Yugoslav Journal of Operations Research
12 (2002), Number 2, 227-236

A TWO-PHASE LINEAR PROGRAMMING APPROACH FOR REDUNDANCY ALLOCATION PROBLEMS

Yi-Chih HSIEH

Department of Industrial Management
National Huwei Institute of Technology
Taiwan, R.O.C.
yhsieh@sparc.nhit.edu.tw

Abstract: Provision of redundant components in parallel is an efficient way to increase the system reliability, however, the weight, volume and cost of the system will increase simultaneously. This paper proposes a new two-phase linear programming approach for solving the nonlinear redundancy allocation problems subject to multiple linear constraints. The first phase is used to approximately allocate the resource by using a general linear programming, while the second phase is used to re-allocate the slacks of resource by using a 0-1 integer linear programming. Numerical results demonstrate the effectiveness and efficiency of the proposed approach.

Keywords: Two-phase linear programming, redundancy allocation.

1. INTRODUCTION

Highly reliable systems can reduce loss of money and time in the real world. Two approaches are generally available to enhance the system reliability, i.e., (i) using highly reliable components constituting the system, and/or (ii) using redundant components in various subsystems in the system (Misra and Sharma [24]). For the former approach, although system reliability can be enhanced, it is occasionally beyond our requirement even the highest reliable elements are used. Although using the latter approach enhances system reliability directly, the cost, weight, and volume of the system increase simultaneously. The redundancy allocation problem is to maximize system reliability subject to specific constraints, e.g. cost, weight and volume etc. The general formulation of this problem can be expressed as:

$$\begin{aligned} &\max \ R_s(x_1, x_2, ..., x_n) \\ &\text{st} \quad \sum_{j=1}^{n} g_{i,j}(x_i) \le b_i, \ i = 1, 2, ..., m \\ &\quad x_i \in \text{positive integer} \end{aligned} \tag{P}$$

where x_i is the number of parallel components in subsystem i.

There are numerous approaches for solving the redundancy allocation problem (P), including:

(i) Heuristics: [6, 14, 16, 29];
(ii) Artificial Algorithms: genetic algorithms [3, 4, 5, 30], simulated annealing [2, 11, 32], and tabu search [8];
(iii) Exact Methods: cutting plane [15], branch-and-bound [25], surrogate constraint method [28], dynamic programming [1, 22], implicit search [9]; and
(iv) Approximate Methods: Lagrange multiplier [33], geometric programming [7, 24], discrete maximum principle [21], sequential simplex search [33], random search [26], boundary search [20, 23], lexicographic search [31], differential dynamic programming [27].

Exact methods can find the optimal solutions for the problems, but they are usually time-consuming when the problem sizes are medium or large. Interested readers are referred to the excellent survey paper by Kuo and Prasad [18].

In this paper, we study the following series redundancy allocation problem with n subsystems and m linear constraints.

$$\text{(P1)} \qquad \max \ R(x_i) = \prod_{i=1}^{n} (1 - q_i^{x_i})$$

$$\text{s.t.} \quad \sum_{i=1}^{n} a_{ij} x_i \le b_j, \ j = 1, 2, ..., m \tag{1}$$

$$x_i \in \text{integer} \tag{2}$$

where $a_{ij} > 0$ denotes the resource requirement associated with each component of the j-th resource, q_i represents the failure probability of components in subsystem i, x_i is the number of components in the subsystem i, and b_j denotes the amount available for the j-th resource.

Previous investigations applied dynamic programming to solve problem (P1) for optimum allocation [1, 10]. Dynamic programming enumerates all possible cases and is time-consuming, particularly when system size is large. In addition to dynamic programming, with the relaxation of integer constraints (2), geometric programming is frequently used to solve problem (P1) (Federowicz and Mazumdar [7] and Misra and Sharma [24]). Once solutions are obtained by geometric programming, one may round off real solutions to the nearest integers. However, the integer solutions are not necessarily optimal any longer. Another drawback of geometric programming is that it is also time-consuming due to the complex transformations (see Misra and Sharma [24]). In a relevant study, Hochbaum [11] converted this redundancy allocation problem

into a 0-1 knapsack problem, indicating that by piecewise linear approximation approach, and the complexity closely resembles that of linear knapsack problem. However, his approach deals with only one single constraint, and it is NP-complete and not attractive in practice. In addition, several local search heuristics, e.g., genetic algorithms, tabu search algorithms and simulated annealing algorithms have also been used to solve related redundancy allocation problems (see Coit and Smith [4, 5], Hsieh, Chen and Bricker [13], Painton and Campbell [27]).

In light of above developments, this paper will propose a simple two-phase linear programming (LP) approach to solve redundancy allocation problem (P1). This proposed approach consists of two main phases, namely:

(i) Phase I: (Approximation stage) Initially, with the linear approximation of the objective function and the relaxation of integer constraints, a general LP is solved for the approximate solution of problem (P1).
(ii) Phase II: (Improving stage) A 0-1 knapsack problem with $n+m$ linear constraints is then solved to improve the real solutions of Phase I to (feasible) integer solutions.

Phase I is used to approximately allocate the available resource by using a general linear programming, while Phase II is to re-allocate the slacks of resource by using a 0-1 integer linear programming. Note that both phases can be easily implemented by general linear programming softwares, e.g., LINDO.

This paper is organized as follows. In Section 2, the linear approximation technique of problem (P1) for Phase I is presented. Section 3 describes a 0-1 knapsack problem with linear constraints for Phase II. An example is also provided to demonstrate the new approach in this section. Numerical results of random test problems are reported in Section 4. Finally, Section 5 briefly summarizes the paper.

2. PHASE I – APPROXIMATION STAGE

Our linearization of problem (P1) in Phase I is based on the following lemma.

Lemma 1. *For* $0 < q_i < 1,\ i = 1,\ldots,n$, $1 - \sum_{i=1}^{n} q_i^{x_i} \le \prod_{i=1}^{n} (1 - q_i^{x_i}) \le \left(1 - \sum_{i=1}^{n} q_i^{x_i} \Big/ n \right)^n$.

Proof: The left inequality holds by mathematical induction. The proof is straightforward and is omitted. The right inequality holds by the arithmetic-geometric mean inequality (Horn and Johnson [12]).

According to Lemma 1, the objective function of problem (P1), i.e., $\max \prod_{i=1}^{n} (1 - q_i^{x_i})$, can be approximated by $\min \sum_{i=1}^{n} q_i^{x_i}$. Moreover, the fact that $q_i^{x_i} \le \max_{1 \le i \le n} q_i^{x_i}$ for any i implies that $\min \sum_{i=1}^{n} q_i^{x_i} \le n \min \max_{1 \le i \le n} q_i^{x_i}$. Thus, problem (P1) can be approximately reformulated as:

$$\min \max_{1\le i\le n} q_i^{x_i}$$

s.t. (1) and (2)

By substituting $\max_{1\le i\le n} q_i^{x_i} = e^{-T}$ into the above formulation, taking the logarithm of $q_i^{x_i} \le e^{-T}$ for all i and relaxing the integer constraints of (2) lead to:

$$\text{Min } e^{-T}$$

$$\text{s.t. } x_i \ln q_i \le -T, \quad i = 1,2,...,n$$

$$\sum_{i=1}^{n} a_{ij} x_i \le b_j, \quad j = 1,2,...,m$$

$$x_i \ge 0, T \ge 0$$

or equivalently, that

(P2) Max T

$$\text{s.t. } T + x_i \ln q_i \le 0, \quad i = 1,2,...,n$$

$$\sum_{i=1}^{n} a_{ij} x_i \le b_j, \quad j = 1,2,...,m$$

$$x_i \ge 0, \ T \ge 0$$

Problem (P2) is a simple LP with $m+n$ linear constraints and it can be solved by general LP softwares, e.g., LINDO. For the special case of $m=1$, the closed form for the optimal solution of (P2) is given below.

Lemma 2. *For $m=1$, the optimal solution for (P2) is $x_i^* = b_1 \Big/ (\sum_{i=1}^{n} a_{i1}/\ln q_i) \ln q_i$, $i = 1,2,...,n$.*

Proof: Firstly, $b_1 \Big/ (\sum_{i=1}^{n} a_{i1}/\ln q_i) \ln q_i$ is a feasible solution of (P2). In the following, we show that no better solution is available for problem (P2).

Since there is only one main constraint with $a_{i1} > 0, b_1 > 0$ when $m=1$, the constraint must be tight (i.e. equality holds) for the optimal solution. Assume that there is a better solution $\tilde{x}$ of (P2) with objective $\tilde{T}$, where $\tilde{T} > T^* = -x_i^* \ln q_i$ for all i. This observation implies that there exists some i such that $\tilde{x}_i \ln q_i = -\tilde{T} < x_i^* \ln q_i = -T^*$ or equivalently that $\tilde{x}_i > x_i^*$. This further implies that $\sum_{k\ne i} a_{k1}\tilde{x}_k < \sum_{k\ne i} a_{k1} x_k^*$. Thus, some k $(k \ne i)$ must exist such that $\tilde{x}_k < x_k^*$, further implying that $-\tilde{T} = \tilde{x}_k \ln q_k > x_k^* \ln q_k = -T^*$. This is a contradiction.

3. PHASE II – IMPROVING STAGE

Once x_i^* is obtained by problem (P2), the solutions can be rounded off to their nearest integers. However, such integer solutions are not necessarily feasible or optimal. Based on the solution of (P2), a simple 0-1 knapsack problem is introduced in the following to obtain a feasible integer solution.

Let $\bar{x}_i = \lfloor x_i^* \rfloor$ (i.e., round down the real solution to integer), then we may solve the following 0-1 knapsack problem for the optimal allocation of unused resources by general LP softwares, e.g. LINDO.

(P3) $\quad \text{Max} \sum_{i=1}^{n} \sum_{k=1}^{J[i]} x_{ik} r_{ik}$

$$\text{s.t.} \sum_{i=1}^{n} \sum_{k=1}^{J[i]} k a_{ij} x_{ik} \le b_j - \sum_{i=1}^{n} a_{ij} \bar{x}_i , \quad j = 1,2,...,m$$

$$\sum_{k=0}^{J[i]} x_{ik} = 1, \quad i = 1,2,...,n$$

$$x_{ik} = 0 \text{ or } 1$$

where $J[i] = \min\left\{3, \max_j \left\lfloor (b_j - \sum_{i=1}^{n} a_{ij}\bar{x}_i)/a_{ij} \right\rfloor\right\}$ and $r_{ik} = \ln\{(1-q_i^{\bar{x}_i+k})/(1-q_i^{\bar{x}_i})\}$. If $x_{ik}^* = 1$ is the optimal solution of problem (P3), then $k + \bar{x}_i$ is the approximation of optimal integer solution for problem (P1). It is clear that problem (P3) is a multiple choice knapsack problem and several typical approaches can be used for solving such a problem (Lin [29]). Notably, (P3) has $n+m$ linear constraints and has at most $4n$ binary variables which subsequently produces a feasible integer solution for problem (P1). Therefore, this approach is more practical than Hochbaum's piecewise linear approximation approach (Hochbaum [11]) when the problem size is large. Next, an example is provided to demonstrate both phases of the new approach.

Example. (Misra and Sharma [24])

$$\max \; R(\mathrm{x}) = (1-0.2^{x_1})(1-0.3^{x_2})(1-0.25^{x_3})(1-0.15^{x_4}) \tag{3}$$

$$\text{s.t.}\; 1.2x_1 + 2.3x_2 + 3.4x_3 + 4.5x_4 \le 56 \tag{4}$$

$$x_1 + x_2 + x_3 + x_4 \le 30 \tag{5}$$

$$x_i \;\; \text{integer}$$

Its corresponding problem (P2) is

(Phase I)

$$\max\ T$$

$$\text{s.t. } T - 1.609438x_1 \le 0$$

$$T - 1.203973x_2 \le 0$$

$$T - 1.386294x_3 \le 0$$

$$T - 1.897120x_2 \le 0$$

$$1.2x_1 + 2.3x_2 + 3.4x_3 + 4.5x_4 \le 56$$

$$x_1 + x_2 + x_3 + x_4 \le 30$$

$$x_i \ge 0,\ T \ge 0$$

For this case, the optimal solution by LINDO is (4.651368, 6.217820, 5.400073, 3.946028). Note that, by nonlinear programming softwares, e.g., GINO, the solution of this example (with nonlinear objective and relaxing the integer constraints) is (5.247578, 6.289277, 5.257760, 3.858401) which is close to that by LINDO. By the improving stage, we have $\bar{x} = (4,6,5,3)$ and $J = (3,3,2,1)$. Its corresponding problem (P3) is:

(Phase II)

$$\max\ 0.001281x_{11} + 0.001537x_{12} + 0.001588x_{13} + 0.000511x_{21} + 0.000664x_{22}$$
$$+0.000710x_{23} + 0.000733x_{31} + 0.000916x_{32} + 0.002874x_{41}$$

$$\text{s.t. } 1.2x_{11} + 2.4x_{12} + 3.6x_{13} + 2.3x_{21} + 4.6x_{22} + 6.9x_{23} + 3.4x_{31} + 6.8x_{32} + 4.5x_{41} \le 6.9$$

$$x_{11} + 2x_{12} + 3x_{13} + x_{21} + 2x_{22} + 3x_{23} + x_{31} + 2x_{32} + x_{41} \le 12$$

$$x_{10} + x_{11} + x_{12} + x_{13} = 1$$

$$x_{20} + x_{21} + x_{22} + x_{23} = 1$$

$$x_{30} + x_{31} + x_{32} = 1$$

$$x_{40} + x_{41} = 1$$

$$x_{ij} = 0 \text{ or } 1$$

By using LINDO to solve problem (P3) again, we have $x_{12}^* = x_{20}^* = x_{30}^* = x_{41}^* = 1$. Therefore, the final solution by the proposed two-phase approach for (P1) herein is (4+2, 6+0, 5+0, 3+1) = (6, 6, 5, 4). Notably, it is optimal for this redundancy allocation problem. We also test a random problem with twenty-five subsystems (variables) in problem (P1), and the results show that the new two-phase approach can obtain the optimal solution within one second (CPU time of Phase I and Phase II). However, the

integer solver, e.g., LINGO requires more than fifteen minutes for the optimal solution by the branch-and-bound algorithm.

4. NUMERICAL RESULTS

To briefly evaluate the performance of the proposed two-phase approach, we execute the following experiments with single linear constraint for problem (P1).

(i) The component weights $w_i\ (=a_{i1})$ are randomly generated from [1, 11] and the maximal available weight in the system is set to $W=\left\{\left\lfloor \sum_{i=1}^{n} w_i \right\rfloor\right\}^{1.3}\ (=b_1)$.

(ii) The failure probability of components in subsystem i is randomly generated from uniform distribution (U) or triangular distribution (T), respectively, with intervals (0, 0.1), (0, 0.2), (0, 0.3), (0, 0.4), and (0, 0.5).

(iii) The number of subsystems n varies from 15, 20 to 25.

(iv) For each case of combination of (i)-(iii), we test 10 random problems by using both the proposed two-phase approach and the branch-and-bound algorithm for comparison.

(v) To improve the optimality of the two-phase approach, we also test $\bar{x}_i=\lfloor x_i^* \rfloor$ and $\bar{x}_i=\lfloor x_i^* \rfloor-1$ in problem (P3), respectively, for comparison.

Table 1 summarizes the numerical results. From Table 1, we observe that:

1. The new two-phase approach is able to obtain the optimal solution for the redundancy allocation problem, especially, when we set $\bar{x}_i=\lfloor x_i^* \rfloor-1$ in problem (P3). This is because that we round off the real solutions by Phase I to the nearest integers and then minus one. Though the integer solutions are not necessarily optimal any longer, but they are generally close to the optimal solutions. Hence with the use of Phase II, the two-phase approach can easily obtain the optimal solutions.
2. The CPU times increase drastically with the increase of problem size n for the branch-and-bound algorithm. For example, it requires 23.45, 109.12 and 512.45 seconds in average for $n=15,\ 20$ and 25, respectively. This is because that the number of branches for the branch-and-bound algorithm increases with the increase of problem size. However, the CPU times are all within 2 seconds for the two-phase approach for $n=15,\ 20$ and 25, respectively.
3. Under the same intervals, there seems no significant difference between uniform distribution and triangular distribution in the test problems for both branch-and-bound algorithm and the new two-phase approach.

Table 1: Numerical results for random test problems.

Test problem		Approaches [(i)]				
		Branch-and-bound		Two-phase approach		
No. of components	Component failure probability Uniform: U Triangular: T	Average CPU time (seconds)	Average No. branches	Average CPU time (seconds)	Average optimality (%) (ii)	Average optimality (%) (iii)
$n = 15$	$U(0,0.1)$	8.0	187.1	<2	100	100
	$U(0,0.2)$	18.2	255.3	<2	100	100
	$U(0,0.3)$	23.1	344.5	<2	100	100
	$U(0,0.4)$	23.6	319.5	<2	70	100
	$U(0,0.5)$	22.8	344.0	<2	40	100
	$T(0,0.1)$	18.3	386.0	<2	100	100
	$T(0,0.2)$	33.8	529.3	<2	100	100
	$T(0,0.3)$	21.5	312.2	<2	100	100
	$T(0,0.4)$	30.8	406.0	<2	100	100
	$T(0,0.5)$	34.4	435.7	<2	100	100
	Total average	23.5	352.0	<2	91	100
$n = 20$	$U(0,0.1)$	15.0	319.3	<2	100	100
	$U(0,0.2)$	104.2	1120.0	<2	100	100
	$U(0,0.3)$	88.8	922.2	<2	100	100
	$U(0,0.4)$	236.1	1525.0	<2	100	100
	$U(0,0.5)$	114.8	1026.0	<2	40	100
	$T(0,0.1)$	14.8	303.6	<2	100	100
	$T(0,0.2)$	146.1	1145.0	<2	100	100
	$T(0,0.3)$	118.8	1380.0	<2	80	100
	$T(0,0.4)$	105.3	1042.0	<2	80	100
	$T(0,0.5)$	147.3	1457.3	<2	60	100
	Total average	109.1	1024.0	<2	86	100
$n = 25$	$U(0,0.1)$	48.7	545.7	<2	70	100
	$U(0,0.2)$	845.4	2853.0	<2	100	100
	$U(0,0.3)$	787.6	2923.1	<2	90	100
	$U(0,0.4)$	569.3	2559.2	<2	70	100
	$U(0,0.5)$	608.6	2517.6	<2	20	100
	$T(0,0.1)$	13.4	139.2	<2	90	100
	$T(0,0.2)$	488.6	2329.5	<2	100	100
	$T(0,0.3)$	360.8	1991.4	<2	100	100
	$T(0,0.4)$	708.8	3955.4	<2	80	100
	$T(0,0.5)$	693.3	2968.7	<2	40	100
	Total average	512.5	2278.3	<2	76	100

(i) Two approaches are used to solve 10 random test problems for each n and component failure probability.

(ii) In Phase II, we set $\bar{x}_i = \lfloor x_i^* \rfloor$ in problem (P3).

(iii) In Phase II, we set $\bar{x}_i = \lfloor x_i^* \rfloor - 1$ in problem (P3).

4. The intervals of failure probabilities for components will effect the CPU times for the branch-and-bound algorithm. For example, when $n = 25$, the mean solving time for test problems with $U(0,0,1)$ is 48.7 seconds, while it is 608.6 seconds for test problems with $U(0,0,5)$. Similar results are for triangular distribution.

5. CONCLUSIONS

This paper has presented a novel two-phase LP approach for solving the typical redundancy allocation problem with multiple linear constraints. The proposed approach is simpler than conventional approaches, e.g. dynamic programming, geometric programming and piecewise linear approximation approaches. Any linear programming softwares, such as LINDO, can be used to implement the LP approach proposed herein. Although no guarantee ensures that the approach proposed herein derives the optimal solutions, limited numerical results demonstrate the efficiency and the effectiveness of the proposed approach. However, one should note that for some few reliability problems none of the optimal integer solutions are near the original approximations. But, as shown, another merit of this two-phase LP approach is that the solution obtained by the proposed approach must be a feasible integer solution. Therefore, the new two-phase approach might also provide a good lower bound for branch-and-bound algorithm or artificial methods, such as genetic algorithms when the problem size is very large.

Acknowledgements: This research is partially supported by National Science Council, Taiwan, under grant No. NSC 90-2218-E-150-007. The author would like to thank an anonymous reviewer for his/her helpful comments and suggestions that greatly improved the presentation of this paper.

REFERENCES

[1] Bellman, H. E., and Dreyfus, E., "Dynamic programming and reliability of multicomponent devices", *Operations Research,* 6 (1958) 300-206.

[2] Cardoso, M. F., Salcedo, R. L., and de Azevedo, S. F., "Non equilibrium simulated annealing: a faster approach to combinatorial minimization", *Industrial Engineering Chemical Research*, 33 (1994) 1908-1918.

[3] Coit, D. W., and Smith, A., "Reliability optimization of series-parallel systems using a genetic algorithm", *IEEE Trans. Reliability*, 45 (1996) 254-260.

[4] Coit, D. W., and Smith, A., "Solving the redundancy allocation problem using a combined neural network / genetic algorithm approach," *Computers and Operations Research*, 23 (1996) 515-526.

[5] Dengiz, B., Altiparmak, F., and Smith, A. E., "Efficient optimization of all-terminal reliable networks using an evolutionary approach", *IEEE Trans. Reliability*, 46 (1997) 18-26.

[6] Dinghua, S., "A new heuristic algorithm for constrained redundancy-optimization in complex systems", *IEEE Trans. Reliability*, 36 (1987) 621-623.

[7] Federowicz, A. J., and Mazumdar, M., "Use of geometric programming to maximize reliability achieved by redundancy", *Operations Research*, 19 (1968) 948-954.

[8] Glover, F., Laguna, M., *Tabu Search,* Kluwer Academic Publishers, 1997.
[9] Geoffrion, A. M., "Integer programming by implicit enumeration and Balas' method", *Soc. Industrial and Applied Mathematics Review*, 9 (1967) 178-190.
[10] Hiller, F. S., and Lieberman, G. J., *An Introduction to Operations Research*, McGraw-Hill, NY, 1995.
[11] Hochbaum, D. S., "A nonlinear knapsack problem", *Operations Research Letters*, 17 (1995) 103-110.
[12] Horn, R. A., and Johnson, C. R., *Matrix Analysis*, Cambridge University Press, London, 1985.
[13] Hsieh, Y. C., Chen, T. C., and Bricker, D. L., "Genetic Algorithms for reliability design problems", *Microelectronics and Reliability* 38 (1998) 1599-1605.
[14] Jianping, L., "A bound heuristic algorithm for solving reliability redundancy optimization", *Microelectronics and Reliability*, 3 (1996) 335-339.
[15] Kelley, J., "The cutting plane method for solving convex program", *J. Soc. Industrial and Applied Mathematics,* 8 (1960) 708-712.
[16] Kim, J. H., and Yum, B. J., "A heuristic method for solving redundancy optimization problems in complex systems", *IEEE Trans. Reliability*, 42 (1993) 572-578.
[17] Kirkpatrick, S. Gelatt Jr., C. D., and Vecchi, M. P., "Optimization by simulated annealing", IBM Research Report RC 9355, 1982.
[18] Kuo, W., and. Prasad, V. R, "An annotated overview of system-reliability optimization", *IEEE Trans. Reliability*, 49 (2000) 176-187.
[19] Lin, Y.H., "A bibliographical survey on some well-known non-standard knapsack problems", *INFORS*, 36 (1998) 274-317.
[20] Misra, K. B., "An algorithm to solve integer programming problems: An efficient tool for reliability design", *Microelectronics and Reliability*, 31 (1991) 285-294.
[21] Misra, K. B., "On optimal reliability design:A review", *System Science*, 12 (1986) 5-30.
[22] Misra, K. B., "Dynamic programming formulation of redundancy allocation problem", *International J. of Mathematical Education in Science and Technology*, 2 (1971) 207-215.
[23] Misra, K. B., and Sharma, U., "An efficient algorithm to solve integer programming problems arising in system reliability design," *IEEE Trans. Reliability*, 40 (1991) 81-91.
[24] Misra, K. B., and Sharma, U., "A new geometric programming formulation for a reliability problem", *International Journal of Control*, 18 (1973) 497-503.
[25] Misra, K. B., and Sharma, U., "Reliability optimization of a system by zero-one programming", *Microelectronics and Reliability*, 12 (1973) 229-233.
[26] Mohan, C., and Shanker, K., "Reliability optimization of complex systems using random search technique", *Microelectronics and Reliability*, 28 (1988) 513-518.
[27] Murray, D. M., and Yakowitz, S. L., "Differential dynamic programming and Newton's method for discrete optimal control problems", *Journal of Optimization Theory and Applications*, 43 (1984) 395-414.
[28] Nakagawa, Y., and. Miyazaki, S., "Surrogate constraints algorithm for reliability optimization problem with two constraints", *IEEE Trans. Reliability*, 30 (1981) 175-180.
[29] Nakagawa, Y., and Miyazaki, S., "An experimental comparison of the heuristic methods for solving reliability optimization problems", *IEEE Trans. Reliability*, 30 (1981) 181-184.
[30] Painton, L., and Campbell, J., "Genetic algorithms in optimization of system reliability", *IEEE Trans. Reliability,* 44 (1995) 172-178.
[31] Prasad, V. R., and Kuo, W., "Reliability optimization of coherent systems", *IEEE Trans. Reliability*, 2000, to be published.
[32] Ravi, V., Burty, B., and Reddy, P., "Nonequilibrium simulated-annealing algorithm applied reliability optimization of complex systems", *IEEE Trans. Reliability*, 46 (1997) 233-239.
[33] Tillman, F. A. Hwang, C. L., and Kuo, W., *Optimization of System Reliability,* Marcel Dekker, 1980.