint of 20, the bridge network should be replaced in favor of a simple series network composed of subsystem blocks 3 and 4, with each block consisting of four parallel and redundant units.

Numerical results of the different algorithms are given in Table 2. The various algorithms were applied to two problems with different cost constraints, 20 and 35. As seen above, the global optimal configuration for the problem with a cost constraint of 20, (0,0,4,4,0), involves only a single path set. For the component reliability and cost parameters used here, most choices of the cost constraint lead to optima which are single path sets. As noted previously, the heuristics are at a disadvantage in these cases. Therefore, the problem with a cost constraint of 35 was run to demonstrate the application of the different algorithms to a problem whose global optimum, (4,3,4,3,1), is a complete bridge network rather than a single path set. The two measures by which we wish to compare their performance are the optimality of the resulting configuration and the amount of computational effort required to reach it.

Of the methods considered here, only total enumeration and the Lawler-Bell algorithm are guaranteed to find the global optimal configuration. The hill-climbing heuristics are dependent on the starting configuration. Although they may not find global optima, they may find good configurations for a small expenditure of effort. Table 2 shows that the heuristics converged on their solutions in around 5-10 steps.

The Boolean method found the optimal solution for the cost = 20 case in a blind application. In a real problem, the effort associated with the Boolean

approach is not determined by the number of evaluations of the structure function. It is determined by the work associated with formulating and then combining the conditions imposed by requiring each cut-set to satisfy the cutoff.

The performance of the genetic algorithm is given as a range of values, since each run is dependent on the initial seed population. The path set coding was very effective at reducing the number of necessary function evaluations by eliminating superfluous configurations. The genetic algorithm was found to converge to the global optimum often, and given the coding scheme used, with a large enough number of generations, we would expect convergence in most cases.

CONCLUSIONS

Both the genetic algorithm and the Boolean procedure have found solutions better than those previously claimed as the global optimum for the cost=20 problem. The price of applying these methods is that they both need more computational power than the hill-climbing heuristics. However, in both cases this is far less computing than a rigorous exhaustive search would require. Neither of these methods guarantees the global optimum, but they have performed better than the hill-climbers. Actually, under certain conditions, the Boolean method can be guaranteed not to find the global optimum. However, we believe that the Boolean method is a good way to generate a starting population for the genetic algorithm. These two methods appear to be perfectly feasible for realistic problems, and the more global character of their search recommends them over the heuristic hill-climbers.

REFERENCES

1. A. MOHAMED, L.M. LEEMIS and A. RAVINDRAN, "Optimization techniques for system reliability: a review," Reliability Engineering and System Safety, 35, 137-146 (1992).

2. F.A. TILLMAN, C. HWANG and W. KUO, Optimization of Systems Reliability, Industrial Engineering Series, vol 4. Marcel Dekker, Inc., New York (1980).

3. D.H. SHI, "A New Heuristic for Constrained Redundancy-Optimization in Complex Systems," IEEE Trans. Reliability, vol R-36, 621-623 (1987).

4. T. KOHDA and K. INOUE, "A Reliability Optimization Method for Complex Systems with the Criterion of Local Optimality," IEEE Trans. Reliability, vol R-31, 109-111 (1982).

5. K.K.AGGARWAL,"Redundancy Optimization in General Systems," IEEE Trans. Reliability, vol R-25, 330-332 (1976).

6. H. SIVARAMAKRISHNAN and A.D. NARASIMHALU,"Correction to 'Redundancy Optimization in General Systems'," IEEE Trans. Reliability, vol R-26, 345 (1977).

7. E.L. LAWLER and M.D. BELL, "A method for solving discrete optimization problems," Oper. Res., 14, 1098-1112 (1966).

8. R. YOUNGBLOOD and L. OLIVEIRA, "A Boolean Approach to Redundancy Optimization," Trans. of the American Nuclear Society, American Nuclear Society, 61, 212 (1990).

9. R. YOUNGBLOOD and L. OLIVEIRA, "Application of an Allocation Methodology," Proc. PSA '89 Int. Topl. Mtg. Probability, Reliability, and Safety Assessment, Pittsburgh, Pennsylvania, 1989, American Nuclear Society, 484-491 (1989).

10. B.L. HULME, "Boolean Methods of Optimization Over Independence Systems," SIAM J. Alg. Disc. Meth., 5, 255-262 (1984).

11. D.E. GOLDBERG, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts (1989).

END

DATE FILMED

3/29/93