Presenting a Series-Parallel Redundancy Allocation Problem with Multi-State Components Using Recursive Algorithm and Meta-Heuristic

Mani Sharifi *a, Majid Saadvandi a, Mohammadreza Shahriari b

a Faculty of Industrial & Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran,
b Faculty of Management & Accounting, South Tehran Branch, Islamic Azad University, Tehran, Iran.

Abstract: Redundancy Allocation Problem (RAP) is one of the most important problems in the field of reliability. This problem aims to increase system reliability, under constraints such as cost, weight, etc. In this paper, we work on a system with series-parallel configuration and multi-state components. To draw the problem nearer to real condition, we merge this problem with discount levels in purchasing components. For calculating sub-systems reliability, we used recursive algorithm. Because redundancy allocation problem belongs to Np. Hard problems, for optimizing the presented model a new Genetic algorithm (GA) was used. The algorithm parameters tuned using Response surface methodology (RSM) and for validation of GA an enumeration method was used.

Keywords: Reliability optimization, Multi-state components, Redundancy Allocation Problem, Recursive algorithm, Genetic Algorithm.

1. Introduction

The simplest model of the RAP (Redundancy Allocation Problem) is to assign the identical components for each subsystem. In mathematical models originally provided for redundant allocation problems, it was assumed that the components of the systems are at binary state. This means that the components have only two states: working or failed. In this paper, we intend to model the problem with the multi-state system components in order to get closer to real-world conditions. The multi-state components have several functional states that start from working and end up to failing.

Fyffe et al. [1] presented the mathematical model of the general RAP problem. Their objective function was to maximize the reliability under weight and cost constraints. They solved the model by using dynamic programming. Ida and Yokota [2, 3] provided a simple GA (Genetic Algorithm) for solving RAP without the possibility of allocating non-identical components to
each subsystem in a series-parallel system of several failed states. Coit and Smith [4] with changes in the objective function, solved the problem using GA. One of the major difficulties in solving RAP with GA is production and selection of infeasible solutions. For this reason, the penalty functions were defined to reduce the chance of selecting these infeasible solutions. Coit and Smith [5] offered an effective penalty function for RAP. Coit and Smith [6, 7] introduced a new model with a solution to RAP. They used GA to solve the proposed new model in parallel-series systems with k-out-of-n: G subsystems. The main characteristic of the proposed algorithm was the presentation of the algorithm chromosome. Coit [8] presented a new model with a solution method for RAPs with parallel-series structure. Tavakkoli-Moghaddam et al. [9] used the GA to solve RAP. The main characteristics of the proposed algorithm were using the design of chromosomes and mutation operators. Tavakkoli-Moghaddam and Safari [10] provided a new model for redundant allocation problems with the possibility of allocating non-homogeneous components to each subsystem and also choosing a redundancy policy for each subsystem. Chambers et al. [11] presented a two-objective model for the RAP in parallel-series systems under assumptions such as non-reparability. Zaretalab et al. [12] solved the model presented by Chmbari with the help of knowledge-based-archive SA (Simulated Annealing) algorithm (Knowledge Based Archive Simulated Annealing). They showed that their proposed meta-heuristic algorithm is better than other algorithms. For the first time, Ushakov introduced UGF (Universal Generating Function) concept and used it to calculate the reliability of systems with multi-state components. Li and Zuo [13] reported that their proposed method (when the number of system components is high) can reduce the computational time significantly in comparison to the UGF method. The proposed method is well known as recursive algorithm. Pourkarim Gilani et al. [14] used a modified Markov process and provided a new method for calculating the reliability of a system with three-state components, which yields a much lower computational time in comparison with UGF method and recursive algorithm. Pourkarim Gilani et al. [15] solved a mathematical model of RAP with subsystems consisting of three-state components using complete numerical methods and GAs that cannot be generalized to other multi-state systems. The main objective of this paper is to consider discount level when the redundant components purchase from the suppliers. So that when the suppliers offer a general discount for each component, then the unit price of each component depends on the total number purchased from
that supplier. In this case, the price is the level of discount considered for the total purchased components [16]. Table 1 contains some new researches in the field.

Insert table 1 here

In this paper, in order to optimize the system reliability, we used recursive method instead of the UGF method due to its faster computing speed. The performance of the recursive algorithm to evaluate the reliability of multi-state systems is satisfied. This method also has a lower computational time than other evaluation methods such as universal generation function. In the conducted researches by Guilani et al. [14] and Li & Zuo [13] this fact is confirmed. Also, because that RAP belongs to Np. Hard problems, the GA was used to obtain the optimal combination of components.

This paper is divided into five sections. The second section defines the problem definitions the third section deals with solving methods. Section four is a numerical example ant the last section deals with conclusion and further studies.

2. Problem Definition

2.1. The Proposed Model

Consider a system consisting of $s$ subsystems that connected serially, and each subsystem has $n_i$ parallel components. The components of each subsystem are multi-state and non-repairable. Also, the price of each component is calculated according to the total amount of the purchase and has discount level. This model aims to find the optimal number and the type of components in each subsystem considering that only one type of component will be assigned to each subsystem from available components type and the objective function is to minimize the system cost.

2.2. Model Assumptions

- Each component is multi-state,
- System parameters such as cost and weight are constant,
- The components are non-repairable,
- Components failures are independent and the failure of each component does not damage the system.
2.3. Nomenclatures

\( i \): Subsystem index, \( i = 1,2,\ldots,S \)

\( S \): Number of subsystems,

\( n_{i,\text{max}} \): Maximum allowable components in subsystem \( i \),

\( j \): Components type index, \( j = 1,2,\ldots,m_{i,\text{max}} \)

\( m_{i,\text{max}} \): Maximum available components type for the subsystem \( i \),

\( c_{ijk} \): Price of components type \( j \) in subsystem \( i \) in discount level \( k \), \( k = 1,2,\ldots,\lambda_{j,\text{max}} \)

\( \lambda_{ijk} \): Discount level \( k \) for components type \( j \) in subsystem \( i \),

\( n_{ijk} \): Maximum purchase amount for discount level \( k \) for components type \( j \) in subsystem \( i \),

\( N_{ij} \): Order value of components type \( j \) in subsystem \( i \),

\( \lambda_{ij,\text{Max}} \): Maximum discount level of components type \( j \) in subsystem \( i \),

\( w_{ijk} \): Weight of components type \( j \) in subsystem \( i \) in discount level \( k \), \( k = 1,2,\ldots,m_{ij} \)

\( W \): Maximum acceptable weight of system,

\( \omega \): Minimum acceptable system performance rate,

\( A(\omega) \): System availability,

\( A_{0} \): Minimum acceptable system availability,
2.4. Mathematical model

The mathematical model of RAP with consideration of the model assumptions is as follow:

\[
\begin{align*}
\text{Min} & \quad Z = \sum_{j=1}^{S} \sum_{j=1}^{m_j,\text{Max}} \left( \sum_{k=1}^{\lambda_{ijk},\text{Max}} c_{ijk} \lambda_{ijk} \right) N_{ij} \\
\text{S.t.:} & \quad \sum_{j=1}^{S} \sum_{j=1}^{m_j,\text{Max}} w_{ij} N_{ij} \leq W \\
& \quad N_{ij} \geq \lambda_{ijk} N_{ij(i-1)} \quad \forall i = 1,2,\ldots,S \\
& \quad \sum_{k=1}^{\lambda_{ijk},\text{Max}} \lambda_{ijk} = 1 \quad \forall i = 1,2,\ldots,S \\
& \quad \sum_{j=1}^{n_j,\text{Max}} N_{ij} \geq 1 \quad \forall i = 1,2,\ldots,S \\
& \quad \sum_{j=1}^{n_j,\text{Max}} N_{ij} \leq n_{i,\text{Max}} \quad \forall i = 1,2,\ldots,S \\
& \quad A(\omega) \geq A_0
\end{align*}
\]

Equation (1) is an objective function that minimizes system cost. Equation (2) is the system weight constraint. Equation (3) defines the discount level, i.e., it establishes the value of purchase, at each discount level after determining the purchase value of each component in each subsystem. Equation (4) ensures that the purchase value for component type \( j \) in subsystem \( i \) should be within the discount levels, and Equation (5) implies that there is at least one component in each subsystem. Equation (6) ensures that the number of components in each subsystem does not exceed the maximum number of acceptable components. Finally, Equation (7) specifies the minimum expected availability of the system.

For calculating the system availability (Equation 7), we used recursive algorithm. This algorithm is presented in the next section.

3. Solving methods
3.1. Recursive Algorithm

3.1.1. The weighted multi-state k-out-of-n: G system

In multi-state system, each component of the system may be in different states, and in each state, the component has specific performance. When a component is completely failed, its performance is 0 [13].

Definition 1: In a system with n components, each component of the system may be in one of the possible \((m+1)\) states.

The component \(i\) \((1 \leq i \leq n)\), when placed in state \(j\), has performance \(g_{ij}\). In this case, the system is in the state \(j\) if the sum of the performances of all components of the system is greater than or equal to \(k_j\). Assume \(\phi\) is the system structure function that indicates the state of the system, and \(G\) is the sum of the performances of all the system components. Based on above definition, we have:

\[
Pr\{\phi \geq j\} = Pr\{G \geq k_j\} \quad (8)
\]

Since state 0 is the worst state in the system, we have:

\[
Pr\{\phi \geq 0\} = 1 \quad (9)
\]

3.1.2. Recursive algorithm

To evaluate the reliability of the multi-state weighted k-out-of-n: G system, using the recursive algorithm, we first introduce the following parameters. These parameters are only used in this section.

- \(n\): Number of components in the system,
- \(M\): State with highest possible performance,
- \(g_{ij}\): The performance of component \(i\) in state \(j\),
- \(p_{ij}\): The probability that component \(i\) is in state \(j\),
- \(q_{ij}\): The performance of the component \(i\) when it is in a state lower than \(j\),
Minimum total performance required to ensure that the system is in state $j$ or higher,

$$R_j^I(k_j, n)$$ The probability that the system is in the state $j$ or higher.

Therefore, the recursive equation for evaluating the distribution of the system state is as follows [13]:

$$R_j^I(k_j, i) = \sum_{r=0}^{M} p_{i,r} R_j^I(k_j - g_{i,r}, i - 1) \quad (10)$$

The partial conditions for this recursive equation are as follows:

$$R_j^I(k_j, 0) = 0 \quad \text{when } 0 < k \leq k_j,$$

$$R_j^I(k_j, i) = 1 \quad \text{when } i \geq 0 \text{ and } k \leq 0. \quad (11)$$

For example, consider a weighted multi-state k-out-of-n: G system with three components. Each component has three possible states 0, 1, and 2. Tables 2 and 3 show the reliability and performance of all components.

Insert tables 2 and 3 here

In this example, $n = 3$, $M = 2$, $k_1 = 5$ and $k_2 = 10$. We obtain the reliability of the system based on equations 10 and 11 as follows [13]:

$$R_1^I(5,3) = \sum_{r=0}^{2} p_{3,0} R_1^I(5 - g_{3,r}, 3 - 1) =$$

$$p_{3,0}. R_1^I(5 - 1, 2) + p_{3,1}. R_1^I(5 - 3, 2) +$$

$$p_{3,2}. R_1^I(5 - 5, 2) = p_{3,0}. R_1^I(4, 2) + p_{3,1}. R_1^I(2, 2) +$$

$$p_{3,2}. R_1^I(0, 2) = p_{3,0}. R_1^I(4, 2) + p_{3,1} + p_{3,2} \quad (12)$$
\[
R_i^l(4, 2) = \sum_{r=0}^{2} p_{2, r} \cdot R_i^l(4 - g_{2, r}, 2 - 1) = \\
p_{2, 0} \cdot R_i^l(4 - 1, 1) + p_{2, 1} \cdot R_i^l(4 - 3, 1) + \\
p_{2, 2} \cdot R_i^l(4 - 4, 1) = p_{2, 0} \cdot R_i^l(3, 1) + \\
p_{2, 1} \cdot R_i^l(1, 1) + p_{2, 2} \cdot R_i^l(0, 1) = p_{2, 0} \cdot p_{1, 2} + \\
p_{2, 1} + p_{2, 2} = 0.4 \times 0.7 + 0.2 + 0.4 = 0.88
\]

\[
R_i^l(5, 3) = p_{3, 0} \cdot R_i^l(4, 2) + p_{3, 1} + p_{3, 2} = \\
p_{3, 0} \times 0.88 + p_{3, 1} + p_{3, 2} = \\
0.3 \times 0.88 + 0.5 + 0.2 = 0.964
\]

\[
R_i^l(10, 3) = \sum_{r=0}^{2} p_{3, r} \cdot R_i^l(10 - g_{3, r}, 3 - 1) = \\
p_{3, 0} \cdot R_i^l(10 - 1, 2) + p_{3, 1} \cdot R_i^l(10 - 3, 2) + \\
p_{3, 2} \cdot R_i^l(10 - 5, 2) = p_{3, 0} \cdot R_i^l(9, 2) + \\
p_{3, 1} \cdot R_i^l(7, 2) + p_{3, 2} \cdot R_i^l(5, 2)
\]

\[
R_i^l(5, 2) = \sum_{r=0}^{2} p_{2, r} \cdot R_i^l(5 - w_{2, r}, 2 - 1) = \\
p_{2, 0} \cdot R_i^l(5 - 1, 1) + p_{2, 1} \cdot R_i^l(5 - 3, 1) + \\
p_{2, 2} \cdot R_i^l(5 - 4, 1) = p_{2, 0} \cdot R_i^l(4, 1) + \\
p_{2, 1} \cdot R_i^l(2, 1) + p_{2, 2} \cdot R_i^l(1, 1) = p_{2, 0} \cdot 0 + \\
p_{2, 1} \cdot (p_{1, 1} + p_{1, 2}) + p_{2, 2} = 0.2 \times (0.2 + 0.7) + 0.4 = 0.58
\]

\[
R_2(7, 2) = p_{1, 2} \cdot p_{2, 2} = 0.7 \times 0.4
\]

\[
R_2(9, 2) = 0,
\]

\[
R_2(10, 3) = p_{3, 1} \cdot 0.28 + p_{3, 2} \cdot 0.58 = \\
0.5 \times 0.28 + 0.2 \times 0.58 = 0.256.
\]

Therefore, the distribution of the system state will be as follows [13]:

\[
\Pr(\phi \geq 0) = 1,
\]

\[
\Pr(\phi \geq 1) = 0.964,
\]

\[
\Pr(\phi \geq 2) = 0.256,
\]

\[
\Pr(\phi = 2) = 0.256,
\]

\[
\Pr(\phi = 1) = 0.964 - 0.256 = 0.708,
\]
Pr(\(\phi \geq 0\)) = 1 - 0.964 = 0.036.

3.2. Genetic Algorithm
In 1975, this algorithm was first introduced by Holland et al. [30] at Michigan University and developed by him and his students. The original idea of this algorithm was derived from Darwinian evolutionary theory in 1895. According to this theory, those creatures survive that are more adaptable to the environment. Information transmitted from each generation to the next generation is enclosed in chromosomes and inherited properties are transmitted in this way. In this algorithm, according to the principle of survival of the best, the better population are combined together, and depending on the suitability of each solution, this solution will be repeated more often in the next generation, and this process will continue to reach the optimal solution.

3.2.1. Algorithm Steps
Step 1: Generate random population including \(n\) chromosome or initial solution,
Step 2: Evaluate the fitness function of each population chromosome,
Step 3: Create a new population based on the following steps:
  • Selection of parent chromosomes by selective methods such as roulette wheel, tournament, randomly, competitive and so on by crossover and mutation operators,
  • Considering a certain value for the probability of crossover operator and then doing a combination operation on parents to create offspring,
  • Considering a certain value for the probability of mutation operator and then using this operation to change one or more genes from a parent chromosome to achieve a new chromosome,
Step 4: Replacing new offspring’s in the new population.

3.2.2. Solution Encoding
The problem chromosome is an \(n_{S \times M}\) matrix presented in Figure 1. In this matrix, \(S\) is the number of subsystems and \(M\) is the maximum type of components. These chromosomes are presented by Tavakkoli-Moghaddam et al. [9].
Assume that the model has three subsystems and four different component types, and in subsystem 1, we have 2 components of type 1, 3 components of type 3 and 1 component of type 4, in subsystem 2, we have 3 components of type 1, 3 components of type 2 and 1 component of type 3 and in subsystem 3, we have 4 components of type 1, 2 components of type 2 and 4 components of type 3; the chromosome matrix of this solution is presented in Figure 2.

3.2.3. Initial population
The initial population is produced randomly and we refer to this initial population by $npop$.

3.2.4. Fitness function
Because of the model constraints, it is possible that the produced chromosome is not feasible. Therefore, the most important problem in using GA for problems with constraints is how to deal with constraints. Penalty functions are one of the first methods to deal with problems with constraints in GA. The penalty functions reduce infeasible solutions, in accordance with the violation ratio of the constraints. In fact, the penalty function turns constrained problems into problems without constraints. Because of the problem constraints, the penalty functions are:

$$ p_1 = \max \{ A_0 - A(\omega), 0 \} $$

(19)

$$ p_2 = \frac{\max \left\{ \sum_{j=1}^{m_{Max}} \sum_{j=1}^{m_{Max}} w_{ij} N_{ij} - W, 0 \right\}}{W} $$

(20)

$$ p_2 = \sum_{i=1}^{S} \left\{ \max \left\{ \sum_{j=1}^{m_{Max}} N_{ij} - n_{i,Max}, 0 \right\} \right\} $$

(21)

Therefore, the general penalty function of the problem is:
\[ P_{Total} = p_1 + p_2 + p_3 \]  \hspace{1cm} (22)

And the fitness function of the model is as follows:
\[ F(x) = f(x).(1 + p_T) \]  \hspace{1cm} (23)

Now, if the equality is satisfied, the value of \( p_T \) is 0 and the fitness function is the same as the objective function.

3.2.5. Crossover operator

In this operator, first, the number of parents is calculated with crossover rate and then the parents are randomly selected using the roulette wheel. To perform the crossover operator, firstly select the parent and then the offspring are created using a uniform crossover operator. The operation of this operator is described in [10]. Intersection operations are performed on parent’s chromosomes so that offspring’s chromosomes are formed. In this operator, for each genome in the parent chromosome, a binary number is randomly generated, which if this number is 1, the genome is replaced in the parent’s chromosomes, and if the number is 0, it is not replaced. The crossover operator in the proposed GA is shown in Figure 3.

Insert figure 3 here

3.2.6. Mutation Operator

In this operator, first, the number of parents is calculated with the mutation rate, and then the parents are randomly selected using the roulette wheel. After selecting the parent, for each genome in the parent chromosome, a random number is generated between 0 and 1 and mutations are performed at a specific mutation rate of the parent chromosome genes. Now, if the generated random number is smaller than the desired mutation rate, the genome in parent chromosome is randomly mutated. If the generated random number is larger than the mutation rate, the gene in the parent chromosome is not mutated [10]. This type of mutation is illustrated in Figure 4. In this Figure, the mutation rate is considered as \( p_M = 0.1 \).
3.2.7. Selection
In this paper, roulette wheel was used to select the population of the next generation. There are two main ideas in this way. First, better chromosomes have better chances of selecting, and secondly, the chances of selecting each chromosome are proportional to their fitness. For each chromosome, the fitness function is calculated and then the cumulative fitness function of the chromosomes is computed. Next a random number is generated between 0 and the cumulative fitness function of the last chromosome. The corresponding number is compared with cumulative fitness function and the chromosome located at the corresponding distance is selected. The implementation of this method is shown in Figure 5.

3.2.8. Stop Criteria
There are many criteria to stop the algorithm, such as the number of algorithms iterations, the improvement of objective function, and so on. In this algorithm, the number of algorithms iteration has been used. This means that this algorithm stops after a certain number of iterations and generation. The algorithm iteration is shown by \( MaxIt \).

3.3. Parameter Tuning
The time of the meta-heuristic algorithms depends on their input parameters. The goal of tuning the algorithms parameters is that the algorithm reaches appropriate solutions at a better time. The method of parameters tuning of the proposed algorithm is as follows. Proposed GA input parameters are population size \( (n_{\text{pop}}) \), intersection crossover probability \( (P_c) \), and probability of mutation \( (P_m) \). RSM (Response Surface Methodology) was used to identify the appropriate values of parameters. We used two-level factorial design method for tuning the algorithm parameters. For each experiment, two levels are considered that are high and low. In addition to the upper and lower limits, axial points as well as a number of central points (herein are 5 central points) are also considered. In this model, considering the three existing parameters, the \( 2^3 \) factorial design is considered. Meanwhile, the stop criterion for parameter tuning is equal to 100
algorithm iterations and the response variable in the model is the system reliability. The input values in addition to the optimal value of each parameter, are presented in Table 4.

4. Numerical example

In this section, we intend to validate and solve the proposed model. For this purpose, we first design a numerical example. It should be noted that the numerical examples used in this paper are taken from [16]. It is assumed that the system consists of 14 subsystems and three price levels to buy components. The maximum acceptable weight for the system is 100, and the maximum number of components in each subsystem is considered 6, and the minimum acceptable availability (reliability) for the system is 0.9 and the minimum acceptable performance rate for the components is 50. It is also assumed that there are four types of components available to allocate in the subsystems.

In Table 5, the values of reliability, weight, and cost for each component are presented in all subsystems at different failure levels. Table 6 shows the performance rates of each component when placed in each of the subsystems, and Table 7 shows the probability that the components will be matched to the performance rates of each of the subsystems when placed in each of the subsystems.

Now, in order to confirm the correct operation of the GA, we solve some small size problems using the precise method of numerical rule, and compare the obtained solutions and its solving time with the solutions and the solving time obtained from the GA. After ensuring that the proposed GA is validated in solving the proposed model, we will solve the proposed model with large size problems with GA.

In this way, we first assume that the problem has five subsystems and then consider a problem with six subsystems. More precisely, it is assumed that the system, once only from the first to fifth subsystems, and once only from the first to sixth subsystems, with the same amount of parameters as in tables 2 to 4. In addition, there are two types of components available to allocate in the subsystems.
The total number of problem solutions is obtained according to the numerical rule method of the following equation:

\[(n_{\text{max}} + 1)^{(S \times T)}\]  

(24)

We also assume that the maximum number of acceptable components in these problems is 2, the maximum acceptable weight is 60, the minimum acceptable components performance rate is 30, and the minimum acceptable reliability for the system is 0.2.

So, according to the Equation (24), the total number of available solutions for the first problem is 59049 and for the second problem is 531441 and the highest amount obtained in these repetitions is chosen as the optimal solution for each problem.

The enumeration method and the proposed GA are programmed for the two mentioned problems in MATLAB 17, and the best solution and the solving time for two problems are presented in Table 8.

Since using the enumeration method, increasing the number of subsystems or the number of components types increases the feasible solutions of the problem, and this causes the enumeration method unable to generate the optimal solutions at the acceptable time for the problem. As we saw, the values of the solutions were the same for both methods of solving two problems, which indicates the validity of the suggested GA for the proposed model. But as the size of the problem enlarges, the solving time of the problem using enumeration method becomes too long and the enumeration method is not a suitable method for solving the problem. Therefore, considering that the proposed GA in small size problems has achieved the optimal solution, it can be used to solve larger size problems.

In order to sensitive analysis, we solved 15 new problems. Assuming that the reliability of the component type 1 in the first subsystem varies from 0.82 to 0.96 and the other parameters of the problem are in accordance with tables 4 to 6. The time and cost of these 15 problem are presented in Table 9.
Also, the optimal solution of the problem number 15 is presented in Figure 6 and the convergence diagram of GA is presented in Figure 7.

Insert figures 6 and 7 here

5. Conclusion and further studies

5.1. Conclusion

In this paper, the reliability of a multi-state RAP was investigated using a recursive method in which there were discounted levels for the purchasing of its components. A single-objective cost optimization model was investigated under various constraints, including reliability, and since the RAP belongs to NP-Hard problems, a GA was used to solve the model. Also, to validate the proposed model, some small size problems were solved using an enumeration method and GA and it was shown that the GA could reach the optimal solution. Finally, a GA was used to solve 15 large-scale problems.

5.2. Further studies

- Provide multi-objective models for solving real-world problems by considering objectives such as discount levels, weight, volume, etc. in the proposed model.
- Consider the incremental discounts instead of all units’ discount in the presented model.
- Consider the problem parameters such as reliability, cost, etc. as probabilistic parameters.
- Consider the time-dependent failure rate rather than constant failure rate.
- Consider repairable components and limits for the number of repairmen to bring the problem closer to real-world conditions.
- …

Acknowledgements

This research was supported by a research project granted by Qazvin branch, Islamic Azad University. We would appreciate the full support of the Deputy of Research of Qazvin branch, Islamic Azad University for their invaluable assistance.

6. References
Mani Sharifi was born in 1973 in Tehran, Iran. Mani is an associate professor at Qazvin Islamic Azad University (QIAU). Mani holds a B.Sc. degree from Qazvin Islamic Azad University in 1996, M.Sc. degree from south Tehran branch Islamic Azad University in 1998, and Ph.D. degree from Tehran Research and Science Islamic Azad University in industrial engineering filed in 2006. He is a member of the industrial engineering department at QIAU. He was the dean of all research centers of QIAU and the Managerial editor of “Journal of Optimization in Industrial Engineering”. His area of interest includes reliability engineering, combinatorial optimization, statistical optimization and fuzzy set theory. He has published a number of papers in journals such as Reliability Engineering and Safety System, Computers and Industrial Engineering and among others.

Majid Saadvandi received M.S. degree in industrial engineering from Islamic Azad University (IAU), Science and Research Branch, Qazvin, Iran, in 2016. His research interests include reliability redundancy allocation problem and fuzzy mathematics.

Mohammadreza Shahriari received his Ph.D. in Operational Research from the Islamic Azad University in 2009. He worked out more than 2 years to find out he how can present and solve a Time-Cost Trade off problem based on time value of money, considering to each crashing. He was also six years in charge of chancellor of Islamic Azad University in Dubai (UAE) and more than one year at the same position in Islamic Azad University in Oxford (UK). He has had another executive academic mandate, like, establishing an Irani University in Germany, Italy, Russia and Oman. After almost running the abovementioned mandate, he again backed to his academic and teaching activities in the university. Now he is busy with teaching of OR (1, 2 and 3) and advanced OR and Reliability in Bachelor and Master level and teaching the innovative field in of Decision Theory. Now, he is Associate Professor of Azad University and is busy with some Ph.D. and master dissertation in field of decision theory and reliability theory.
Table 1. Some of recent studies on reliability area.

Table 2. Components states probabilities \( (p_{ij}) \) [1]

Table 3. Components states performance \( (g_{ij}) \) [1]

Table 4. GA optimal values.

Table 5. Components reliability, cost, and weight

Table 6. Components performance rate

Table 7. Components performance rate correspondence probabilities

Table 8. The cost and time calculated by the numerical rule and the GA

Table 9. The reliability and the cost of 15 solved problem

Figure 1. Model Chromosome

Figure 2. Sample chromosome.

Figure 3. Crossover operator.

Figure 4. Mutation operator.

Figure 5. Roulette wheel method.

Figure 6. Optimal solution of the problem number 15.

Figure 7. Convergence diagram of GA for problem no. 15.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Year</th>
<th>State</th>
<th>Algorithm</th>
<th>Repairable</th>
<th>Objective</th>
<th>Parameter setting</th>
<th>Failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garg et al. [17]</td>
<td>2013</td>
<td>Binary</td>
<td>Bee colony</td>
<td>-</td>
<td>Single</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Levitin et al. [18]</td>
<td>2013</td>
<td>Multi-state</td>
<td>GA</td>
<td>-</td>
<td>Single</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Maatouk et al. [19]</td>
<td>2013</td>
<td>Multi-state</td>
<td>GA</td>
<td>✓</td>
<td>Single</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Ebrahimipour et al. [21]</td>
<td>2013</td>
<td>Binary</td>
<td>Fuzzy inference</td>
<td>-</td>
<td>Single</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Authors</td>
<td>Year</td>
<td>State</td>
<td>Type</td>
<td>Repair</td>
<td>Number</td>
<td>Constant</td>
<td>Method</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>---------</td>
<td>-----------------------</td>
<td>----------</td>
<td>--------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>Liu et al. [22]</td>
<td>2013</td>
<td>Multi-state</td>
<td>Imperfect repair</td>
<td>✓</td>
<td>Single</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Khalili-Damghani et al. [23]</td>
<td>2014</td>
<td>Binary</td>
<td>e-constraint</td>
<td>-</td>
<td>Multiple</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Sharifi et al. [24]</td>
<td>2015</td>
<td>Multi-state</td>
<td>Markov model</td>
<td>-</td>
<td>Single</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Mousavi et al. [25]</td>
<td>2015</td>
<td>Multi-state</td>
<td>CE-NRGA</td>
<td>-</td>
<td>Multiple</td>
<td>Taguchi</td>
<td>Constant</td>
</tr>
<tr>
<td>Zaretalab et al. [12]</td>
<td>2015</td>
<td>Multi-state</td>
<td>MOSA</td>
<td>-</td>
<td>Multiple</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Miriha et al. [26]</td>
<td>2017</td>
<td>Binary</td>
<td>NSGA-II MOEA/D</td>
<td>-</td>
<td>Multiple</td>
<td>Taguchi</td>
<td>Time dependent</td>
</tr>
<tr>
<td>Guilani et al. [27]</td>
<td>2017</td>
<td>Multi-state</td>
<td>SPEA-II NSGA-II</td>
<td>-</td>
<td>Multiple</td>
<td>No</td>
<td>Constant</td>
</tr>
<tr>
<td>Hadipour et al. [28]</td>
<td>2018</td>
<td>Binary</td>
<td>NSGA-II NRGA</td>
<td>✓</td>
<td>Single</td>
<td>Taguchi</td>
<td>Constant</td>
</tr>
<tr>
<td>Guilani et al. [29]</td>
<td>2018</td>
<td>Binary</td>
<td>Simulation</td>
<td>-</td>
<td>Single</td>
<td>-</td>
<td>Time dependent</td>
</tr>
</tbody>
</table>

Table 2. Components states probabilities ($p_{ij}$)

\[
\begin{array}{ccc}
  j = 0 & j = 1 & j = 2 \\
  \hline
  i = 0 & 0.1 & 0.2 & 0.7 \\
  i = 1 & 0.4 & 0.2 & 0.4 \\
  i = 2 & 0.3 & 0.5 & 0.2 \\
\end{array}
\]

Table 3. Components states performance ($g_{ij}$)

\[
\begin{array}{ccc}
  j = 0 & j = 1 & j = 2 \\
  \hline
  i = 0 & 1 & 2 & 3 \\
  i = 1 & 1 & 3 & 4 \\
  i = 2 & 1 & 3 & 5 \\
\end{array}
\]

```
\[
\begin{bmatrix}
  N_{11} & N_{21} & \cdots & N_{M1} \\
  N_{11} & N_{22} & \cdots & N_{M2} \\
  \vdots & \vdots & \ddots & \vdots \\
  N_{1S} & N_{2T} & \cdots & N_{MT} \\
\end{bmatrix}
\]
```

Figure 1. Model Chromosome
Figure 2. Sample chromosome.

Figure 3. Crossover operator.

Figure 4. Mutation operator.

Figure 5. Roulette wheel method.

Table 4. GA optimal values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Optimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>npop</td>
<td>50</td>
<td>100</td>
<td>75</td>
</tr>
<tr>
<td>$p_c$</td>
<td>0.4</td>
<td>0.7</td>
<td>0.55</td>
</tr>
<tr>
<td>$p_m$</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 5. Components reliability, cost, and weight

<table>
<thead>
<tr>
<th>Components Type</th>
<th>Parameters</th>
<th>Number of sub-systems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>R</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>Component type 1</td>
<td>Component type 2</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Subsystems</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>C_1</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>C_2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>C_3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>n_1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n_2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>R</td>
<td>0.93</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 6. Components performance rate
Table 7. Components performance rate correspondence probabilities

<table>
<thead>
<tr>
<th>Subsystems</th>
<th>Component type 1</th>
<th>Component type 2</th>
<th>Component type 3</th>
<th>Component type 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>0.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td>0.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 8. The cost and time calculated by the numerical rule and the GA

<table>
<thead>
<tr>
<th>Solving Method</th>
<th>First Problem</th>
<th>Second problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost</td>
<td>Time, s</td>
</tr>
<tr>
<td>Enumeration method</td>
<td>34</td>
<td>66.949</td>
</tr>
<tr>
<td>GA</td>
<td>34</td>
<td>5.379</td>
</tr>
</tbody>
</table>
Table 9. The reliability and the cost of 15 solved problem

<table>
<thead>
<tr>
<th>Problem</th>
<th>Reliability of component type 1 in sub-system 1</th>
<th>System cost</th>
<th>Solving time (Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.82</td>
<td>282</td>
<td>17.21</td>
</tr>
<tr>
<td>2</td>
<td>0.83</td>
<td>299</td>
<td>17.44</td>
</tr>
<tr>
<td>3</td>
<td>0.84</td>
<td>287</td>
<td>16.46</td>
</tr>
<tr>
<td>4</td>
<td>0.85</td>
<td>306</td>
<td>17.00</td>
</tr>
<tr>
<td>5</td>
<td>0.86</td>
<td>301</td>
<td>16.51</td>
</tr>
<tr>
<td>6</td>
<td>0.87</td>
<td>285</td>
<td>17.48</td>
</tr>
<tr>
<td>7</td>
<td>0.88</td>
<td>288</td>
<td>17.30</td>
</tr>
<tr>
<td>8</td>
<td>0.89</td>
<td>291</td>
<td>16.97</td>
</tr>
<tr>
<td>9</td>
<td>0.90</td>
<td>287</td>
<td>17.03</td>
</tr>
<tr>
<td>10</td>
<td>0.91</td>
<td>284</td>
<td>17.05</td>
</tr>
<tr>
<td>11</td>
<td>0.92</td>
<td>290</td>
<td>17.37</td>
</tr>
<tr>
<td>12</td>
<td>0.93</td>
<td>289</td>
<td>16.97</td>
</tr>
<tr>
<td>13</td>
<td>0.94</td>
<td>289</td>
<td>17.18</td>
</tr>
<tr>
<td>14</td>
<td>0.95</td>
<td>292</td>
<td>16.12</td>
</tr>
<tr>
<td>15</td>
<td>0.96</td>
<td>290</td>
<td>17.79</td>
</tr>
</tbody>
</table>

Figure 6. Optimal solution of the problem number 15.
Figure 7. Convergence diagram of GA for problem no. 15.