International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:2, 2013

Improved Ant Colony Optimization for Solving Reliability Redundancy Allocation Problems

Phakhapong Thanitakul, Worawat Sa-ngiamvibool, Apinan Aurasopon, Saravuth Pothiya

***Abstract*—**This paper presents an improved ant colony optimization (IACO) for solving the reliability redundancy allocation problem (RAP) in order to maximize system reliability. To improve the performance of ACO algorithm, two additional techniques, i.e. neighborhood search, and re-initialization process are presented. To show its efficiency and effectiveness, the proposed IACO is applied to solve three RAPs. Additionally, the results of the proposed IACO are compared with those of the conventional heuristic approaches i.e. genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO). The experimental results show that the proposed IACO approach is comparatively capable of obtaining higher quality solution and faster computational time.

***Keywords*—**Ant colony optimization, Heuristic algorithm, Mixed-integer nonlinear programming, Redundancy allocation problem, Reliability optimization.

I. Introduction

RELIABILITY optimization is very important in the real world system such as electric power, communication, and computer systems. Normally two major methods have been used to achieve higher system reliability. The first method is increasing the reliability of system components, the system reliability can be improved to some degree, but the required reliability enhancement may be never attainable even though the most currently reliable elements are used. The second method is using redundant components in various subsystems, this method requires one to choose the optimal element combination and redundancy-levels; the system reliability can be also enhanced, but the cost, weight, volume, etc. will be increased as well. Such problems of maximizing system reliability through redundancy and component reliability choices are called the ‘‘reliability-redundancy allocation problem’’. The redundancy allocation problem (RAP) may be the most common problem in the design for reliability approach [1]. It involves setting reliability objectives for components or subsystems in order to meet the resource consumption constraint, e.g. the total cost. Hence, RAP is becoming an increasingly important tool in the initial stages of or prior to the planning, designing and control of systems. The reliability design problems include the integer problem and mixed-integer reliability problem. The major focus of recent work in the RAPs has been on the development of heuristic/meta-heuristic algorithms for redundancy allocation [2], [3]. But a few approaches were proposed for the mixed-integer reliability problems of optimizing both the redundancy and component reliability [4] in literature. These system reliability problems are either subjected to the linear constraints or to the nonlinear constraints [5], [6].

Recently, an ant colony optimization (ACO) approach has been proposed [7], [8] and successfully applied to the combinatorial optimization problem such as traveling salesman problem (TSP) [9], [10], quadratic assignment problem [11], vehicle routing problem [12] and job-shop scheduling problem. Therefore, this paper presents an improved ACO to solve RAPs where formulated as maximizing reliability subject to cost constraint. This improved ant colony optimization (IACO) which introduces addition improvement procedures, i.e. neighborhood search and re-initialization.

The remainder of this paper is organized as follows: in Section II, problem formulation and assumptions of system is described. In Section III, basic principle of convention ACO, application of ACO for solving RAP, principle of a proposed IACO approach and its application to this problem are presented. Section IV, illustrative examples and simulation results are given. Finally, discussion and conclusion are revealed in Section V.

II. Problem Formulation

A. Acronyms and Notations

Acronyms

ACO	ant colony optimization
IACO	Improved ant colony optimization
GA	genetic algorithm
PSO	particle swarm optimization
RAP	redundancy allocation problem
TSP	traveling salesman problem

Notations

n	number of subsystems (units)
m	number of constraints
i	index for subsystem (unit), $i = 1,2,...,n$
j	index for redundancy values
r_i	reliability of a component for system (unit) i that uses identical redundancy
x_i	variable to denote redundancy at subsystem (unit) i, $x_i \in Z^+$
x_{ij}	j th value of x_i

Phakhapong Thanitakul is with the department of electrical and computer, faculty of engineering, Mahasarakarm University, Mahasarakarm, 44150, Thailand (Tel: 66-43-754321-40 Ext 3010-3058; e-mail: p_thanitakul@yahoo.com).

Worawat Sa-ngiamvibool, Apinan Aurasopon, and Saravuth Pothiya are with the Department of Electrical and Computer, Faculty of Engineering, Mahasarakarm University, Mahasarakarm, 44150, Thailand (e-mail: wor_nui@yahoo.com, aurasopon@yahoo.com, saravuthpothiya@yahoo.com).

$x = (x_1, x_2, ..., x_n)$ a general solution vector

x^* final best feasible solution

l_i lower limit of x_i

$l = (l_1, l_2, ..., l_n)$ lower limit vector

$N_i^0 \equiv \{l_i, l_i + 1, ..., u_i\}$ set of initial redundancy levels for subsystem (unit) i

N_i set of available redundancy levels for subsystem (unit) at an iteration, $N_i \subset N_i^0$

$S = \underset{i=1}{\overset{n}{\times}} N_i^0$ search space

$\|A\|$ number of elements in set A

B. Redundancy Allocation Problem

The system consists of n units or subsystems. In subsystem i, at least l_i number of active components are required to function, which constitutes the lower bound of the redundancy level for that subsystem and is pre-specified. The upper bound u_i of the redundancy level for the unit i is either given in advance or can be obtained by solving the system constraints, if linear.

To solve a RAP is all about to find the optimal allocation of components which maximizes reliability of the system satisfying the given resource constraints. The problem can be formulated as the following non-linear integer-programming problem [13].

$$\text{Maximize } R_s(x) \tag{1}$$

$$\text{Subject to } g_y(x) = \sum_{i=1}^{n} g_{yi}(x_i) \le b_y, \quad y = 1,2,...,m \tag{2}$$

$$l_i \le x_i \le u_i, \quad i = 1,2,...,n$$

$$l_i, u_i, x_i \in Z^+ \quad i = 1,2,...,n$$

Assumptions

(1). The system and all its subsystems are s-coherent.
(2). The states of the components are mutually s-independent, i.e., the failure of a component is independent of that of others.
(3). The components and the system can have only two states viz. good or failed.
(4). Failed components do not damage the system and are not repaired.
(5). Component attributes such as reliability, cost, weight etc. are fixed.

III. Improved Ant Colony Optimization

A. Basic Principle of Ant Colony

Inspired by the collective behavior of a real ant colony, Marco Dorigo first introduced the Ant System (AS) in his Ph.D. thesis (1992) and the study was further continued by Dorigo et al. [7], [8]. The characteristics of an artificial ant colony include positive feedback, distributed computation and the use of a constructive greedy heuristic. Positive feedback accounts for rapid discovery of good solutions, distributed computation avoids premature convergence and the greedy heuristic helps to find acceptable solutions in the early stages of the search process. In order to demonstrate the AS approach, the authors apply this approach to the classical TSP, asymmetric TSP, quadratic assignment problem (QAP) and job-shop scheduling. The AS shows very good results in each applied area. More recently, Dorigo and Gambardella have worked on extended versions of the AS paradigm. ACO is one of the extensions and has been applied to the symmetric and asymmetric TSP with excellent results [9], [10]. The Ant System has also been applied with success to other combinatorial optimization problems such as the vehicle routing problem [11].

ACO is an algorithm which was inspired by the behavior of real ants. Ethnologists have studied how blind animals such as ants capable of finding the shortest path from food sources to the nest without using visual cues. They are also capable of adapting to changes in the environment. For example, finding a new shortest path once the old one is no longer feasible due to a new obstacle. The studies of ethnologists reveal that such capabilities are essentially due to communicating information among individuals regarding path to decide the direction. Ants deposit a certain amount of pheromone while walking and each ant probabilistically prefers to follow a direction rich in pheromone rather than a poorer one [12].

B. ACO for Solving RAP

The ACO approach to redundancy allocation in complex systems involves a colony of artificial ants moving on each unit by selecting redundancy levels. Each ant represents a design of the entire system, a collection x_i of components in parallel $(l_i \le x_i \le u_i)$ for n units. Thus, a complete solution (ant) is a vector $x = (x_1, x_2, ..., x_n)$.

In general, the procedure of ACO algorithm can describe as follows: m ants are initially positioned at the nest. Each ant will choose a possible route as a solution. In fact, each ant builds a feasible solution (called a tour) by repeatedly applying a stochastic greedy search, called, *the state transition rule*. Once all ants have terminated their tours, the following steps are performed:

The amount of pheromone is modified by applying the *global updating rule*. Ants are guided, in building their tours, by both heuristic information and pheromone information. Naturally, a redundancy level with a high amount of pheromone is a desirable choice. The pheromone updating rules are designed so that they tend to give more pheromone to edges, which should be visited by ants.

The detail of ACO algorithm can be described in the following steps [14].

Step 1 Initialization
Set NC = 0 /* NC: cycle counter */
For every combination (i,j).
Set an initial value $\tau_{ij}(0) = \tau_0$ and $\Delta\tau_{ij} = 0$
End

Step 2 Construct feasible solutions
For k=1 to m /m: number of ants/
For i=1 to n/* n: number of subsystem*/
Choose a redundancy level with transition probability given by (3).
End
Evaluate objective function
Check constraints
End
Update the best solution
Step 3 Global updating rule
For every combination (i,j)
For k=1 to m
Find $\Delta\tau_{ij}^{k}$ according to (7)
End
Update $\Delta\tau_{ij}$ according to (6)
End
Update the trail values according to (5)
Update the transition probability according to (3)
Step 4 Next search
Set NC = NC+1
For every combination (i,j)
$\Delta\tau_{ij} = 0$
End
Step 5 Termination
If (NC < NC_{max})
Then
Goto step 2
Else
Stop
End

1. State Transition Rule

The state transition rule of the ant colony is given in (3). This equation represents the probability that ant k selects a redundancy level j for subsystem i:

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{m=1}^{M_i}\left[\tau_{im}(t)\right]^{\alpha}\left[\eta_{im}(t)\right]^{\beta}} \quad (3)$$

where τ_{ij} is the pheromone intensity and η_{ij} is the heuristic information between subsystem i and redundancy level j, respectively. In addition, α is the relative importance of the trail and β is the relative importance of the heuristic information. The problem specific heuristic information is:

$$\eta_{ij} = \frac{1}{C_{ij}} \quad (4)$$

where C_{ij} represents the associated cost. Therefore, the redundancy level of subsystem with smaller cost has greater probability to be chosen.

2. Global Updating Rule

During the solution construction, it is no guarantee that an ant will construct a feasible solution, which obeys the reliability constraint. The pheromone updating treats the unfeasible solution. The amount of pheromone, deposited by ants, is set to a high value if the generated solution is feasible. On other hand, this value is set to a low value if it is infeasible. Therefore, this value depends on the solution quality. Infeasibility can be handled by assigning the penalty which proportion to the amount of reliability violations. In case of feasible solution, an additional penalty is introduced to improve its quality.

Following the above remarks, the trail intensity is updated as follows:

$$\tau_{ij}(new) = \rho\tau_{ij}(old) + \Delta\tau_{ij} \quad (5)$$

ρ is a coefficient such that $(1-\rho)$ represents the evaporation of trail and $\Delta\tau_{ij}$ is:

$$\Delta\tau_{ij} = \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \quad (6)$$

where m is the number of ants and $\Delta\tau_{ij}^{k}$ is given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q\cdot penalty_k\cdot C_k & \text{if the } k^{th} \text{ ant chooses redundancy level } j \text{ for subsystem } i \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where Q is a positive number, and $penalt\ y_k$ is defined as follows:

$$penalt\ y_k = \left(\frac{C_k}{C^*}\right)^{a} \quad (8)$$

C_k is the cost obtained by the k^{th} ant, C^* is the best obtained solution. Parameter a represents the relative importance of penalties.

C. IACO for Solving RAP

A major weakness of conventional ACO algorithm is stagnation that is all ants take the same position. If this problem situation occurs, the algorithm may be trapped in a local optimal point. To alleviate the stagnation problem of conventional ACO algorithms, two improvement procedures are applied in order to improve the ant colony optimization method to guarantee the diversity of ants. This approach is called improved ant colony optimization (IACO). The additional procedures are a specific improvement algorithm (called *neighborhood search*) and *re-initializations* [14].

The neighborhood search algorithm is shown at step 3 in IACO's algorithm, and it proceeds to change in turn each redundancy level of chosen subsystem by another redundancy level. For each subsystem, redundancy levels are indexed in ascending order in accordance with their reliability. A solution $S = \{u, v, ...\}$ indicates that subsystem 1 uses redundancy level with index u, subsystem 2 uses redundancy level with index v, etc.

During searching, the search process gets the repeated solution for a long time. That means the process could not be found the better solutions or escaped from this solution. So this solution could be either a local or global solution. In case the global solution is known, the search process will be stopped as it has been gotten the best solution. On the other hand the global solution is unknown. The algorithm will assume that this solution is a local solution. If the process is stroke on local solution for a long time, the re-initialization process will be applied. This mechanism helps the process to continue searching and find the better solutions.

The detail of IACO algorithm can be described in the following steps.

Step 1 Initialization
Set NC = 0 /* NC: cycle counter */
For every combination (i,j)
Set an initial value $\tau_{ij}(0) = \tau_0$ and $\Delta\tau_{ij} = 0$
End
Step 2 Construct feasible solutions
For k=1 to m /* m: number of ants */
For i=1 to n/* n: number of subsystem*/
Choose a redundancy level with transition probability given by (3)
End
Evaluate objective function
Check constraints
End
Update the best solution
Step 3 Apply the neighborhood search
For k=1 to m
For i=1 to (2*n)
If i = odd
Change the chosen redundancy level of subsystem i with level p by level p+1
Else
Change the chosen redundancy level of subsystem i with level p by level p-1
End
Calculate cost C_k
If $C_k \geq C_o$
Except for exchanging
Record the obtained solution
Else
Do not except for exchanging
End
Calculate cost C_k
End
Update the best solution
Step 4 Global updating rules
For every combination (i,j)
For k=1 to m
Find $\Delta\tau_{ij}^k$ according to (7)
End
Update $\Delta\tau_{ij}$ according to (6)
End
Update the trail values according to (5)
Update the transition probability according to (3)
Step 5 Next search
Set NC = NC+1
For every combination (i,j)
$\Delta\tau_{ij} = 0$
End
Step 6 Re-initializations
If the best solution has not been improved for a long time
Then
Set an initial value $\tau_{ij}(0) = \tau_0$
End
Step 7 Termination
If (NC < NC_{max})
Then
Goto step 2
Else
Print the best feasible solution
Stop
End

IV. Numerical Examples and Computations

To evaluate the performance of proposed approach in solving the mixed-integer nonlinear reliability design problems, three test problems are solved. We conducted tests on the fixed data problems involving 4 and 5-unit complex structures subject to linear constraints [15] as well as a 4-stage series system [15] with non-linear constraints. All optimization methods (GA, PSO, ACO, and IACO) were implemented in MATLAB® package and the simulation cases done on a Intel® Core2 Duo 1.66 GHz Laptop with 2 GB RAM under Windows XP. Each studied systems was run 30 times with differential random initial solutions. In order to evaluate the performance of each technique, the best, worst, average, and standard deviation of the generation costs and the average of computational time to get near optimum solution are used for evaluation.

Example 1: 4-Unit system (Fig. 1) with two linear constraints. The problem data is given in Table I.

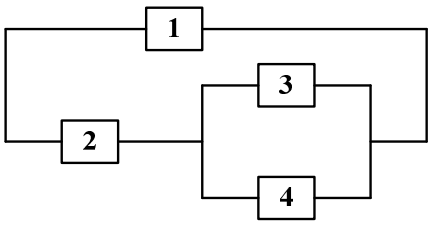

Fig. 1 4-Unit system

TABLE I
Subsystem Data for 4-Unit System

i	1	2	3	4
r_i	0.80	0.75	0.70	0.65
c_i	6	4	3	2
w_i	9	4	4	3

Maximize $R_s(x) = R_1 + Q_1R_2R_4 + Q_1R_2R_3Q_4$

subject to $g_1 = \sum_{i=1}^{4} c_i x_i \leq 30$

$g_2 = \sum_{i=1}^{4} w_i x_i \leq 40$

$x_i \in Z^+, \quad i = 1,2,3,4.$

$l=(1,1,1,1)$ $u=(3,4,6,7)$ $\|S\|=504$

The best solution found is $x^*=(3,1,1,1)$ for which $R^*=0.997370$. The algorithm was run for the problem 30 times. The results show in the Table II.

TABLE II
RESULTS OF EXAMPLE 1 USING FOUR ALGORITHMS

Algorithm	Max	Average	Min	Std.	CPU Time
GA	0.997370	0.997262	0.997086	0.00013	4.68
PSO	0.997370	0.9973643	0.997086	0.00003	0.74
ACO	0.997370	0.9973432	0.996115	0.00013	0.65
IACO	0.997370	0.997370	0.997370	0.00000	0.06

Example 2. 5-Unit Bridge system (Fig. 2) with one linear constraint
Maximize

$$R_s(x)=R_5(1-Q_1Q_3)(1-Q_2Q_4)+Q_5\{1-(1-R_1R_2)(1-R_3R_4)\}$$

subject to $g(x)=\sum_{i=1}^{5}c_ix_i\le 20$

$$x_i\in Z^+,\ \ i=1,2,3,4,5$$

$$l=(1,1,1,1,1) \qquad u=(5,4,5,4,10) \qquad \|S\|=4000$$

The problem data is given in Table III.

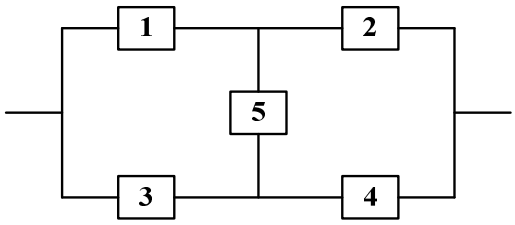

Fig. 2 5-Unit bridge system

TABLE III
SUBSYSTEM DATA FOR 5 UNIT BRIDGE SYSTEM

i	1	2	3	4	5
r_i	0.70	0.85	0.75	0.80	0.90
c_i	2	3	2	3	1

The problem data is given in Table III. The best feasible solution found is $x^*=(3,2,2,1,1)$ for which $R^*=0.993216$. The algorithm was run for the problem 30 times. The results of the computation time of the algorithm are as in the Table IV.

TABLE IV
RESULTS OF EXAMPLE 2 USING FOUR ALGORITHMS

Algorithm	Max	Average	Min	Std.	CPU Time
GA	0.993216	0.993109	0.990775	0.00032	1.46
PSO	0.993216	0.993163	0.991779	0.00022	0.86
ACO	0.993216	0.993115	0.992096	0.00029	1.45
IACO	0.993216	0.993216	0.993216	0.00000	0.148

Example 3. 4-Stage series system with non-linear constraints [16] Consider a 4-stage series system. Stage 1 does not allow component redundancy, but choosing a more reliable component at that stage can enhance its reliability. There are six component choices with reliabilities $r_1(1),r_1(2),...,r_1(6)$ at stage 1. The stages 2 and 4 have identical redundancies in parallel. However, stage 3 has a 2-out-of-n: G configuration.

Maximize $R_s(x)=\prod_{i=1}^{4}R_i(x_i)$

Subject to

$$g_1(x)=10\exp\left(\frac{0.02}{1-R_1(x_1)}\right)+10x_2+6x_3+15x_4\le 150$$

$$g_2(x)=10\exp(0.5x_1)+4\exp(x_2)+2(x_3+\exp(0.25x_3))+6x_4^2\le 750$$

$$g_3(x)=40x_1^2+6\exp(x_2)+3x_3\exp(0.25x_3)+8x_4^2\le 750$$

$x_i\ge 1,\ i=1,2,\ldots,4,$ where stage reliabilities are

$R_1(x_1)=0.94,0.95,0.96,0.965,0.97,0.975,$ for $x_1=1,2,\ldots,6,$ resp.

$$R_2(x_2)=1-(1-0.75)^{x_2}=1-(0.25)^{x_2}$$

$$R_3(x_3)=\sum_{k=2}^{x_3}\binom{x_3}{k}(0.9)^k(0.1)^{x_3-k}$$

$$R_4(x_4)=1-(1-0.95)^{x_4}=1-(0.05)^{x_4}$$

$l=(1,1,2,1)$ $u=(4,3,11,4)$ $\|S\|=480$

The best feasible solution found is $x^*=(3,3,5,3)$ for which $R^*=0.944447$. Table V summarizes the results of the computation time of the algorithm.

TABLE V
RESULTS OF EXAMPLE 3 USING FOUR ALGORITHMS

Algorithm	Max	Average	Min	Std.	CPU Time
GA	0.944447	0.935039	0.935039	0.0014	0.56
PSO	0.944447	0.942636	0.942636	0.0003	0.94
ACO	0.944447	0.935039	0.935039	0.0011	2.44
IACO	0.944447	0.944447	0.944447	0.0000	0.27

The comparison results of the proposed IACO with other three methods (GA, PSO and ACO) are given in Tables II, IV and V. The results show that the IACO succeeds in finding the best solution. The maximum, average, and minimum reliability achieved by IACO for 30 runs are much better than those of other methods. Besides, the lowest standard deviation (Std.) and the average computational time indicate that IACO has performance higher robustness and faster than other methods.

V. CONCLUSION

In this paper, the proposed IACO algorithm has been proposed to solve the reliability allocation problem. To improve the search process, two techniques, i.e. neighborhood

search and re-initialization are added to the ACO. Three examples of RAP have been used to evaluate the performance of IACO. Additionally, the results of IACO are compared with those obtained by the conventional heuristic approaches, i.e. GA, PSO and ACO. Studied results confirm that the proposed IACO is much superior to other conventional methods in terms of high-quality solution, stable convergence characteristic, and good computation efficiency.

REFERENCES

[1] Kuo W, Prasad VR. "An annotated overview of system-reliability optimization." *IEEE Transactions on Reliability*, vol. 49 (2), pp. 176-187, 2000.
[2] Chen TC, You PS. "Immune algorithm based approach for redundant reliability problems." *Computers in Industry*, vol. 56, pp. 195-205, 2005.
[3] Coit DW, Smith AE. "Reliability optimization of series-parallel systems using a genetic algorithm." *IEEE Transactions on Reliability* vol. 45, pp. 254-260, 1996.
[4] Liang YC, Chen YC. "Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm." *Reliability Engineering and System Safety* vol. 92, pp.323-331, 2007.
[5] Hsieh YC. "A linear approximation for redundant reliability problems with multiple component choices." *Computers and Industrial Engineering* vol. 44, pp. 91-103, 2003.
[6] Onishi J, Kimura S, James RJW, Nakagawa Y. "Solving the redundancy allocation problem with a mix of components using the improved surrogate constraint method." *IEEE Transactions on Reliability* vol. 56(1), pp.94-101, 2007.
[7] Colorni A, Dorigo M, Maniezzo V. "Distributed optimization by ant colonies." *In: Proceedings of the European conference on artificial life*; pp. 134-142, 1991.
[8] Dorigo M, Maniezzo V, Colorni A. "Ant system: optimization by a colony of cooperative agents." *IEEE Trans Syst Man Cybernet B: Cybernet* vol. 26(1), pp.29-41, 1996.
[9] Dorigo M, Gambardella LM. "Ant colonies for the traveling salesman problem. Bio systems, vol. 43, pp. 73-81, 1997.
[10] Dorigo M, Gambardella LM. "Ant colony system: a cooperative learning approach to the traveling salesman problem." *IEEE Trans Evol Comput* vol. 1(1), pp.53-66, 1997.
[11] Gambardella LM, Taillard E, Dorigo M. "Ant colonies for the quadratic assignment problem." *J Operat Res Soc* vol. 50, pp. 167-176, 1999.
[12] Bell J, McMullen P. "Ant colony optimization techniques for the vehicle routing problem." *Adv Eng Inform* vol. 18, pp. 41-48, 2004.
[13] Wei-Chang Yeh, Tsung-Jung Hsieh, "Solving reliability redundancy allocation problems using an artificial bee colony algorithm," *Computer & Operations Research*, vol.38, pp. 1465-1473, 2011.
[14] Kanyapat W., Paramote W, "Reliability optimization of topology communication network design using an improved ant colony optimization," *Computer and electrical engineering*, vol. 35, pp.730-747, 2009.
[15] Manju Agarwal, Vikas K. Sharma, "Ant colony approach to constrained redundancy optimization in binary systems," *Applied Mathematical Modeling*, vol. 34, pp. 992-1003, 2010.
[16] W. Kuo, V.R. Prasad, F.A. Tillman, C.L. Hwang, "Optimal Reliability Design-fundamentals and Applications," *Cambridge Press*, Cambridge, 2001.