Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm

David W. Coit, Member IEEE
Rutgers University, Piscataway
Alice E. Smith, Member IEEE
University of Pittsburgh, Pittsburgh

Key Words — **Genetic algorithm, Combinatorial optimization, Redundancy allocation problem, Reliability design**

Summary & Conclusions — **A problem-specific genetic algorithm (GA) is developed and demonstrated to analyze *series-parallel* systems and to determine the optimal design configuration when there are multiple component choices available for each of several *k*-out-of-*n*:G subsystems. The problem is to select components and redundancy-levels to optimize some objective function, given system-level constraints on reliability, cost, and/or weight. Previous formulations of the problem have implicit restrictions concerning the type of redundancy allowed, the number of available component choices, and whether mixing of components is allowed. GA is a robust evolutionary optimization search technique with very few restrictions concerning the type or size of the design problem. The solution approach was to solve the dual of a nonlinear optimization problem by using a dynamic penalty function. GA performs very well on two types of problems: 1) redundancy allocation originally proposed by Fyffe, Hines, Lee, and 2) randomly generated problem with more complex *k*-out-of-*n*:G configurations.**

1. INTRODUCTION

Acronym[1]

GA	genetic algorithm
B&L	Bulfin & Liu [11]
N&M	Nakagawa & Miyazaki [9].

This paper describes the use of a GA to solve the redundancy allocation problem for a *series-parallel*[2] system. In this problem formulation, there is a specified number of subsystems and, for each subsystem, there are multiple component choices which can be selected (assuming an unlimited supply of each), and used in *parallel*. For those systems designed using off-the-shelf component types, with known cost, reliability, and weight, the system design and component selection become a combinatorial optimization problem. The consumer electronics industry is one such example where new system designs are composed largely of standard component types (*eg*, microcircuits, transistors, resistors) with known characteristics. The problem is then to select the optimal combination of parts and redundancy-levels to meet reliability & weight constraints collectively at a minimum cost, or alternatively, to maximize reliability, given cost & weight constraints. The *series-parallel* system with *k*-out-of-*n*:G subsystem redundancy, addressed in this paper, is a common representation for many system design problems.

The GA optimization approach was used in this research. It is one of a family of heuristic optimization techniques, which include simulated annealing, Tabu search, and evolutionary strategies [1]. GA have been demonstrated to converge to the optimal solution for many diverse difficult problems, although optimality cannot be guaranteed. The ability of GA to find good solutions efficiently often depends on properly customizing the encoding, breeding operators, and fitness measures to the specific engineering problem.

1.1 Previous Research

The redundancy allocation problem for *series-parallel* systems is difficult. Chern [2] showed that the problem is NP-hard. Optimization approaches to determine optimal or very-good solutions have included dynamic programming, integer programming, mixed integer and nonlinear programming, and heuristics. A summary of work in this area is presented in Tillman, Hwang, Kuo [3, 4].

Bellman [5] and Bellman & Dreyfus [6, 7] used dynamic programming to maximize reliability for a system, given a single cost-constraint. For each subsystem, there was only one component choice so the problem was to identify the optimal levels of redundancy. Fyffe, Hines, Lee [8] also used a dynamic programming approach and solved a more difficult design problem. They considered a system with 14 subsystems and with constraints on cost & weight. For each subsystem, there were three or four component choices each with different reliability, cost, and weight. To accommodate multiple constraints, they used a Lagrangian multiplier within the objective function. While their formulation provided a selection of components, the search space was artificially restricted to consider only solutions where the same component type is used in *parallel*.

Nakagawa & Miyazaki [9] showed that the use of a Lagrangian multiplier with dynamic programming is often inefficient. Instead of Lagrangian multipliers, they used a surrogate constraints approach, and demonstrated their algorithm by solving 33 variations of the Fyffe problem. Of the 33 problems, their N&M algorithm produced optimal solutions for 30 of them. For the other cases, the algorithm did not lead to a feasible solution.

Another approach has been to use integer programming wherein it is necessary to restrict artificially the search space and to prohibit mixing of different components within a

[1]The singular & plural of an acronym are always spelled the same.
[2]The terms, *series* & *parallel* are used in their logic-diagram sense, irrespective of the schematic-diagram or physical-layout.

subsystem. For problems to maximize reliability, given nonlinear but separable constraints, many variations of the problem can be transformed into an equivalent integer programming problem. This was demonstrated by Ghare & Taylor [10], who used a branch-and-bound approach. Bulfin & Liu [11] also used integer programming. They formulated the problem as a knapsack problem using surrogate constraints (approximated by Lagrangian multipliers found by subgradient optimization [12]). They formulated the Fyffe problem and its variations as integer programs and determined the optimal solution to all 33 problems investigated by N&M. They solved only problems with 1-out-of-n:G subsystem redundancy.

Other examples of integer programming solutions are described by Misra & Sharma [13], Gen, Ida, Tsujimura, Kim [14], and Gen, Ida, Lee [15]. Misra & Sharma [13] present a fast algorithm to solve integer programming problems like those of Ghare & Taylor [10]. Refs [14, 15] formulate the problem as a multi-objective decision-making problem with distinct goals for reliability, cost, and weight.

There have been several effective uses of mixed-integer and nonlinear programming to solve the redundancy allocation problem. Several notable examples are Tillman, Hwang, Kuo [16, 17] wherein component reliability is treated as a continuous variable and component cost is expressed as a function of reliability and other parameters.

While the redundancy allocation problem has been studied in great detail, two areas which have not been sufficiently analyzed are the implications of mixing functionally similar components within a *parallel* subsystem and the use of k-out-of-n:G redundancy (with $k > 1$). In practice many system designs use different (yet functionally similar) components in *parallel*. For example, airplanes often use a primary electronic gyroscope and a secondary mechanical gyroscope working in *parallel*, and most new automobiles have a redundant (spare) tire with different size and weight characteristics forming a 4-out-of-5:G standby redundant system. The power of a GA is that it can easily be adapted to many diverse design scenarios including those with mixing of components, k-out-of-n:G redundancy, and more complex forms of redundancy.

1.2 Use of Genetic Algorithms in Reliability

GA have been used to solve many difficult engineering problems and are particularly effective for combinatorial optimization problems with large, complex search spaces. Within the reliability field, however, there have been very few examples of their use. For a fixed design configuration and known incremental decreases in component failure rates and their associated costs, Painton & Campbell [18, 19] used a GA to find maximum reliability solutions to satisfy specific cost constraints. Their algorithm is flexible and can be formulated to optimize reliability, mean time to failure (MTTF), the 5th percentile of the MTTF distribution or availability. Ida, Gen, Yokota [20] used a GA to solve a redundancy-allocation problem where there are several failure modes. This problem had previously been solved by both nonlinear programming and integer programming.

Coit & Smith [21] analyzed a *series-parallel* system with 8 subsystems and 10 unique component choices for each subsystem. The search space had $> 10^{30}$ unique solutions and the GA readily converged to a good solution after analyzing less than $4 \cdot 10^{-26}$ of the search space. An interesting feature of this work is that neural network approximations to subsystem reliability, instead of exact solutions, were used.

Notation

R, C, W [reliability, cost, weight] constraint

s number of subsystems

i index for subsystems: $i=1,\ldots,s$ unless otherwise specified

m_i number of available component choices for subsystem i

$r_{i,j}$, $c_{i,j}$, $w_{i,j}$ [reliability, cost, weight] of component j available for subsystem i

$x_{i,j}$ number of component-j used in subsystem i

$\boldsymbol{x}_i$ $(x_{i,1}, x_{i,2}, \ldots, x_{i,m_i})$

t_j number of good component-j

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

n_i $\sum_j x_{i,j}$: number of components used in subsystem i

$n_{\max}$ maximum number of components in *parallel* (user specified)

k_i minimum number of components in *parallel* required for subsystem i to operate

$\boldsymbol{k}$ $(k_1, k_2, \ldots, k_s)$

$R_i(\boldsymbol{x}_i|k_i)$ reliability of subsystem i, given k_i

$C_i(\boldsymbol{x}_i)$, $W_i(\boldsymbol{x}_i)$ [cost, weight] of subsystem i

$\boldsymbol{v}_q$ vector encoding of solution q

p population size

$\boldsymbol{\lambda}$ Lagrangian multiplier vector

$P(\boldsymbol{\lambda},\boldsymbol{v}_q)$, $f(\boldsymbol{\lambda},\boldsymbol{v}_q)$ [penalty, fitness] for member-q of the population

$\sum_i$, $\prod_i$ [sum, product] over i from 1 to s

$\sum_j$, $\prod_j$ [sum, product] over j from 1 to m_i.

Other, standard notation is given in "Information for Readers & Authors" at the rear of each issue.

Assumptions

1. Components and the system are 2-state (good, bad).
2. The component reliabilities are known and deterministic.
3. Failures of individual components are s-independent.
4. All redundancy is active: failure rate while not in use is the same as when in use.
5. Failed components do not damage the system, and are not repaired. ◄

2. PROBLEM FORMULATION

Given the redundancy allocation problem for the *series-parallel* system, the problem formulation is to maximize reliability (Problem P1) or to minimize cost (Problem P2) given constraints, and k_i is specified for each subsystem. Expansion of the problem to include more than 2 constraints can be easily accommodated by the GA, unlike dynamic programming formulations.

Problem P1

$$\max_{x_1,\ldots,x_s} \left\{ \prod_i R_i(x_i|k_i) \right\}$$

subject to:

$$\sum_i C_i(x_i) \le C, \sum_i W_i(x_i) \le W. \quad \blacktriangleleft$$

System reliability can be expressed as a function of the $x_{i,j}$ by (1). In this general form, it is not possible to determine a linearly equivalent objective function (or constraint) as is done in integer programming formulations of this problem [10, 11, 13-15].

$$R_s(x_1,\ldots,x_s|k) = \prod_i R_i(x_i|k_i)$$

$$= \prod_i \left[[1 - \sum_{l=0}^{k_i-1} \sum_{t\in\tau} \prod_j \text{binm}(t_j;\, r_{i,j},\, x_{i,j}) \right], \quad (1)$$

τ is such that $\sum_j t_j = l$

Within P1 & P2, system weight & cost are often defined as linear functions primarily because this is a reasonable representation of the cumulative effect of component cost & weight. In the examples in section 4, cost & weight constraints are linear; however this is not a requirement. Nonlinear constraints and even nonseparable constraints can be handled by GA, unlike integer programming formulations where constraints must be separable.

The size of the search space (N) is very large even for small & moderately sized problems. Assuming an upper bound on the number of redundant components ($n_{\max}$), the number of unique system representations is [22]:

$$N = \prod_i [\binom{m_i + n_{\max}}{m_i} - \binom{m_i + k_i - 1}{m_i}]. \quad (2)$$

Fyffe Problem

The best known problem of the form P1 is in Fyffe, Hines, Lee [1], with $k_i = 1$, $C = 130$, $W = 170$. In the optimal solution, the weight constraint is tight and the system-cost = 119. Figure 1a shows the optimal solution for subsystems 7, 8; the numbers within the rectangle represent the optimal component choices. For these two subsystems, the collective reliability is 99.06%, the cost is 20 and the weight is 30.

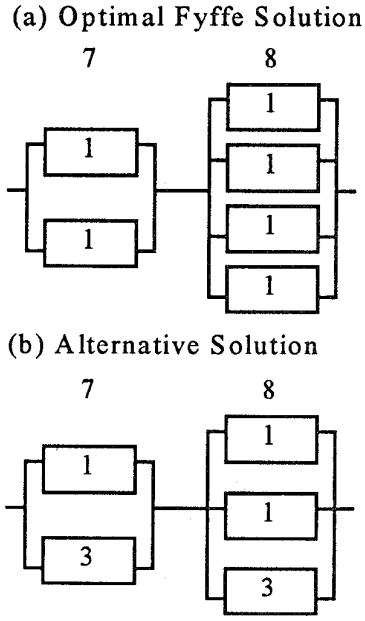

Figure 1. Alternative Solutions to Subsystems 7 & 8

Figure 1b shows an alternative solution wherein the collective reliability is 99.14%, the cost is 21 and the weight is 30; this has slightly higher reliability at the same weight and an increase in cost of 5%. But, since the cost constraint in the optimal solution had excess capacity, a change to this alternative solution for subsystems 7. 8, while leaving everything else the same, improves the system reliability and still is feasible. ◄

An advantage of GA is that there are very few restrictions on the form of the solutions. GA thoroughly examines the search space and readily identifies design configurations, analogous to that depicted in figure 1b, which will improve the final solution, but would not be identified using prior dynamic programming, integer programming, or nonlinear programming formulations of this problem.

3. GENETIC ALGORITHM IMPLEMENTATION

A genetic algorithm is a stochastic optimization technique (patterned after natural selection in biological evolution) as introduced by Holland [23] and further described by Goldberg [24]. GA is characterized by the steps:

1. Encode the solutions
2. Generate an initial population
3. Select parent solutions for breeding
   a. Crossover breeding operator
   b. Mutation operator
   c. Cull inferior solutions

4. Repeat step 3 until termination criteria are met.

An effective GA depends on complementary crossover & mutation operators. The effectiveness of the crossover operator dictates the rate of convergence, while the mutation operator prevents the algorithm from prematurely converging to a local optimum. The number of children & mutants produced each generation are tunable parameters that are held constant during a specific trial.

3.1 Solution Encoding

Traditionally, solution encodings have been a binary string [23, 24]. For combinatorial optimization, an encoding using integer values can be more efficient [25]. We took this approach in this analysis. Each possible solution to the redundancy allocation problem is a collection of n_i parts in *parallel* ($k_i \leq n_i \leq n_{\max}$) for each subsystem. The n_i parts can be chosen in any combination from among the m_i available components. The m_i components are indexed in descending order in accordance with their reliability (1 represents the most reliable, m_i represents the least reliable). The solution encoding is a vector representation with $s \cdot n_{\max}$ positions. Each subsystem is represented by n_{max} positions with the components which form a particular solution listed according to their reliability index. An index of $m_i + 1$ is assigned to a position where an additional component was not used ($n_i < n_{max}$). The subsystem representations are then placed adjacent to each other to complete the vector representation.

Encoding Example

The system has:

$s = 3$,

$m_1 = 5$, $m_2 = 4$, $m_3 = 5$

$n_{\max} = 5$ (predetermined).

$v_q = (1\ 1\ 6\ 6\ 6|2\ 2\ 3\ 5\ 5|4\ 6\ 6\ 6\ 6)$

represents a prospective solution with —

- 2 of the most reliable components used in *parallel* for subsystem #1;
- 2 of the second most reliable and 1 of the third most reliable components used in *parallel* for subsystem #2;
- 1 of the fourth most reliable components used for subsystem #3. ◄

3.2 Initial Population

For a given p, the initial population was determined by randomly selecting p solution vectors. For each solution, s integers between k_i and $n_{\max}$ were randomly selected to represent n_i for a particular subsystem. Then, n_i parts were randomly and uniformly selected from among the m_i available components (assuming an unlimited supply of each available component-type). The chosen components were sequenced by their reliability. Previous experimentation [21] indicated that a population size of 40 converged quickly and produced good solutions. In general, the minimum effective population-size grows with problem-size.

3.3 Objective Function

The objective function was the sum of the reliability (P1) or cost (P2) and a dynamic penalty function determined by the relative degree of infeasibility. It is important to search through the infeasible region, particularly for highly constrained problems, because the optimal solution can most efficiently be reached via the infeasible region, and often, good feasible solutions are a product of breeding between a feasible and an infeasible solution. To provide an efficient search through the infeasible region, and to assure that the final best solution is feasible, a dynamic penalty function based on the squared constraint-violation was defined to increase the penalty incrementally for infeasible solutions as the search progresses. The penalty function used here is based on the research in [26], and increased monotonically with generation number, g. For P2, the objective & penalty functions, used for this analysis, are defined in (3). Exact reliability estimates are used for each k-out-of-n:G subsystem by using (1). The formulation for P1 is analogous.

$$f(\lambda, v_q) = \left[\sum_i C_i(x_i) \right] + P(\lambda, v_q). \tag{3}$$

3.4 Crossover Breeding Operator

The crossover breeding operator provides a thorough search of regions of the sample space already demonstrated to produce good solutions. For this GA, parents were selected based on the ordinal ranking of their objective function. A uniform random number, U, between 1 and $\sqrt{p}$ was selected and the solution with the ranking closest to U^2 is selected as a parent — following the selection procedure of [27]. The crossover operator retained all identical genetic information from both parents and then randomly selected, with equal probability, from either of the two parents for components which differed. Because the solution encodings were ranked from most to least reliable, matches were common. This crossover operator is a variation of the uniform crossover operator which is superior [28] to traditional crossover strategies for analyzing combinatorial problems.

3.5 Mutation Operator

The mutation operator performs random perturbations to selected solutions. A predetermined number of mutations within a generation is set for each GA trial. Each value within the solution vector (which was randomly selected to be mutated) was changed with probability = mutation-rate. A mutated component was changed to an index of $m_i + 1$ with 50% probability and to a randomly chosen component, from among the m_i choices, with 50% probability.

3.6 Evolution

A 'survival of the fittest' strategy was used. After crossover breeding, the p best solutions from among the previous generation and the new child vectors were retained to form the next generation. The fitness measure was the objective function value. Mutation was then performed after culling inferior solutions from the population. The best solution within the population was never chosen for mutation — to assure that the optimal solution was never altered via mutation. This is a form of elitist selection [29]. The GA was terminated after a preselected number of generations although the optimal solution was often reached much earlier.

4. EXAMPLES[3]

The GA was used to analyze two problems with very good results.

[3]The number of significant figures is not intended to imply any accuracy in the estimates, but to illustrate the arithmetic.

4.1 Example 1

GA was implemented on the 33 variations of the Fyffe, Hines, Lee problem [8] which were attempted by N&M in the form of P1.

$C = 130;$

W is varied incrementally from 191 to 159.

Because of the stochastic nature of GA, 10 trials were performed for 1200 generations each and the best solution from among the 10 trials was used as the final solution. The maximum reliability identified by GA was used to compare its performance to other algorithms.

Considering component mixing, the 'search space size' $> 7.6 \cdot 10^{33}$ from (2). For this problem, 18 children and 22 mutations were produced each generation and the mutation rate was 0.05. As the weight constraint was incrementally lowered to solve new variations of the problem, the GA performance improved by increasing the severity of the penalty function. Table 1

Table 1. GA Performance for Fyffe [8] Problem
[MPI = Maximum Possible Improvement = $(GA_{max} - N\&M_{max})/(1 - N\&M_{max})$]

Weight Constraint	N&M			GA with 10 Trials				
	Reliability	Cost	Weight	Max Rel	Mean Rel	Min Rel	Std Dev	MPI(%)
191	.9864	130	191	.9867	.9862	.9854	.000511	2.21
190	.9854	132*	189	.9857	.9855	.9852	.000183	2.05
189	.9850	131*	188	.9856	.9850	.9838	.000603	4.00
188	.9847	129	188	.9850	.9848	.9842	.000253	1.96
187	.9840	133*	186	.9844	.9841	.9835	.000305	2.50
186	.9831	129	186	.9836	.9833	.9827	.000285	2.96
185	.9829	129	185	.9831	.9826	.9822	.000373	1.17
184	.9822	126	184	.9823	.9819	.9812	.000385	0.56
183	.9815	130	182	.9819	.9814	.9812	.000271	2.16
182	.9815	130	182	.9811	.9806	.9803	.000301	-2.16
181	.9800	128	181	.9802	.9801	.9800	.000074	1.00
180	.9796	126	180	.9797	.9793	.9782	.000479	0.49
179	.9792	127	179	.9791	.9786	.9780	.000370	-0.48
178	.9772	123	177	.9783	.9780	.9764	.000548	4.82
177	.9772	123	177	.9772	.9771	.9770	.000105	0.00
176	.9764	125	176	.9764	.9760	.9751	.000536	0.00
175	.9744	121	174	.9753	.9753	.9753	.000000	3.52
174	.9744	121	174	.9744	.9732	.9716	.000938	0.00
173	.9723	122	173	.9738	.9732	.9719	.000851	5.42
172	.9720	123	172	.9727	.9725	.9712	.000455	2.50
171	.9700	119	170	.9719	.9712	.9701	.000760	6.33
170	.9700	119	170	.9708	.9705	.9695	.000564	2.67
169	.9675	121	169	.9692	.9689	.9684	.000343	5.23
168	.9666	120	168	.9681	.9674	.9662	.000789	4.49
167	.9656	117	167	.9663	.9661	.9657	.000290	2.03
166	.9646	116	166	.9650	.9647	.9636	.000497	1.13
165	.9621	118	165	.9637	.9632	.9627	.000481	4.22
164	.9609	116	164	.9624	.9620	.9609	.000669	3.84
163	.9602	114	163	.9606	.9602	.9592	.000466	1.01
162	.9589	112	162	.9591	.9587	.9579	.000569	0.49
161	.9565	111	161	.9580	.9572	.9561	.000815	3.45
160	.9546	110	159	.9557	.9556	.9554	.000126	2.42
159	.9546	110	159	.9546	.9538	.9531	.000501	0.00

*infeasible solution

compares the GA results and the corresponding results from N&M. In table 1, MPI is the fraction that the best feasible solution achieved of the maximum possible improvement, considering that 'reliability $\leq$ 1'.

The GA produced feasible final solutions for all 33 problems while the N&M model yielded feasible solutions for only 30 of the 33. The GA produced a solution with higher reliability than the N&M or B&L model for 27 of the 33 problems. It was possible to obtain values for system reliability higher than the previously determined "optimal" solutions because GA allows component mixing within a specific subsystem, while the other algorithms do not. In all instances where there was higher reliability, the improvement was small ($< 5\%$). However, in high reliability applications, even very small improvements in reliability are often difficult to obtain. In four of the problem variations, the GA yielded precisely the same solution as the N&M and B&L algorithms, and in 2 of the 33 problems, the GA produced a solution which was very close, but at a lower reliability.

4.2 Example 2

Table 2 defines the component choices. This is a P2 problem and the objective is to minimize cost, given reliability & weight constraints for a system with 2 subsystems.

$k_1 = 4, k_2 = 2.$

Cost, weight, reliability component values were selected randomly; however, the underlying distributions were chosen so that the marginal cost of improved reliability was generally, but not in every specific case because of random selection, an increasing function with cost. This problem is more difficult than section 4.1 in several respects:

- both subsystems are k-out-of-n:G with $k > 1$;
- for each subsystem, there are 10 distinct components choices.

Considering that components mixing is allowed, this is a difficult combinatorial problem with over $1.9 \cdot 10^9$ unique solutions ($n_{max} = 8$).

Table 2. Component Choices

Design Alternative j	Subsystem i 1 (k=4) R	C	W	2 (k=2) R	C	W
1	.981	95	52	.931	137	83
2	.933	86	94	.917	132	96
3	.730	80	32	.885	127	94
4	.720	75	92	.857	122	93
5	.708	61	41	.836	100	95
6	.699	45	33	.811	59	63
7	.655	40	98	.612	54	65
8	.622	36	96	.432	41	49
9	.604	31	83	.389	36	33
10	.352	26	66	.339	30	51

Twenty GA trials were performed for 6 cases; the cases differed by changing the reliability & weight constraints as

Table 3. GA Performance for 6 Cases

	Problem Description				GA Performance Over 20 Trials				
Case	R	W	Global Minimum	Previous Best*	Minimum Cost	Average Cost	Average # Gen.	Number Optimal	Number Feasible
1	.975	650	727	770	727	727.25	988.65	18/20	20/20
2	.975	600	736	770	736	736.90	570.95	11/20	20/20
3	.975	550	747	871	747	747.00	662.30	20/20	20/20
4	.95	600	656	711	656	656.00	309.10	20/20	20/20
5	.95	550	661	711	661	661.00	268.00	20/20	20/20
6	.95	500	661	**	661	680.80	226.85	18/20	20/20

*The lowest minimum cost possible from the N&M and B&L formulations adapted for k-out-of-n:G
** Other algorithms do not produce any feasible solutions

Table 4. Optimal Solutions

				Subsystem 1 (k=4)*				Subsystem 2 (k=2)*		
Case	Cost	Reliability	Weight	1	6	7	8	6	9	10
1	727	.9750	640	4	1	0	1	4	0	1
2	736	.9768	577	4	2	0	0	4	0	1
3	747	.9819	545	5	0	0	0	4	1	0
4	656	.9506	558	4	0	1	0	4	0	0
5,6	661	.9537	493	4	1	0	0	4	0	0

*Amount of each design alternative, from table 2.

shown in table 3. For this problem, 15 children and 25 mutations were produced each generation and the mutation rate was 0.25. In each case GA was terminated after 1200 generations, even if the optimal solution had not been reached. Table 4 presents the optimal solutions for each problem case (obtained by complete enumeration).

The results show that the GA consistently converged (89%) to the optimal solution. The results yield minimum costs which are between 4.4% and 14.2% better than that could be obtained from any of the previously presented integer programming [10, 11, 13-15] or dynamic programming [8, 9] formulations of the problem (which do not allow component mixing).

Two cases merit further discussion:

- Case 6 is a highly constrained problem. There are only 9 feasible solutions (out of $1.9 \cdot 10^9$), all of which require the mixing of components within a subsystem. A competing algorithm which did not allow mixing of components would not be able to identify any feasible solutions, whereas the GA identified the optimal solution in 18 out of 20 trials.
- In case 2, only 11 of 20 trials converged to optimal, but the average solution differed from the optimal only by 0.12% and the worst solution differed by only 0.27%. ◄

Table 3 presents the average number of generations for the GA to converge to its final solution. Remembering that each generation includes 40 solutions and the search space from (2), these convergence results show the algorithm converges with a percent effort ratio (PER) [12] of less than 0.0021% of the search space for a single GA run. This compares to a PER of 3.9% for an integer programming solution to the redundancy allocation problem using the Lawler & Bell techniques [30], and a PER ranging from 0.013% to 10.41% for the Misra & Sharma algorithm [13].

These integer programming algorithms guarantee convergence to an optimal solution over a smaller search space while the GA can not guarantee that the optimal solution will be reached.

REFERENCES

[1] C.R. Reeves (*Ed*), *Modern Heuristic Techniques for Combinatorial Problems*, 1993; John Wiley & Sons.

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

[3] F.A. Tillman, C.L. Hwang, W. Kuo, *Optimization of System Reliability*, 1980; Marcel Dekker.

[4] F.A. Tillman, C.L. Hwang, W. Kuo, "Optimization techniques for system reliability with redundancy: A review", *IEEE Trans. Reliability*, vol R-26, 1977 Aug, pp 148-155.

[5] R.E. Bellman, *Dynamic Programming*, 1957; Princeton Univ. Press.

[6] R.E. Bellman, E. Dreyfus, "Dynamic programming and reliability of multicomponent devices", *Operations Research*, vol 6, 1958 Mar-Apr, pp 200-206.

[7] R.E. Bellman, E. Dreyfus, *Applied Dynamic Programming*, 1962; Princeton Univ. Press.

[8] D.E. Fyffe, W.W. Hines, N.K. Lee, "System reliability allocation and a computational algorithm", *IEEE Trans. Reliability*, vol R-17, 1968 Jun, pp 64-69.

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

[10] P.M. Ghare, R.E. Taylor, "Optimal redundancy for reliability in series system", *Operations Research*, vol 17, 1969 Sep, pp 838-847.

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

[12] M. Fisher, "The Lagrangian relaxation method for solving integer programming problems", *Management Science*, vol 27, 1981 Jan, pp 1-18.

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

[14] M. Gen, K. Ida, Y. Tsujimura, C.E. Kim, "Large-scale 0-1 fuzzy goal programming and its application to reliability optimization problem", *Computers and Industrial Eng'g*, vol 24, 1993, pp 539-549

[15] M. Gen, K. Ida, J.U. Lee, "A computational algorithm for solving 0-1 goal programming with GUB structures and its application for optimization problems in system reliability", *Electronics & Communications in Japan*, part 3, vol 73, 1990 Mar, pp 88-96.

[16] F.A. Tillman, C.L. Hwang, W. Kuo, "Determining component reliability and redundancy for optimum system reliability", *IEEE Trans. Reliability*, vol R-26, 1977 Aug, pp 162-165.

[17] C.L. Hwang, F.A. Tillman, W. Kuo, "Reliability optimization by generalized Lagrangian-function and reduced-gradient methods", *IEEE Trans. Reliability*, vol R-28, 1979 Oct, pp 316-319.

[18] L. Painton, J. Campbell, "Identification of components to optimize improvements in system reliability", *Proc. SRA PSAM-II Conf. System-based Methods for the Design & Operation of Technological Systems & Processes*, 1994 Mar, 10-15 - 10-20.

[19] L. Painton, J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Trans. Reliability*, vol 44, 1995 Jun, pp. 172-178.

[20] K. Ida, M. Gen, T. Yokota, "System reliability optimization with several failure modes by genetic algorithm", *Proc. 16th Int'l Conf. Computers and Industrial Engineering*, 1994 Mar, pp 349-352.

[21] D.W. Coit, A.E. Smith, "Use of a genetic algorithm to optimize a combinatorial reliability design problem", *Proc. 3rd Int'l Eng'g Research Conf*, 1994 May, pp 467-472.

[22] W. Feller, *An Introduction to Probability Theory*, 1968; John Wiley & Sons.

[23] J. Holland, *Adaptation in Natural and Artificial Systems*, 1975; Univ. of Michigan Press.

[24] D.E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, 1989; Addison Wesley.

[25] J. Antonisse, "A new interpretation of schema notation that overturns the binary encoding constraint", *Proc. 3rd Int'l Conf. Genetic Algorithms*, 1989, pp 86-91.

[26] A.E. Smith, D.M. Tate, "Genetic optimization using a penalty function", *Proc. 5th Int'l Conf. Genetic Algorithms*, 1993, pp 499-505.

[27] D.M. Tate, A.E. Smith, "A genetic approach to the quadratic assignment problem", *Computers and Operations Research*, 1994, vol 22, pp 73-83.

[28] G. Syswerda, "Uniform crossover in genetic algorithms", *Proc. 3rd Int'l Conf. Genetic Algorithms*, 1989, pp 2-9.

[29] J.J. Grenfenstette, "Optimization of control parameters for genetic algorithms", *IEEE Trans. Systems, Man, Cybernetics*, vol SMC-16, 1986, pp 122-128.

[30] E.L. Lawler, M.D. Bell, "A method for solving discrete optimization problems", *Operations Research*, vol 14, 1966 Nov-Dec, pp 1098-1112.

(Continued on page 266)

[12] "[13] part II: Application to *good-as-old* failure data", Working Paper 1994-4746-02.
[13] R. Guo, C.E. Love, "Estimating a linear-spline approximation for semi-parametric modeling with proportional hazard rates"; Simon Fraser University.

AUTHORS

Dr. Renkuan Guo; Dep't of Statistical Sciences; Univ. of Cape Town; Rondebosch 7700, Rep. of SOUTH AFRICA.
Internet (e-mail): rguo@uctvms.uct.ac.za

Renkuan Guo completed his undergraduate studies (1970) in Mechanical Engineering at Tsinghua University, Beijing, then received his MSc (1986) in Statistics and PhD (1989) in Statistics from Iowa State University. He is a Senior Lecturer in the Department of Statistical Sciences at the University of Cape Town. Dr Guo's research interests are in statistical modeling and analysis of repairable systems, and has written a variety of papers in this field.

Dr. C. E. Love; Faculty of Business Adm; Simon Fraser University; Burnaby; British Columbia V5A 1S6 CANADA.
Internet (e-mail): love@sfu.ca

Ernie Love obtained his B. Eng (Chemical, 1966) from McMaster University, MBA (1969) from McMaster University, and PhD (1975) in Operations Research from the University of London. His research interests are in optimization of stochastic systems, and he has published in a variety of international journals on these issues. In addition to his current research in preventive maintenance and reliability, Dr. Love is researching several aspects of quality deployment within manufacturing environments. Prior to obtaining his PhD, Dr. Love worked as an Engineer for Shell Oil Company and Union Carbide Ltd.

Manuscript received 1995 November 3

Publisher Item Identifier S 0018-9529(96)04776-8 ◄TR►

Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm

(*continued from page* 260)

AUTHORS

David W. Coit; Dep't of Industrial Eng'g; Rutgers Univ; POBox 909; Piscataway, New Jersey 08855-0909 USA.

David W. Coit received a BS (1980) in Mechanical Engineering from Cornell University, an MBA (1988) from Rensselaer Polytechnic Institute, and an MS (1993) in Industrial Engineering from the University of Pittsburgh. He is a PhD candidate at the University of Pittsburgh. From 1980 to 1992, he was a reliability engineer and project manager at IIT Research Institute, Rome, New York where he established reliability programs, analyzed the reliability of engineering designs, and developed statistical models to predict reliability of electronic components for client companies. His research involves reliability optimization, stochastic optimization techniques, and industrial applications for artificial neural networks. Mr. Coit is a student member of IEEE, IIE, and INFORMS.

Dr. Alice E. Smith; 1031 Benedum Hall; Dep't of Industrial Eng'g; Univ. of Pittsburgh, Pittsburgh, Pennsylvania 15261 USA.
Internet (e-mail): aesmith@engrng.pitt.edu

Alice E. Smith is Assistant Professor - Industrial Engineering. After 10 years of industrial experience with Southwestern Bell Corporation, she joined the faculty of the University of Pittsburgh in 1991. Her research interests are in modeling & optimization of complex systems using computational intelligence techniques, and her research has been sponsored by Lockheed Martin Corp, the Ben Franklin Technology Center of Western Pennsylvania, and the National Science Foundation, from which she was awarded a CAREER grant in 1995. She is an Associate Editor of *INFORMS J. Computing* and of *Engineering Design and Automation*, and a registered Professional Engineer in the state of Pennsylvania. Dr. Smith is a Member of IEEE, ASEE, and INFORMS, and a Senior Member of IIE and SWE.

Manuscript received 1995 June 1

Publisher Item Identifier S 0018-9529(96)04434-X ◄TR►