Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Reliability–redundancy allocation problem with cold-standby redundancy strategy

Mostafa Abouei Ardakan [*], Ali Zeinal Hamadani

Department of Industrial and Systems Engineering, Isfahan University of Technology, 84156-83111 Isfahan, Iran

ARTICLE INFO

Article history:
Received 18 September 2013
Received in revised form 10 December 2013
Accepted 15 December 2013
Available online 21 January 2014

Keywords:
Reliability optimization
Reliability–redundancy allocation problem
Cold-standby redundancy strategy
Genetic algorithm

ABSTRACT

This paper considers the mixed-integer non-linear optimization of reliability–redundancy allocation problem (RRAP) to determine simultaneous reliability and redundancy level of components. In the RRAP, it is necessary to create a trade-off between component reliabilities and the number of redundant components with the aim of maximizing system reliability through component reliability choices and component redundancy levels. RRAPs have been generally formulated by considering an active redundancy strategy. A large number of solution methods have been developed to deal with these problems. In this paper, a cold-standby strategy for redundant components is used, for the first time, to model the RRAP; a modified genetic algorithm is developed to solve the proposed non-linear mixed-integer problem; and three famous benchmark problems are used for comparison. The results indicate that the cold-standby strategy exhibits a better performance and yields higher reliability values compared to the previous studies.

1. Introduction

High-reliability systems play crucial roles in modern industry. System reliability may be enhanced by: (a) raising component reliability, (b) providing redundant components in parallel, (c) using a combination of enhanced component reliability and redundant components provisioned in parallel, and (d) reassigning interchangeable components [1]. The second and third options are called redundancy allocation problem (RAP) and reliability–redundancy allocation problem (RRAP), respectively. In RAPs, there are discrete component choices with known characteristics such as reliability, cost, and weight, where the aim is to find the optimal/near optimal number of redundancies in each subsystem in order to maximize the overall system reliability subject to some constraints. The reliability–redundancy allocation problem (RRAP) is the problem of maximizing system reliability through component reliability choices and component redundancy, which forms a difficult but realistic optimization problem in which component reliability is not given but treated as a design variable while component cost, weight, volume, etc. are defined in advance as increasing non-linear functions of component reliability [2].

In reliability studies, either of two different strategies, called active and standby, may be considered for determining how the redundant components must be used. In the active strategy, all redundant components simultaneously start to operate from time zero although only one is required at any particular time. The standby redundancy may take one of the three cold, warm, or hot variant forms. In the cold variant, the redundant components are protected from operational stresses associated with system operation so that no component fails before its start. The components in the warm-standby redundancy are affected by operational stresses more than those in the cold variant. Finally, in the hot-standby redundancy, component failure does not depend on whether the component is idle or in operation. The mathematical formulation for the hot-standby

* Corresponding author. Tel.: +98 9131564886.
E-mail address: m.abouei@in.iut.ac.ir (M. Abouei Ardakan).

1569-190X/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2013.12.013

strategy is the same as that with the active redundancy case. In the standby redundancy strategy, the redundant components are sequentially used in the system at failure times of operating components by switching to one of the redundant components in order to continue system operation [3,4].

Activation of a standby redundant component in case of online component failure in standby strategies requires a switching system that is based on one of two scenarios. In the first scenario ($S1$), the failure detection and switching hardware/software is continually monitoring system performance to detect a failure and to activate the redundant component. It is assumed that switch failure can occur at any time and that switch reliability is a non-increasing function of time ($\rho_i(t)$) which does not depend on the number of switching required. In the second scenario ($S2$), failure of switching will happen with a constant probability, (ρ_i), when the switch is required [5].

Reliability–redundancy allocation problems (RRAP) have been generally formulated by considering the active redundancy strategy and using different solution methods. For this purpose, exact optimization methods such as dynamic programming [6], branch-and-bound approach [6,7], and implicit enumeration [8] have been used to maximize system reliability. Hikita et al. [9] developed a surrogate constraint method to solve the RRAP. They developed a dynamic programming to solve single-constrained surrogate problems. However, this method requires that either the objective function be separable or the surrogate problem be formulated as a multi-stage decision making problem. Their approach is only useful for special structures which include parallel-series and series–parallel designs [10].

Since RAP and RRAP have been proved to be NP-hard optimization problems [11,12], they are too difficult and time-consuming to solve using traditional optimization methods, especially when the problem size is large. For this reason, numerous meta-heuristic algorithms such as genetic algorithm [4,13–17], Tabu search [18,19], ant colony optimization [20–23], artificial immune system [10,24,25], artificial neural networks [26], artificial bee colony algorithm [27,28], particle swarm optimization [29], Memetic algorithm [30] and a combination of these algorithms [31] have been widely employed over the past decade for solving reliability optimization problems.

Recently, Yeh and Hsieh [28] developed a penalty guided artificial bee colony algorithm (ABC) for solving the reliability–redundancy allocation problems. In order to improve the solutions, they also proposed a local search for their ABC algorithm. Hsieh and You [10] developed a two-phase approach based on immune algorithm to solve the same problem. According to this approach, an immune algorithm (IA) is developed in the first phase to solve the problem, followed by a new procedure in the second phase to improve the solutions. Wu et al. [32] presented an improved particle swarm optimization (IPSO) algorithm for the RRAP. Zou et al. [33] developed an effective global harmony search algorithm (EGHS) to solve the RRAP. The EGHS algorithm combined the harmony search algorithm (HS) with the concepts of swarm intelligence in particle swarm optimization (PSO) algorithm to solve the problem. Wang and Li [34] proposed a differential evolution (DE) algorithm combined with a harmony search (HS) algorithm.

Valian and Valian [35] proposed a cuckoo search (CS) algorithm and used some well-known benchmark problems to test the performance of the algorithms in solving the reliability redundancy-allocation problem. In a different study, Valian et al. [36] presented an improved cuckoo search algorithm by enhancing the accuracy and convergence rate of the cuckoo search algorithm. Afonso et al. [37] proposed a modified version of the imperialist competitive algorithm (ICA) and compared the results obtained from their proposed ICA with the best-known results of different benchmarks reported in the literature to demonstrate the capability of their proposed algorithm.

As it is clear from the literature, all the previous studies of the reliability–redundancy allocation problem have considered the active strategy for redundant components and their efforts have been directed at improving system reliability by developing novel, modified, or combined meta-heuristic algorithms. In the present study, however, the cold-standby strategy is used for the first time and a new mathematical formulation is presented for the problem under the non-linear constraints of weight, cost, and volume.

The rest of the paper is organized as follows. In Section 2, the formulation of the RRAP, the mathematical formulation of the cold-standby strategy, and three benchmark problems are presented. Section 3 presents the GA developed for solving the proposed non-linear models. Section 4 considers the experimental results to demonstrate the advantages of the cold-standby strategy and the efficiency of the proposed methodology. Finally, conclusions are presented in Section 5.

2. Formulation of the reliability–redundancy allocation problem

The objective of reliability optimization is to improve system reliability. The reliability–redundancy allocation problems are useful for system designs that are largely assembled and manufactured using off-the-shelf components, and that have high reliability requirements [29]. A reliability–redundancy allocation problem is generally formulated with the objective of maximizing system reliability under some non-linear constraints. Therefore, in this paper, the RRAP is considered with the objective of maximizing system reliability subject to the multiple non-linear constraints of weight, cost, and volume. The mixed-integer non-linear programming model of RRAP is generally formulated as follows:

$$\text{Maximize } R_s = f(r, n) \tag{1}$$

$$\text{subject to } g_j(r, n) \leqslant l_j \quad j = 1, 2, \ldots, k \tag{2}$$

$$0 \leqslant r_i \leqslant 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \leqslant i \leqslant m$$

where R_s is the reliability of the system; g_j is the jth constraint; $r = (r_1, r_2, r_3, \ldots, r_m)$ is the vector of the component reliabilities for the system; $n = (n_1, n_2, n_3, \ldots, n_m)$ is the vector of the redundancy allocation for the system; r_i and n_i are the reliability and the number of components in the ith subsystem, respectively; $f(\cdot)$ is the objective function for the overall system reliability; l is the resource limitation vector; and m is the number of subsystems in the system. The goal is to determine the number of components and each component's reliability in each subsystem in order to maximize the reliability of the whole system. The problem belongs to the category of constrained mixed-integer non-linear optimization problems.

2.1. System reliability with cold-standby redundancy strategy

Coit [5] presents a mathematical formulation as in Eq. (3) for the reliability of a subsystem with cold-standby strategy and perfect switching. This equation is a general form that is appropriate for any distribution of component time-to-failure.

$$R_i(t) = r_i(t) + \sum_{x=1}^{n_i-1} \int_0^t r_i(t-u) f_i^{(x)}(u)\,du \tag{3}$$

where $r_i(t)$ is the reliability at time t for the component used in subsystem i; n_i is the number of components in subsystem i, and $f_i^{(x)}(t)$ is the pdf for the xth failure arrival for subsystem i, i.e. the sum of x iid component failure times.

In order to determine the reliability of the subsystems with imperfect switching, $R_i(t)$, Eqs. (4) and (5) are presented for two scenarios as follows [5]:

Scenario 1 ($S1$): Continuous detection and switching:

$$R_i(t) = r_i(t) + \sum_{x=1}^{n_i-1} \int_0^t \rho_i(u) r_i(t-u) f_i^{(x)}(u)\,du \tag{4}$$

Scenario 2 ($S2$): Switch activation only in response to a failure:

$$R_i(t) = r_i(t) + \sum_{x=1}^{n_i-1} \rho_i^x \int_0^t r_i(t-u) f_i^{(x)}(u)\,du \tag{5}$$

where $\rho_i(t)$ and ρ_i are failure-detection/switching reliabilities at time t for $S1$ and $S2$, respectively. This paper only investigates the continuous detection and switching ($S1$). As mentioned by Coit [5], it is difficult to determine a closed form for equations similar to Eq. (4). Therefore, a convenient lower bound on subsystem reliability, $\widetilde{R}_i(t)$, can be determined as follows:

$$\widetilde{R}_i(t) = r_i(t) + \rho_i(t) \sum_{x=1}^{n_i-1} \int_0^t r_i(t-u) f_i^{(x)}(u)\,du \tag{6}$$

This is an estimation for Eq. (4) because $\rho_i(t) \leqslant \rho_i(u)$ for all $u \leqslant t$.

Coit [5] has developed these mathematical models for calculating the standby strategy in a general form. He also used Eq. (6) with Erlang distribution for the redundancy allocation problem. However, as all the non-linear constraints in the reliability–redundancy allocation problem are derived by considering the exponential time-to-failure [38], in this paper Eq. (6) has been extended based on the exponential distribution and implemented in this problem. If the component time-to-failure is exponential, then Eq. (6) can be represented by considering the occurrences of subsystem failures as a homogeneous Poisson process prior to the n_ith failure. In this case, the reliability of the subsystem is the probability that there are strictly less than n_i failures, which is Poisson distributed. Therefore,

$$\int_0^t r_i(t-u) f_i^{(x)}(u)\,du = \frac{e^{-\lambda_i t} (\lambda_i t)^x}{x!} \tag{7}$$

and

$$\widetilde{R}_i(t) = r_i(t) + \rho_i(t) \sum_{x=1}^{n_i-1} \frac{e^{-\lambda_i t} (\lambda_i t)^x}{x!} \tag{8}$$

where λ_i is the component failure rate (the exponential distribution parameter) and t is the mission time. Unlike in all previous studies, r_i is not a decision variable in the present study because the component failure rate needs to be known in the cold-standby strategy for the component reliability, r_i, to be easily calculated. Therefore, λ_i and n_i are two decision variables in the model and r_i is calculated on the basis of λ_i. In the next subsections, three famous benchmarks in the reliability–redundancy optimization are presented.

2.2. Problem 1 (P1): Series system

The first problem ($P1$) is a mixed-integer non-linear programming problem for a series system with five subsystems. The problem formulation is as follows:

$$Maximize\ f(r,n) = \prod_{i=1}^{m} R_i(n_i,t) \tag{9}$$

$$Subject\ to:$$

$$g_1(r,n) = \sum_{i=1}^{m} w_i \cdot v_i^2 \cdot n_i^2 \leqslant V \tag{10}$$

$$g_2(r,n) = \sum_{i=1}^{m} \alpha_i \cdot \left(-\frac{1000}{\ln r_i}\right)^{\beta_i} \cdot [n_i + e^{0.25 n_i}] \leqslant C \tag{11}$$

$$g_3(r,n) = \sum_{i=1}^{m} w_i \cdot n_i \cdot e^{0.25 n_i} \leqslant W \tag{12}$$

$$0 \leqslant r_i \leqslant 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \leqslant i \leqslant m \tag{13}$$

Fig. 1 shows this series system. The same problem was considered by [9,10,25,28,32,35–37,39,40].

In Eq. (9), the objective function contains the redundancy level and the reliability of the component used in each subsystem to achieve maximum system reliability. $R_i(n_i, t)$ in (9) is calculated based on Eq. (8). Eqs. (10)–(12) consider the limitations on system volume, cost, and weight, respectively. The constraint given by (10) is a combination of weight, redundancy allocation, and volume; (11) is a cost constraint; and (12) is a weight constraint [37]. In this context, V is the upper limit of the sum of the subsystem's product of volume and weight, C is the upper limit on the cost of the system, and W is the upper limit on the weight of the system. Furthermore, w_i is the weight of each component in subsystem i, v_i is the volume of each component in subsystem i, and c_i is the cost of each component in subsystem i; $n_i \in Z^+$, where Z^+ is the discrete space of positive integers, and $0 \leqslant r_i \leqslant 1$, $r_i \in R$, $1 \leqslant i \leqslant m$. The parameters β_i and α_i are physical features of the ith system component.

2.3. Problem 2 (P2): Series–parallel system

The second problem ($P2$) is a mixed-integer non-linear programming problem for a series–parallel system with five subsystems. The problem formulation is as follows:

$$Maximize\ f(r,n) = 1 - (1 - R_1(t) \cdot R_2(t))(1 - (R_3(t) + R_4(t) - R_3(t) \cdot R_4(t)) \cdot R_5(t)) \tag{14}$$

$$Subject\ to:$$

$$g_1(r,n) \leqslant V; \quad g_2(r,n) \leqslant C; \quad g_3(r,n) \leqslant W$$

$$0 \leqslant r_i \leqslant 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \leqslant i \leqslant m$$

where R_i is calculated based on Eq. (8) and the condition on the variables are the same as those in $P1$. Fig. 2 shows this series–parallel system. This same problem was considered by [9,10,25,28,32,34–37,40]. It need be mentioned that the reliability function for this system as $f(r,n) = 1 - (1 - R_1R_2)(1 - (1 - R_3)(1 - R_4)R_5)$ reported by Chen [25] and Afonso et al. [37] is wrong, and that the right problem formulation is as in Eq. (14) [32].

2.4. Problem 3 (P3): Complex (bridge) system

The third test problem ($P3$) is a mixed-integer non-linear programming problem for a complex bridge structure with five subsystems. The problem formulation is as follows:

$$\begin{aligned} Maximize\ f(r,n) = {} & R_1(t) \cdot R_2(t) + R_3(t) \cdot R_4(t) + R_1(t) \cdot R_4(t) \cdot R_5(t) \cdot + R_2(t) \cdot R_3(t) \cdot R_5(t) \\ & - R_1(t) \cdot R_2(t) \cdot R_3(t) \cdot R_4(t) - R_1(t) \cdot R_2(t) \cdot R_3(t) \cdot R_5(t) - R_1(t) \cdot R_2(t) \cdot R_4(t) \cdot R_5(t) \\ & - R_1(t) \cdot R_3(t) \cdot R_4(t) \cdot R_5(t) - R_2(t) \cdot R_3(t) \cdot R_4(t) \cdot R_5(t) + 2R_1(t) \cdot R_2(t) \cdot R_3(t) \cdot R_4(t) \cdot R_5(t) \end{aligned} \tag{15}$$

$$Subject\ to:$$

$$g_1(r,n) \leqslant V; \quad g_2(r,n) \leqslant C; \quad g_3(r,n) \leqslant W$$

$$0 \leqslant r_i \leqslant 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \leqslant i \leqslant m$$

where R_i is calculated based on Eq. (8) and the conditions on the variables are the same as those in $P1$. Fig. 3 shows this complex system. This same problem was also considered by [9,10,25,29,32–37,40].

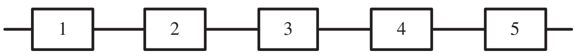


Fig. 1. Series system (test problem 1).

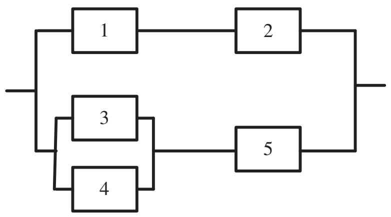


Fig. 2. Series–parallel system (test problem 2).

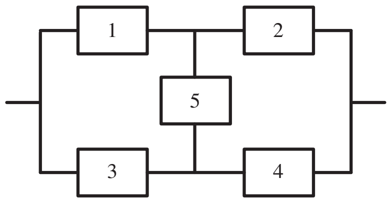


Fig. 3. Complex bridge system (test problem 3).

3. Solution method: Genetic algorithm

Genetic algorithm, as one of the most effective and powerful evolutionary algorithms, was first introduced by Holland [41]. In this paper, a modified version of this algorithm is developed to solve the proposed reliability–redundancy allocation problems. To use the genetic algorithm in its best form, features of the algorithm should be initially designed based on the problem traits. Therefore, the following subsections will describe the proper design of the genetic algorithm to be implemented in the proposed model. These features include the chromosome structure (solution encoding), fitness function, initial population, selection of parents for generating a new population, crossover, mutation, and stopping criteria.

In order to produce the initial population, *Pop* chromosomes are generated randomly and legally. Furthermore, the GA process is terminated after a predefined number of iterations (Max*Gen*). Other features of the algorithm are as follows:

3.1. Chromosome definition

In the proposed GA, the solution encoding (chromosome) is presented as a $2 \times s$ matrix where s is the number of subsystems and the first and second rows of this matrix represent the selected failure rate, λ_i, for the components in each subsystem and the redundancy level for the subsystems, respectively. Fig. 4 presents a chromosome structure considered for this problem with $s = 5$.

This figure demonstrates a solution in which the first subsystem ($s = 1$) uses two components with a failure rate equal to 0.001 in parallel, the second subsystem uses four components with a failure rate equal to 0.045, and the last subsystem uses one component with a failure rate equal to 0.00105. Fig. 5 presents the structure of this solution in the series system ($P1$).

3.2. Fitness function

The fitness function is equal to the sum of the objective function (reliability) and the penalty of constraint violation. In other words, the problem constraint is added to the objective function in such a way that if one solution exceeds the constraint limit, a relatively large amount of penalty is added to the objective function. This penalty provides the feasibility of the final solution while keeping the search in the infeasible space of the problem. The search in the infeasible space leads to an appropriate diversity for the genetic algorithm. In this paper, in order to add the penalties of constraints violation to the objective function, first the objective function is transformed to minimization of unreliability then additive penalty function is applied as Eq. (16). Now, because of minimization structure of the objective function, these added penalties decrease the probability of selecting the violated solution during the algorithm process. In this way, after some iteration, the violated solutions are expected to be eliminated from the population.

$$\begin{array}{l c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \textit{Failure rate} & \left[\begin{array}{ccccc} .001 & .045 & .0094 & .0205 & .00105 \\ 2 & 4 & 2 & 3 & 1 \end{array}\right] \\ \text{Number of components} & \end{array}$$

Fig. 4. Chromosome representation (solution encoding).

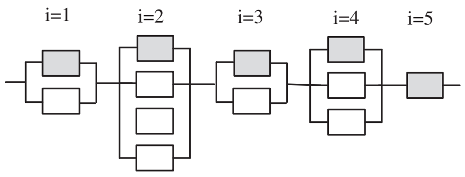


Fig. 5. Chromosome representation (solution encoding).

$$\textit{Minimize} \quad UR_s = (1 - R_s) + \sum_{i=1}^{3} P_i(x) \tag{16}$$

In this equation, R_s and UR_s are the total reliability and unreliability of the system, respectively; $P_i(x)$ is the penalty value related to the ith constraint which takes a positive value in case of violation. In this paper, violation of a constraint like $g_i(r,n) \leqslant l_i$ is performed through the function $\frac{g_i(r,n)}{l_i} - 1$. Therefore, $P_i(x) = M \times \max\left\{0, \frac{g_i(r,n)}{l_i} - 1\right\}$, where M is a sufficiently large number.

3.3. Selection

In order to select the required chromosomes for crossover and mutation operators, tournament selection is used based on the following steps. The fitness function is calculated for all existing chromosomes (*Pop*) in the present population and then, from the *Pop* chromosomes, *k* chromosomes are selected randomly and compared with each other based on the fitness function. A chromosome with the largest fitness function is selected as the parent for generating a new population. This procedure will be repeated *Pop* times until *Pop* parents are selected for the next generation. After the selection process, the crossover and mutation operators will be used.

3.4. Crossover

The crossover operator is applied with a predefined rate of r_C. By using two crossover operators, four offspring will be generated from each two selected parents. From these four offspring, two premier ones will be selected based on their fitness values to be transferred to the next generation. As a result, there will be *Pop* new chromosomes at the end of the crossover operation. In order to produce these four offspring from the two selected parents, three crossover operators are defined, two of which are used for each test problem.

These three operators are the double-point crossover, a modified version of the max–min crossover [4], and a new crossover operator. The double-point and max–min crossover are shown in Figs. 6 and 7, respectively. In the max–min crossover, the subsystems with the lowest and highest reliabilities amongst the candidate solutions are determined and all the relative genes for each parent are exchanged with the same genes in the other parent. The third operator performs as follows:

Consider x^P as a selected chromosome in the present population and x_i^P as the ith component of x^P. Also, suppose that x^{Best} is the best chromosome in the population and that x_i^{Best} represents the ith component of this solution. The new chromosome will be defined based on following equation:

$$x_i^{new} = x_i^P + rand \cdot \left(x_i^{Best} - x_i^P\right); \quad \forall i \tag{17}$$

In this equation, *rand* is a random value.

3.5. Mutation

After the crossover operation, the mutation operator will be employed with a predefined rate of r_M. The main purpose of applying the mutation operator is to increase diversity and to avoid being trapped into a local optimum. In order to perform the mutation, r_M. 100% of the chromosomes are randomly selected and the gene values are changed with a probability of P_M.

$$\begin{array}{l} \textit{Failure rate:} \\ \textit{Number of components} \end{array} \begin{bmatrix} .001 & .045 & .0094 & .0205 & .00105 \\ 2 & 4 & 2 & 3 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} .001 & .045 & .0514 & .095 & .00105 \\ 2 & 4 & 1 & 1 & 1 \end{bmatrix}$$
$$\begin{array}{ccccc} & & \uparrow & & \uparrow \end{array}$$

$$\begin{array}{l} \textit{Failure rate:} \\ \textit{Number of components} \end{array} \begin{bmatrix} .041 & .005 & .0514 & .095 & .0137 \\ 5 & 2 & 1 & 1 & 6 \end{bmatrix} \Rightarrow \begin{bmatrix} .041 & .005 & .0094 & .0205 & .0137 \\ 5 & 2 & 2 & 3 & 6 \end{bmatrix}$$
$$\begin{array}{ccccc} & & \uparrow & & \uparrow \end{array}$$

Fig. 6. Double point crossover operator.

$$\begin{array}{lccccccccccc}
\textit{Sub-systems reliab.:} & .85 & .95 & .87 & .90 & .70 & & .85 & .80 & .87 & .90 & .99 \\
\textit{Failure rate:} & .001 & .045 & .0094 & .0205 & .00105 & \Rightarrow & .001 & .005 & .0094 & .0205 & .0137 \\
\textit{Number of components} & 2 & 4 & 2 & 3 & 1 & & 2 & 2 & 2 & 3 & 4 \\
 & & \uparrow & & & \uparrow & & & \uparrow & & & \uparrow \\
 & & \max & & & \min & & & & & &
\end{array}$$

$$\begin{array}{lccccccccccc}
\textit{Sub-systems reliab.:} & .96 & .80 & .65 & .45 & .99 & & .96 & .80 & .65 & .90 & .70 \\
\textit{Failure rate:} & .041 & .005 & .0514 & .095 & .0137 & \Rightarrow & .041 & .005 & .0514 & .0205 & .0105 \\
\textit{Number of components:} & 5 & 2 & 1 & 1 & 4 & & 5 & 2 & 1 & 3 & 1 \\
 & & & & \uparrow & \uparrow & & & & & \uparrow & \uparrow \\
 & & & & \min & \max & & & & & &
\end{array}$$

Fig. 7. Max–min crossover operator.

$$\begin{array}{lccccccccccc}
\textit{Sub-systems reliab.:} & .85 & .95 & .87 & .90 & .70 & & .85 & .75 & .87 & .90 & .92 \\
\textit{Failure rate:} & .001 & .045 & .0094 & .0205 & .00105 & \Rightarrow & .001 & .0201 & .0094 & .0205 & .003 \\
\textit{Number of components} & 2 & 4 & 2 & 3 & 1 & & 2 & 1 & 2 & 3 & 3 \\
 & & \uparrow & & & \uparrow & & & \uparrow & & & \uparrow \\
 & & \max & & & \min & & & & & &
\end{array}$$

Fig. 8. Max–min mutation operator.

Table 1
Crossover and mutation operators in each test problem.

Operators problem no.	Crossover			Mutation		
	Double point	Max–min	New	Simple	Max–min	New
*P*1	☑	☑		☑	☑	
*P*2	☑		☑	☑		☑
*P*3	☑		☑	☑		☑

Note that the new gene's values are randomly generated between the lower and upper bounds of the variables. Then, the fitness function is calculated for the mutated chromosome and compared with the fitness function of the pre-mutation chromosome. If the fitness function of the new chromosome is better than the previous one, the previous chromosome will be replaced by the newly generated offspring; otherwise, the previous chromosome remains as the superior one. In this paper, the max–min mutation operator developed by Tavakkoli-Moghaddam et al. [4] is used. In this operator, for each candidate solution, the subsystems with the highest and lowest reliabilities are randomly mutated. The values of genes for these subsystems are randomly changed at a mutation rate of P_M. Furthermore, another mutation operator is used which is the same as the max–min mutation with the only difference that in this operator, two random subsystems, rather than subsystems with maximum and minimum reliabilities, are selected to be mutated. As an example, the max–min mutation is depicted in Fig. 8.

Crossover and mutation operators transfer one generation to the next. For this purpose, the *Pop* best solutions amongst the previous generation and the new offspring are retained to form the next generation. As already mentioned, a combination of the crossover and mutation operators is used for each test problem. Table 1 demonstrates this combination for each test problem.

4. Experimental results

In this Section, the results obtained by the proposed GA are presented for the three benchmark problems described in Section 2. The simulations were implemented for each problem in MATLAB environment on an Intel Core 2 Duo CPU2.66 GHz PC with 3 GB of RAM under a 32-bits Windows operating system. Preliminary numerical tests were used to set the values of the genetic algorithm parameters. Different data were randomly generated and used to calibrate the parameters. Once the values of the parameters had been set for these preliminary data, they were used for the variations of the problem instances to be solved in this paper. The following parameters were used in the proposed GA: a population size of 500, maximum generation (Max*Gen*) equal to 40 runs, $k = 4$, $r_C = 0.8$, $r_M = 0.3$, and $P_M = 0.2$. The switching reliability is considered to be 0.99 in the mission time, it means that $\rho_i(t) = 0.99$. This value of switching reliability is used in the previous studies such as [2,4,5].

In order to obtain a better understanding of the proposed algorithm, each test problem was evaluated over 5 independent runs. The input parameters for the series and complex (bridge) systems ($P1$ and $P3$) are shown in Table 2and those for the series–parallel system ($P2$) are presented in Table 3.

Table 2
Data used in series and complex bridge systems.

Stage	$10^5 \cdot \alpha_i$	β_i	$w_i \cdot v_i^2$	w_i	V	C	W
1	2.330	1.5	1	7	110	175	200
2	1.450	1.5	2	8			
3	0.541	1.5	3	8			
4	8.050	1.5	4	6			
5	1.950	1.5	2	9			

Table 3
Data used in series–parallel systems.

Stage	$10^5 \cdot \alpha_i$	β_i	$w_i \cdot v_i^2$	w_i	V	C	W
1	2.500	1.5	2	3.5	180	175	100
2	1.450	1.5	4	4.0			
3	0.541	1.5	5	4.0			
4	0.541	1.5	8	3.5			
5	2.100	1.5	4	4.5			

Table 4
Different runs of proposed GA for series system.

Parameter	Run #1	Run #2	Run #3	Run #4	Run #5	STD
R_s	0.96957251	0.96949763	0.96950107	0.96820142	**0.96957758**	5.98581E−04
λ_i	0.00026619	0.00026791	0.00026121	0.00028628	0.00026841	
	0.00011796	0.00012267	0.00011843	0.00011721	0.00011931	
	0.00008811	0.00009014	0.00009354	0.00010897	0.00008840	
	0.00037080	0.00035360	0.00035951	0.00032495	0.00036600	
	0.00025370	0.00025964	0.00026252	0.00028245	0.00025356	
n	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	
r	0.76629197	0.76497899	0.77012204	0.75105245	0.76459335	
	0.88873470	0.88455745	0.88831636	0.88939536	0.88752892	
	0.91565985	0.91380683	0.91069820	0.89676142	0.91539527	
	0.69018313	0.70215628	0.69801819	0.72256119	0.69350544	
	0.77592300	0.77133168	0.76911345	0.75393410	0.77603145	
Slack (g1)	27	27	27	27	27	
Slack (g2)	0.00000511	0.00132501	0.00009664	0.00016629	0.00002478	
Slack (g3)	7.51891824	7.51891824	7.51891824	7.51891824	7.51891824	

Table 5
Different runs of proposed GA for series–parallel system.

Parameter	Run #1	Run #2	Run #3	Run #4	Run #5	STD
R_s	**0.999988249**	0.999988243	0.999988183	0.999984630	0.999988224	1.60776E−06
λ_i	0.000192558	0.000184630	0.000190962	0.000218509	0.000187741	
	0.000171006	0.000161603	0.000155218	0.000099861	0.000163980	
	0.000096320	0.000099501	0.000124554	0.000202271	0.000100256	
	0.000106807	0.000106898	0.000091938	0.000289820	0.000105741	
	0.000144491	0.000152923	0.000152832	0.000133194	0.000148347	
n	(3, 3, 2, 1, 3)	(3, 3, 2, 1, 3)	(3, 3, 1, 2, 3)	(3, 2, 2, 3, 3)	(3, 3, 1, 2, 3)	
r	0.824846726	0.831411488	0.826163745	0.803716235	0.828829523	
	0.842816570	0.850778906	0.856228310	0.904963136	0.848758789	
	0.908173083	0.905289070	0.882890356	0.816873233	0.904605868	
	0.898699000	0.898616953	0.912161652	0.748398488	0.899657410	
	0.865463014	0.858195436	0.858273837	0.875295691	0.862131961	
Slack (g1)	62	62	53	18	53	
Slack (g2)	0.000064145	0.000182880	0.000020252	0.000151000	0.000001979	
Slack (g3)	6.104140278	6.104140278	7.110848840	0.583959096	7.110848840	

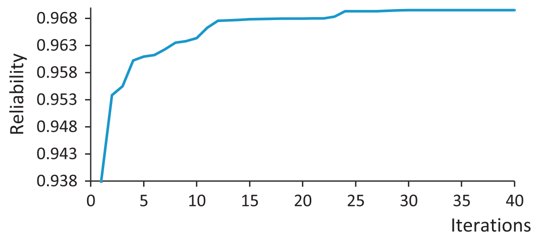
Table 6
Different runs of proposed GA for complex bridge system.

Parameter	Run #1	Run #2	Run #3	Run #4	Run #5	STD
R_s	0.99997304	0.99997402	0.99997369	0.99997413	0.99997405	4.47953E−07
λ_i	0.00021390	0.00022154	0.00022860	0.00021744	0.00022566	
	0.00018236	0.00014970	0.00016632	0.00015412	0.00014950	
	0.00013199	0.00014554	0.00013647	0.00014232	0.00013862	
	0.00029705	0.00031615	0.00029787	0.00031802	0.00032495	
	0.00031301	0.00028430	0.00030351	0.00026897	0.00024029	
n	(3, 3, 3, 3, 1)	(3, 3, 3, 3, 1)	(3, 3, 3, 3, 1)	(3, 3, 3, 3, 1)	(3, 3, 3, 3, 1)	
r	0.80743162	0.80128617	0.79564329	0.80457234	0.79799199	
	0.83330218	0.86096397	0.84677624	0.85717305	0.86113517	
	0.87635075	0.86455106	0.87243372	0.86734683	0.87055869	
	0.74301048	0.72894917	0.74239993	0.72759162	0.72256091	
	0.73124088	0.75254340	0.73821892	0.76416666	0.78639900	
Slack (g1)	18	18	18	18	18	
Slack (g2)	0.00004397	0.00000001	0.00000805	0.00006867	0.00001866	
Slack (g3)	4.26476980	4.26476980	4.26476980	4.26476980	4.26476980	

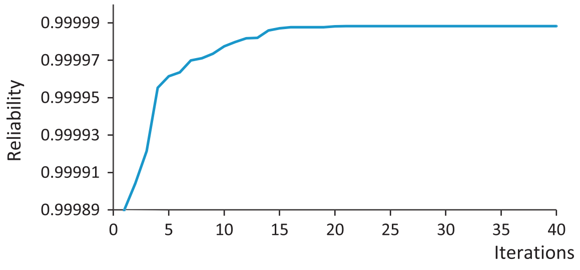
Tables 4–6 show the simulation results of all the five runs of the proposed GA for the benchmarks $P1$, $P2$ and $P3$, respectively. Furthermore, these Tables provide the standard deviation (STD) values of the objective functions in the 5 different runs for each test problem. Clearly, the STDs are small, indicating the robustness of the proposed GA in all the test problems.

In Tables 4–6, λ_i is the exponential distribution parameter (failure rate) for the components of subsystem i, n_i is the number of components in subsystem i, and r_i is the reliability at mission time (1000 h) for the components used in subsystem i which is calculated in terms of λ_i. The slack represents unused resources.

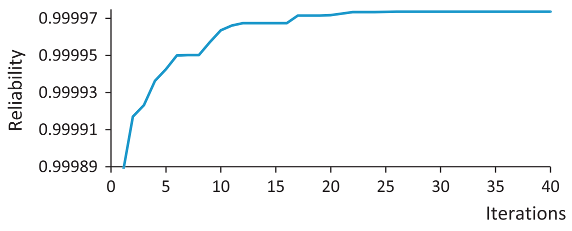
In Fig. 9(a–c), the trends show the progress of the best run among the 5 runs in each problem. During the early iterations, the proposed GA exhibits a fast convergence rate while in the last iterations, it stagnates slowly, which indicates no fitness improvements occurring during the latter evolutionary process.



(a) Series system



(b) Series-parallel system



(c) Complex bridge system

Fig. 9. Optimization curves of three test problems using proposed GA.

The maximum reliability obtained by the cold-standby strategy (see Tables 4–6) was then used to compare its performance with those reported in the literature. Tables 7–9 present a comparison between the best results obtained in this paper and those reported in previous studies. Maximum Possible Improvement (MPI) index was used to measure the significance of the improvements made by using the cold-standby strategy as compared to the previous best-known solutions that considered the active strategy. This index, which has been used in many previous studies including Yeh and Hsieh [28], Coelho [29], Wu et al. [32], Zou et al. [33], Wang and Li [34], Valian and Valian [35], and Valian et al. [36], is given by:

$$\text{MPI }(\%) = [R_s(\mathit{New\ Approach}) - R_s(\mathit{Other})]/[1 - R_s(\mathit{Other})] \tag{18}$$

where $R_s(\mathit{New\ Approach})$ represents the best system reliability obtained by the proposed approach and $R_s(\mathit{Other})$ represents the best system reliability obtained by any other method reported in the literature. Tables 6–9 indicate that the cold-standby

Table 7
Comparison of best result for the series system with other results presented in literature.

Parameter	Hikita et al. [39]	Hikita et al. [9]	Hsieh et al. [40]	Chen [25]	Hsieh and You [10]	Wu et al. [32]	Yeh and Hsieh [28]	Valian and Valian [35] and Valian et al. [36]	Afonso et al. [37]	Abouei and Hamadani
R_s	0.931451	0.931363	0.931578	0.931678	0.931682340	0.931680	0.9316820	0.931682387	0.93167939	0.96957758
n	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)
r	0.774887	0.777143	0.779427	0.779266	0.779462304	0.78037307	0.779399	0.779416938	0.779874	0.76459335
	0.870065	0.867514	0.869482	0.872513	0.871883456	0.87178343	0.871837	0.871833278	0.872057	0.88752892
	0.898549	0.896696	0.902674	0.902634	0.902800879	0.90240890	0.902885	0.902885082	0.903426	0.91539527
	0.716524	0.717739	0.714038	0.710648	0.711350168	0.71147356	0.711403	0.711393868	0.71096	0.69350544
	0.791368	0.793889	0.786896	0.788406	0.787861587	0.78738760	0.787800	0.787803712	0.786902	0.77603145
MPI (%)	55.6194495	55.6763502	55.5370735	55.4719950	55.4691663	55.4706915	55.4693879	55.4691357	55.4710891	
Slack (g1)	27	27	27	27	27	27	27	27	27	27
Slack (g2)	0.108244	0.0000	0.121454	0.001559	0.0000005284	0.000101	−0.0002184[a]	0.000000265	0.000099	0.00002478
Slack (g3)	7.518918	7.518918	7.518918	7.518918	7.518918	7.518918	7.5189182	7.518918241	7.518918	7.51891824

[a] Infeasible.

Table 8
Comparison of best result for the series–parallel system with other results presented in literature.

Parameter	Hikita et al. [39]	Hsieh et al. [40]	Chen [25]	Yeh and Hsieh [28]	Wu et al. [32]	Hsieh and You [10]	Valian and Valian [35] and Valian et al. [36]	Wang and Li [34]	Afonso et al. [37]	Abouei and Hamadani
R_s	0.999968750	0.999974180	0.999976580	0.999977310	0.999976640	0.999976649	0.999976649	0.999976650	0.999976610	**0.999988249**
n	(3, 3, 1, 2, 3)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(3, 3, 2, 1, 3)
r	0.83819295	0.785452	0.812485	0.8197457	0.81918526	0.819591561	0.819927087	0.819596000	0.82201264	0.824846726
	0.85506525	0.842998	0.843155	0.8450080	0.84366421	0.844951068	0.845267657	0.845000000	0.84365640	0.842816570
	0.87885933	0.885333	0.897385	0.8954581	0.89472992	0.895428548	0.895491554	0.895514000	0.89129092	0.908173083
	0.91140223	0.917958	0.894516	0.9009032	0.89537628	0.895522339	0.895440692	0.895519000	0.89869886	0.898699000
	0.85035522	0.870318	0.870590	0.8684069	0.86912724	0.868490229	0.868318775	0.868456000	0.86824939	0.865463014
MPI (%)	62.3979	54.4901	49.8264	48.2121	49.6975	49.6781	49.6781	49.6760	49.7620	–
Slack (g1)	53	40	40	40	40	40	40	40	40	62
Slack (g2)	0.000011	1.194440	0.002627	-1.469522	0.000561	0.000000	0.0000161	0.0000070	0.000396	0.00006415
Slack (g3)	7.110849	1.609289	1.609289	1.609289	1.609289	1.609289	1.6092890	1.6092890	1.609289	6.10414028

Table 9
Comparison of best result for the complex bridge system with other results presented in literature.

Parameter	Hikita et al. [9]	Hsieh et al. [40]	Chen [25]	Coelho [29]	Wu et al. [32]	Hsieh and You [10]	Zou et al. [33]	Valian and Valian [35] and Valian et al. [36]	Wang and Li [34]	Afonso et al. [37]	Abouei and Hamadani
R_s	0.99978937	0.99987916	0.99988921	0.99988957	0.99988963	0.9998893505	0.9998896	0.99988964	0.99988964	0.99988963	**0.99997413**
n	(3,3, 2, 3, 2)	(3, 3, 3, 3, 1)	(3, 3, 3, 3, 1)	(3, 3, 2, 4, 1)	(3, 3, 2, 4, 1)	(3, 3, 3, 3, 1)	(3, 3, 2, 4, 1)	(3, 3, 2, 4, 1)	(3, 3, 2, 4, 1)	(3, 3, 2, 4, 1)	(3, 3, 3, 3, 1)
r	0.81448276	0.814090	0.812485	0.826678	0.82868361	0.816624176	0.82983999	0.82809404	0.82812427	0.82764257	0.80457234
	0.82138254	0.864614	0.867661	0.857172	0.85802567	0.868767396	0.85798911	0.85800449	0.85784692	0.85747845	0.85717305
	0.89615152	0.890291	0.861221	0.914629	0.91364616	0.858748781	0.91333926	0.91416292	0.91420816	0.91419677	0.86734683
	0.71309103	0.701190	0.713852	0.648918	0.64803407	0.710279379	0.64674479	0.64790779	0.64808425	0.64927379	0.72759162
	0.81409107	0.734731	0.756699	0.715290	0.70227595	0.753429200	0.70310972	0.70456598	0.70411685	0.704092	0.76416666
MPI (%)	87.7155	78.5875	76.6451	76.5690	76.5563	76.6155	76.5626	76.5541	76.5541	76.5563	–
Slack (g1)	27	18	18	5	5	18	5	5	5	5	18
Slack (g2)	0.00001	0.376347	0.001146[a]	0.00116843[b]	0.00000359	0.00000	0.00000594	0.000079290	0.000000593	0.00004428	0.00006867
Slack (g3)	10.572475	4.26477	4.26477	1.560466	1.56046629	4.26477	1.56046629	1.560466288	1.560466000	1.56046629	4.26476980

[a] In Chen [25] it was reported 0.001494.
[b] In Coelho [29] it was reported 0.00000359.

strategy used for the redundant components led to remarkable improvements in the reliability levels of all the three benchmark problems compared to those achieved in other studies which considered the active strategy. The MPI index for our approach led to values greater than 55.4%, 48.2%, and 76.5% for *P*1, *P*2, and *P*3, respectively. Also, comparison with the values obtained in other studies such as Yeh and Hsieh [28], Coelho [29], Wu et al. [32], Zou et al. [33], Wang and Li [34], Valian and Valian [35], and Valian et al. [36], reveals that the values obtained in the present study are rather large, indicating the advantage of the cold-standby strategy compared to its active counterpart.

5. Concluding remarks

The reliability–redundancy allocation problems (RRAPs) have been traditionally formulated by considering the active strategy. In this paper, the RRAP was considered with a cold-standby strategy for redundant components. In order to evaluate the efficiency of the proposed strategy, three well-known benchmark problems were considered and a new mathematical formulation was developed for calculating the value for system reliability under the cold-standby strategy. The problems were formulated as non-linear integer programming models subject to a number of given non-linear constraints. In general, RRAPs are not easy to solve, especially for large size instances. In this paper, a modified version of the genetic algorithm (GA) was, therefore, developed as an effective meta-heuristic algorithm for RRAP. Numerical results demonstrated that the cold-standby redundancy strategy led to a higher reliability compared to its active counterpart. This strategy was shown to offer greater flexibility for system designers and reliability analysts and led to considerable improvements in the reliability of complex systems. For future studies, developing other solution methodologies may be suggested for the proposed strategy to achieve better results. Another interesting extension may be the case where both active and cold-standby strategies are simultaneously used in the system.

References

[1] W. Kuo, R. Prasad, An annotated overview of system reliability optimization, IEEE Trans. Reliab. 49 (2) (2000) 176–187.
[2] D.W. Coit, Maximization of system reliability with a choice of redundancy strategies, IIE Trans. 35 (6) (2003) 535–544.
[3] C.E. Ebling, An Introduction to Reliability Maintainability Engineering, McGraw-Hill, New York, 1997.
[4] R. Tavakkoli-Moghaddam, J. Safari, F. Sassani, Reliability optimization of series–parallel systems with a choice of redundancy strategies using a genetic algorithm, Reliab. Eng. Syst. Saf. 93 (2008) 550–556.
[5] D.W. Coit, Cold standby redundancy optimization for non-repairable systems, IIE Trans. 33 (2001) 471–478.
[6] W. Kuo, R. Prasad, F.A. Tillman, C.L. Hwang, Optimal Reliability Design: Fundamentals and Applications, Cambridge University Press, Cambridge, 2001.
[7] W. Kuo, H. Lin, Z. Xu, W. Zhang, Reliability optimization with the Lagrange multiplier and branch-and-bound technique, IEEE Trans. Reliab. 36 (1987) 624–630.
[8] V.R. Prasad, W. Kuo, Reliability optimization of coherent system, IEEE Trans. Reliab. 49 (2000) 323–330.
[9] M. Hikita, Y. Nakagawa, H. Harihisa, Reliability optimization of systems by a surrogate constraints algorithm, IEEE Trans. Reliab. 41 (1992) 473–480.
[10] Y.C. Hsieh, P.C. You, An effective immune based two-phase approach for the optimal reliability–redundancy allocation problem, Appl. Math. Comput. 218 (2011) 1297–1307.
[11] M.S. Chern, On the computational complexity of reliability redundancy allocation in a series system, Oper. Res. Lett. 11 (1992) 309–315.
[12] C. Ha, W. Kuo, Reliability redundancy allocation: an improved realization for non convex nonlinear programming problems, Eur. J. Oper. Res. 171 (2006) 24–38.
[13] P. Giuggioli, M. Marseguerra, E. Zio, Multi objective optimization by genetic algorithms: application to safety systems, Reliab. Eng. Syst. Saf. 72 (1) (2001) 59–74.
[14] S. Martorell, A. Sanchez, S. Carlos, V. Serradell, Alternative and challenges in optimizing industrial safety using genetic algorithms, Reliab. Eng. Syst. Saf. 86 (1) (2004) 25–38.
[15] J.E. Ramirez, D.W. Coit, A. Konank, Redundancy allocation for series–parallel systems using a max–min approach, IEE Trans. 36 (9) (2004) 891–898.
[16] M. Marseguerra, E. Zio, S. Martorell, Basics of genetic algorithms optimization for RAMS applications, Reliab. Eng. Syst. Saf. 91 (9) (2006) 977–991.
[17] I.D. Lins, E.L. Droguett, Redundancy allocation problems considering systems with imperfect repairs using multi-objective genetic algorithms and discrete event simulation, Simul. Model. Pract. Theory 19 (2011) 362–381.
[18] S.K. Konak, A.E. Smith, D.W. Coit, Efficiently solving the redundancy allocation problem using tabu search, IIE Trans. 35 (6) (2003) 515–526.
[19] M. Ouzineb, M. Nourelfath, M. Gendreau, Tabu search for the redundancy allocation problem of homogenous series–parallel multi-state systems, Reliab. Eng. Syst. Saf. 93 (8) (2008) 1257–1272.
[20] Y.C. Liang, A.E. Smith, An ant colony optimization algorithm for the redundancy allocation problem, IEEE Trans. Reliab. 53 (3) (2004) 417–423.
[21] N. Nahas, M. Nourelfath, Ant system for reliability optimization of a series system with multiple-choice and budget constraints, Reliab. Eng. Syst. Saf. 87 (1) (2005) 1–12.
[22] M. Samrout, F. Yalaoui, E. Chatelet, N. Cheboo, New methods to minimize the preventive maintenance cost of series–parallel systems using ant colony optimization, Reliab. Eng. Syst. Saf. 89 (3) (2005) 346–354.
[23] N. Nahas, M. Nourelfath, D.A. Kadi, Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series–parallel systems, Reliab. Eng. Syst. Saf. 92 (2) (2007) 211–222.
[24] T.C. Chen, P.S. You, Immune algorithms-based approach for redundant reliability problems with multiple component choices, Comput. Ind. 56 (2) (2005) 195–205.
[25] T.C. Chen, IAs based approach for reliability redundancy allocation problems, Appl. Math. Comput. 182 (2) (2006) 1556–1567.
[26] A. Habib, R. Alsieidi, G. Youssef, Reliability analysis of a consecutive r-out-of-n: F system based on neural networks, Chaos, Soliton. Fract. 39 (2009) 610–624.
[27] T.J. Hsieh, W.C. Yeh, Penalty guided bees search for redundancy allocation problems with a mix of components in series–parallel systems, Comput. Oper. Res. 39 (2012) 2688–2704.
[28] W.C. Yeh, T.J. Hsieh, Solving reliability redundancy allocation problems using an artificial bee colony algorithm, Comput. Oper. Res. 38 (2011) 1465–1473.
[29] L.S. Coelho, An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications, Reliab. Eng. Syst. Saf. 94 (4) (2009) 830–837.
[30] A. Pourdarvish, Z. Ramezani, standby redundancy allocation in a multi-level series system by memetic algorithm, Int. J. Reliab. Qual. Saf. Eng. 20 (3) (2013) 1340007 (16 pages).
[31] M. Ouzineb, M. Nourelfath, M. Gendreaua, An efficient heuristic for reliability design optimization problems, Comput. Oper. Res. 37 (2012) 223–235.

[32] P. Wu, L. Gao, D. Zou, S. Li, An improved particle swarm optimization algorithm for reliability problems, ISA Trans. 50 (2011) 71–81.
[33] D. Zou, L. Gao, S. Li, P. Wu, An effective global harmony search algorithm for reliability problems, Expert Syst. Appl. 38 (2011) 4642–4648.
[34] L. Wang, L. Li, A coevolutionary differential evolution with harmony search for reliability–redundancy optimization, Expert Syst. Appl. 39 (2012) 5271–5278.
[35] E. Valian, E. Valian, A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems, Eng. Optim. 1080 (2012), http://dx.doi.org/10.1080/0305215X.2012.729055.
[36] E. Valian, S. Tavakoli, S. Mohanna, A. Haghi, Improved cuckoo search for reliability optimization problems, Comput. Ind. Eng. 64 (2013) 459–468.
[37] L.D. Afonso, V.C. Mariani, L.S. Coelho, Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability–redundancy optimization, Expert Syst. Appl. 40 (2013) 3794–3802.
[38] A.K. Dhingra, Optimal apportionment of reliability & redundancy in series systems under multiple objectives, IEEE Trans. Reliab. 41 (4) (1992) 576–582.
[39] M.Y. Hikita, Y. Nakagawa, K. Nakashima, H. Narihisa, Reliability optimization of system by a surrogate constraints algorithm, IEEE Trans. Reliab. 7 (1978) 325–328.
[40] Y.C. Hsieh, T.C. Chen, D.L. Bricker, Genetic algorithms for reliability design problems, Microelectron. Reliab. 38 (1998) 1599–1605.
[41] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Michigan, 1975.