

# An evolutionary computation approach to solving repairable multi-state multi-objective redundancy allocation problems

Madjid Tavana<sup>1,2</sup> · Kaveh Khalili-Damghani<sup>3</sup> · Debora Di Caprio<sup>4,5</sup> · Zeynab Oveisi<sup>3</sup>

Received: 20 July 2016 / Accepted: 25 October 2016 / Published online: 15 November 2016  
© The Natural Computing Applications Forum 2016

**Abstract** The redundancy allocation problem (RAP) is an optimization problem for maximizing system reliability at a predetermined time. Among the several extensions of RAPs, those considering multi-state and repairable components are the closest ones to real-life availability engineering problems. However, despite their practical implications, this class of problems has not received much attention in the RAP literature. In this paper, we propose a multi-objective nonlinear mixed-integer mathematical programming to model repairable multi-state multi-objective RAPs (RMMRAPs) where a series of parallel systems experiencing repairs, partial failures, and component degrading through time is considered. The performance of

a component depends on its state and may decrease/increase due to minor and major failures/repairs which are modeled by a Markov process. The proposed RMMRAP allows for configuring multiple components and redundancy levels in each sub-system while evaluating multiple objectives (i.e., availability and cost). A customized version of the non-dominated sorting genetic algorithm (NSGA-II), where constraints are handled using a combination of penalty functions and modification strategies, is introduced to solve the proposed RMMRAP. The performance of the proposed NSGA-II and that of an exact multi-objective mathematical solution procedure, known as the epsilon-constraint method, are compared on several benchmark RMMRAP instances. The results obtained show the relative dominance of the proposed customized NSGA-II over the epsilon-constraint method.

✉ Kaveh Khalili-Damghani  
kaveh.khalili@gmail.com

Madjid Tavana  
tavana@lasalle.edu;  
http://tavana.us/

Debora Di Caprio  
dicaper@mathstat.yorku.ca

Zeynab Oveisi  
zeynab.oveisi@yahoo.com

<sup>1</sup> Business Systems and Analytics Department, Distinguished Chair of Business Systems and Analytics, La Salle University, Philadelphia, PA 19141, USA

<sup>2</sup> Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, 33098 Paderborn, Germany

<sup>3</sup> Department of Industrial Engineering, South-Tehran Branch, Islamic Azad University, Tehran, Iran

<sup>4</sup> Department of Mathematics and Statistics, York University, Toronto, Canada

<sup>5</sup> Polo Tecnologico IISS G. Galilei, Bolzano, Italy

**Keywords** Redundancy allocation problem · Availability computation · Repairable component · Multi-state component · Markov process · Multi-objective programming · NSGA-II

## 1 Introduction

The redundancy allocation problem (RAP) is a combinatorial optimization problem with multiple resource constraints aimed at improving the system reliability. The RAP was first introduced by Misra and Ljubojevic [16]. RAPs are generally formulated as nonlinear integer problems and usually characterized by a considerable computational complexity.

A series–parallel RAP consists of a set of sub-systems which are connected in series where it is possible to select multiple component choices and redundancy levels in

parallel in each sub-system. The components of each sub-system may be either binary- or multi-state [10, 30]. The entire system may also consist of either two or multiple states. Binary systems are assumed to be either functional or non-functional [21]. However, a real system may go through several states ranging from full performance to complete failure carrying out infinite partial performances during its service time. That is, the state of a real system may be either downgraded or improved during its service time depending on several factors such as aging, maintenance, and workload. The transition between the states can be modeled using stochastic and Markov processes.

In a multi-state series–parallel system, two different types of redundancy strategies may be considered. The first one occurs when the components of the system are binary state, but each type of component may have a different nominal performance level. In this case, the system performance depends on the performance levels of the selected components and on these components being functional or non-functional. The second type of redundancy strategy applies to systems with multi-state components where the system performance is completely determined by the states of the selected components [21]. In this paper, we consider this second type of redundancy problem for multi-state series–parallel systems.

RAPs and their extensions are NP-hard problems [2]. A wide range of solution procedures including heuristics and meta-heuristics [1, 5, 12, 13, 17–19, 22, 23, 25, 29], hybrid methods [3, 20, 24], and multi-objective optimization [7] have been proposed to solve these problems. Yu and Gen [28] provided a comprehensive review of the genetic algorithm (GA) approaches that have been used to solve various reliability optimization problems in the literature.

To solve multi-objective system reliability optimization problems, Li et al. [11] proposed a procedure consisting of two steps: (1) generating non-dominated solutions using a multi-objective evolutionary algorithm; (2) clustering similar solutions by means of a self-organizing map and identifying the most efficient solutions by data envelopment analysis. Khalili-Damghani and Amiri [7] proposed a hybrid approach to solve binary-state multi-objective reliability RAPs. Their approach combines the efficient epsilon-constraint with the multi-start partial bound enumeration algorithm and data envelopment analysis.

In this paper, a multi-objective mixed-integer nonlinear programming is proposed to solve repairable multi-state multi-objective RAPs (RMMRAPs).

A RMMRAP is a series–parallel system where components are allowed to go through several performance states. We assume that one can select multiple component choices and redundancy levels in parallel in each sub-system. The performance of a component may deteriorate or improve

based on minor and major failures and repairs, respectively. The partial failure/repair model is parameterized by a Markov process.

RMMRAPs are based on practical real-world problems and, as such, have many practical implications. However, to the best of our knowledge, so far they have not been adequately studied in the RAP literature.

Due to conflicting objective functions, we use a multi-objective decision-making method to solve the proposed RMMRAP model. Generating non-dominated solutions on the Pareto front is an insightful way to study the problem since it imposes no restriction on the priorities of the objective functions. Thus, a customized version of the NSGA-II method is introduced to generate a set of non-dominated solutions on the Pareto front for RMMRAP instances. The performance of the proposed NSGA-II method on several benchmark instances is compared with that of an epsilon-constraint method using a set of well-known multi-objective comparison metrics.

The paper proceeds as follows. Section 2 provides some preliminaries including: a brief introduction to the principles of multi-objective decision making, the basic NSGA-II method, and the component availability analysis using a Markov process. Section 3 introduces and describes the proposed RMMRAP, the customization of the epsilon-constraint method, and a customized version of the NSGA-II method. Test problems, software implementation, and comparison metrics are presented in Sect. 4. The results obtained for the test problems are reported and discussed in Sect. 5. Finally, Sect. 6 concludes and suggests future research directions.

## 2 Preliminaries

A multi-objective decision-making (MODM) problem considering minimum objective functions can be presented as follows [6]:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & x \in S = \{x \in R^n | F(x) \leq b, x \geq 0\} \end{cases} \quad (1)$$

where  $f(x)$  represents a finite number of objective functions,  $F(x) \leq b$  represents a finite number of constraints,  $S$  is the feasible solution space, and  $x$  is the  $n$ -vector of decision variables,  $n \in Z^+$ .

The type and the timing of the preference articulations play a central role in the decision-making process. MODM solution procedures have been classified into four groups [6]. In particular, Hwang and Masud [6] showed that when assuming posterior preference information on the priority of objective functions, it is desirable to generate non-dominated solutions on the Pareto front of MODM problem.

Mavrotas [15] introduced the efficient epsilon-constraint method which is an improved version of a classical non-dominated solution generation method known as the epsilon-constraint method. The efficient epsilon-constraint method has been widely used to solve MODM problems [7, 8, 17, 26].

### 2.1 Non-dominated sorting genetic algorithm (NSGA-II) method

The NSGA-II method, initially introduced by Deb et al. [4], is a well-known multi-objective evolutionary algorithm. This method combines different techniques such as elitism, fast non-dominated sorting, and diversity maintenance along with the Pareto-optimal front and has been widely and successfully employed to solve several engineering, management, and combinatorial optimization problems. Details regarding the structure of the standard NSGA-II algorithm can be found in Deb et al. [4].

### 2.2 Component performance and availability

According to the general Markov model for a generic multi-state system introduced by Lisnianski and Levitin [14], any component  $h$  in the multi-state system can have  $k_h$  different states corresponding to different performance levels belonging to the set  $g_h = \{g_{h,1}, \dots, g_{h,k_h}\}$ . The indexing of the set  $g_h$  reflects the performance level of the component  $h$ , that is:

$$\forall l, q = 1, \dots, k_h, \quad l < q \quad \leftrightarrow \quad g_{h,l} < g_{h,q}. \tag{2}$$

The current state of the component  $h$  and the corresponding value of the component performance level  $G_h(t)$  at any instant  $t$  are random variables.  $G_h(t)$  takes values in  $g_h$ , that is,  $G_h(t) \in g_h$ . Thus, given an operational time period  $[0, T]$ , the evolution of the performance level  $G_h(t)$  of component  $h$  follows a discrete-state continuous-time Markov stochastic process.

Lisnianski and Levitin [14] model deals with minor and major failures and repairs of components. Given a generic repairable multi-state component  $h$ , a minor failure causes a downgrading from a state  $q$  to the adjacent lower state  $q - 1$ , while a major failure causes a downgrading from a state  $q$  to a state  $1 < q - 1$ . Similarly, a minor repair implies a return of the component from a state  $l$  to the adjacent higher state  $l + 1$ , while a major repair a return of the component from a state  $l$  to a state  $q > l + 1$ .

The state probabilities  $p_q(t)$ , with  $q = 1, \dots, k_h$ , define the probabilities that at the instant  $t > 0$  the component  $h$  will be in state  $q$  and can be obtained from the following system of differential equations.

$$\begin{aligned} \frac{dp_{k_h}(t)}{dt} &= \sum_{l=1}^{k_h-1} \mu_{l,k_h} p_l(t) - p_{k_h}(t) \sum_{l=1}^{k_h-1} \lambda_{k_h,l} \\ \frac{dp_q(t)}{dt} &= \sum_{l=q+1}^{k_h} \lambda_{l,q} p_l(t) + \sum_{l=1}^{q-1} \mu_{l,q} p_l(t) \\ &\quad - p_q(t) \left( \sum_{l=1}^{q-1} \lambda_{q,l} + \sum_{l=q+1}^{k_h} \mu_{q,l} \right) \quad \text{for } 1 < q < k_h \\ \frac{dp_1(t)}{dt} &= \sum_{l=2}^{k_h} \lambda_{l,1} p_l(t) - p_1(t) \sum_{l=2}^{k_h} \mu_{1,l} \end{aligned} \tag{3}$$

where

- $p_{k_h}(0) = 1, p_{k_h-1}(0) = \dots = p_1(0) = 0$  are the initial conditions representing the fact that at time  $t = 0$  the component  $h$  is new, that is, the component  $h$  is in state  $k_h$  where no failure has occurred yet;
- $\mu_{l, q}$  is the repair rate from state  $l$  to state  $q$ , where  $l, q = 1, \dots, k_h$  and  $l < q$ ;
- $\lambda_{q, l}$  is the failure rate from state  $q$  to state  $l$ , where  $l, q = 1, \dots, k_h$  and  $l < q$ ;

If  $D$  is a given demand level, that is, a reference component performance level, the acceptable states for the component  $h$  are all states  $q$  such that  $g_{h,q+1} \geq D > g_{h,q}$ . This leads to the definition of instantaneous availability of component  $h$  which is calculated as follows [28]:

$$A_h(t) = \sum_{e=q+1}^{k_h} p_e(t). \tag{4}$$

Equation (4) says that the instantaneous availability of component  $h$  at time  $t$  is given by the sum of all the state probabilities of the component from state  $q + 1$  to state  $k_h$ . That is, at a given demand level  $D$ ,  $A_h(t)$  calculates the probability of component  $h$  going through all working states which allow the component to meet the minimum required performance level.

### 3 Modeling RMMRAPs

Repairable multi-state multi-objective redundancy allocation problems (RMMRAPs) consist of series–parallel systems experiencing repairs, partial failures, and component degrading through time. RMMRAPs comprise configuring multiple components and redundancy levels in each subsystem while evaluating multiple objectives (i.e., availability and cost). In RMMRAPs, the performance of a component depends on its state and may decrease/increase due to minor and major failures/repairs which are modeled by a Markov process.

The proposed RMMRAP model is based on the following main assumptions.

- Multi-objectives (i.e., availability and cost) are considered for planning the RMMRAPs.
- All components are assumed to go through several states from perfect performance to complete failure.
- All components are assumed to be repairable.
- The performance level is a discrete-state continuous-time Markov stochastic process.
- Cost and weight of all components are assumed to be known, deterministic, and time independent.
- The availability of each component is assumed to be time dependent.
- The demand level is assumed to be constant and hence time independent.
- There are several redundant component choices within a sub-system.
- The redundancy strategy is assumed to be active.
- The total number of components in each sub-system is bounded.
- Final designs must conform to the cost, weight, and availability constraints.

The main notations used in the proposed RMMRAP are listed below:

$m$	Number of sub-systems
$i$	Index for the sub-systems, $i = 1, 2, \dots, m$
$n_i$	Total number of components used in sub-system $i$
$h$	Index for the components of subsystem $i$ , $h = 1, 2, \dots, n_i$
$j$	Index for the component types in each sub-systems, $j = 1, 2, \dots, n$
$D$	Demand level
$A_{ij}(t)$	Availability of a component of type $j$ in sub-system $i$ at the time $t$
$c_{ij}$	Cost of a component of type $j$ in sub-system $i$
$w_{ij}$	Weight of a component of type $j$ in sub-system $i$
$A_s(t)$	Overall availability of the series-parallel system at time $t$
$C_s$	Overall cost of the series-parallel system
$A_o$	System-level constraint: lower bound limit for availability (a number between 0 and 1)
$C_o$	Overall cost allowed by the system
$W_o$	Overall weight allowed by the system
$a_i$	Number of available component choices for sub-system $i$
$x_{ij}$	Number of components of type $j$ used in sub-system $i$
$n_{\max}$	Maximum number of components allowed to be in parallel
$n_{\min}$	Minimum number of components allowed to be in parallel

Figure 1 provides the schematic view of a series-parallel system whose sub-systems consist of repairable multi-state components, displaying the decision variables of the proposed RMMRAP model and the solution encoding that will be used to solve the model with NSGA-II.

More precisely, the proposed model considers a series-parallel system formed of  $m$  subsystems (Fig. 1a). Each sub-system  $i$  consists of  $n_i$  components which are allowed to be of  $n$  different types (Fig. 1b). Thus, a generic  $h$ th component is assigned a type  $j$ , where  $j = 1, \dots, n$ , and it is assumed to have the same states as the rest of the components of the same type. That is, the possible states for a component and the corresponding performance levels depend on the type of the component (Fig. 1c).

Through time, any component in any sub-system goes through minor and major failures and repairs while the demand level  $D$  is kept constant. Figure 1d represents the state-space diagram for a repairable multi-state component. The state probabilities and the availability of a component are calculated as in Lisnianski and Levitin [14] (see Sect. 2.2) and extended to the corresponding component type. That is,  $A_h(t) = A_{h'}(t) = A_{ij}(t)$  whenever  $h$  and  $h'$  are two components of type  $j$  in sub-system  $i$ .

The multi-objective model proposed for the RMMRAP is the following:

$$\text{Max } A_s(t) = \prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij}(t))^{x_{ij}} \right) \quad (5)$$

$$\text{Min } C_s = \sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot c_{ij} \quad (6)$$

s.t.

$$\prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij}(t))^{x_{ij}} \right) \geq A_o, \quad (7)$$

$$\sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot c_{ij} \leq C_o, \quad (8)$$

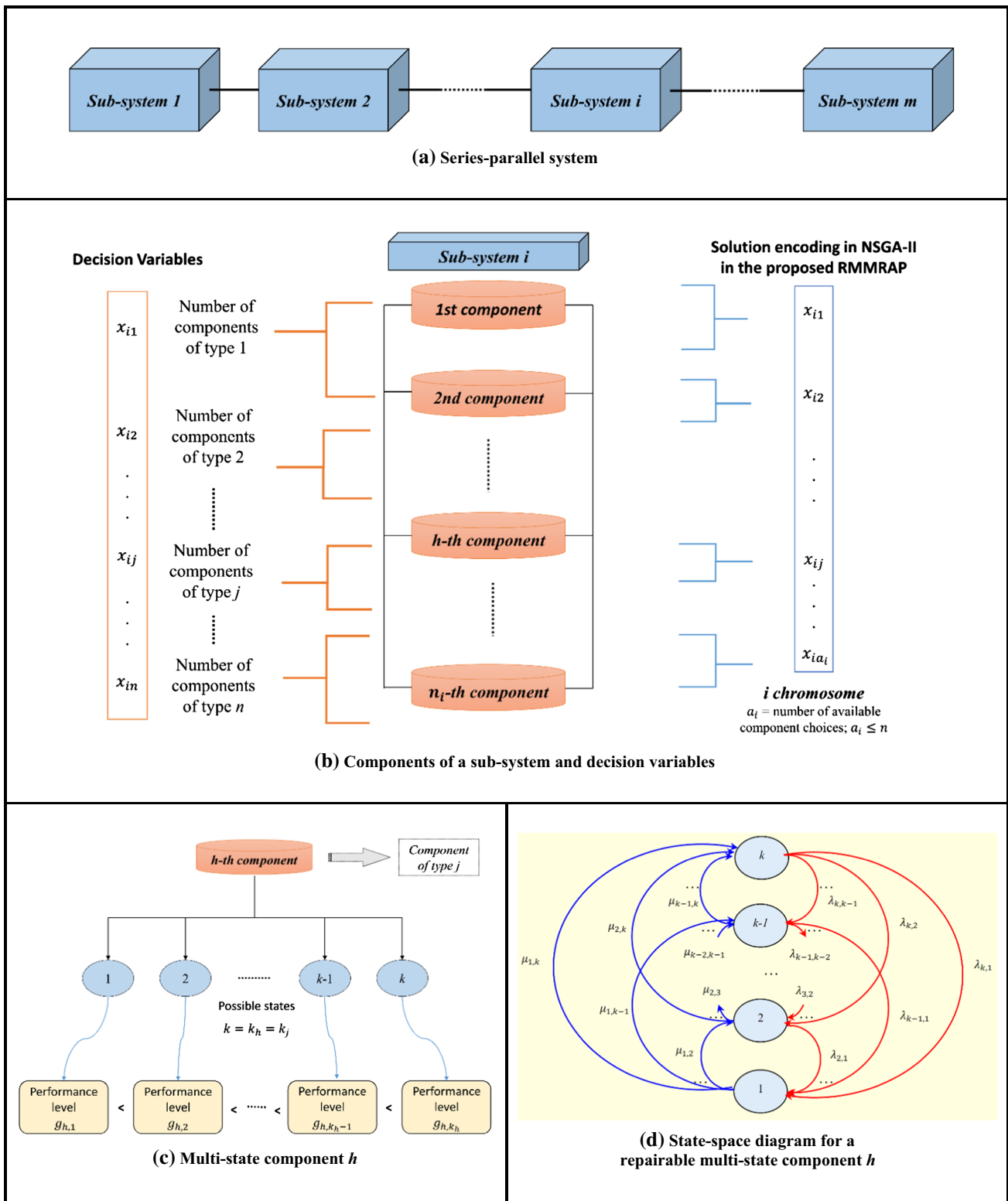
$$\sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot w_{ij} \leq W_o, \quad (9)$$

$$\sum_{j=1}^{a_i} x_{ij} \leq n_{\max}; \quad i = 1, 2, \dots, m; \quad (10)$$

$$\sum_{j=1}^{a_i} x_{ij} \geq n_{\min}; \quad i = 1, 2, \dots, m; \quad (11)$$

$$x_{ij} \in \mathbb{Z}^+; \quad i = 1, 2, \dots, m; \quad j = 1, \dots, n. \quad (12)$$

Objective functions (5) and (6) seek to optimize the system availability and system cost, respectively. In particular, objective function (5) calculates the availability of



**Fig. 1** Proposed framework

a series–parallel system which consists of  $m$  sub-systems and, in each sub-system  $i$ , it is possible to configure in parallel a number  $x_{ij}$  of components of type  $j$ , for every

$j$  varying from 1 (the sub-system is formed of at least one type of components) to a predetermined value  $a_i$ . Constraint (7) specifies the lower bound for the system



availability. Constraints (8) and (9) place an upper bound on the system cost and weight, respectively. Constraints (10) and (11) state the minimum and maximum number of components allowed in each sub-system. Constraint (12) defines the decision variables to be positive integers.

The right-hand side of constraint (7) represents the system lower bound limit for availability. Clearly, introducing constraint (7) imposes the model to seek configurations which have at least an availability value greater than or equal to  $A_o$ . In our model, the value  $A_o$  is assumed to be given, but it could alternatively be customized and set through managerial insights for different systems. From an optimization viewpoint, the direct consequence of introducing constraint (7) is that the proposed Model (5)–(12) seeks preferred solutions among all the non-dominated solutions on the Pareto front of the RMMRAPs. Thus, it is first necessary to generate a desired section of the Pareto front from the final non-dominated solution set of Model (5)–(12).

### 3.1 Handling RMMRAP using the epsilon-constraint method

Model (5)–(12) is transformed into Model (13)–(19) using the efficient epsilon-constraint method proposed by Mavrotas [15].

$$\text{Max } Z = \prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij}(t))^{x_{ij}} \right) - \varepsilon \times \frac{S_c}{r_c} \quad (13)$$

s.t.

$$\prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij}(t))^{x_{ij}} \right) \geq A_o, \quad (14)$$

$$\sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot c_{ij} + S_c = C_o - (C_o - C_1) \times e_2, \quad (15)$$

$$\sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot w_{ij} \leq W_o, \quad (16)$$

$$\sum_{j=1}^{a_i} x_{ij} \leq n_{\max}; \quad i = 1, 2, \dots, m; \quad (17)$$

$$\sum_{j=1}^{a_i} x_{ij} \geq n_{\min}; \quad i = 1, 2, \dots, m; \quad (18)$$

$$x_{ij} \in \mathbb{Z}^+; \quad i = 1, 2, \dots, m; \quad j = 1, \dots, n. \quad (19)$$

where  $C_1$  is the minimum cost allowed by the system,  $S_c$  is the slack variable of the cost objective function,  $r_c$  is the range of the cost objective function,  $e_2$  is a parameter belonging to the interval  $[0, 1]$ , and  $\varepsilon$  is a very small positive constant value.

Note that the term  $\frac{S_c}{r_c}$  in Eq. (13) is to be minimized due to the negative multiplier before it. Also,  $r_c$  is the range of the cost objective function and is a fixed parameters. Thus, Model (13)–(19) aims at reducing the value of  $S_c$  in the second term of objective function (13). As a consequence, given the presence of  $S_c$  in Eq. (15), the cost of the system will be as near as to the upper bound of cost in Eq. (15). This will lead to stronger efficient solutions in terms of cost objective function while maximizing the availability of the system expressed by Eq. (13). Moreover, note that the term  $\frac{S_c}{r_c}$  in Eq. (13) normalizes the slack variable of the cost objective function. Since both  $0 \leq \frac{S_c}{r_c} \leq 1$  and  $0 \leq A_o \leq \prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij}(t))^{x_{ij}} \right) \leq 1$  hold true, adding  $\prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij}(t))^{x_{ij}} \right)$  and  $-\varepsilon \times \frac{S_c}{r_c}$  is meaningful (as both these terms vary in the same range).

Finally, it must be noted that the simultaneous use of constraints (14) and (15) together with the objective function in Eq. (15) may lead to some infeasibility issues in Model (13)–(19). Fortunately, the efficient epsilon-constraint method proposed by Mavrotas [15] has a key advantage with respect to the classic epsilon-constraint method, that is, it can recognize infeasible loops and break them. So, no extra computational efforts are imposed to the model when considering Eqs. (14)–(15).

More formally, consider the parametric Model (13)–(19) to be solved using a predetermined step size for  $e_2$ . In the first iteration,  $e_2$  is set equal to zero. Hence, the term  $C_o - (C_o - C_1) \times e_2$  is equal to  $C_o$ . As the iterations go on, the value of  $e_2$  increases and, consequently, the value of the term  $C_o - (C_o - C_1) \times e_2$  decreases from  $C_o$  toward  $C_1$ . The decreasing behavior of the right-hand side of constraint (15) may lead to some infeasibilities in Model (13)–(19) when incorporating the role of constraint (14). Indeed, constraint (14) imposes the model to generate configurations that have an availability higher than  $A_o$ , while in each iteration the model is also imposed to generate a solution with smaller total cost, as the  $C_o - (C_o - C_1) \times e_2$  is decreasing while iterations go on. Using the efficient epsilon-constraint method, when the model is recognized infeasible due to such a conflict of constraints, the next iterations are discarded, as they cannot improve the right-hand side of constraint (15). So, no extra computational efforts are imposed to the model.

### 3.2 Customizing NSGA-II for RMMRAP

In this section, a customized version of NSGA-II, initially introduced by Deb et al. [4], is structured to solve the proposed RMMRAP.

### 3.2.1 Solution encoding

As for any other evolutionary algorithm, a feasible solution to the proposed RMMRAP is represented by a chromosome, while a population can be interpreted as a set of feasible solutions. Thus, to correctly encoding a solution, a chromosome must be a string of genes signifying the available components in all the sub-systems.

Figure 2 shows a schematic representation of the chromosome used to encode the solution to RMMRAP. This chromosome is composed of  $m$  sub-strings one per each sub-system. The first  $a_1$  genes represent the first sub-system, the next  $a_2$  genes represent the second sub-system, and so on. The sub-string of genes corresponding to the sub-system  $i$  will be referred to as *chromosome i*.

### 3.2.2 Determining the initial population

The size of the initial population is generated randomly to be 50. After generating the initial population, each chromosome is checked for feasibility based on the minimum and maximum number of components allowed in each sub-system.

### 3.2.3 Handling the constraints

In order to handle the constraints of the RMMRAP, a combination of the modification and penalty strategies is employed.

**3.2.3.1 Penalty strategy** Constraints (7)–(9) are handled by incorporating a penalty strategy. Two sets of violation values are calculated for each chromosome  $i$ . The first set of violations has an effect on the availability objective, while the second one has an effect on the cost objective.

The first set of violations relative to chromosome  $i$  is given by:

$$AV_{i1} = \text{Max} \left\{ A_o - \prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij})^{x_{ij}} \right), 0 \right\} \quad (20)$$

$$CV_{i1} = \text{Max} \left\{ \frac{\sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot c_{ij}}{C_o} - 1, 0 \right\} \quad (21)$$

$$WV_{i1} = \text{Max} \left\{ \frac{\sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot w_{ij}}{W_o} - 1, 0 \right\} \quad (22)$$

where  $AV_{i1}$ ,  $CV_{i1}$ , and  $WV_{i1}$  are, respectively, the violation values for the availability, cost, and weight of chromosome

$i$  relative to the first objective. Therefore, the availability objective is dynamically penalized as follows:

$$A'_{si} = A_{si} - (\text{max}\{AV_{i1}, CV_{i1}, WV_{i1}\} \times it) \quad (23)$$

where  $A'_{si}$  is the penalized availability function for the violated chromosome  $i$  and it is the iteration number. The second set of violations is relative to chromosome  $i$  is given by:

$$AV_{i2} = \text{Max} \left\{ A_o - \prod_{i=1}^m \left( 1 - \prod_{j=1}^{a_i} (1 - A_{ij})^{x_{ij}} \right), 0 \right\} \times M \quad (24)$$

$$CV_{i2} = \text{Max} \left\{ \sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot c_{ij} - C_o, 0 \right\} \quad (25)$$

$$WV_{i2} = \text{Max} \left\{ \sum_{i=1}^m \sum_{j=1}^{a_i} x_{ij} \cdot w_{ij} - W_o, 0 \right\} \quad (26)$$

where  $AV_{i2}$ ,  $CV_{i2}$ , and  $WV_{i2}$  are, respectively, the violation values of the availability, cost, and weight of chromosome  $i$  for the second objective. Hence, the cost objective is dynamically penalized as follows:

$$C'_{si} = C_{si} + (\text{max}\{AV_{i2}, CV_{i2}, WV_{i2}\} \times it) \quad (27)$$

where  $C'_{si}$  is the penalized cost function for the violated chromosome  $i$  and it is the iteration number.

**3.2.3.2 Modification strategy** Constraints (10) and (11) are handled using the modification strategy. The chromosomes violating the condition  $n_{\min} \leq \sum_{j=1}^{a_i} x_{ij} \leq n_{\max}$ , for  $i = 1, \dots, m$ , are modified by using the methods proposed by Khalili-Damghani et al. [9]. This modification strategy allows to determine which one of the component types of the violated chromosome  $i$  should be pruned by introducing a cost–benefit heuristic. Following again Khalili-Damghani et al. [9], a cost–benefit ratio is associated with each component type of the proposed RMMRAP model. This ratio is obtained dividing the availability of the component type by the sum of its cost with its weight, that is:

$$\text{Ratio}_{ij}(t) = \frac{A_{ij}(t)}{c_{ij} + w_{ij}} \quad (28)$$

The components with a lower  $\text{Ratio}_{ij}(t)$  have a higher pruning priority, while the components with a higher  $\text{Ratio}_{ij}(t)$  have a higher adding priority in the case of a shortage. That is, after ordering all the ratios from the

**Fig. 2** Structure of the proposed chromosome

$x_{11}$	$x_{12}$	...	$x_{1a_1}$	$x_{21}$	$x_{22}$	...	$x_{2a_2}$	...	$x_{m1}$	$x_{m2}$	...	$x_{ma_m}$
Sub-system 1				Sub-system 2					Sub-system m			

**Table 1** Test problem configuration

Case	Max component type	Number of sub-systems	Component attributes						
			Component cost	Component weight	Number of component's states	Performance of components	System cost	System weight	Availability
Small size	3	3	$U[2, 15]$	$U[2, 10]$	$U[2, 5]$	$U[0, 100]$ or $U[0, 10]$ or $U[0,1]$	$U[C_i, C_u]$	$U[W_i, W_u]$	$U[0.95,1]$
Medium size	3	6							
Large size	4	13							
Component type $j$		Minor repair	Major repair	Minor failure	Major failure				
1		$U[50, 100]$	$U[0, 40]$	$U[0.7, 2]$	$U[0.1, 0.4]$				
2		$\frac{1}{2} * U[50, 100]$	$\frac{1}{2} * U[0, 40]$	$U[0.7, 2]$	$U[0.1, 0.4]$				
3		$\frac{1}{2} * U[50, 100]$	$\frac{1}{2} * U[0, 40]$	$2 * U[0.7, 2]$	$2 * U[0.1, 0.4]$				
4		$\frac{1}{2} * U[50, 100]$	$\frac{1}{2} * U[0, 40]$	$2 * U[0.7, 2]$	$2 * U[0.1, 0.4]$				

Time span\* = [0, 0.1, 0.2], \* Time intervals in which availability of components and system are calculated

lowest to the highest one, the redundant components are those whose ratios are at the top of the list while the components corresponding to ratios at the bottom of the list can be used to supply a shortage of components.

### 3.2.4 Fitness function and non-dominated ranking

The fitness functions are calculated based on the penalized objective functions. All the solutions in a population are assigned a rank according to a non-dominated ranking procedure: Assign Rank 1 to non-dominated solutions; omit these solutions from the population; assign Rank 2 to other non-dominated solutions; omit these solutions; and so on until when all the solutions in the current population receive a rank.

### 3.2.5 Genetic operators

**3.2.5.1 Crossover operator** Double-point crossover is applied based on a predefined cross-rate. In the double-point crossover, two different points between 1 and the chromosome length minus 1 are selected randomly. Parents are divided into three separate parts by these two points. Each child is formed based on a selection of a mid-piece from one of the parents and two other pieces from the other parent.

**3.2.5.2 Mutation operator** First, an initial step size is determined. A binary random number is then generated. If the generated number is one, the step size is added to the value of the selected gene for mutation. Otherwise, the step

size is subtracted from the value of the selected gene. In this method, an exception is intended to avoid a negative number for the values of genes. If the value of the selected gene for mutation is zero, then the step size will be added regardless of the random number value.

### 3.2.6 Stopping criterion

In the proposed customized version of NSGA-II, the number of iterations works as the stopping criterion.

## 4 Test problems, software implementation, and comparison metrics

Thirty test RMMRAP instances belonging to three classes of test problems, namely small-size, medium-size, and large-size problems, have been used to test the performance of the proposed algorithm. The efficient epsilon-constraint and the proposed NSGA-II methods have been implemented on all instances and their performances compared using multi-objective metrics.

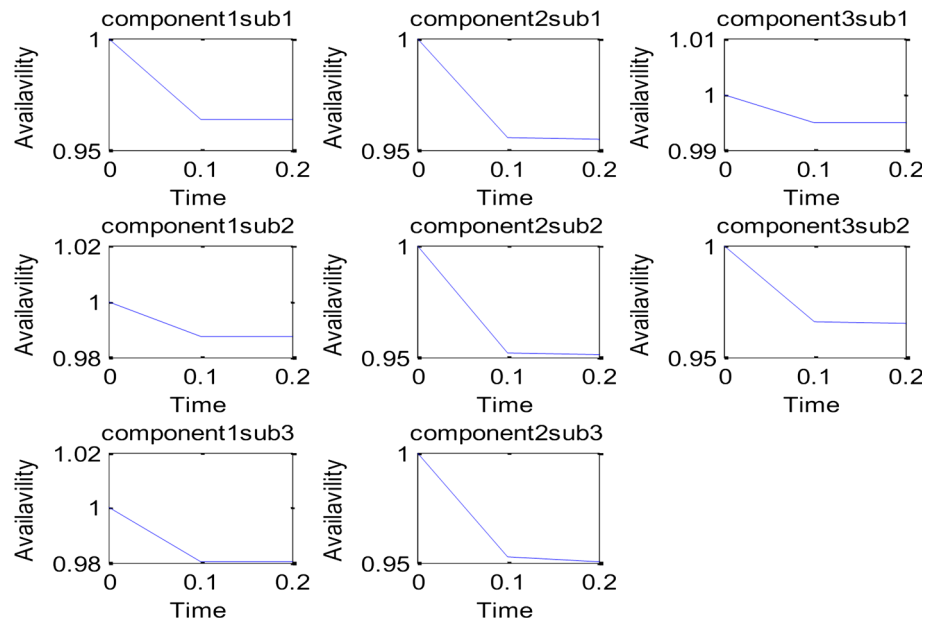
### 4.1 Test problems

The characteristics of the test instances are reported in Table 1.

These characteristics have been simulated using uniform probability density functions.  $U[a, b]$  in Table 1 means that the associated parameters have been generated using a uniform distribution function with parameters  $a$  and  $b$ .



**Fig. 3** Availability graphs for the components in the test problems



## 4.2 Software–hardware implementation

The test instances have been simulated with MS-Excel. The LINGO software was used to code the efficient epsilon-constraint method, while the proposed customized version of NSGA-II was coded using the MATLAB software. All codes were run on a laptop with MS-Windows 8.0, with 4 GB of RAM and 1.8 GHz Core i7 CPU.

## 4.3 Comparison metrics

In order to analyze and compare the accuracy and the diversity of the epsilon-constraint and the NSGA-II methods with respect to the Pareto front of the RMMRAP, several metrics proposed by Yu and Gen [28] have been used [i.e., number of non-dominated solutions (NNSs), error ratio (ER), generational distance (GD), and spacing metric (SM)]. The interested reader may refer to Khalili-Damghani and Amiri [7], Khalili-Damghani et al. [9], and Tavana et al. [27] for a detailed review of these metrics. Moreover, the reference set (RS) was retrieved for all runs since the real Pareto front cannot be achieved through direct enumeration for the test problems. The RS contains the non-dominated solutions for all the methods in all the runs. We performed 20 runs for each method and retrieved a suitable estimation of the RS.

## 5 Results

The results are presented in three subsections. The first subsection presents the results obtained when calculating the availability of the components. The second subsection

discusses the results relative to a single run of the NSGA-II and epsilon-constraint methods for all test problems comparing them to the RS. The third subsection describes the results obtained for the computational CPU time for the NSGA-II and epsilon-constraint methods.

### 5.1 Results for the availability values of components

The differential Eq. (3) is coded in MATLAB for each component. The availability graphs of the components are plotted and illustrated in Fig. 3.

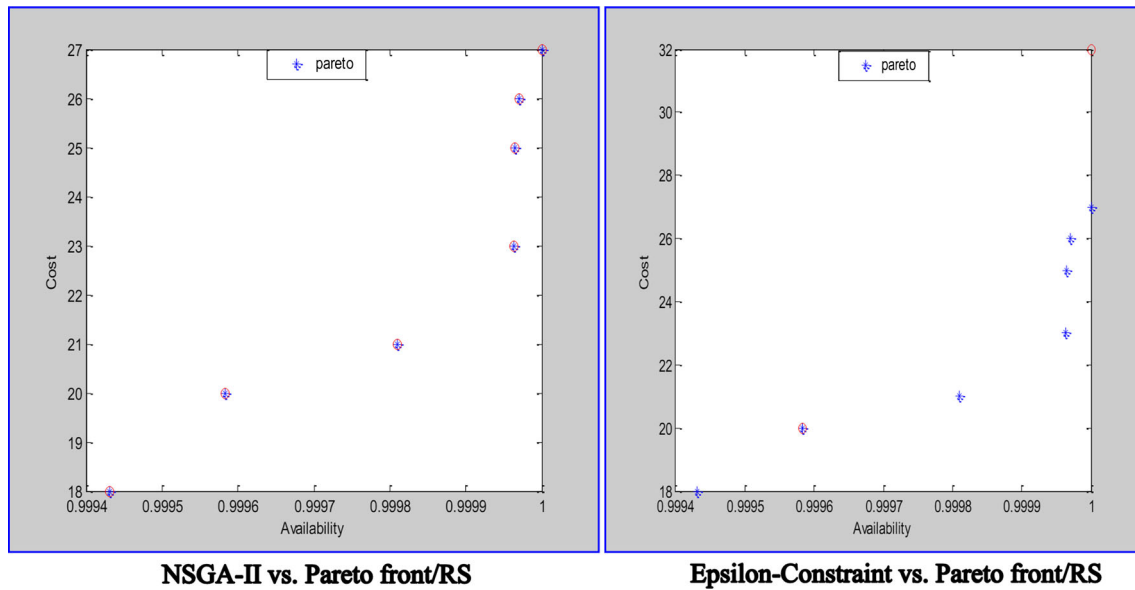
As shown in Fig. 3, the availability value of each component takes the maximum value at time zero and decreases over time. This is due to the fact that the components are in their best state at the start time. As time goes on, the components become less available due to component aging, until their availability becomes constant. This component behavior can be observed in all the test problems.

### 5.2 Results of the proposed NSGA-II and epsilon-constraint methods

In this subsection, the result of the customized NSGA-II and the epsilon-constraint methods is presented for all test problems.

#### 5.2.1 Results for the small-size test problems

The solutions generated for the small-size test problems by implementing the proposed customized version of NSGA-II and the epsilon-constraint method together with the RS



**Fig. 4** Pareto front for the small-size test problems

**Table 2** Comparison metrics for the test problems

Case	Accuracy metric						Diversity metric	
	NNS		ER		GD		SM	
	$\epsilon$ -Constraint	NSGA-II	$\epsilon$ -Constraint	NSGA-II	$\epsilon$ -Constraint	NSGA-II	$\epsilon$ -Constraint	NSGA-II
Small size	1	7	0.5	0	2.5	0	3.5355	0
Medium size	0	20	1	0.1667	18.5	0.1667	24.7487	0.4815
Large size	N/A	5	N/A	0.8980	N/A	0.2449	N/A	0.4345

are represented in Fig. 4, while the corresponding accuracy and diversity metrics are displayed in Table 2.

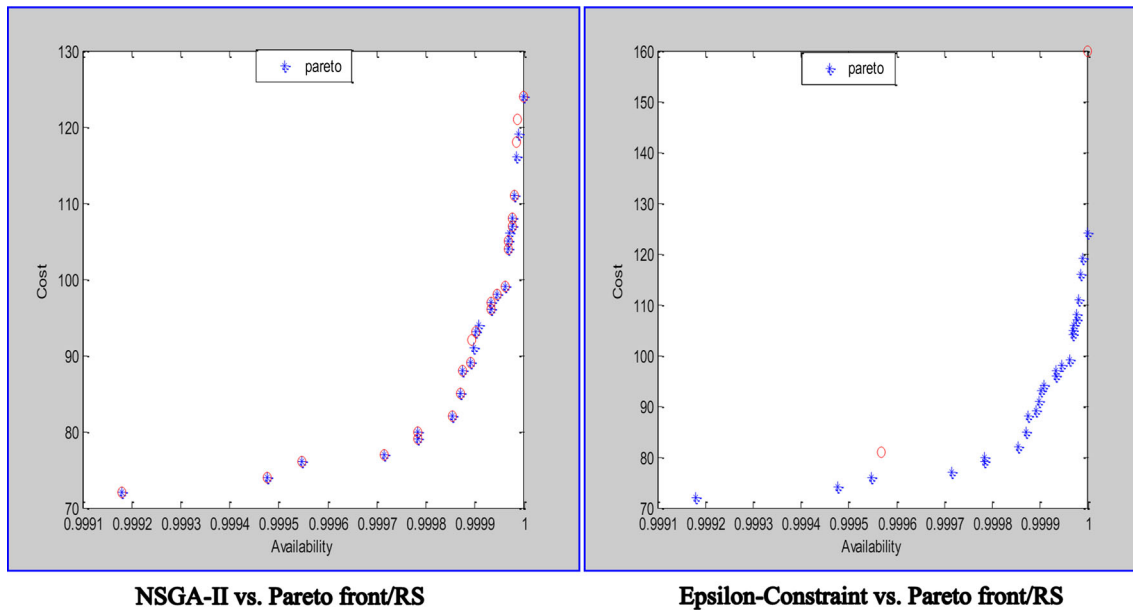
Relative to the small-size test problems, Table 2 shows that the NNS for the NSGA-II method is much higher than the NNS for the epsilon-constraint method. That is, the number of non-dominated solutions generated by the NSGA-II method is higher than that generated by the epsilon-constraint method. The ER measure is zero for the customized NSGA-II method and much smaller than the ER measure for the epsilon-constraint method. The zero value obtained for ER when implementing NSGA-II indicates the fact that the NSGA-II method allows for a complete convergence of the solutions toward the Pareto front in the considered small-size test problems. The GD value in the NSGA-II method not only is much smaller than the GD value in the epsilon-constraint method, but equals zero. Therefore, the distance between the Pareto front/RS and the solution set generated in the customized NSGA-II method is minimal. The SM value in the NSGA-II method is zero and much smaller than that in the epsilon-constraint method,

which means that the non-dominated solutions obtained by the NSGA-II method are considerably more uniformly distributed than those obtained by the epsilon-constraint method.

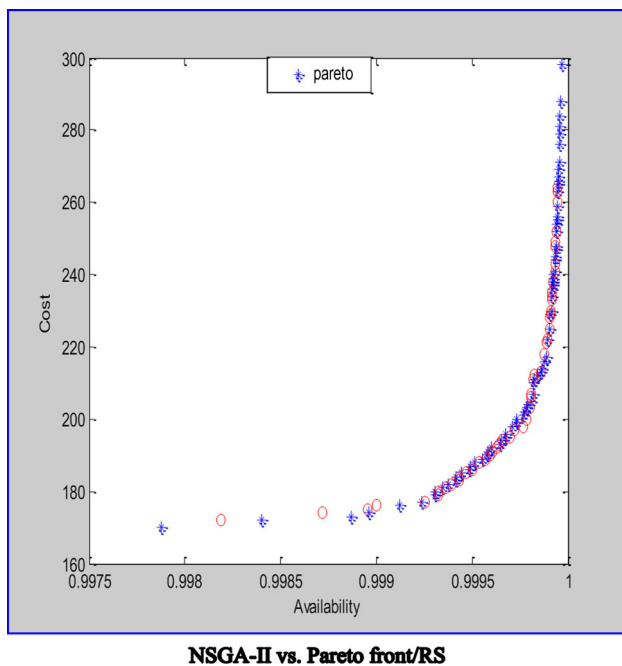
### 5.2.2 Results for the medium-size test problems

The solutions generated for the medium-size test problems for the NSGA-II and the epsilon-constraint methods together with the RS are represented in Fig. 5, while the corresponding accuracy and diversity metrics are displayed in Table 2.

As given in Table 2, in the medium-size case, the NNS for the customized NSGA-II method is significantly higher than the NNS for the epsilon-constraint method. The ER in the NSGA-II is very low compared to the ER in the epsilon-constraint method. The GD value in the customized NSGA-II method is considerably smaller than the GD value in the epsilon-constraint method. The SM measure in the NSGA-II is very small in comparison with the SM measure in the epsilon-constraint.



**Fig. 5** Pareto front for the medium-size test problems



**Fig. 6** Pareto front for the large-size test problems

5.2.3 Results for the large-size test problems

Figure 6 represents the solutions generated for the large-size test problems for the NSGA-II and the epsilon-constraint methods together with the RS. The accuracy and diversity metrics for the large-size test problems are displayed again in Table 2.

The epsilon-constraint method was unable to solve the problem for the large-size instances. Therefore, the graph

of Fig. 6 and the comparison metrics of Table 2 show solutions and values relative only to the NSGA-II method. Although the NNS for the NSGA-II method for the large instances is low, the GD and the SM metrics are acceptable. The GD measure shows that the distance between the solution sets obtained by the NSGA-II method and the RS is very small. Finally, the SM value shows that the non-dominated solutions obtained by the NSGA-II are diverse.

5.3 Computational CPU time

Table 3 reports the average CPU time of the test problems for the NSGA-II method. The NSGA-II was run ten times and the average computing time reported.

Table 4 presents the average CPU time of the test problems for the epsilon-constraint method. The CPU time in the epsilon-constraint method consists of two average time calculations. The first average time corresponds to the time needed for solving the differential equations using the MATLAB software, and the second average time accounts for the time necessary to solve the problem with the LINGO software.

It must be observed that a comparison of the computational CPU time between the two methods is not feasible. In fact, in order to solve the problem with the epsilon-constraint method, we must first solve the differential equations in MATLAB. Thus, considering that some time is needed to change from one software to the other, the real time for solving the test problems is actually larger than the sum of the times which are presented in Table 3. In addition, although the sum of the average times for the epsilon-

**Table 3** CPU time for the customized NSGA-II

Case	Number of iteration	Average of CPU time
Small size	50	11.402933
Medium size	500	118.3679657
Large size	1000	239.5948666

**Table 4** CPU time for the epsilon-constraint method

Case	Average time in MATLAB	Average time in LINGO
Small size	4.109219	4.5
Medium size	4.708507	4
Large size	6.462567	N/A

constraint method is less than the corresponding times for NSGA-II method for the small and medium-size test problems, the NNS for the epsilon-constraint method is very small while the other accuracy and diversity metrics for the large-size test problems are not even acceptable.

## 6 Conclusions and future research direction

We proposed a multi-objective mixed-integer nonlinear programming to solve repairable multi-state multi-objective redundancy allocation problems (RMMRAPs). Despite being among the extensions of RAPs which are the closest to real-life availability engineering problems, to the best of our knowledge, this class of problems has not been receiving the adequate attention in the RAP literature.

A customized version of the NSGA-II method was introduced in order to generate a set of non-dominated solutions on the Pareto front for RMMRAPs. Several test instances were designed and simulated to compare the proposed NSGA-II method to an epsilon-constraint method on several benchmark instances by using a set of well-known multi-objective comparison metrics. The Pareto front was generated for all instances together with a reference set (RS) using both solution procedures based on several runs. The performance of both solution procedures was compared with the RS using diversity and accuracy metrics.

The proposed customized version of the NSGA-II method showed better accuracy and diversity when compared to the epsilon-constraint method. In particular, although the total computational times for the epsilon-constraint method were less than the corresponding times for NSGA-II method for small- and medium-size test problems, the NNS obtained by the epsilon-constraint

method was very small and the other accuracy and diversity metrics for large-size test problems were not even acceptable.

Future work may include the selection of other redundancy strategies such as hot-standby or cold-standby strategies in place of the active redundancy strategy addressed in this study. Furthermore, additional objective functions (e.g., weight) could be considered in future formulations of the problem. Considering dynamic and different demand levels of performance could also provide interesting extensions of the model proposed in this study.

**Acknowledgements** The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions.

## References

1. Agarwal M, Gupta R (2006) Genetic search for redundancy optimization in complex systems. *J Qual Maint Eng* 12(4):338–353
2. Chern MS (1992) On the computational complexity of reliability redundancy allocation in a series system. *Oper Res Lett* 11(5):309–315
3. Coit DW, Smith AE (1996) Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. *Comput Oper Res* 23(6):515–526
4. Deb K, Pratap A, Agarwal S, Meyarivan TA (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6(2):182–197
5. Garg H, Sharma SP (2013) Multi-objective reliability-redundancy allocation problem using particle swarm optimization. *Comput Ind Eng* 64(1):247–255
6. Hwang CL, Masud ASM (1983) Multiple objective decision making methods and applications. Springer, New York
7. Khalili-Damghani K, Amiri M (2012) Solving binary-state multi-objective reliability redundancy allocation series-parallel problem using efficient epsilon-constraint, multi-start partial bound enumeration algorithm, and DEA. *Reliab Eng Syst Saf* 103:35–44
8. Khalili-Damghani K, Tavana M, Sadi-Nezhad S (2012) An integrated multi-objective framework for solving multi-period project selection problems. *Appl Math Comput* 219(6):3122–3138
9. Khalili-Damghani K, Tavana M, Abtahi AR (2013) A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems. *Reliab Eng Syst Saf* 111:58–75
10. Kuo W, Zuo MJ (2003) Optimal reliability modeling: principles and applications. Wiley, Hoboken
11. Li Z, Liao H, Coit DW (2009) A two-stage approach for multi-objective decision making with applications to system reliability optimization. *Reliab Eng Syst Saf* 94(10):1585–1592
12. Liang YC, Lo MH (2012) A variable neighborhood search algorithm with novel archive update strategies for redundancy allocation problems. *Eng Optim* 44(3):289–303
13. Liang YC, Smith E (2004) An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Trans Reliab* 53(3):417–423
14. Lisnianski A, Levitin G (2003) Multi-state system reliability assessment, optimization and applications. World Scientific, Singapore

15. Mavrotas G (2009) Effective implementation of the  $\epsilon$ -constraint method in multi-objective mathematical programming problems. *Appl Math Comput* 213(2):455–465
16. Misra KB, Ljubojevic MD (1973) Optimal reliability design of a system: a new look. *IEEE Trans Reliab* 22(5):255–258
17. Mousavi SM, Alikar N, Niaki STA, Bahreininejad A (2015) Two tuned multi-objective meta-heuristic algorithms for solving a fuzzy multi-state redundancy allocation problem under discount strategies. *Appl Math Model* 39(22):6968–6989
18. Mousavi SM, Alikar N, Niaki STA, Bahreininejad A (2015) Optimizing a location allocation-inventory problem in a two-echelon supply chain network: a modified fruit fly optimization algorithm. *Comput Ind Eng* 87:543–560
19. Mousavi SM, Alikar N, Niaki STA (2016) An improved fruit fly optimization algorithm to solve the homogeneous fuzzy series-parallel redundancy allocation problem under discount strategies. *Soft Comput* 20(6):2281–2307
20. Onishi J, Kimura S, James RJW, Nakagawa Y (2007) Solving the redundancy allocation problem with a mix of components using the improved surrogate constraint method. *IEEE Trans Reliab* 56(1):94–101
21. Ramirez-Marquez JE, Coit DW, Konak A (2004) Redundancy allocation for series-parallel systems using a max-min approach. *IIE Trans* 36(9):891–898
22. Safari J (2012) Multi-objective reliability optimization of series-parallel systems with a choice of redundancy strategies. *Reliab Eng Syst Saf* 108:10–20
23. Safari J, Tavakkoli-Moghaddam R (2010) A redundancy allocation problem with the choice of redundancy strategies by a memetic algorithm. *J Ind Eng Int* 6(11):6–16
24. Sheikhalishahi M, Ebrahimipour V, Shiri H, Zaman H, Jaihoo-nian M (2013) A hybrid GA–PSO approach for reliability optimization in redundancy allocation problem. *Int J Adv Manuf Technol* 68(1–4):317–338
25. Tavakkoli-Moghaddam R, Safari J, Sassani F (2008) Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm. *Reliab Eng Syst Saf* 93:550–556
26. Tavana M, Khalili-Damghani K, Abtahi AR (2013) A new variant of fuzzy multi-choice knapsack for project selection problem. *Ann Oper Res* 206(1):449–483
27. Tavana M, Abtahi AR, Khalili-Damghani K (2014) A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems. *Exp Syst Appl* 41(4):1830–1846
28. Yu X, Gen M (2010) *Introduction to evolutionary algorithms*. Springer, London
29. Zhao JH, Liu Z, Dao MT (2007) Reliability optimization using multiobjective ant colony system approaches. *Reliab Eng Syst Saf* 92:109–120
30. Zio E (2009) Reliability engineering: old problems and new challenges. *Reliab Eng Syst Saf* 94:125–141