J. Ind. Eng. Int., 6 (11), 6-16, Spring 2010
ISSN: 1735-5702

A redundancy allocation problem with the choice of redundancy strategies by a memetic algorithm

J. Safari[1]; R. Tavakkoli-Moghaddam[2]

[1] *Assistant Prof., Dep. of Industrial Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran*

[2] *Professor, Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*

Received: 11 June 2008; Revised: 14 August 2008; Accepted: 2 September 2008

Abstract: This paper proposes an efficient algorithm based on memetic algorithm (MA) for a redundancy allocation problem without component mixing (RAPCM) in a series-parallel system when the redundancy strategy can be chosen for individual subsystems. Majority of the solution methods for the general RAPCM assume that the type of a redundancy strategy for each subsystem is pre-determined and known a priori. In general, active redundancy has traditionally received greater attention; however, in practice both active and cold-standby redundancies may be used within a particular system design. The choice of the redundancy strategy then becomes an additional decision variable. Thus, the problem is to select the best redundancy strategy, component and redundancy level for each subsystem in order to maximize the system reliability under system-level constraints. Due to its complexity and NP-hardness, it is so difficult to optimally solve such a problem by using traditional optimization tools. To validate the performance of the proposed MA in terms of solution quality, a number of test problems are examined and the robustness of this algorithm is then discussed. Finally, the related results are reported and it is shown that the proposed MA performs well.

Keywords: *Memetic algorithm; Redundancy strategies; Redundancy allocation problem; Series-parallel systems*

1. Introduction

The main goal of reliability engineering is to improve system reliability. In the initial design activity, redundancy allocation is a direct way of enhancing the system reliability. The redundancy allocation problem (RAP) involves the simultaneous selection of components and a system-level design configuration that can collectively meet all design constraints in order to optimize some objective functions, such as system cost and/or reliability (Coit and Smith, 1995). In this problem, there are several types of different components with different levels of cost, reliability, weight, and other characteristics.

The RAP is first modeled by Fyffe *et al.* (1968) who solved this problem with a dynamic programming algorithm. In general, the RAP is known to be an NP-hard problem (Chern, 1992) and solved by using a number of optimization approaches for different formulations, as summarized in Kuo *et al.* (2001) and Gen and Yun (2006). The problem can be classified into two main groups.

1) RAP without component mixing (RAPCM): Those problems where a mix of components within a subsystem is not allowed; that is, a single component type is available for redundancy.
2) RAP with a mix of components (RAPMC): Those problems where a mix of components is allowed within a subsystem; that is, the numerous component types are available for redundancy in parallel.

The RAPMC is first solved by Coit and Smith (1995, 1996). They used a genetic algorithm (GA) to solve this problem in a series-parallel system with *k*-out-of-*n*:*G* subsystems. Coit and Smith (1996) solved the RAPMC with a hybrid algorithm. This hybrid algorithm is constructed by a combination of GA and neural network approaches. Hsieh (2002) developed a linear programming approach to approximate the integer nonlinear RAPMC. Ramirez-Marquez *et al.* (2004) reformulated the objective of this problem maximizing the minimum subsystem reliability, and then solved using integer programming. Tavakkoli-Moghaddam and Safari (2007) represented a new mathematical model for the RAPMC. There are a number of methods for solving of the RAPMC, such as tabu search (Kulturel-Konak *et al.*, 2003), ant colony optimization (Liang and Smith, 2004), simulated annealing (Kim *et al.*, 2004), immune algorithm (Chen and You, 2005), heuristic method (You and Chen, 2005), variable neighborhood descent algorithm (Liang and Wu, 2005), variable neighborhood search (Liang and Chen, 2007), hybrid algorithm (Nahas *et al.*, 2007), and an exact

*Corresponding Author Email: jalalsafari@yahoo.com
Tel.: +98 912 370 5300

method based on the improved surrogate constraint method (Onishi *et al.*, 2007).

In general, the RAPCM has been formulated by considering active redundancy. Coit (2001) presented a problem formulation and solution methodology for the this problem when the redundancy strategy of all subsystems is cold-standby. Choosing a particular redundancy strategy for each subsystem is much more realistic and provides a better tool for the designers.

Coit and Liu (2000) presented a new problem formulation and solution method for the RAPCM when a system design includes multiple subsystems designed with either active or cold-standby redundancy.

This solution method assumes that the redundancy strategy (i.e., active or cold-standby) for each subsystem is pre-determined. However, the choice of redundancy strategy (i.e., active or cold-standby) for each subsystem is much more realistic and provides a better tool for the designers. Unfortunately, the RAPCM is not considered sufficiently when the redundancy strategy can be chosen for individual subsystems. Coit (2003) presented an optimal solution for the RAPCM when there are some subsystems using active redundancy, cold-standby redundancy, or selecting the best redundancy strategy. Tavakkoli-Moghaddam *et al.* (2008) applied a GA to solve this problem.

This paper presents an efficient algorithm based on memetic algorithm (MA) for solving of the RAPCM when the redundancy strategy can be chosen for individual subsystems. Thus, the problem is to select the best redundancy strategy, component, and redundancy level for each subsystem in order to maximize the system reliability under system-level constraints.

In general, the RAPCM has been considered for various system structures, such as series, parallel, network, and parallel-series. In this paper, we consider the series-parallel system that is a common system structure used in most system designs.

Chern (1992) showed that even a simple RAPCM in series systems with linear constraints is an NP-hard problem, prompting researchers to develop meta-heuristic methods to achieve near-optimal solutions of acceptable quality in reasonable computational time. Meta-heuristic methods are very efficient in solving such complex discrete optimization problems. These methods provide more flexibility and require fewer assumptions regarding the objective function and associated constraints.

The paper is organized as follows. Sections 2 and 3 present a review of the redundancy strategies and formulate the problem, respectively. Section 4 proposes a meta-heuristic method based on MA for solving the RAPCM when either active or cold-standby redundancy can be selected for individual subsystems. A numerical example and the computational results are reported in Section 5 to demonstrate the efficiency of this proposed methodology, and the conclusions are presented in Section 6.

2. Redundancy strategies

There are two types of redundancy strategies, namely active and standby. If all redundant components operate simultaneously from time zero, even though the system needs only one at any given time, such an arrangement is called active redundancy. There are three variants of the standby redundancy, namely cold, warm, and hot. In the cold standby redundancy, the component does not fail before it operates. In the warm standby redundancy, the component is more prone to failure before operation than the cold standby components. In the hot standby redundancy, the failure pattern of component does not depend on whether the component is idle or in operation.

The mathematical models for hot standby and active redundancy arrangements are the same. In the standby redundancy arrangement, the redundant components are sequentially used in the system at component failure times. Each redundant component in the standby arrangement can operate only when it is switched on. When the component in operation fails, one of the redundant units is switched on to continue the system operation (Ebling, 1997).

In the standby redundancy, there are two scenarios, namely Case 1 and Case 2, in first detecting failure and then switching to good components. For Case 1, the failure detection and switching hardware or software continually monitors the system performance. When it detects a failure it activates a redundant component. For Case 2, a failure is only possible when a switch is required. At any time the switch is required, there is a constant probability (ρ_i) that the switching will be successful (Coit and Liu, 2000).

This paper considers redundancy strategies consisting of only active (i.e., hot) and cold-standby redundancy. The approach used in this paper categorizes all subsystems into four sets according to the following definitions:

N Set of all subsystems with no redundancy,

A Set of all subsystems with active redundancy,

S Set of all subsystems with cold-standby redundancy,

$A\&S$ Set of all subsystems with active or cold-standby redundancy.

3. Problem formulation

Following is the mathematical model of the RAPCM for the series-parallel system when the redundancy strategy can be chosen for individual subsystems with s subsystems and two separable linear constraints. The integer nonlinear programming problem is presented as follows (Coit, 2003)0.

3.1. Assumptions

The main assumptions of the presented model are given bellow:

- The states of components and the related system have either good or bad options,
- The component attributes (i.e., reliability, cost and weight) are known and deterministic,
- Two redundancy strategies (i.e., active redundancy cold standby) are considered,
- There is no component repair or preventive maintenance,
- Failures of components are independent events,
- Failed components do not damage the system,
- The components within the same subsystem are the same type.

3.2. Notations

S Number of subsystems,

n_i Number of components used in subsystem i, ($i = 1, 2, \ldots, s$),

N Set of n_i , ($n_1, n_2 , ..., n_s$),

$n_{\max, i}$ Upper bound for n_i , ($n_i \le n_{\max, i} \quad \forall i$) ,

m_i Number of available component choices for a subsystem i, ($i = 1, 2, \ldots, s$),

z_i Index of component choice used for a subsystem i, ($z_i \in \{1, 2, \ldots, m_i\}$),

z Set of z_i , ($z_1, z_2 , ..., z_s$),

t Mission time,

$R(t; z, n)$ System reliability at time t for designing vectors z and n,

$r_{ij}(t)$ Reliability at time t for the j-th available component for subsystem i,

λ_{ij} , k_{ij} Scale and shape parameters for the Gamma distribution $f_{ij}(t) = \dfrac{\lambda_{ij}^{k_{ij}} \; t^{k_{ij}-1} \; e^{-\lambda_{ij} t}}{\Gamma(k_{ij})}$,

C, W System-level constraint limits for cost and weight,

c_{ij} , w_{ij} Cost and weight for the j-th available component for subsystem i,

$\rho_i(t)$ Failure-detection/switching reliability at time t (Case 1: Continuous detector / switch operation),

ρ_i Failure-detection/switching reliability at time t (Case 2: Switch active only in response to a failure).

3.3. Mathematical model

$$\text{Max} \quad R(t; z, n) \tag{1}$$

Subject to:

$$\sum_{i=1}^{s} c_{iz_i} n_i \le C \qquad n_i \in \{1, 2, ..., n_{\max,i}\} \tag{2}$$

$$\sum_{i=1}^{s} w_{iz_i} n_i \le W \qquad z_i \in \{1, 2, ..., m_i\} \tag{3}$$

The objective function (1) determines the redundancy strategy, component type, and the quantity of components in each subsystem to achieve the maximum system reliability. Constraints (2) and (3) consider the available cost and weight, respectively. To calculate $R(t; z, n)$, Eqs. (4) and (5) are presented for the system reliability in two cases as follows:

Case 1: Continuous detector/switch operation:

$$R(t;z,n)=\prod_{i\in A}\left(1-\left(1-r_{iz_i}(t)\right)^{n_i}\right)$$
$$\times\prod_{i\in S}\left(r_{iz_i}(t)+\sum_{j=1}^{n_i-1}\int_0^t\rho_i(u)f_{iz_i}^{(j)}(u)r_{iz_i}(t-u)du\right)$$
$$\times\prod_{i\in N}r_{iz_i}(t) \quad (4)$$

Case 2: Switch active only in response to a failure:

$$R(t;z,n)=\prod_{i\in A}\left(1-\left(1-r_{iz_i}(t)\right)^{n_i}\right) \quad (5)$$
$$\times\prod_{i\in S}\left(r_{iz_i}(t)+\sum_{j=1}^{n_i-1}\rho_i^j\int_0^t f_{iz_i}^{(j)}(u)r_{iz_i}(t-u)du\right)\times\prod_{i\in N}r_{iz_i}(t)$$

The exact techniques for reliability optimization problems are not necessarily desirable because it is very hard to obtain the exact solution. Because of the difficulties of applying exact techniques, a major focus of this paper is to attempt reliability optimization using the proposed MA. This algorithm can be considered as a very practical tool to solve such complex problems successfully.

4. Memetic algorithms

In the field of evolutionary computation, it is well known that the effectiveness of evolutionary algorithms can be enhanced by incorporating problem-dependant local search heuristics. These approaches are often called memetic algorithms, which are similar to genetic algorithms but usually incorporate neighborhood search techniques such as local search, tabu search, or simulated annealing (Moscato, 1989). However, the performance of the MA highly depends on the structure of the problem under consideration. In this paper, we propose a MA to solve the RAPCM when the redundancy strategy can be chosen for individual subsystems as described in the following subsections.

4.1. Solution encoding

Each possible solution (i.e., phenotype) to this problem is a collection of redundancy strategies, selected components, and n_i parts in parallel ($n_i \le n_{\max,i}$) for each subsystem. n_i parts can be chosen only in one combination amongst the m_i available components. The solution encoding is a $3\times s$ matrix. The first, second and third rows represent the redundancy strategy, type of selected components and the number of selected components, respectively.

The subsystem representations are then placed into adjacent columns to complete the matrix representation. Fig. 1 shows an example of encoding solution with s=14. This matrix represents a prospective solution with two of the first component used in parallel with active redundancy for the first subsystem; two of the second component used in parallel with cold standby redundancy for the second subsystem, and the like.

4.2. Initial population

The initial population is determined by two methods. The one is the new heuristic method, which is described as below, and another is the random method.

In this algorithm, the population size of the initial population is 8. One of the initial populations is found by the heuristic method and the others are found by a random method.

4.2.1. Heuristic method

Step 0: Find the component type in each subsystem that is the maximum reliability at time t. For example, we have:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
{2	2	4	3	2	4	3	2	2	3	3	1	1	3}

		Subsystem													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Redundancy strategy	1	A	S	N	A	S	S	N	N	N	S	S	A	A	S
Type of component selected	2	1	2	4	3	4	2	1	3	2	1	3	1	2	4
Number of component	3	2	2	1	5	3	2	1	1	1	4	2	3	1	2

Figure 1: Encoding solution.

Step 1: Allocate one type of component, which is found in Step 0 in each subsystem.

Step 2: Report the solution and stop.

4.3. Fitness function

The fitness function is defined as the summation of the objective and a dynamic penalty function determined by the relative degree of infeasibility. To provide an efficient search through the infeasible region but to ensure that the final best solution is feasible, the modified dynamic penalty function proposed by Coit and Smith (1996) is adopted.

4.4. Crossover breeding operator

Parents are selected at random, and then the proposed crossover breeding operator is applied with crossover probability (or rate) of 0.55. In this paper, we find this rate to be best according to our experimental results. One offspring is produced by a uniform crossover breeding operator. This operator first generates a random crossover mask, and then exchanges the respective subsystems between parents according to the mask. A crossover mask is simply a binary $s\times1$ matrix (Gen and Cheng, 1997). Another offspring is produced by the modified uniform crossover operator. For this operator, the crossover mask generates as follows:

1. Subsystems with the lowest and highest reliability among the candidate solutions are determined.
2. All bits in the crossover mask for these two subsystems consist of zeros and ones.

4.5. General mutation operator

The mutation operator performs random perturbations to the selected solutions. Mutation results maintain diversity of solution. This procedure avoids premature convergence to local optimum and facilitates jumps in the solution space. In this algorithm, as depicted in Fig. 2, first a random matrix in size of solution matrix is created.

Each value within the solution matrix is altered at random if the related value in the mutation matrix is smaller than the mutation rate. In this algorithm, the mutation rate is 0.33. This rate also works well for most MA solutions.

4.6. Max-Min mutation operator

Because the considered system structure is series-parallel, then the reliability of system is calculated by product of all subsystems. In each system, the subsystem with min of reliability within all subsystems has a bad effect in reliability of system. Max-Min mutation operator is trying to bring to closer the reliability of subsystems. For each candidate solution, this operator selects subsystems with the highest reliability and lowest reliability among the all subsystems and then randomly mutates the number of allocated components in each selected subsystem. Fig. 3 shows a typical max-min mutation used in the proposed MA.

4.7. Local search for the RAPCM

Local search (LS) algorithms are the basal component in MA. LS algorithms are improvement heuristics that search in the neighborhood of the current solution for a better one. If a better solution is found, the current solution is replaced and the neighborhood search is started again. If no further improvement can be made, a local optimum is found (i.e., there is no better solution in the neighborhood of the current solution) (Merz and Katayamab, 2004).

To avoid the local maximum quickly, the LS designed for the proposed MA is applied in each generation. After each generation, each solution can be classified as follow:

1.Feasible solution

a. Additional component can be added to any subsystem because the remaining resources are available for any component.

b. No additional component can be added to any subsystem because the remaining resources are unavailable for any component.

2.Infeasible Solution: No additional component can be added to any subsystem because the remaining resources are unavailable for any component.

To extend the search space so as to find better quality solutions, one of the three following LSs is employed for each solution, which is found in each generation.

4.7.1. Method 1

If the solution is feasible and additional component can be added to any subsystem the method 1 is applied.

Step 0: Find the subsystem with the minimum reliability at time *t* in all subsystems.

Step 1: Add one component in subsystem found in Step 0, if and only if the solution quality can be improved and the constraint is satisfied; else, go to Step 2.

Step 2: Report the solution and stop.

4.7.2. Method 2

If the solution is feasible and no additional component can be added to any subsystem, Method 2 is applied.

Step 0: Find the subsystem with the maximum reliability at time *t* in all subsystems.

Step 1: Find the subsystem with the minimum reliability at time *t* in all subsystems.

Step 2: Remove one component in subsystem found in Step 0 and add one component in subsystem found in Step 1, if and only if the solution quality can be improved and the constraint is satisfied; else, go to Step 3.

Step 3: Report the solution and stop.

4.7.3. Method 3

If the solution is infeasible the method 3 is applied.

Step 0: Find the subsystem with the maximum reliability at time *t* in all subsystems.

Step 1: Remove one component in subsystem found in Step 0 if and only if the solution quality can be improved and the constraint is satisfied; else, go to Step 2.

Step 2: Report the solution and stop.

4.8. Selection

After operator breeding, the *p* best solutions among the previous generation and the new child are retained to form the next generation.

4.9. Stopping condition

The proposed MA is terminated after a pre-selected number of generations. A reasonable number of generations is 30.

5. A numerical example

To evaluate the performance of the proposed MA, a typical example taken from Fyffe *et al.* (1968)0 is first solved. A series–parallel system is connected by 14 parallel subsystems, and each subsystem has three or four components of choice.

Time to failure distribution for all components is exponential. Component cost, weight, and the exponential distribution parameters are given in Table 1. The objective is to maximize the system reliability at t=100 hours, given the constraints for the system cost (C=130 max) and the system weight (W=170 max).

For each subsystem, active or cold-standby redundancy can be used, and the reliability of a switch at 100 hours is 0.99 for all subsystems. The maximum number of the allocated components is 6 (i.e., $\sum_{j=1}^{4} n_{ij} \le 6$) within any subsystem.

Because of the stochastic nature of the proposed MA, four experiments are performed for 30-generation each and the best feasible solution amongst the four is selected and reported as the final and best solution. The maximum reliability identified by the proposed MA is shown in Table 2. In this table, the proposed MA is compared with optimal solution.

To compare the performance of the proposed MA, 33 test problems are solved. These problems are a modified version of the problems given in Nakagawa and Miyazaki (1981). The data sets of these problems are shown in Table 1 and various weights of the available resource from 159 to 191 are considered.

The computational results of this proposed algorithm is shown in Table 3. As shown in this table, the computational result of the proposed MA is better than the GA represented by Tavakkoli-Moghaddam *et al.* (2008). The standard deviation of the fitness function is an important measure of the algorithm robustness.

In Fig. 4 for each problem, the standard deviation of fitness function of the proposed method is very low. This implies that the proposed approach is robust and credible. The standard deviation of the fitness function is an important measure of the algorithm robustness.

Table 1: Parameter setting for the given problem.

i	Choice 1 (j = 1)				Choice 2 (j = 2)				Choice 3 (j = 3)				Choice 4 (j = 4)			
	λ_{ij}	k_{ij}	c_{ij}	w_j	λ_{ij}	k_{ij}	c_{ij}	w_j	λ_{ij}	k_{ij}	c_{ij}	w_j	λ_{ij}	k_{ij}	c_{ij}	w_j
1	0.00532	2	1	3	0.000726	1	1	4	0.00499	2	2	2	0.00818	3	2	5
2	0.00818	3	2	8	0.000619	1	1	10	0.00431	2	1	9	-	-	-	-
3	0.0133	3	2	7	0.0110	3	3	5	0.0124	3	1	6	0.00466	2	4	4
4	0.00741	2	3	5	0.0124	3	4	6	0.00683	2	5	4	-	-	-	-
5	0.00619	1	2	4	0.00431	2	2	3	0.00818	3	3	5	-	-	-	-
6	0.00436	3	3	5	0.00567	3	3	4	0.00268	2	2	5	0.000408	1	2	4
7	0.0105	3	4	7	0.00466	2	4	8	0.00394	2	5	9	-	-	-	-
8	0.0150	3	3	4	0.00105	1	5	7	0.0105	3	6	6	-	-	-	-
9	0.00268	2	2	8	0.000101	1	3	9	0.000408	1	4	7	0.000943	1	3	8
10	0.0141	3	4	6	0.00683	2	4	5	0.00105	1	5	6	-	-	-	-
11	0.00394	2	3	5	0.00355	2	4	6	0.00314	2	5	6	-	-	-	-
12	0.00236	1	2	4	0.00769	2	3	5	0.0133	3	4	6	0.0110	3	5	7
13	0.00215	2	2	5	0.00436	3	3	5	0.00665	3	2	6	-	-	-	-
14	0.0110	3	4	6	0.00834	1	4	7	0.00355	2	5	6	0.00436	3	6	9

Table 2: Comparison between the MA and optimal solutions.

		Optimal solution			Proposed algorithm	
i	z_i	n_i	Redundancy	z_i	n_i	Redundancy
1	3	4	Active	1	3	Active
2	1	2	Cold-standby	1	2	Active
3	4	3	Active	4	2	Cold-standby
4	3	3	Cold-standby	2	2	Cold-standby
5	2	3	Active	3	2	Active
6	2	2	Cold-standby	4	2	Cold-standby
7	1	2	Cold-standby	3	2	Active
8	3	2	Cold-standby	1	3	Cold-standby
9	1	2	Cold-standby	3	2	Active
10	2	3	Cold-standby	2	3	Active
11	3	2	Cold-standby	3	2	Cold-standby
12	4	2	Cold-standby	4	2	Cold-standby
13	2	2	Active	1	2	Active
14	3	2	Cold-standby	3	2	Active
Computational time			715 (Sec.)			526 (Sec.)
System reliability			0.9863			0.9719
Resources consumed		Weight	170			170
		Cost	123			106

Table 3: Results of the proposed MA and its comparison with the GA (Tavakkoli-Moghaddam *et al.*, 2008).

Problem	Weight Constraint	GA (Tavakkoli-Moghaddam et al., 2008)		Proposed MA								Note
		Best feasible solution	Standard deviation	Trial 1	Trial 2	Trial 3	Trial 4	Best feasible solution			Standard deviation	
				Reliability	Reliability	Reliability	Reliability	Reliability	Cost	Weight		
1	159	0.9641	0.0150	0.9439	0.963	0.9564	0.9691	0.9691	109	159	0.0108	☒
2	160	0.9629	0.0211	0.9688	0.9592	0.9604	0.9506	0.9688	102	160	0.0074	☒
3	161	0.9636	0.0290	0.9583	0.9464	0.9663	0.9602	0.9663	105	160	0.0083	☒
4	162	0.9564	0.0197	0.9643	0.941	0.9513	0.9609	0.9643	100	164	0.0105	☒
5	163	0.9675	0.0180	0.9642	0.9575	0.9698	0.9483	0.9698	107	163	0.0093	☒
6	164	0.9619	0.0208	0.9606	0.9678	0.9538	0.948	0.9678	113	165	0.0086	☒
7	165	0.9454	0.0125	0.9564	0.9632	0.9522	0.9614	0.9632	105	164	0.0050	☒
8	166	0.9647	0.0180	0.9567	0.9613	0.967	0.9574	0.967	96	165	0.0047	☒
9	167	0.9614	0.0273	0.956	0.9604	0.9592	0.9639	0.9639	103	167	0.0033	☒
10	168	0.9669	0.0270	0.956	0.9556	0.9621	0.9687	0.9687	99	168	0.0062	☒
11	169	0.9602	0.0137	0.9695	0.9491	0.9588	0.9591	0.9591	98	169	0.0083	
12	170	0.9705	0.0150	0.9616	0.9537	0.9691	0.9719	0.9719	106	170	0.0082	☒
13	171	0.9639	0.0047	0.9609	0.9562	0.9647	0.9581	0.9647	109	171	0.0037	☒
14	172	0.9608	0.0048	0.9685	0.9525	0.9661	0.9521	0.9685	111	172	0.0087	☒
15	173	0.9717	0.0210	0.9588	0.9514	0.9767	0.963	0.9767	96	172	0.0106	☒
16	174	0.9707	0.0136	0.9676	0.9586	0.9603	0.9549	0.9676	107	172	0.0053	
17	175	0.9681	0.0120	0.9693	0.9576	0.97	0.9689	0.97	109	174	0.0059	☒
18	176	0.9703	0.0083	0.9605	0.9629	0.9661	0.9708	0.9708	117	176	0.0045	☒
19	177	0.9738	0.0144	0.9758	0.9805	0.9617	0.9723	0.9758	122	176	0.0080	
20	178	0.9734	0.0117	0.9615	0.9661	0.9497	0.9713	0.9661	113	177	0.0092	
21	179	0.9756	0.0046	0.9839	0.951	0.976	0.9795	0.9839	115	178	0.0148	☒
22	180	0.9834	0.0144	0.9792	0.9839	0.976	0.9749	0.9839	120	179	0.0040	☒
23	181	0.9741	0.0086	0.9708	0.9831	0.9775	0.9774	0.9831	113	179	0.0050	☒
24	182	0.9797	0.0065	0.9841	0.9643	0.9823	0.9809	0.9841	128	181	0.0092	☒
25	183	0.9737	0.0082	0.9756	0.9823	0.9767	0.9818	0.9823	117	182	0.0034	☒
26	184	0.9804	0.0246	0.9807	0.9847	0.978	0.9866	0.9866	184	130	0.0039	☒
27	185	0.9844	0.0136	0.9841	0.9714	0.9721	0.9772	0.9841	117	185	0.0059	
28	186	0.9821	0.0080	0.9839	0.9696	0.9797	0.9684	0.9839	114	185	0.0076	☒
29	187	0.9815	0.0098	0.9751	0.9816	0.9853	0.9849	0.9853	129	187	0.0047	☒
30	188	0.9771	0.0163	0.9874	0.9772	0.9845	0.9716	0.9874	122	188	0.0071	☒
31	189	0.9861	0.0067	0.9812	0.9847	0.9828	0.9721	0.9847	123	188	0.0056	
32	190	0.9863	0.0092	0.987	0.9804	0.9862	0.9587	0.987	126	188	0.0132	☒
33	191	0.9856	0.0107	0.9727	0.9836	0.978	0.9865	0.9865	127	190	0.0061	☒

Note: ☒ represents the best solution found by the proposed MA is superior to the best solution found by the GA (Tavakkoli-Moghaddam , 2008).

Figure 2: Example of the general mutation.

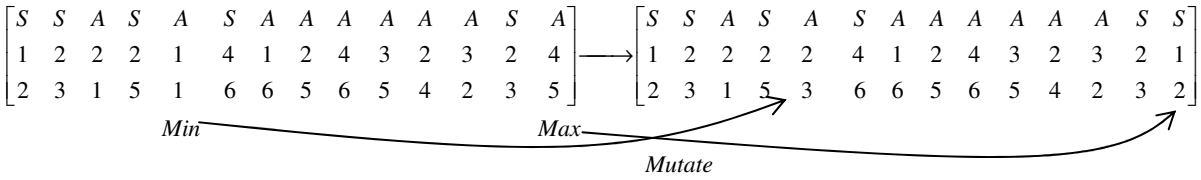

Figure 3: Example of the Max-Min mutation.

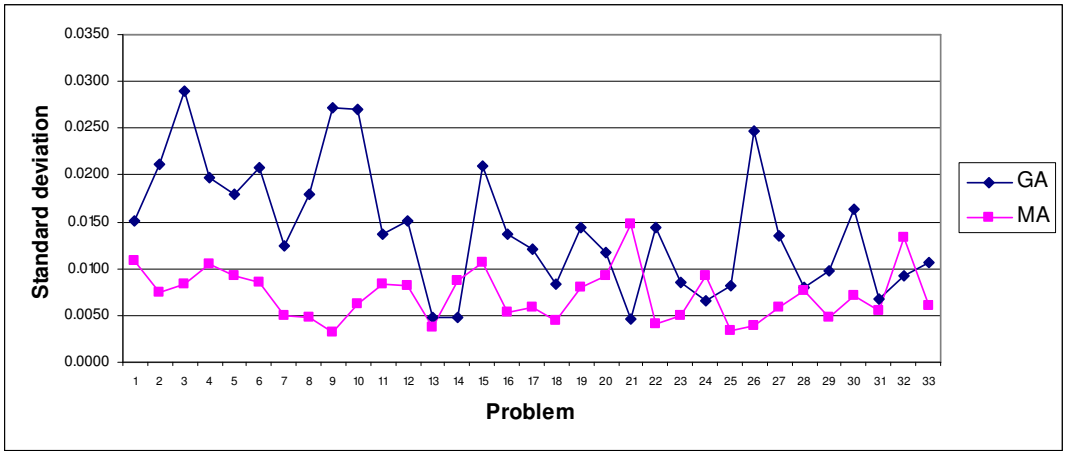

Figure 4: Standard deviation of the fitness functions.

In Fig. 5 for each problem, the standard deviation of fitness function of the proposed method is very low. This implies that the proposed approach is robust and credible. Because of the computational results and the standard deviation of fitness function of the proposed MA is better than GA, then the proposed MA is better than GA, which is represented by Tavakkoli-Moghaddam *et al.* (2008).

6. Conclusion

In this paper, we have proposed a memetic algorithm (MA) to solve a novel, mathematical model of a redundancy allocation problem without component mixing (RAPCM) for a series-parallel system, where either active or cold-standby redundancy can be selected for individual subsystems. The choice of redundancy strategies for each subsystem is much more realistic and provides a better tool for the designers.

In general, this problem is not easy to solve in real cases, especially for large sizes. This is the motivation of using meta-heuristic methods for solving such a hard and complex problem. We have thus proposed the MA for solving this problem, which are more flexible in the sense that the practitioners are not limited to a single solution. Finally, it is showed that the computational results of our proposed MA are near-optimal solutions and the related result of this algorithm is better than the genetic algorithm (GA). All future researches can be classified in two main groups as follows:

1. Develop the new mathematical model. As shows in Section 3.1, in the presented mathematical model, there are a lot of assumptions. We can develop this mathematical model by deleting these assumptions.

2. Use another solution method. There are a lot of meta heuristics that can be useful for solution, such as tabu search, ant colony system, simulated annealing, immune algorithm, heuristic method, variable neighborhood descent algorithm, variable neighborhood search, and hybrid algorithms.

References

Chen, T.; You, P., (2005), Immune algorithms-based approach for redundant reliability problems with multiple component choices. *Computers in Industry*, 56(2), 95-205.

Chern, M. S., (1992), On The computational complexity of reliability redundancy allocation in a series system. *Operations Research Letters*, 11(5), 309-315.

Coit, D. W., (2001), Cold standby redundancy optimization for nonrepairable systems. *IIE Transactions*, 33(6), 471-478.

Coit, D. W., (2003), Maximization of system reliability with a choice of redundancy strategies. *IIE Transactions*, 35(6), 535-544.

Coit, D. W.; Liu, J., (2000), System reliability optimization with k-out-of-n subsystems. *International Journal of Reliability, Quality & Safety Engineering*, 7(2), 129-143.

Coit, D. W.; Smith, A., (1995), *Optimization approaches to the redundancy allocation to the redundancy allocation problem for series-parallel systems.* Proceedings of the Fourth Industrial Engineering Research Conference, Nashville TN, 342-349.

Coit, D. W.; Smith, A., (1996), Penalty guided genetic search for reliability design optimization. *Computers & Industrial Engineering*, 30(4), 895-904.

Coit, D. W.; Smith, A., (1996), Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, 45(2), 254-260.

Coit, D. W.; Smith, A., (1996), Solving the redundancy allocation problem using a combined neural network / genetic algorithm approach. *Computers & Operations Research*, 23(6), 515-526.

Ebling, C. E., (1997), *An introduction to reliability maintainability engineering*. McGraw-Hill, New York, USA.

Fyffe, D. E.; Hines, W. W.; Lee, N. K., (1968), System reliability allocation and a computational algorithm. *IEEE Transactions on Reliability*, 17 64-69.

Gen, M.; Cheng, R., (1997), *Genetic algorithm and engineering design*. John Wiley & Sons, Inc, New York, USA.

Gen, M.; Yun, Y. S., (2006), Soft computing approach for reliability optimization: State-of-the-art survey. *Reliability Engineering and System Safety*, 91(9), 1008-1026.

Kim, H.; Bae, C.; Park, S., (2004), Simulated annealing algorithm for redundancy optimization with multiple component choices. In: *Advanced Reliability Modeling*, Proceedings of the 2004 Asian International Workshop, World Scientific, 237-244.

Kulturel-Konak, S.; Smith, A.; Coit, D. W., (2003), Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions*, 35(6), 515-526.

Kuo, W.; Prasad, V. R.; Tillman, F. A.; Hawang, C., (2001), *Optimal reliability design fundamental and application*. Cambridge University Press, Inc, London, UK.

Hsieh, Y. C., (2003), A Linear approximation for redundant reliability problems with multiple component choices. *Computers and Industrial Engineering*, 44(1), 91-103.

Liang, Y. C.; Chen, Y. C., (2007), Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm. *Reliability Engineering and System Safety*, 92(3), 323-331.

Liang, Y.-C.; Smith, A., (2004), An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Trans Reliability*, 53(3), 417-423.

Liang, Y.-C.; Wu, C. C., (2005), A Variable neighborhood descent algorithm for the redundancy allocation problem. *Industrial Engineering & Management System*, 4(1), 109-116.

Merz, P.; Katayamab, K., (2004), Memetic algorithms for the unconstrained binary quadratic programming problem. *BioSystems,* 78(1), 99-118.

Moscato, P., (1989), *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.* Technical Report No. 790, Caltech Concurrent Computation Program, California Institute of Technology.

Nakagawa, Y.; Miyazaki, S., (1981), Surrogate constraints algorithm for reliability optimization problems with two constraints. *IEEE Trans Reliability*, 30(2), 175-180.

Nahas, N.; Nourelfath, M.; Ait-Kadi, D., (2007), Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series-parallel systems. *Reliability Engineering and System Safety*, 92(2), 211-222.

Onishi, J.; Kimura, S.; James, R. T. W.; Nakagawa, Y., (2007), Solving the redundancy allocation problem with a mix of components using the improved surrogate constraint method. *IEEE Transactions on Reliability*, 56(1), 94-101.

Ramirez-Marquez, J.; Coit, D. W.; Konak, A., (2004), Redundancy allocation for series-parallel systems using a max–min approach. *IIE Trans*, 36(9), 891-898.

Tavakkoli-Moghaddam, R.; Safari, J., (2007), A New mathematical model for a redundancy allocation problem with mixing components

redundant and choice of redundancy strategies. *Applied Mathematical Sciences*, 45(1), 2221-2230.

Tavakkoli-Moghaddam, R.; Safari, J.; Sassani, F., (2008), Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm. *Reliability Engineering and System Safety*, 93(4), 550-556.

You, P. -S.; Chen, T. C., (2005), An efficient heuristic for series–parallel redundant reliability problems. *Computers & Operations Research,* 32(8), 2117-2127.