Scientia Iranica E (2017) 24(3), 1547–1560

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
www.scientiairanica.com

Solving the redundancy allocation problem of k-out-of-n with non-exponential repairable components using optimization via simulation approach

P. Azimi*, M. Hemmati and A. Chambari

Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran.

Received 26 May 2015; received in revised form 30 September 2015; accepted 5 September 2016

KEYWORDS
Redundancy allocation problem; k-out-of-n systems; Meta-heuristic algorithms; Simulation methods; Enterprise Dynamic (ED) software.

Abstract. In this article, a new model and a novel solving method are provided to address the non-exponential redundancy allocation problem in series-parallel k-out-of-n systems with repairable components based on Optimization Via Simulation (OVS) technique. Despite the previous studies, in this model, the failure and repair times of each component were considered to have non-negative exponential distributions. This assumption makes the model closer to the reality where the majority of used components have greater chance to face a breakdown in comparison to new ones. The main objective of this research is the optimization of Mean Time to the First Failure (MTTFF) of the system via allocating the best redundant components to each subsystem. Since this objective function of the problem could not be explicitly mentioned, the simulation technique was applied to model the problem, and different experimental designs were produced using DOE methods. To solve the problem, some meta-Heuristic Algorithms were integrated with the simulation method. Several experiments were carried out to test the proposed approach; as a result, the proposed approach is much more real than previous models, and the near optimum solutions are also promising.

1. Introduction

The increase in system reliability has been one of the most appealing areas for the engineers and designers, and the utilization of redundant components is one of the common approaches in the development of the systems. Each system is formed by putting different components together wherein their way of relation with each other depends on their function in the system. Different kinds of the system structures could be systems with parallel, series, series-parallel, parallel-series components, and bridge network structures [1]. In this paper, series-parallel structures are used; consequently, it is one of those system structures designed by allocating redundant components in parallel to the components of a system with series structure [2]. Reliability is the probability of proper function of system in a certain time interval, and the reliability of the whole system is a combination of reliability of its individual components. Two approaches are proposed for increasing the reliability of system. The first approach is to increase the reliability of system components; the second is to use the redundant components in addition to the main components in parallel [3]. Due to economic and technological limitations, the best and most efficient method of increasing system reliability is the second approach by using the redundant components with the main components. For this reason, this approach is used in this article as well [4]. Redundancy strategies

*. *Corresponding author. Tel./Fax: +982833665275*
E-mail addresses: p.azimi@yahoo.com (P. Azimi); M.hemmati84@yahoo.com (M. Hemmati); amir.chambari@gmail.com (A. Chambari)

are categorized into active and standby strategies. In an active redundancy strategy, it is assumed that all of the redundant components are implemented together from time zero, whereas in the standby strategy, only the components operate. Hence, the redundant components are idle until the active component fails. Thus, whenever a component in operation fails, one of the redundant components should be switched on. In this paper, the active redundancy is considered. After the first article by Fyffe et al. [5], in which they studied the redundancy allocation problem in series-parallel systems, a great number of researchers have tried to develop this knowledge. The investigations of Coelho [6], Zou et al. [7], Ramirez-Marquez and Coit [8], Nahas et al. [9], Safaei et al. [10], Liang and Chen [11], Ha and Kuo [12], Soltani et al. [13], Liang et al. [14], Juang et al. [15], Zhang et al. [16], and Chambari et al. [17] are of those investigations into the redundancy allocation problem in series-parallel systems. For more information about the redundancy allocation problems, readers are referred to a work by Soltani [18].

With a glance at the trend of the studies in redundancy allocation problem, it is clearly seen that this problem has not been addressed in the repairable systems. The lack of investigations into the redundancy allocation problem in series-parallel systems with repairable components and subsystems by Kuo and Wan [19], Guedenko and Ushakov [20], Elegbede and Adjallah [21], and Ying-Shen et al. [22] is compensated after some finished research works in the literature of redundancy allocation problem. Since the structure of the system has a significant effect on system reliability, the reparability or non-reparability of system components or subsystems is another issue affecting the proper function of a system in its operating duration. The system reparability means that it is possible to restart the system by the required repairs of failures. Whenever a system is repairable, "availability" is used instead of reliability. Availability is the percentage of the time during which a repairable system appropriately does the defined tasks [19]. In this paper, reparability components are considered.

For the multi-objective redundancy allocation problem, Chambari et al. [23], for non-repairable components, considered a large series-parallel system with two objectives: maximization of system reliability and minimization of cost. Garg et al. [24] managed to maximize system reliability and minimize cost by Particle Swarm Optimization (PSO) in a series-parallel system. Zoulfaghari et al. [25] formulated bi-objective redundancy allocation problem with two objectives, including maximization of the system availability and minimization of cost of the system considering reparability and non-reparability components. Furthermore, Li and Lin [26] considered three-objective models, whose objectives include reliability, total cost, and total weight.

Since a different word other than reliability is defined as a functional parameter of the system for repairable systems, the other notions regarding the system function are also different in repairable systems. Mean Time To the Failure (MTTF) or survivability in non-repairable systems shows the mean durability of system lifetime, while the components of a repairable system are repairable and the system could restart after repair, and the MTTF is the mean time of system failure for the first time (MTTFF) [27].

The aim of this manuscript is to represent a bi-objective model developed for the series-parallel system to allocate redundant components in systems to repairable components in order to maximize MTTFF of the system and minimize system cost under the constraints of total cost, weigh of the system, and sum of components within the system. The redundancy allocation problem is the non-linear polynomial optimization problem which is of NP-hard class [28]. It has been addressed by different methods and with the extension of solution space, the absolute solutions in solving these problems are inefficient; the meta-heuristic approaches have been preferred instead in recent years. Then, since the proposed model belongs to an NP-hard class of optimization problems, an effective Multi-Population Genetic Algorithm (MPGA) is implemented to solve the model.

The rest of the paper is organized as follows. Section 2 defines multi-objective optimization more precisely and describes the problem definition and basic assumptions in Section 3. Section 4 develops the proposed methods for solving the reliability optimization problem. Section 5 describes computational results and provides analysis of the results for a set of test problems. Finally, Section 6 concludes the paper and all remarks.

2. Multi-objective optimization

Multi-Objective Optimization Problem (MOOP) refers to the problems in which two or more objectives must be optimized simultaneously. Often, such objectives are in conflict with each other and are expressed in different units. Because of their nature, the final solution to MOOP is not a single one but a set of solution known as Pareto-solutions [29,30]. When such solutions are represented in the objective function space, the graph produced is called the Pareto-front or the Pareto-optimal set. A general formulation of a MOOP consists of a number of objectives with a number of inequality and equality constraints. The problem can be mathematically written as $\min\{f_1(x), f_2(x), \cdots, f_v(x)\}$ subject to $g_l(x) \leq 0$; $l = 1, 2, \cdots, L$ and $h_k(x) = 0$; $k = 1, 2, \cdots, K$, where x is vector of decision variables,

$f_v(x)$ is the ith objective function, and $g_l(x)$ and $h_k(x)$ are constraints vectors. In the function set, some of the objectives are often in conflict with others; some have to be minimized, while others are to be maximized. The constraints limit feasible region X, and any point $x \in X$ is categorized as feasible solutions. There is rarely a situation in which all $f_v(x)$ function values have an optimum in X at common point x. Therefore, in the absence of preference information, solutions to multi-objective problems are compared using the notion of Pareto dominance. In a minimization problem for all objectives, solution x_1 dominates solution x_2 (also written as $x_1 > x_2$), if and only if the two following conditions are true:

- x_1 is no worse than x_2 in all objectives, i.e. $f_v(x_1) \leq f_v(x_2)$; $\exists v \in \{1, 2, \cdots, V\}$;
- x_1 is strictly better than x_2 for at least one objective, i.e. $f_v(x_1) < f_v(x_2)$; $\exists v \in \{1, 2, \cdots, V\}$.

Then, a solution is said to be Pareto-optimal if it is not dominated by any other possible solution, as described above. Thus, the Pareto-optimal solutions to a multi-objective optimization problem form the Pareto-front or Pareto-optimal set [31].

3. Problem definition

In this article, k-out-of-n system containing S independent subsystems, as in Figure 1, is considered. Since this structure is more realistic, it is the most important and the most utilized structure in different studies [32]. In this structure, a system is arranged with S subsystems beside each other; in each subsystem n_i $(i = 1, 2, \cdots, S)$, different components are arranged in parallel. There are different types of component choices per subsystem, and the components within the same subsystem are of the same type.

3.1. Assumptions

- The components of each subsystem are arranged in parallel;
- The lifelong of elements (components) is non-exponentially distributed;

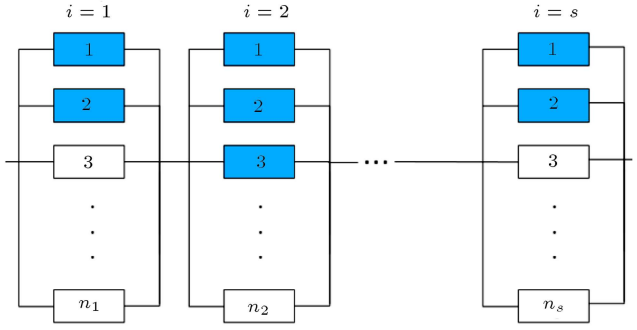

Figure 1. The structure of a k-out-of-n system containing s independent sub-systems.

- The components are repairable and the repair rate of elements is also non-exponential;
- Mixture of components is allowed within a subsystem;
- The components allocated to each subsystem are of the same type;
- A certain number of components are utilized within each subsystem;
- There is one repairman able to repair all the components.

3.2. Mathematical model of the problem

According to the assumptions, the applied symbols, indices, and mathematical models including objective function and the constraints of the problem are as follows:

3.2.1. Symbols, indices, and variables of the problem

i	Index of subsystem
j	Index of component choice used for subsystems i
x_{ij}	Number of selected components of type j for subsystem i
λ_{ij}	Failure rate of component of type j for subsystem i
μ_{ij}	Repair rate of component of type j for subsystem i
S	Number of subsystems
n_i	Number of components allocated to subsystem i
c_{ij}	Cost of component j available for subsystem i
w_{ij}	Weight of component j available for subsystem i
W_s	Weight of the system
N_i	Upper bound for n_i
N_s	Maximum number of components used in the whole system
y_{ij}	Binary variable to choose/not choose one component type for the subsystem i
M	A big number

3.2.2. Mathematical model

$$\max Z = \min\{\text{MTTF}_{S_i}\}, \tag{1}$$

$$\min C_s = \sum_{i=1}^{S} \sum_{j=1}^{n_i} c_{ij} \times x_{ij}, \tag{2}$$

s.t.

$$\sum_{i=1}^{S}\sum_{j=1}^{n_i} w_{ij} \times x_{ij} \leq W_s, \tag{3}$$

$$\sum_{j=1}^{n_i} y_{ij} = 1, \qquad 0 \leq x_{ij} \leq N_i.y_{ij} \qquad \forall i = 1, \cdots, S, \tag{4}$$

$$\sum_{i=1}^{S}\sum_{j=1}^{n_i} x_{ij} \leq N_s, \tag{5}$$

$$k_i \leq \sum_{j=1}^{n_i} x_{ij} \leq N_i \qquad \forall i = 1, \cdots, S, \tag{6}$$

$$y_{ij} \in \{0,1\}k_i, \qquad x_{ij} \geq 0, \text{int}, \tag{7}$$

where:

$$\text{MTTF}_{S_i} = f(x_{ij}, \lambda_{ij}, \mu_{ij}).$$

Eq. (1) is the main objective function or Mean MTTFF of the system. The second objective function of the problem is based on the minimization of the total cost of the system in Eq. (2); Eq. (3) is the system weight constraint; constraints required to choose only one component type are provided in Eq. (4); Eq. (5) is the constraint of number of components of the whole system; Eq. (6) is the constraint of lower and upper bounds for the number of components allocated to each subsystem; for the proper function of each subsystem, it is required that, at least, k_i out of n_i components operates. And finally, the constraint of components' type and their extents is provided in Eq. (7).

The aim of this article is to increase MTTFF of a series-parallel system. In this system, there are components arranged in parallel in each subsystem, and a subsystem operates in spite of failures and repairs until all the components fail. This system with its series subsystems fails for the first time, i.e. the first failure occurs whenever one of the subsystems completely fails [8]. Hence, mean time of the first failure of the system is the mean time within which one of the subsystems completely fails. Therefore, the increase of mean time of the first failure of the system is the minimum value of the mean failure of each subsystem. The algorithm for the calculation of MTTFF of a subsystem with parallel and reparable components is provided in [33].

4. Methods for reliability optimization

The methods for reliability optimization are classified into three main categories of exact, approximate, heuristic and meta-heuristic. These are explained as follows [4,18]:

- **Exact methods:** In these methods, the optimal solution is calculated for the reliability optimization problems. Rate of the computations of the exact method exponentially increases according to the increase of the problem quantity. Some of the exact methods are dynamic programming, gradient method, linear programming, and integer programming;
- **Approximate method:** In these methods, the model of the problem is approximated due to the complexity of the models, and the optimal solution to the approximate model is calculated using the exact methods. Some of the important approximate methods are the geometric programming, Lagrange multiplier method, random search and lexicographic method;
- **Heuristic and meta-heuristic methods:** Considering the long duration of computation, the heuristic and meta-heuristic methods are proposed. These methods provide a near optimal solution in a proper computational time.

4.1. The proposed solving methods

4.1.1. The Multi-Population Genetic Algorithm (MPGA)

The two-stage approach, which we call MPGA, is illustrated in Figure 2. In the first stage, the GA evolves based on the combined objective. The solutions at the end of the first stage are rearranged and divided into subpopulations, then they start to evolve separately. The steps of each stage are described in the following subsection. Essentially, this approach uses a modification of Multi-Objective Genetic Algorithm (MOGA) in stage one and a modification of Vector Evaluated Genetic Algorithm (VEGA) in stage two. The general pseudo-code of the proposed MPGA is described in Figure 2.

Encoding

The first step in running the Genetic Algorithm is to represent the solution or design the chromosome. The chromosomes are so designed to estimate the main limitations of the problem as much as possible. The chromosome designed in this study is a $T \times N$ matrix where T represents the type and number of selected components, and N is the number of subsystems. Each column represents a subsystem, and the value of each cell in the first row represents the type; in the second row, the value of each cell represents the number of components in the relevant subsystem.

For example, Figure 3 shows the layout of a system with four subsystems, where in the first subsystem, there are four components of type two; in the second subsystem, there is one component of type three; in the third subsystem, there are three components of type

Algorithm: The main procedure of MPGA

Stage 1

1. **Representation:** Encoding the problem into a search solution.
2. **Initialization:**
   (a) **Parameters setting:** Set the number of population (N), the number of sub-population (N_s), number of individuals in each sub-population (n), Max_Gen 1,2 (the total number of generations of stage 1 and stage 2), probability of crossover (P_c), probability of mutation (P_m)
   (b) **Initial population generation:** Randomly generate an initial population.
3. counter←0
4. **while** counter $<$ Max_Gen1 **do**
5. **for** $i = 1$ to N do
6. **Selection**
7. Crossover: Select $N_s \times P_c$ pairs of parents from current population, and perform crossover on the parents.
8. Mutation: Select $N_s \times P_{am}$ chromosome from current population and mutate the individual bits.
9. Simulation of the first function
10. Evaluation of combined: Evaluating the fitness function for each of the chromosome use combine the min-max method with the weighting method (to combine them into a scalar fitness function) to generate various Pareto solution.
11. Finding Pareto solution: Efficient solutions of current population are copied into an archive that stores the best non-dominated front obtained.
12. Elitist: Binary tournament selection and Elitist selection are employed to reproduce the next generation.
13. Turing criteria.
14. **End for**
15. counter←counter+1
16. **End while**

Stage 2

17. counter←0
18. Re-initialization
19. Dividing population
20. **while** counter$<$ Max_Gen2 **do**
21. **for** $i = 1$ to N_s do
22. **Selection**
23. Crossover
24. Mutation
25. Simulation of the first function
26. Evaluation of combined values in the function
27. Finding Pareto solutions: Efficient solutions of current population are copied into an archive that stores the best non-dominated front obtained.
28. Merge sub-population.
29. Elitist: Certain selection use non-dominated sorting and Elitist selection is employed to reproduce the next generation.
30. **End for**
31. counter←counter+1
32. **End while (Stop)**

Figure 2. The general pseudo-code of the proposed MPGA.

	Subsystem			
Type of component selected	2	3	2	1
Number of component	4	1	3	2

Figure 3. Encoding solution as a chromosome representation.

two; in the fourth subsystem, there are two components of type one.

Initialization

In this article, different populations are generated instead of using only the initial population and running the steps of genetic algorithm on it. The chromosome structure is the same as in all of these populations, but each population could have its own steps of selection, generation, and mutation; after some generations, we exchange the chromosomes within them by Elitism Algorithms. Since the efficiency of different models of Genetic Algorithm heavily depends on data types, the proposed method minimizes the risk of exposure to local minimum due to its possibility to use various methods in each population.

Selection

Selection is an operation to select two parent strings for generating new offspring. Let X_t^i be the ith solution (I between 1 and population size) in the tth generation, and $f(x_t^i)$ be the performance measure of solution X_t^i. Each solution X_t^i is selected as a parent string according to the selection probability $P(x_t^i)$. The following method is used:

$$P(x_t^i) = \frac{[f_t^w - f(x_t^i)]^2}{\sum_{k=1}^{N}[f_t^w - f(x_t^k)]^2}, \tag{8}$$

where f_t^w is the worst value of objective at generation t.

Crossover

For crossover operation, we follow steps below considering the condition of feasibility of the generated offsprings:

1. The first chromosome of the population is selected;
2. Generate a random number r between 0 and 1;
3. If $r < P_c$, then choose that algorithm for the crossover operation;

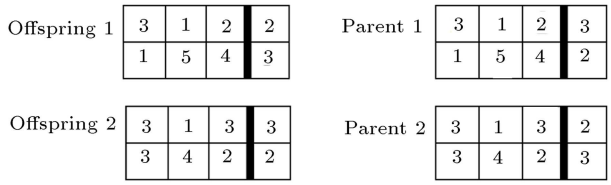

Figure 4. An example of the crossover process to fine new solutions.

4. Now, pair the selected chromosomes randomly; generate a random number of "cross point" per chromosome in the range of $\{1, \cdots, N\}$, where N is the number of subsystems;
5. Copy the bits of 1 through cross point in the chromosome of the first parent directly into the genes of the first off-spring;
6. The bits of cross *point+1* through N of the second parent are transited to the first off-spring according to their arrangement in the second parent;
7. Repeat steps 6 and 7 for the generation of the second off-spring;
8. Repeat the above steps to generate the other off-spring chromosomes.

The crossover process is shown in Figure 4 schematically.

Mutation

The most important function of mutation operator is to avoid the convergence and local optimum and searching in intact spaces of the problem. The task of mutation of a chromosome is to change its genes, and it has different methods depending on the type of coding. We follow the steps below concerning the mutation on the chromosomes:

1. The first chromosome in the population is selected;
2. Generate a random number r between 0 and 1;
3. If $r < P_m$, then the chromosome is mutated, i.e. one of the genes of the component type or the number of components is selected and replaced with another value being randomly and feasibly generated. Figure 5 illustrates this operator graphically.

Evaluation of the main objective function

A series-parallel system fails for the first time when one of the subsystems fails, i.e. if each subsystem of the system fails first, it results in the failure of the whole system. Hence, in order to calculate the MTTFF of the system, it is required to simulate the MTTFF of each subsystem, and the MTTFF value of the subsystem which has the earliest failure time is considered as the MTTFF of the whole system. Since all the relations and equations for calculating the MTTFF of a system are possible by failure probabilities and exponential repair by mathematical and statistical analyses; it is not possible by any kind of repair or analysis other than non-exponential repair, and in this way, the simulation approach is applied to calculate this value. It should be noted that the fitness of each chromosome is equal to their mean fitness in n times of simulation of that chromosome. Therefore, we are able to calculate the fitness rate of each chromosome without any unrealistic assumption.

Figure 5. Example of the mutation process to avoid local optimums.

Evaluation of the combined objective

One of the most famous methods of multi-objective optimization (MOOP) is "weighted aggregation". In this method, a linear combination of objective functions with non-negative different weights is used to form "aggregated function" as below:

$$F(x) = \sum_{i=1}^{m} w_i f_i(x) \qquad \sum_{i=1}^{m} w_i = 1; \qquad w_i \geq 0. \quad (9)$$

In the above formula, $F(x)$ is aggregated function, and w_i is lth non-negative weight related to lth objective function. There are different methods for aggregating objective function and producing aggregated function $F(x)$. The best method to estimate aggregation of functions is "Dynamic Weighted Aggregation" method (DWA), because this method shows better ability to estimate concave Pareto-fronts. This method is defined for two objective functions as below (it can be generalized easily to more than two-objective functions):

$$(w_1(t), w_2(t)) = (|\sin(2\pi t/R)|, \qquad 1 - w_1(t)), \quad (10)$$

where t is tth sub-population ($t = 1, 2, \cdots, Ns$) and $R = 200$.

MOOPs are usually solved by scalarization (it means converting the problem with multiple objectives into a single objective or a family of single objective optimization problems). The major trouble of weighting method is the need for scalarizing multiple objectives in sum weighting, and in this regard, we employ one of the most widely used multi-objective methods called Min-Max. In Min-Max approach, we use the idea of minimizing the distance of every solution from the best possible solution f^*. In other way, if $f_i(x)$ is the ith objective function and f_i^* is the best available solution for $f_i(x)$, then the quantity of f^* will be $f^* = (f_1^*, f_1^*, \cdots, f_m^*)^T$. So, the following function is minimized subject to the constraint of the problem:

$$\min \left[\sum_{i=1}^{m}\left(\frac{f_i(x)-f_i^*}{f_i^*}\right)^p\right]^{\frac{1}{p}},$$

$$\text{s.t.} \quad X \in S,$$

$$1 \le p \le \infty. \tag{11}$$

Exponent p shows various ways for measuring the distance. The most widely applied values of p are 1 for the simplest formulation and 2 for the Euclidean distance, Andersson [34]. The major challenge in this approach is to find the value of p that will maximize the satisfaction of Decision-Makers (DM). Another shortcoming of this method is that there is only one output, and DM has to accept it as a final solution. Thus, in this phase, through mixing the weighting and Min-Max method, two problems will be solved, one of which is the mono-solution of Min-Max, and the other is the problem of scalarzing in weighting method. Consequently, there would be two-objective functions defined by:

$$\left[w\left(\frac{f_1(x)-f_1^*}{f_1^*}\right)^p+(1-w)\left(\frac{f_2(x)-f_2^*}{f_2^*}\right)^p\right]^{\frac{1}{p}}, \tag{12}$$

where $f_1(x)$ and $f_2(x)$ indicate each individual minima of each respective objective function, and $0 < w < 1$. w denotes the weight (or relative importance) of a number of setups and the usage rate. In this paper, p value is determined as 1. Because the scales of the objectives are different, the normalization process has been applied to objective values of this method. Note that all the examples solved here have used 10 different seeds for every algorithm, and the minimum solution in all runs is f^* for each objective. These are substituted in Eq. (13).

Elitist

The best solution of each objective and the best one of the combined objective function are preserved in each generation. For each generation, if the best solution is worse than the preserved one, a randomly selected string will be replaced with the preserved.

Turning criterion

After a certain number of generations, the algorithm switches to the next stage. In stage 2, the populations of the solution from stage 1 will be rearranged based on their performance of each objective.

Re-initialization

Assume N objectives to be optimized, as discussed above, the solutions from the first stage are rearranged and $N + 1$ sub-populations will be created and evolved separately. The first stage through Nth sub-populations is for N objectives, and $(N + 1)$th subpopulation is for the combined objective.

Selection, Crossover, and Mutation

The same selection, crossover, and mutation procedures used in Stage 1 are applied to each subpopulation.

Elitist strategy

Although each sub-population evolves separately, the elitist strategy searches for the best solution to each objective and combined objective across all subpopulations. $N + 1$ solutions will be stored and will replace the worst solution of each objective and the combined objective.

Stopping criteria

A test run indicates that the algorithm does not show significant improvement after 2,000 generations. However, to consider the error due to the randomness and have more promising results, a larger generation number should be used as the stopping criteria. In addition, most literature reviews have used 3,000 generations to stop the algorithm. Therefore, 3,000 generations are used in this study as stopping criteria. A flow chart of the proposed MPGA is depicted in Figure 6.

4.1.2. Weighted sum Multi-Objective Genetic Algorithm (WMOGA)

One of the most famous methods in multi-objective optimization is "weighted aggregation". The two objectives are usually formulated in a weighted sum approach. The details of the proposed WMOGA are presented in Figure 7.

4.1.3. Non-dominated Sorting Genetic Algorithm II (NSGA-II)

As a well-known Multi-Objective Evolutionary Algorithm (MOEA), the NSGA-II has been the most widely used and has been proven to do well on various real-world application problems [35]. The pseudo-code of NSGA-II is presented in Algorithm 1 in Figure 8. We used NSGA-II in our research, since there have been many investigations ensuring that NSGA-II can often converge to Pareto-optimal set, and the obtained solutions can often spread well over the Pareto-optimal set. NSGA-II takes the fast non-dominated-sort mechanism to ensure the well convergence, shown in Algorithm 2, Figure 8. For details of NSGA-II, one can refer to [36]. The general pseudo-code of the NSGA-II is described in Figure 8.

5. Computational results

In order to conduct the experiment, we implement the proposed algorithms in Matlab 7.8 software, ED 8.1 to simulate the problem. All computations are on a PC with Intel Pentium 4, 1.67 GHz processor, 4 GByte memories with windows 7 Professional Operating System.

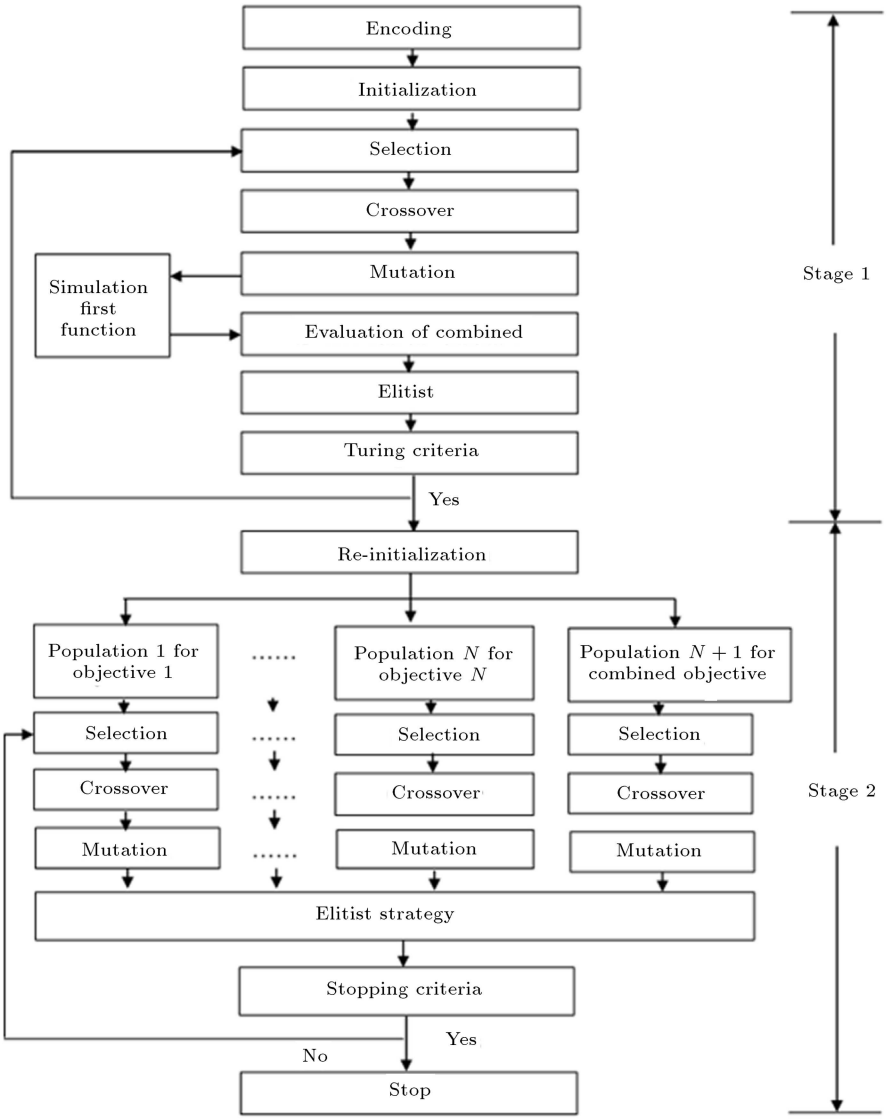

Figure 6. A two-stage Multi-Population Genetic Algorithm (MPGA).

Test problems: The experiments were implemented over 30 test problems. For all experiments, the following assumptions were considered:

Data generation: Also, in this article, a series of experiments are designed where the following values are considered for the variety of distributions:

- The component type one has exponential distribution: λ for the failure (mean value) in the range of (0.06, 0.25) and μ for the repair value in the range of (0.033, 0.167) are considered, respectively;
- The component type two has Erlang distribution: for the failure of the components, the value of α in the range of (0.3, 0.9) and $\beta = 2$; for the repair, the value of α in the range of (0.1, 0.6) and $\beta = 2$ are considered, respectively;
- The component type three has Weibull distribution: for the failure of the components, the value of α in the range of (0.5, 0.9) and $\beta = 0.5$ and for the repair, the value of α in the range of (0.1, 0.5) and $\beta = 0.5$ are considered, respectively.

In order to explain the efficiency of the proposed algorithms, the illustrative examples are designed with 30 experiments. The number of subsystems in the problem is 5, 15, and 20 subsystems based on which the random problems are produced in small, medium, and large sizes. For each size, 10 sample problems are generated. In these problems, the maximum number of the components in each subsystem is between 5-10, the minimum number of components for each subsystem is between 2-4, the volume and cost of each component are between 200-300 and 100-500, and the maximum volume available for the system is between 9000-13000.

1. **Representation:** Encode the problem into a search solution
2. **Initialization:**
   - **Parameters setting:** Set the number of population ($popsize$).Max_Gen (the total number of generations), probability of crossover (P_c), probability of mutation (P_m)
   - **Initial population generation:** Randomly generate an initial population.
3. Assign weight to each objective
4. Counter $\leftarrow 0$
5. **While** counter $<$ Max_Gen or maximum CPU time **do**
6. **for** $j = 1$ to $popsize$ do
7. **Evaluation**: To evaluate the fitness function for each solution, the min-max method must be combined with the weighting method. The resulting values are scalars which can be used to generate the Pareto front.
8. **Find Pareto solutions:** Efficient solutions of current population are copied into an archive that stores the best non-dominated front obtained
9. **Selection/Elitist:** Roulette wheel selection and Elitist selection are employed to reproduce the next generation.
10. **Crossover operation:** Select $popsize \times P_c$ pairs of parents from current population:
    a. Determine the pairs of parents among the parent chromosomes.
    b. Apply the crossover operator to produce two offsprings corresponding to each pair.
    c. Replace each offspring in the population instead of the parents.
11. **Mutation operation:** Select $popsize \times P_m$ chromosome from current population, and mutate the individual bits
12. **Update archive:** Efficient solutions of current population are copied into an archive that stores the best non-dominated front obtained.
13. **Generate next generation:**
14. **End for**
15. counter$\leftarrow$counter+1
16. **End while**

Figure 7. The pseudo-code of the WMOGA.

Algorithm 1. The pseudo-code of NSGA-II

Step 1: Set the parent vector $P = \emptyset$, the offspring vector $Q = \emptyset$, the collect vector $R = \emptyset$, and the generation number $t = 0$.
Step 2: Initialize the parent vector P_0.
Step 3: **While** $t <$ the terminate generation number **do**
(1) Combine the parent and offspring population via $R_t = P_t \cup Q_t$.
(2) Sort all solutions of R_t to get all non-dominated fronts F=fast-non-dominated-sort (R_t) where $F = (F_1, F_2, \cdots)$.
(3) Set c and $i = 1$.
(4) **While** the parent population size $|P_{i+1}| + |F_i| < N$ **do**
(a) Calculate crowding-distance of F_i.
(b) Add the ith non-dominated front F_i to the parent pop P_{t+1}.
(c) $i = i + 1$.
end while
(5) Sort the F_i according to the crowding distance.
(6) Fill the parent pop P_{t+1} with the first $N - |P_{t+1}|$ elements of F_i.
(7) Generate the offspring population to Q_{t+1}.
(8) Set $t = t + 1$.
End while
Step 4: The population in vector P is the non-dominated solutions.

Algorithm 2. The Pseudo-code for the function: Fast non-dominated /sort (P)

Step 1: For each population p in the P, we get the solutions which p dominates and save these solutions into S_p. We also need to calculate the n_p, which is the the number of solutions that dominate P.
Step 2: Find the solutions whose $n_p = 0$ and add them to the first front F_1.
Step 3: Initialize the front counter $i = 1$.
Step 4:
While F_i is not empty **do**
Set the temp vector $Q = \emptyset$.
for each $p \in F_i$ **do**
for each $q \in S_p$ do
$n_p = n_p - 1$.
If $n_p = 0$ then add q to the Q
end for
end for
$i = i + 1$ and the solutions in Q compose the F_i.
End while

Figure 8. The pseudo-code of the NSGA-II.

Comparator algorithms: Performance of the developed MPGA was compared with two Meta heuristics developed for the multi-objective redundancy allocation problem, discussed in this paper in a set of problems.

5.1. Parameters settings

This subsection tries to find the optimal parameter setting in the algorithms. There are several parameters that may influence the performance of the algorithms. For example, the larger population size may find better solution quality but cost higher computational expense. It should be noted that changing these parameters may result in different outcomes than those achieved in this research. When the number of populations is larger, it may have better diversity. However, it may also be a trade-off to cut the number of generations. Moreover, the crossover and mutation operator is also considered, because it may offer better solution quality. The parameters' setting of algorithms is indicated in Table 1.

5.2. Performance measures

To evaluate the performances of the proposed MPGA,

Table 1. Parameter setting.

Parameter setting MPGA		Parameter setting NSGA-II		Parameter setting WMOGA	
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
Population size (N)	500	Popsize	500	Popsize	600
Number of sub-population (N_s)	50	Max_Gen	750	Max_Gen	800
n	10	Crossover rate	0.7	Crossover rate	0.6
Max_Gen phase 1	400	Mutation rate	0.3	Mutation rate	0.2
Max_Gen phase 2	600	Elitist	Top 20% of population	Elitist	Top 20% of population
Crossover rate	0.6				
Mutation rate	0.4				
Elistist	Top 20% of population				

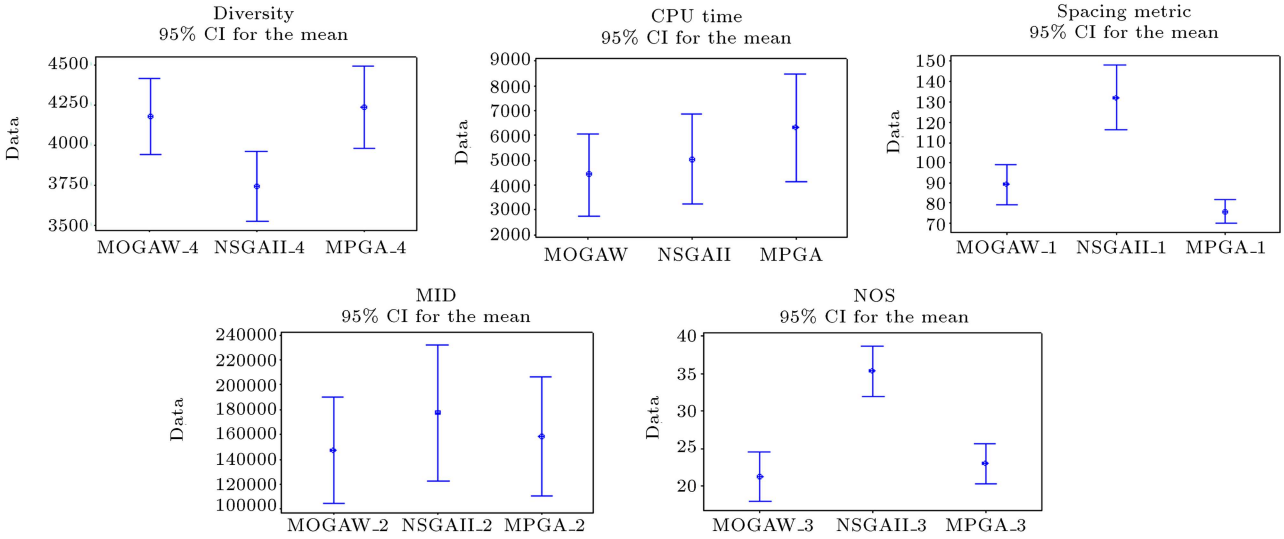

Figure 9. Comparability of MPGA in comparison with NSGA-II and WMOGA on diversity, CPU time, MID, spacing, and NOS metrics.

five standard metrics of multi-objective algorithms are applied as follows:

- Diversity: Measures the extension of the Pareto front in which bigger value is better [23];
- Spacing: Measures the standard deviation of the distances among solutions of the Pareto front in which smaller value is better [23];
- Mean Ideal Distance (MID): Measures the convergence rate of the Pareto front to a certain point (0, 0) in which smaller value is better [23];
- Number Of found Solutions (NOS): Counts the number of the Pareto solutions in Pareto optimal front in which bigger value is better;
- The computational (CPU) time of running the algorithms to reach near optimum solutions.

The experiments are implemented on 30 test problems. Furthermore, to eliminate uncertainties of the solutions obtained, each problem is used three times under different random environments. Then, the averages of these three runs are treated as the ultimate responses. Then, we compare the proposed MPGA algorithm with WMOGA and NSGA-II as the most applicable Pareto-based MOEAs in the test problem to demonstrate the performance of the proposed algorithm to solve the multi-objective optimization problems.

To evaluate the performance of the proposed MPGA, Table 2 reports the multi-objective metrics' amounts on 30 test problems, in which "NAS" shows that the algorithm cannot find Pareto front in the reported time.

The algorithms are statistically compared based on the properties of their obtained solutions via the analysis of variance (ANOVA) test. These outputs are reported in Tables 3 to 7 in terms of defined metrics. In order to clarify our statistical results, interval-plots are represented in Figure 9.

Based on the statistical outputs in Tables 3 and 6 along with Figure 9, NSGA-II shows better

Table 2. Evaluation of non-dominated solution for algorithms grouped by problem size and index type.

Problem no.	CPU Time			Spacing metric			MID			NOS			Diversity		
	WMOGA	NSGAII	MPGA	WMOGA	NSGAII	MPGA	WMOGA	NSGAII	MPGA	WMOGA	NSGAII	MPGA	WMOGA	NSGAII	MPGA
P1	511	551	468	54	87	51	36177	43539	40388	10	24	13	3605	3608	3375
P2	466	514	581	57	84	59	40698	43304	41446	12	22	18	3500	3265	3361
P3	435	525	505	54	90	57	37564	41965	40008	12	27	16	3643	2887	3631
P4	529	498	506	53	84	60	39188	43587	40159	14	26	16	3704	3203	3522
P5	469	572	436	52	90	49	35951	44781	40206	11	29	16	3315	3257	3367
P6	451	575	561	56	85	54	38052	40629	39698	14	28	20	3656	2947	3308
P7	426	420	577	55	86	52	35003	40362	40157	11	24	17	3422	3375	3773
P8	501	412	489	59	83	50	35622	43768	41349	12	28	13	3323	2787	3715
P9	494	467	503	82	110	58	49656	51269	50928	21	25	21	3617	3417	4178
P10	915	1190	1482	75	104	80	44878	54642	48350	21	24	22	4278	3304	3563
P11	1079	1215	1595	78	105	83	47157	46195	49202	20	39	23	3328	3425	4297
P12	865	1271	1450	73	96	73	44595	47183	48562	14	33	19	3704	3557	4077
P13	838	1007	1528	77	93	80	46780	46929	47838	16	31	19	4219	3436	4219
P14	913	1045	1696	82	103	83	48738	43714	53699	20	34	23	4005	3502	3883
P15	3730	5263	6052	82	106	79	46195	49526	50788	21	37	21	3895	3254	3568
P16	3486	5156	8050	83	95	79	48658	46145	48988	18	38	23	3708	3522	4096
P17	3888	4317	8296	119	180	82	229771	275696	225875	15	35	18	4838	4453	4735
P18	3202	5841	8993	105	169	88	220649	289771	261871	21	36	28	5053	4058	4371
P19	3555	4822	8974	111	171	82	224140	277417	261938	23	37	28	4114	4694	4015
P20	3386	4563	8439	102	161	91	229502	287780	234874	18	40	18	4114	4430	4310
P21	8915	9190	9482	109	165	91	225910	270478	243206	29	35	18	4248	4148	4546
P22	9079	9215	9595	103	164	92	222076	280034	262142	28	39	28	5213	4791	4058
P23	8865	9271	9450	116	171	86	223147	282323	247898	24	38	21	5161	4627	4685
P24	8838	9007	9528	105	177	89	224107	285924	235584	19	35	27	4973	4666	4409
P25	8913	9045	9696	112	194	72	306678	380782	363201	37	55	37	4980	3592	5298
P26	11730	13263	14052	126	172	93	314191	377458	337583	35	48	39	5000	4277	4851
P27	11486	13156	16050	121	191	92	304098	384322	333531	35	50	29	4736	3893	5462
P28	11888	12317	16296	127	175	92	310736	381013	349602	37	45	31	4360	3743	5339
P29	11202	13841	16993	121	188	94	303129	386547	339830	33	44	32	4569	4282	5437
P30	11555	12822	16974	127	180	88	304583	391846	366935	36	53	35	5004	3897	5504

performances in terms of NOS, while WMOGA has a better performance in terms of CPU time.

Moreover, Tables 4 to 7 along with Figure 9 show the comparability of MPGA in comparison with NSGA-II and WMOGA on MID, spacing, and diversity metrics, in which the algorithms have no significant differences and statistically work the same. It is required to be mentioned that this conclusion is confirmed at 95% confidence level. Based on the outputs in Table 2, there is the increasing of size of problems in test problems 22 and 30; in test problem 30, NSGA-II and WMOGA cannot find Pareto front,

Table 3. Analysis of variance for the time metric.

Source	DF	SS	MS	F	P
Algorithms	2	55606000	278030	1.09	0.340
Error	87	221143724	254188		
Total	89	226704324			

Table 4. Analysis of variance for the spacing metric.

Source	DF	SS	MS	F	P
Algorithms	2	51401	25701	28.00	0.000
Error	87	79855	918		
Total	89	131256			

Table 5. Analysis of variance for the MID metric.

Source	DF	SS	MS	F	P
Algorithms	2	139856556	699282	0.40	0.669
Error	87	1.508E+12	173386		
Total	89	1.525E+12			

Table 6. Analysis of variance for the NOS metric.

Source	DF	SS	MS	F	P
Algorithms	2	3529.9	1764.9	25.98	0.000
Error	87	5910.6	67.9		
Total	89	9440.5			

Table 7. Analysis of variance for the diversity metric.

Source	DF	SS	MS	F	P
Algorithms	2	4291874	2145937	5.38	0.006
Error	87	34708077	398943		
Total	89	38999951			

respectively. However, in these large sizes, MPGA can find Pareto front. The MPGA algorithm performs better performance in the terms and CPUT metric. These features conclude robustness of the proposed MPGA in large-sized problems in the area of multi-objective optimization problems.

6. Conclusions

In this article, a new method is provided to model and solve the Redundancy Allocation Problem in the k-out-of-n series-parallel systems with non-exponential repairable components based on OVS technique. A bi-objective function was used to model the system. The first one was optimizing the Mean Time To the First Failure (MTTFF) of the system, and the second one was minimizing the total costs. The components are assumed to have non-exponential breakdowns and repair times, which are the main contributions of this research. A model with non-exponential components is closer to the reality where the memoryless property is rare to be found among the components. In general, the used components have shorter expected life time in comparison to new ones. Also, we have no practical reasons to assume exponential distributions for the components of repair times. When the stochastic events, such as the failures and repair times, are non-exponential, the MTFFF function cannot be written explicitly. This is the simulation technique which enables us to model and solve the model, where all stochastic events with any kind of distribution function could be modeled easily. In order to demonstrate applicability of the proposed solving algorithm (MPGA), the multi-objective reliability problems were applied. The proposed algorithm was able to improve the quality of the obtained solutions by taking the specific advantages of the multi-population Genetic Algorithm and also using the simulation techniques.

The results show the capability of the MPGA algorithm to solve the multi-objective problems. To justify the proposed algorithm, NSGA-II and WMOGA algorithms have been implemented to evaluate the performance of the proposed MPGA. The results show the efficiency of MPGA in the large-size problems. For future research, one may compare the proposed MPGA with other multi-objective algorithms (e.g., MOPSO or MOTS) in various optimization problems.

References

1. Kapur, K.C. and Lamberson, L.R., *Reliability in Engineering Design*, New York, John Wiley & Sons (1987).
2. Tekiner-Mogulkoc, H. and Coit, D.W. "System reliability optimization considering uncertainty: minimization of the coefficient of variation for series-parallel systems", *Reliability, IEEE Transactions on*, **60**(3), pp. 667-674 (2011).
3. Garg, H. "An efficient biogeography based optimization algorithm for solving reliability optimization problems", *Swarm and Evolutionary Computation*, **24**, pp. 1-10 (2015).
4. Kuo, W. and Prasad, V.R. "An annotated overview of system-reliability optimization", *IEEE Transaction on Reliability*, **2**, pp. 176-187 (2000).
5. Fyffe, D.E., Hines, W.W. and Lee, N.K. "System reliability allocation and a computational algorithm", *IEEE Transaction on Reliability*, pp. 64-69 (1968).
6. Coelho, L.D.S. "Reliability-redundancy optimization by means of a chaotic differential evolution approach", *Chaos, Solitons and Fractal*, pp. 594-602 (2009).
7. Zou, D., Gao, L., Li, S. and Wu, J. "An effective global harmony search algorithm for reliability problems", *Expert Systems with Applications*, **38**(4), pp. 4642-4648 (2011).
8. Ramirez-Marquez, J.E. and Coit, D.W. "A heuristic for solving the redundancy allocation problem for multi-state series-parallel system", *Reliability Engineering and System Safety*, pp. 314-349 (2004).
9. Nahas, N., Nourelfath, M. and Ait-Kadi, D. "Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series-parallel systems", *Reliability Engineering and System Safety*, pp. 211-222 (2007).
10. Safaei, N., Tavakkoli-Moghaddam, R. and Kiassat, C. "Annealing-based particle swarm optimization to solve the redundant reliability problem with multiple component choices", *Applied Soft Computing*, **12**(11), pp. 3462-3471 (2012).
11. Liang, Y.C. and Chen, Y.C. "Redundancy allocation of series parallel systems using a variable neighborhood

search algorithm", *Reliability Engineering and System Safety*, pp. 323-331 (2007).

12. Ha, C. and Kuo, W. "Reliability redundancy allocation: An improved realization for non-convex nonlinear programming problems", *European Journal of Operation Research*, pp. 24-38 (2006).
13. Soltani, R., Sadjadi, S.J. and Tofigh, A.A. "A model to enhance the reliability of the serial parallel systems with component mixing", *Applied Mathematical Modelling*, **38**, pp. 1064-1076 (2013).
14. Liang, Y.C., Lo, M.H. and Chen, Y.C. "Variable neighborhood search for redundancy allocation problems", *IMA Journal of Management Mathematics*, pp. 135-155 (2007).
15. Juang, Y.S., Lin, S.S. and Kao, H.P. "A knowledge management system for series-parallel availability optimization and design", *Expert System and Applications*, pp. 181-193 (2008).
16. Zhang, E., Wu, Y. and Chen, Q. "A practical approach for solving multi-objective reliability redundancy allocation problems using extended bare-bones particle swarm optimization", *Reliability Engineering & System Safety*, **127**, pp. 65-76 (2014).
17. Chambari, A., Najafi, A.A., Rahmati, S.H.A. and Karimi, A. "An efficient simulated annealing algorithm for the redundancy allocation problem with a choice of redundancy strategies", *Reliability Engineering & System Safety*, **119**, pp. 158-164 (2013).
18. Soltani, R. "Reliability optimization of binary state non-repairable systems: A state of the art survey", *International Journal of Industrial Engineering Computations*, **5**(3), pp. 339-364 (2014).
19. Kuo, W. and Wan, R. "Recent advances in optimal reliability allocation", *IEEE Transaction on System, Man and Cybernetics-Part a: System and Humans*, pp. 143-156 (2007).
20. Guedenko, B. and Ushakov, I., *Probabilistic Reliability Engineering*, A Wiley-Interscience Publication, New York (1995).
21. Elegbede, C. and Adjallah, K. "Availability allocation to repairable systems with genetic algorithms: A multi-objective formulation", *Reliability Engineering and System Safety*, **82**, pp. 319-330 (2003).
22. Ying-Shen, J., Shui-Shun, L. and Hsing-Pei, K. "A knowledge management system for series-parallel availability optimization and design", *Expert Systems with Applications*, **34**, pp. 181-193 (2008).
23. Chambari, A., Rahmati, S.H.A. and Najafi, A.A. "A bi-objective model to optimize reliability and cost of system with a choice of redundancy strategies", *Computers & Industrial Engineering*, **63**(1), pp. 109-119 (2012).
24. Garg, H., Rani, M., Sharma, S.P. and Vishwakarma, Y. "Bi-objective optimization of the reliability-redundancy allocation problem for series-parallel system", *Journal of Manufacturing Systems*, **33**(2), pp. 353-367 (2014).
25. Zoulfaghari, H., Hamadani, A.Z. and AboueiArdakan, M.A. "Bi-objective redundancy allocation problem for a system with mixed repairable and non-repairable components", *ISA Transactions*, pp. 17-24 (2014).
26. Li, Z., Liao, H. and Coit, D.W. "A two-stage approach for multi-objective decision making with applications to system reliability optimization", *Reliability Engineering and System Safety*, **94**, pp. 1585-1592 (2009).
27. Lai, C-D. and Lin, G.D. "Mean time to failure of systems with dependent components", *Applied Mathematics and Computation*, **246**(1), pp. 103-111 (2014).
28. Najafi, A.A., Karimi, H., Chambari, A. and Azimi, F. "Two metaheuristics for solving the reliability redundancy allocation problem to maximize mean time to failure of a series-parallel system", *Scientia Iranica*, **20**(3), pp. 832-838 (2013).
29. Garg, H. and Sharma, S.P. "Multi-objective reliability-redundancy allocation problem using particle swarm optimization", *Computers & Industrial Engineering*, **64**, pp. 247-255 (2013).
30. Khalili-Damghani, K., Abtahi, A.R. and Tavana, M. "A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems", *Reliability Engineering & System Safety*, **111**, pp. 58-75 (2013).
31. Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA- II", In *Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI) Conference*, pp. 849-858 (2000).
32. Coit, D.W. and Liu, J. "System reliability optimization with k-out-of-n subsystems", *International Journal of Reliability, Quality and Safety Engineering*, **7**(2), pp. 129-142 (2000).
33. Amiri, M. and Ghassemi-Tari, F. "A methodology for analyzing the transient availability and survivability of a system with repairable components", *Applied Mathematics and computation*, pp. 300-307 (2007).
34. Andersson, J. "A survey of multi-objective optimization in engineering design", In Technical Report LiTH-IKP-R-1097, Department of Mechanical Engineering, Linköping University, Linköping, Sweden (2000).
35. CoelloCoello, C.A., Van Veldhuizen, D.A. and Lamont, G.B., *Evolutionary Algorithms for Solving Multi Objective Problems*, Kluwer Academic Publishers (2002).
36. Srinivas, N. and Deb, K. "Multi objective optimization using non-dominated sorting in genetic algorithms", *Evolutionary Computation,* MIT Press Journals, **2**(3), pp. 221-248 (1994).

Biographies

Parham Azimi was born in 1974 in Tehran. He received his PhD degree in Industrial Engineering in Sharif University of Technology. He has been a university tutor at the Faculty of Industrial and

Mechanical Engineering at Qazvin Islamic University since 2004. His research interests are optimization via simulation, graph labeling problems, and simulation modeling.

Mojtaba Hemmati received his BS and MS degrees in Industrial Engineering from Qazvin Islamic Azad University, Iran, where he is currently a PhD student. His research interests include applications of operations research in redundancy allocation problems, queuing system, and more specifically, modeling and solution methods including exact procedures, simulation methods and artificial intelligence techniques.

Amirhossein Chambari received his MS degree in Industrial Engineering from Islamic Azad University, Qazvin, Iran, in 2010, where he is currently a PhD student. He currently works as purchase expert of the oil Co., Tehran, Iran. His research interests include reliability engineering, facility location-allocation, simulation methods, and artificial intelligence techniques.