Cite as *Proceedings of the Fourth Industrial Engineering Research Conference* (IERC) May 1995

OPTIMIZATION APPROACHES TO THE REDUNDANCY ALLOCATION PROBLEM FOR SERIES -PARALLEL SYSTEMS

David W. Coit and Alice E. Smith[1]
Department of Industrial Engineering
1031 Benedum Hall
University of Pittsburgh
Pittsburgh, PA 15261
412-624-9830
aesmith@engrng.pitt.edu

ABSTRACT

The redundancy allocation problem involves the selection of components and the appropriate levels of redundancy to maximize reliability or minimize cost of a series-parallel system given design constraints. Different optimization approaches have been previously applied to the problem including dynamic programming, integer programming, and mixed integer and nonlinear programming. However these approaches can only solve sub-classes of the problem. This paper presents a review of the different optimization approaches and presents a new approach, a genetic algorithm (GA) which can solve the general class of the redundancy allocation problem. The GA is demonstrated on two different problems and compared with the other techniques.

INTRODUCTION

Determination of an optimal or near optimal system design is very important to economically produce new systems which meet and exceed customers' expectations for reliability, quality and performance. When developing a new system, there are detailed engineering specifications which prescribe minimum levels of reliability, maximum weight, etc. The redundancy allocation problem involves the simultaneous evaluation and selection of components and a system-level design configuration which can collectively meet all design constraints, and at the same time, optimize some objective function, usually system cost or reliability. There have been many successful attempts to determine optimal design configurations for different formulations of the redundancy allocation problem. However, many of these optimization techniques have included both explicit and implicit assumptions which unnecessarily restrict the form of the possible design configurations.

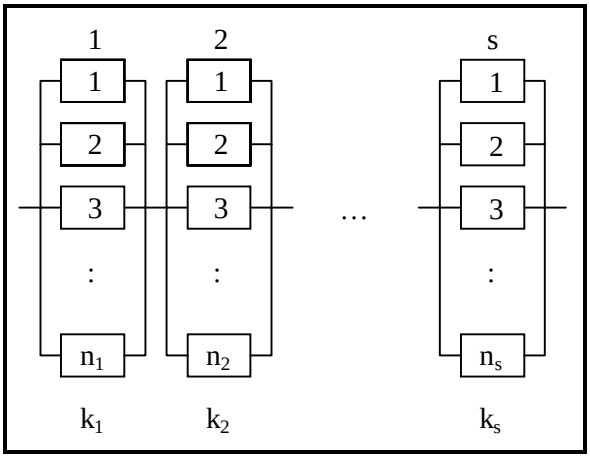

FIGURE 1: SERIES-PARALLEL SYSTEM CONFIGURATION

For many design problems, there are discrete component choices available for each subsystem

[1] Corresponding author.

(i=1,...,s) with known reliability, cost and weight characteristics. Figure 1 presents a typical example of a series-parallel system with k-out-of-n subsystem reliabilities. For each subsystem i, a minimum of k_i components must be chosen, with replacement, from among m_i available choices. Additionally, there is the option of adding more levels of redundancy (> k_i) to improve the system reliability as an alternative to using a more reliable, and more costly, component. It then becomes a combinatorial problem to select the optimal combination of components and level of redundancy for each of the s subsystems. Also, it is often beneficial to use a mix of different component types within a subsystem's parallel structure, analogous to a knapsack problem, thereby creating a large number of possible solutions even for relatively small problems.

The redundancy allocation problem has been previously analyzed using dynamic programming (DP), integer programming (IP), and mixed integer and nonlinear programming (MINLP). Each of these techniques has been demonstrated to be successful for specific sub-classes of the problem. The use of a genetic algorithm (GA) is a new approach to analyze this problem in the general case and it is demonstrated in this paper. GAs are a stochastic optimization technique, designed to mimic the process of evolution within populations of a species. They have been used to solve diverse types of industrial engineering and operations research problems with large and non-convex search areas, but rarely have been used to solve reliability optimization problems.

The general problem formulation is as follows for a problem to maximize reliability where C and W are system level constraints on cost and weight, and $R_i(\mathbf{x}_i|k_i)$, $C_i(\mathbf{x}_i)$ and $W_i(\mathbf{x}_i)$ represent the reliability, cost and weight of subsystem i. For each subsystem, there are m_i component choices. A formulation to minimize cost, given constraints on reliability and weight, is analogous but more difficult because of a more highly constrained search area and the nonlinear reliability constraint.

$$\max \quad \prod_{i=1}^{s} R_i\left(\mathbf{x}_i \middle| k_i\right) \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{s} C_i(\mathbf{x}_i) \le C$$

$$\sum_{i=1}^{s} W_i(\mathbf{x}_i) \le W$$

where

$\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{i,m_i})$, $n_i = \sum_{j=1}^{m_i} x_{ij}$

x_{ij} = number of the j^{th} available component used in subsystem i

DIFFERENT OPTIMIZATION APPROACHES

Dynamic Programming Formulations

DP solutions to the redundancy allocation problem are presented in Bellman and Dreyfus [1], Fyffe, Hines and Lee [2] and Nakagawa and Miyazaki [3]. The Fyffe, Hines and Lee formulation uses a Lagrangian multiplier (λ) within the objective function to reduce the number of problem constraints to one, and alternatively, the Nakagawa and Miyazaki formulation uses a surrogate constraint to combine the constraints into one. As one example, the problem formulation of the Fyffe, Hines and Lee formulation is as follows.

$$\max \quad \left[\prod_{i=1}^{s} R_i\left(\mathbf{x}_i \middle| k_i = 1\right)\right] e^{-\lambda \sum_{i=1}^{s} W_i(\mathbf{x}_i)} \tag{2}$$

$$\text{subject to} \quad \sum_{i=1}^{s} C_i(\mathbf{x}_i) \le C$$

The DP recursion formula is as follows where the final solution is found at $f_s(C)$ for the optimal value of λ.

$$f_1(\alpha) = \max_{\mathbf{x_1} | C_1(\mathbf{x_1}) \le \alpha} \left\{ R_1(\mathbf{x_1} | k_1) e^{-\lambda W(\mathbf{x_1})} \right\} \tag{3}$$

$$f_i(\beta) = \max_{\mathbf{x_i} | \Sigma C_i(\mathbf{x_i}) \le \beta} \left\{ R_i(\mathbf{x_i} | k_i) e^{-\lambda W(\mathbf{x_i})} f_{i-1}(\beta - C_i(\mathbf{x_i})) \right\} \tag{4}$$

for $i = 2, \ldots, s$

To use this DP algorithm, it is necessary to assume a λ value and then successively solve the recursion formulas starting with the function $f_1(\alpha)$, $f_2(\beta)$ and continuing on until $f_{s-1}(\beta)$. Then, $f_s(C)$ is the optimal solution for that particular λ. If $\Sigma\, W_i(\mathbf{x_i}) = W$, then this is the final solution. If not, then λ is incrementally changed, and the recursion formulas repeated, until the equality holds and the final solution is found.

Each of the published DP formulation examples assume all k_i to be one, and only maximize over $\mathbf{x_i}$ vectors of

the form $\mathbf{x_i} = (0,...,0,n_i,0,...,0)$. This limits the search space to subsystems where one of the available components is used exclusively and the problem is to chose which one. This makes for a more efficient search but often prevents the true optimal solution from being found, and for particularly constrained problems, does not allow for the identification of any feasible solutions.

Two problems with DP formulations using Lagrangian multipliers (λ) are the difficulty in accommodating more than two constraints and the potential inefficiency involved in searching over λ, particularly for highly constrained and infeasible solutions. The surrogate constraints approach of Nakagawa and Miyazaki remedied many of these problems, but their model was inefficient compared to equivalent IP models, and would not always converge to a feasible final solution.

Integer Programming Formulations

IP formulations of the problem include those proposed by Ghare and Taylor [4], Misra and Sharma [5], Gen et al [6] and Bulfin and Liu [7]. IP models require that the objective function and constraints are separable, either directly or by transformation. Ghare and Taylor presented a branch-and-bound technique for solving the redundancy allocation problem when there is only one component choice for each subsystem, and demonstrated the technique on several large problems. Bulfin and Liu showed that the IP models also work well when there are multiple component choices available for each subsystem. Their approach was to combine the constraints into one surrogate constraint and then to solve the problem as a knapsack problem. Misra and Sharma present an efficient algorithm for solving the redundancy allocation problem as an IP.

The large majority of the IP formulations (a notable exception is Misra [8]) either assume that $k = 1$ or only use examples with $k = 1$, and all of the IP formulations prohibit the mixing of different (but functionally similar) components within the same parallel subsystem structure. While problems with $k > 1$ can be accommodated by IP, it is doubtful that IP would be successful for any non-trivial problem with multiple component choices and mixing of components as well as $k > 1$. A typical IP formulation follows for a problem to maximize reliability of a system with only one component choice per subsystem, $k_i \geq 1$, and linear cost and weight constraints (r_i, c_i and w_i equal component reliability, cost and weight, respectively). n_{max} is the maximum number allowed in parallel.

$$\max \quad \sum_{i=1}^{s}\sum_{j=k_i}^{n_{\max}} a_{ij} y_{ij} \tag{5}$$

$$\text{subject to} \quad \sum_{i=1}^{s}\sum_{j=k_i}^{n_{\max}} c_i y_{ij} \leq d_1 \left(= C - \sum_{i=1}^{s} k_i c_i \right)$$

$$\sum_{i=1}^{s}\sum_{j=k_i}^{n_{\max}} w_i y_{ij} \leq d_2 \left(= W - \sum_{i=1}^{s} k_i w_i \right)$$

$$j = k_i+1, \ldots, n_{max} \quad \forall\, i$$

$$\text{where} \quad y_{ij} = \begin{cases} 1, \text{ if } k_i \leq j \leq x_{ij} \\ 0, \text{ otherwise} \end{cases}$$

$$a_{ij} = \ln\left(1 - \sum_{l=0}^{k_i-1} \binom{j}{l} r_i^{\,l} \left(1-r_i\right)^{j-l}\right) - \ln\left(1 - \sum_{l=0}^{k_i-1} \binom{j-1}{l} r_i^{\,l} \left(1-r_i\right)^{j-l-1}\right) \tag{6}$$

Equation 6 represents the logarithm of subsystem i reliability (with k_i specified) with j identical components in parallel minus the logarithm of subsystem i reliability with j - 1 identical components in parallel. This definition of y_{ij} and a_{ij} provides for the linear objective function in (5) which maximizes the logarithm of the system reliability (which then maximizes the system reliability).

For IP problems with multiple component choices (but without mixing), the problem is modeled similarly and the number of 0-1 decision variables is $s \times \Pi(m_i \times (n_{max} - k_i))$. For many redundancy allocation problems, IP models give very good solutions very efficiently, particularly when compared to equivalent DP formulations [7]. There are two primary deficiencies with the IP formulations. First, a formulation which considers mixing of different components within a subsystem will cause a combinatorial explosion of decision variables. Second, when the problem is to minimize cost given stringent constraints on reliability and weight, it becomes very difficult to identify any feasible solutions.

Mixed Integer and Nonlinear Programming

When the redundancy allocation problem is formulated as a MINLP [9, 10, 11], it normally takes a different form than that previously discussed. In typical MINLP formulations, component reliability is expressed as a

variable, cost is defined as a smooth function of reliability and the constraints are nonlinear functions of several decision variables. The objective is therefore to determine the optimal level of reliability and the amount of redundancy. This is the best approach for new system designs when the constituent components are being designed specifically for this new system and the optimal reliability levels represent the reliability to be "designed in" to the new components or used as a specified value for subcontractors or suppliers.

There are several drawbacks with using MINLP for many design problems. First, for many designs, components must be selected from a discrete list of available components, which have known reliability, cost and weight. In these cases, determination of the optimal reliability level is useful only as a goal when selecting which components to use. Second, it is not always possible or practical to determine a differentiable function for component cost as it relates to reliability.

GENETIC ALGORITHM IMPLEMENTATION

A GA is a stochastic optimization technique patterned after natural selection in biological evolution as initially described by Holland [12]. The GA approach is very flexible, can accommodate both discrete and continuous functions, and can investigate a larger search space than the corresponding DP and IP formulations by explicitly considering the option of mixing component types within a particular subsystem in conjunction with $k > 1$. Deficiencies with the GA approach are that there are several search parameters which must be found experimentally, solution encoding, crossover and mutation must be formulated and the GA can not guarantee convergence to the optimal, although it can be demonstrated that it consistently does converge.

An initial population of p solutions is randomly selected to form the first generation of the GA. Crossover and mutation operators are then performed on the population members to produce subsequent generations. An effective GA depends on complementary crossover and mutation operators. The crossover operator dictates the rate of convergence, while the mutation operator prevents the algorithm from prematurely converging. Each member of the population is evaluated in accordance with its "fitness", which is used as basis for selecting parent solutions and for culling inferior solutions from the population. The GA methodology can be characterized by the following operators.

- encoding of solutions
- selection of an initial population
- crossover breeding operator
- mutation operator
- culling of inferior solutions

Solution Encoding

Each possible solution to the redundancy allocation problem is a collection of n_i parts in parallel ($k_i \le n_i \le n_{max}$) 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 (i.e., 1 representing the most reliable, etc.). The solution encoding is a vector with s x n_{max} positions. An index of $m_i + 1$ is assigned to a position where an additional component was not used (i.e., $n_i < n_{max}$). The subsystem representations are then placed adjacent to each other to complete the vector representation.

As an example, consider a system with $s = 3$, $m_1 = 5$, $m_2 = 4$, $m_3 = 5$ and n_{max} predetermined to be 5. The vector

$$v_k = (1\ 1\ 6\ 6\ 6 \mid 2\ 2\ 3\ 5\ 5 \mid 4\ 6\ 6\ 6\ 6)$$

represents a prospective solution with two of the most reliable components used in parallel for the first subsystem; two of the second most reliable and one of the third most reliable component used in parallel for the second subsystem; and one of the fourth most reliable component used for the third subsystem.

Initial Population

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 the number of parts in parallel (n_i) for a particular subsystem. Then, n_i parts were randomly and uniformly selected from among the m_i available components (with replacement). The chosen components were then sequenced according to their reliability.

Objective Function

The objective function ($f(\boldsymbol{\lambda}, \mathbf{v_k})$) was defined as the sum of the search objective (cost or reliability) and a

dynamic penalty function ($P(\boldsymbol{\lambda},\mathbf{v_k})$) determined by the relative degree of infeasibility. It was 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 but to assure that the final best solution is feasible, a dynamic penalty function was defined to incrementally increase the penalty for infeasible solutions as the search progresses. It is important not to increase the penalty too rapidly or the GA will likely converge to a local optima. When minimizing system cost, the objective function and penalty function are defined as follows.

$$f(\boldsymbol{\lambda}, \mathbf{v}_k) = \sum_{i=1}^{s} C_i(\mathbf{x}_i) + P(\boldsymbol{\lambda}, \mathbf{v}_k) \tag{7}$$

$$P(\boldsymbol{\lambda}, \mathbf{v}_k) = \left(\lambda_c \min\left[0, \prod_{i=1}^{s} R_i(\mathbf{x}_i | k_i) - R\right]\right)^2 + \left(\lambda_w \min\left[0, W - \sum_{i=1}^{s} W_i(\mathbf{x}_i)\right]\right)^2 \tag{8}$$

where

$$\lambda_c = \lambda_{c1} + \left\lfloor g/g' \right\rfloor \lambda_{c2}$$

$$\lambda_w = \lambda_{w1} + \left\lfloor g/g' \right\rfloor \lambda_{w2}$$

g is the generation number and g' is the interval whereby the penalty function increases. For the examples which follow, g' was found experimentally to be effective when equal to 40. That is, every 40 generations, the penalty on infeasible solutions is increased, leading the search more towards the feasible region.

The Lagrangian multiplier parameters (λ_{c1}, λ_{c2}, λ_{w1} , λ_{w2}) and constants which need to be scaled based on the problem's constraints and complexity. While it is important to scale these parameters, the GA is robust and not sensitive to small alterations of these constants. As the search progresses, λ_c and λ_w are incrementally increased, thereby forcing the solutions into the feasible region. The values of λ_{c2} and λ_{w2} are determined empirically considering the severity of the constraints. The danger in increasing λ_c and λ_w too rapidly is that the algorithm will more readily find feasible solutions, as desired, but will less likely converge to the optimal solution.

Crossover Breeding Operator

The reproductive process of crossover produces child solution vectors which maintain certain properties of both parents. As in natural evolution, more fit parents are more likely to breed, and child vectors are often better solutions than either parent individually. A uniform crossover breeding operator was used to search areas 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 was selected as a parent. Two parents are chosen to produce each child solution vector. 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 ordered from most to least reliable, matches were common. For example, consider the child vector ($\mathbf{v}_c$) produced by the two parent vectors ($\mathbf{v}_1$, $\mathbf{v}_2$) as follows.

$\mathbf{v}_1$ = (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 components
$\mathbf{v}_c$ = (2 1 6 7 7 3 2 7 7 7) - random selection
$\mathbf{v}_c$ = (1 2 6 7 7 2 3 7 7 7) - final arrangement

Mutation Operator

The mutation operator performs random perturbations to selected solutions. A predetermined number of mutations within a generation is decided for each GA trial. The mutation operator is then based on a mutation rate, normally assigned a value of 0.05 - 0.30. Each value within the solution vector is changed with probability equal to the mutation rate. If selected to be mutated, it is 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. All mutated solutions are maintained within the population for at least one generation. This assures that the mutated solutions have the opportunity to breed with other solutions.

Evolution

A survival of the fittest strategy was employed. After crossover breeding, the *p* best solutions from among the previous generation and the new child vectors were retained to form the next generation. 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 it was never altered via mutation and to improve the rate of convergence. The GA was then terminated after a preselected number of generations, normally 1,200 generations.

EXAMPLES

The first example is that posed by Fyffe, Hines and Lee [2] with 33 problem variations from Nakagawa and Miyazaki [3]. The problem objective is to maximize reliability for a system with 14 subsystems and three or four component choices for each. There is a cost constraint of 130 and the weight constraint is varied incrementally from 191 to 159 to form the 33 variations. Ten different trials were performed for each variation of the problem. Considering component mixing, the size of the search space is greater than 7.6×10^{33}. For this problem, *p* was 40, the GA had 18 children and 22 mutations produced each generation and the mutation rate was 0.05.

The GA and IP produced feasible final solutions for all 33 problems while the DP model yielded feasible solutions for only 30 of the 33. The GA produced a solution with higher reliability than the DP or IP models for 27 of the 33 problems. 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, DP and IP yielded precisely the same solution, and in 2 of the 33 problems, the GA produced a solution which was close, but at a lower reliability compared to the other models.

The second example analyzed is defined by the component choices in Table 1. The objective is to minimize cost given reliability and weight constraints for a system designed with two subsystems. For the first subsystem k_1 is defined to be four, and for the second subsystem, k_2 is two. This problem is more difficult from the previous problem in several respects. First, both subsystems are k-out-of-n redundancy with k > 1. Second, for each subsystem, there are ten distinct components choices. Considering that the mixing of components is allowed, there are $> 1.9 \times 10^9$ unique solutions (with $n_{max} = 8$).

Six different cases of the second example were analyzed by changing the reliability and weight constraints. Table 2 describes the six different cases and the results of the analysis. For this problem, the GA produced 15 children and 25 mutations each generation and the mutation rate was 0.25. The results show that the GA consistently converged (i.e., 89%) to the optimal solution. Additionally, the results yield minimum costs which are between 4.4% and 14.2% better than that obtained from the previously presented IP [10,12,13,14] or DP [1,8] formulations of the problem.

Case 6 from Table 2 is a highly constrained problem. There are only a total of nine feasible solutions (out of 1.9×10^9), all of which require the mixing of components within a subsystem. The DP and IP models, which do not allow mixing of components, would not be able to identify any feasible solutions. The algorithm converges with a percent effort ratio (PER) [5] of less than 0.0021% of the search space for all cases. This compares to a PER ranging from 0.013% to 10.41% for the algorithm proposed by Misra and Sharma [5]. It should be noted that the Misra and Sharma algorithm guarantees convergence to the optimal solution while the GA can not guarantee that, although it has been demonstrated to consistently reach the optimal solution.

CONCLUSIONS

The redundancy allocation problem has previously been solved using DP, IP and MINLP models. There are distinct differences between these various optimization approaches, and different classes of the problem are better suited to one particular model. This paper describes the use of a GA to analyze this problem. The GA was tested on two problems and compared with the corresponding results from DP and IP. The GA was demonstrated to be very flexible and few restrictions are needed on the form of potential solutions. In the sample problems, the GA produced results consistently better than alternative approaches; however, because of the stochastic nature of the GA search, it can not guarantee convergence to the optimal solution.

The natural extension of this work is to adapt the GA to problems where the component reliability values are

governed by a probability density function. Constraints would then be based on a lower bound value for system reliability or cost. This represents a more realistic model of the design process. Also, it allows for the explicit consideration of risk-averse system designers and users.

REFERENCES

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

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

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

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

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

[6] 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 Engineering*, vol 24, 1993, pp 539-549

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

[8] K. B. Misra, U. Sharma, "An efficient approach for multiple criteria redundancy optimization problems", *Microelectronics and Reliability*, vol 31, 1991, pp 303-321.

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

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

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

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

BIOGRAPHICAL SKETCHES

David W. Coit is a Ph.D. candidate at the University of Pittsburgh. He has a B.S. in Mechanical Engineering from Cornell University, an M.B.A. from Rensselaer Polytechnic Institute and an M.S.I.E. from the University of Pittsburgh. He was previously employed at IIT Research Institute where he developed reliability prediction models, and developed and implemented reliability programs for industrial companies. His research interests are reliability optimization, implementation of neural networks for manufacturing processes and heuristic optimization search algorithms.

Alice E. Smith is Assistant Professor of Industrial Engineering at the University of Pittsburgh. Her research interests are in modeling and optimization using computational intelligence techniques. Her research has been sponsored by the National Science Foundation, Lockheed Martin Corporation and the Ben Franklin Technology Center of Western Pennsylvania. She is a Senior Member of IIE and a registered Professional Engineer.

TABLE 1: COMPONENT CHOICES FOR PROBLEM 2

Design Alternative (*j*)	Subsystem (*i*)					
	1 (k=4)			2 (k=2)		
	R	C	W	R	C	W
1	0.981	95	52	0.931	137	83
2	0.933	86	94	0.917	132	96
3	0.730	80	32	0.885	127	94
4	0.720	75	92	0.857	122	93
5	0.708	61	41	0.836	100	95
6	0.699	45	33	0.811	59	63
7	0.655	40	98	0.612	54	65
8	0.622	36	96	0.432	41	49
9	0.604	31	83	0.389	36	33
10	0.352	26	66	0.339	30	51

TABLE 2: GA PERFORMANCE FOR DIFFERENT CASES OF PROBLEM 2

Problem Description					GA Performance				
Case	Reliability Constraint	Weight Constraint	Global Minimum	Best IP or DP*	Minimum Cost	Average Cost	Average # gen	Number Optimal	Number Feasible
1	0.975	650	727	770	727	727.25	988.65	18/20	20/20
2	0.975	600	736	770	736	736.90	570.95	11/20	20/20
3	0.975	550	747	871	747	747.00	662.30	20/20	20/20
4	0.95	600	656	711	656	656.00	309.10	20/20	20/20
5	0.95	550	661	711	661	661.00	268.00	20/20	20/20
6	0.95	500	661	none**	661	680.80	226.85	18/20	20/20

* This is the lowest minimum cost possible from the IP formulation adapted for k-out-of-n.

** Other algorithms do not produce any feasible solutions.