Reliability Engineering and System Safety 111 (2013) 58–75

Contents lists available at SciVerse ScienceDirect

Reliability Engineering and System Safety

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

A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems

Kaveh Khalili-Damghani [a,*], Amir-Reza Abtahi [1,b], Madjid Tavana [2,c]

[a] Department of Industrial Engineering, South-Tehran Branch, Islamic Azad University, Tehran, Iran
[b] Department of Knowledge Engineering and Decision Sciences, Faculty of Economic Institutions Management, University of Economic Sciences, Tehran, Iran
[c] Business Systems and Analytics, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA

ARTICLE INFO

Article history:
Received 28 October 2011
Received in revised form
1 October 2012
Accepted 26 October 2012
Available online 5 November 2012

Keywords:
Multi-objective redundancy allocation problem
Meta-heuristics
Dynamic self-adaptive multi-objective particle swarm optimization
ε-constraint method
NSGA-II

ABSTRACT

In this paper, a new dynamic self-adaptive multi-objective particle swarm optimization (DSAMOPSO) method is proposed to solve binary-state multi-objective reliability redundancy allocation problems (MORAPs). A combination of penalty function and modification strategies is used to handle the constraints in the MORAPs. A dynamic self-adaptive penalty function strategy is utilized to handle the constraints. A heuristic cost-benefit ratio is also supplied to modify the structure of violated swarms. An adaptive survey is conducted using several test problems to illustrate the performance of the proposed DSAMOPSO method. An efficient version of the epsilon-constraint (AUGMECON) method, a modified non-dominated sorting genetic algorithm (NSGA-II) method, and a customized time-variant multi-objective particle swarm optimization (cTV-MOPSO) method are used to generate non-dominated solutions for the test problems. Several properties of the DSAMOPSO method, such as fast-ranking, evolutionary-based operators, elitism, crowding distance, dynamic parameter tuning, and tournament global best selection, improved the best known solutions of the benchmark cases of the MORAP. Moreover, different accuracy and diversity metrics illustrated the relative preference of the DSAMOPSO method over the competing approaches in the literature.

1. Introduction

The utilization of redundancy is one of the most important attributes in meeting high-level reliability. The problem is to select the feasible design configuration that optimizes the measurement functions such as reliability, cost, weights, and risk [10]. This is called the reliability redundancy allocation problem (RAP) which was first introduced by Misra and Ljubojevic [17]. A series-parallel system is characterized through a predefined number of sub-systems which are connected serially. Multiple component choices and redundancy levels are available to connect in parallel for each sub-system [10]. A given component may have a binary-state or a multi-state in the RAPs [13]. In binary-state RAP, the problem of a proper structure can be handled by increasing the reliability of components or supplying parallel redundant components at some stages [10]. In some other cases, called multi-state systems, the states of a given component may follow more than two different levels, ranging from perfectly working to completely failed [1].

The RAP is assumed to be a NP-hard (non-deterministic polynomial-time hard) problem [3]. The application and the development of the meta-heuristic procedures are assumed to be useful to properly solve NP-hard problem. Different heuristic and meta-heuristic methods such as Evolutionary Computation methods, variable neighborhood search, ant colony optimization, and particle swarm optimization (PSO) were proposed in this area ([4,7,10,14,15,19,21,25]). Gen and Yun [7] surveyed the Genetic Algorithm-based (GA-based) approaches for various reliability optimization problems. Konak et al. [11] presented an overview and tutorial describing GA developed specifically for problems with multiple objectives. Li et al., 2009 [15] proposed a two-stage approach for solving multi-objective system reliability optimization problems. In the first stage, a Multi-Objective Evolutionary Algorithm (MOEA) generated non-dominated solutions. Then, a Self-Organizing Map (SOM) was used to cluster similar solutions.

Abbreviations: ANOVA, analysis of variance; AUGMECON, efficient epsilon-constraint method; CI, confidence intervals; *c*TV-MOPSO, customized time-variant multi-objective particle swarm optimization; DM, decision maker; DiM, diversification metric; DSAMOPSO, dynamic self-adaptive multi-objective particle swarm optimization; ER, error ratio; *GA*, Genetic Algorithms; GD, generational distance; MODM, multi-objective decision making; MOEA, Multi-Objective Evolutionary Algorithm; *MORAP*, multi-objective reliability redundancy allocation problem; NNS, number of non-dominated solutions; NSGA-II, non-dominated sorting genetic algorithm; PSO, particle swarm optimization; RAP, redundancy allocation problem; RS, reference set; SM, spacing metric; SOM, self-organizing map; Std. Dev., standard deviation

* Corresponding author. Tel.: +98 912 3980373; fax: +98 21 77868749.
E-mail addresses: kaveh.khalili@gmail.com (K. Khalili-Damghani), amir_abtahi@yahoo.com (A.-R. Abtahi), tavana@lasalle.edu (M. Tavana).
[1] Tel.: +98 912 1887920; fax: +98 21 44453640.
[2] Tel.: +215 951 1129; fax: +267 295 2854.

0951-8320/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ress.2012.10.009

Finally, a DEA method is presented to select the most efficient solutions in each cluster.

We present an efficient dynamic self-adaptive multi-objective particle swarm optimization (DSAMOPSO) method to solve the binary-state multi-objective reliability redundancy allocation problems (MORAPs). The proposed method serves different properties which improve the process of re-generating the original Pareto front of the MORAP with minimum error and maximum diversity. The proposed procedure utilizes dynamic parameter-tuning, self-adaptive penalty functions, heuristic-based modification, and evolutionary-based re-production operators to search the solution space of the MORAPs efficiently.

An adaptive survey is conducted to find out the performance of the DSAMOPSO method in comparison with other well-known approaches in the literature. First, different sets of test problems and a well-known benchmark case of the MORAP are selected. Then, three different procedures as well as the DSAMOPSO method are customized and supplied to re-generate the Pareto front of the considered cases under controlled conditions. The proposed procedures are a modified version of the epsilon-constraint (AUGMECON) method, a well-known MOEA method called the non-dominated sorting genetic algorithm-II (NSGA-II), and a customized time-variant multi-objective particle swarm Optimization (cTV-MOPSO) method.

The properties of the DSAMOPSO method such as fast-ranking, elitism, crowding distance, and a global best selection procedure makes it competitive among the other procedures. Moreover, a dynamic self-adaptive penalty function, which considers the iteration of the algorithm and swarm situation, respectively, results in the relative preference of the DSAMOPSO method over the competing approaches in the literature. The experimental results and statistical analysis reveal that the DSAMOPSO method outperforms the competing methods.

The rest of the paper is arranged as follows. A brief literature of the multi-objective decision making (MODM) concepts, the epsilon-constraint method, the NSGA-II method, the particle swarm optimization (PSO) method, and its multi-objective version are presented in Section 2. The MORAP definition, the main customization of the AUGMECON method, the NSGA-II method, the cTV-MOPSO method and the DSAMOPSO method are presented in Section 3. In Section 4, different test problems and a well-known benchmark case of the MORAP, parameter-tuning, software implementation, and comparison metrics are introduced. The results from the experimental results and statistical analysis are presented in Section 5. In Section 6, we present our conclusions and future research directions.

2. Literature review

Formally, a multi-objective decision making (MODM) model considers a vector of decision variables, objective functions, and constraints. Decision makers (DMs) attempt to optimize the objective functions. Since this problem rarely has a unique solution, DMs are expected to choose a solution from a set of efficient solutions. Generally, a MODM problem with minimum objective functions can be formulated as follows:

$$(MODM)\quad \begin{cases} \min f(x) \\ s.t.\quad x \in S = \{x \in R^n \,|\, g(x) \le b, x \ge 0\} \end{cases} \tag{1}$$

where, $f(x)$ represents k conflicting objective functions, $g(x) \le b$ represents m constraints, S is the feasible solution space, and x is the n-vector of decision variables, $x \in R^n$ [9].

During the decision-making process, some DM's preference articulation may be required. The type and the time of preference articulation play a critical role in the actual decision-making method. Under this consideration, the methods for solving MODM problems have been systematically classified into four classes [9]. In one of the aforementioned classes, when there is a posterior preference articulation of information on the priority of objective functions, generating non-dominated solutions on the Pareto front of the MODM problem is desirable. The methods in this class strictly deal with the constraints and do not consider the preferences of the DMs. These are also called non-dominated solution generation methods. The ε-constraints method proposed by Chankong and Haimes [2] is a non-dominated solution generation method used to solve MODM problems.

2.1. Epsilon-constraint method

There are also methods that produce an entire efficient set for a special kind of MODM problems including linear programming. These methods can provide a representative subset of the Pareto set which in most cases is adequate. In this method, DM chooses one objective out of n to be optimized. The remaining objectives are constraints to be less than or equal to given target values. In mathematical terms, DM letting $f_j(x), j \in \{1,...,k\}$ be the objective function chosen to be optimized, we have the following problem $P(\varepsilon_j), j \in \{1,...,k\}$:

$$\min\left\{f_j(x), j \in \{1,...,k\}; f_i(x) \le \varepsilon_i, \forall i \in \{1,...,k\}, i \ne j; x \in S\right\}. \tag{2}$$

where, S is the feasible solution space.

One advantage of the ε-constraints method is that it is able to achieve efficient points in a non-convex Pareto curve. Therefore, the DM can vary the upper bounds ε_i to obtain weak Pareto optima. Clearly, this is also a drawback of this method (i.e., the DM has to choose appropriate upper bounds for the $\varepsilon_i$values). Moreover, the method is not particularly efficient as the number of the objective functions increases. Several research projects are dedicated to improving the ε-constraint method [16]. The traditional ε-constraint method tries to obtain the efficient solutions in a problem through parametrical variations in the RHS of the constraints. More formally, the ε-constraint method has three points that need attention in its implementation: (a) the calculation of the range of the objective functions over the efficient set; (b) the guarantee of efficiency of the obtained solution and (c) the increased solution time for problems with several (more than two) objective functions. Mavrotas [16] tried to address these three issues with a novel version of the ε-constraint method. The aforementioned drawbacks of the classical MODM procedures motivated us to develop the heuristic and the meta-heuristic procedures proposed in this study.

2.2. Particle swarm optimization

In nature, birds seek food by considering their personal experience and the knowledge of the other birds in the flock. This idea motivated Kennedy and Eberhart [12] to propose the PSO method. More formally, the PSO method is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. The mechanism used to search the solution space in the PSO differs from the evolutionary computations. The simplicity and the applicability of the PSO method have added to the popularity of this method for solving a large number of engineering and management optimization problems.

2.2.1. Mathematics of the PSO

Let us consider a swarm which includes m particles which are seeking the optimum value of the objective function(s) in an n-dimensional search space, each particle having a vector of

position $\vec{X}_i=(x_{i1},x_{i2},...,x_{in}),\quad i=1,2,...,m$ which is associated with a solution, a vector of velocity $\vec{V}_i=(v_{i1},v_{i2},...,v_{in}),\quad i=1,2,...,m$ which determines the movement value of a particle in each dimension to improve its current position, and a vector of particle best position $\vec{P}_i=(p_{i1},p_{i2},...,p_{in}),\quad i=1,2,...,m$ which is associated with most fitted positions of a particle from the first step of the algorithm. It is notable that the fitness of a position can easily be calculated considering the objective function of the optimization problem. A vector of the global best particle $\vec{P}_g=\left(p_{g1},p_{g2},...,p_{gn}\right)$ is reserved for knowledge sharing among all particles of a swarm. Using the aforementioned notations, each argument of the velocity and position vector for each particle in the swarm is updated through the multiple iterations of the algorithm using the following model:

$$\vec{V}_i(t+1)=W\times\vec{V}_i(t)+C_1r_1\left(\vec{P}_i-\vec{X}_i(t)\right)+C_2r_2\left(\vec{P}_g-\vec{X}_i(t)\right),\quad i=1,2,...,m,$$

$$\vec{X}_i(t+1)=\vec{X}_i(t)+\vec{V}_i(t+1),\quad i=1,2,...,m \tag{3}$$

where, W is the inertia weight and determines the tendency of a particle to maintain its previous exploration direction [20], C_1 and C_2 are cognitive and social factors, respectively; $r_1,r_2\in[0,1]$ are the random numbers, and t represents the iteration number.

2.2.2. Multi-objective PSO

The multi-objective procedures, including the meta-heuristics, should supply two main properties. The first one is generating high quality non-dominated solutions on the Pareto front of the MODM problem. The second one concerns a proper diversity for the generated solutions on the Pareto front of MODM problem. Simplicity and successfulness of the simple PSO among the other meta-heuristic procedures resulted in developing its multi-objective variants. The developed procedures in this area are roughly categorized in five main classes of which are aggregating, lexicographic, sub-population, Pareto-based, and Combined [18].

2.3. Non-dominated Sorting genetic algorithm (NSGA-II)method

The NSGA-II method, initially introduced by Deb et al. [6], is a well-known MOEA. The NSGA-II method contains several characteristics such as elitism, fast non-dominated sorting and diversity maintenance along with the Pareto-optimal front. The NSGA-II method has been successfully used in a wide range of engineering, management and combinatorial optimization problems. The NSGA-II has also been used as a tool for validating comparison reference for other new developed/extended meta-heuristics. In this paper, the NSGA-II method is selected as a validated comparison reference to assess the performance of the proposed DSA-MOPSO method. A brief description of the customized NSGA-II method is provided here.

Each chromosome in the population is sorted into each front based on the non-domination. The first front contains only the non-dominant chromosomes; the chromosomes in the second front are dominated by the chromosomes in the first front and this pattern is repeated. Each chromosome in a front is assigned a ranking value in accordance with its front. The chromosomes in the first front are given a fitness value of 1, the chromosomes in the second front are assigned a fitness value of 2, and so forth.

A measure, called the crowding distance, is calculated for each individual in the NSGA-II population. The crowding distance is a measure of how close an individual is to its neighbors. The large average crowding distance will result in a better diversity in the population. Parents are selected from the population by using a binary tournament selection procedure based on their rank and crowding distance. The fitness of an individual, which belongs to a given rank, is less than others in the same rank with a greater crowding distance. The selected population generates offspring through crossover and diversification operators. All individuals, including current population and the off-springs, are sorted again based on non-domination. Only the best N individuals are selected, in which N is the population size. The selection is based on the rank and crowding distance on the last front. The details of the NSGA-II method and the modified NSGA-II method for the fuzzy classifier design can be found in Ref. [6].

3. Problem definition and modeling the MORAP

Let us revisit the same model proposed by Khalili-Damghani and Amiri [10]. In the binary-state MORAP, a set of objective functions such as reliability, cost, weight, and volume, are to be optimized considering a set of constraints like cost, weight, and volume. The basic assumption for a binary-state MORAP are as follows: (a) all components are assumed to be non-repairable; (b) all components are assumed to have binary states (i.e. working/fail); and (c) the functioning and physical properties (i.e. reliability, volume, weight, and cost) of all components are assumed to be known, deterministic, and time-independent. The following notations and parameters are used in the binary-state MORAP:

m	Number of sub-systems
i	Index of sub-systems, $i=1,2,..,m$
j	Index of components in each sub-systems, $j=1,2,..,n$
r_{ij}	Reliabilityofcomponent$j$ in sub-system i
c_{ij}	Costofcomponent$j$ in sub-system i
w_{ij}	Weightofcomponent$j$ in sub-system i
R_s	Overall reliability of the series−parallel system
C_s	Overallcostoftheseries - parallelsystem
W_s	Overallweightoftheseries - parallelsystem
C_o	Allowedcostofsystem
W_o	Allowedweightofsystem
a_i	Number of available component choices for sub-system i.
x_{ij}	Quantityofcomponent$j$ used in sub-system i
n_i	Total number of components used in sub-system i
$n_{\max}$	Maximumnumberofcomponentswhichcanbeinparallel
$n_{\min}$	Minimumnumberofcomponentswhichcanbeinparallel

Models (4)–(11) represent a binary-state MORAP in which (4)–(6) are allocated to describe the reliability, cost, and weight objective functions, respectively. The constraint (7) holds the maximum allowed cost of the system while constraint (8) is written for the weight of the system. Constraints (9) and (10) guarantee the minimum and maximum allowed number of components in each sub-system. The set of constraints (11) guarantees that the decision variables are positive integers.

$$\text{Max}R_s=\prod_{i=1}^{m}\left(1-\prod_{j=1}^{a_i}(1-r_{ij})^{x_{ij}}\right) \tag{4}$$

$$\text{Min}C_s=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij} \tag{5}$$

$$\text{Min}W_s=\sum_{i=1}^{m}\sum_{j=1}^{n}w_{ij}x_{ij} \tag{6}$$

$$\text{s.t.}\quad\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}\le C_o \tag{7}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}x_{ij} \le W_o \tag{8}$$

$$\sum_{j=1}^{a_i} x_{ij} \le n_{\max}, \quad i=1,2,...,m \tag{9}$$

$$\sum_{j=1}^{a_i} x_{ij} \ge n_{\min}, \quad i=1,2,...,m \tag{10}$$

$$x_{ij} \in Z^+, \quad i=1,2,...,m; \quad j=1,2,...,n \tag{11}$$

As mentioned earlier, a series-parallel system is a compound design that uses both series and parallel connections. Fig. 1, presents the schematic view of a series-parallel system.where n_i is the number of components in each sub-system, and m is the number of sub-systems.

3.1. Efficient epsilon-constraint (AUGMECON) method

The application of the AUGMECON method [16] on Models (4)–(11) results in the following models:

$$\text{Max } R_s+\beta\times(S_2/r_2+S_3/r_3) \tag{12}$$

$$\text{s. t.} \quad C_s+S_2=\varepsilon_2, \quad \varepsilon_2\in[C_s^-,C_s^+] \tag{13}$$

$$W_s+S_3=\varepsilon_3, \quad \varepsilon_3\in[W_s^-,W_s^+] \tag{14}$$

$$X\in S \tag{15}$$

where $r_i, i=2,3$ represent the range of the objective i which has been calculated from the lexicographic payoff table of the original binary-state MORAP (i.e., using the Ideal and Nadir value of C_s^+,C_s^-,W_s^+,W_s^-). $X\in S$is the feasible region of the original binary-state MORAP (i.e., relations (7)–(11)) and β is a small positive number (usually between 0.001 and 0.000001). In real-world problems with k objectives, similar to ours which has 3 conflicting objectives, Models (4)–(11) results in k-1 ε-constraints.

3.2. Non-dominated sorting genetic algorithm II for the MORAP optimization

In this section we briefly describe the customized NSGA-II method for the MORAP. Since the MORAP models (4)–(11) are concerned with the optimization of three objectives (i.e. reliability, cost, and weight), the conventional NSGA-II method proposed by Deb et al. [6] cannot be utilized. The NSGA-II method should be customized to generate several non-dominated solutions for the MORAP with three objective functions and several constraints. The operating mechanism of the NSGA-II method in reproduction, crossover, diversification, and selection has been widely discussed [6]. Therefore, we illustrate the main customization of the modified NSGA-II method here.

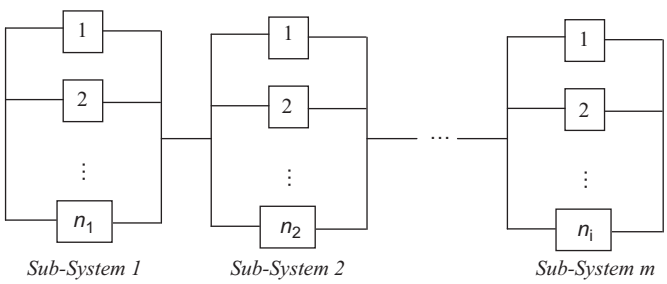

Fig. 1. Series-parallel system.

3.2.1. Handling the constraints in the NSGA-II method

A combination of the modification and penalty strategies are considered in the customized NSGA-II method to handle the MORAP constraints. Constraints (7)–(8) in the MORAP are handled using a penalty strategy. In this strategy, the violation value of each candidate solution in the population is calculated as follows:

$$o_{i1}=\text{Max}\left\{0,\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}-C_o\right\} \tag{16}$$

$$o_{i2}=\text{Max}\left\{0,\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}x_{ij}-W_o\right\} \tag{17}$$

where c_{ij}, x_{ij}, w_{ij}, C_o, and W_o have the same definition as the MORAP (4)–(11), and o_{i1} and o_{i2} are violation values of the cost and weight of chromosome i in the population, respectively. Then, the cost and weight objectives for the violated chromosomes in the population are penalized as follows:

$$C'_{si}=C_{si}+o_{i1}\times M \tag{18}$$

$$W'_{si}=W_{si}+o_{i2}\times M \tag{19}$$

where C'_{si} and W'_{si} are penalized objectives for the violated chromosome i in the population, and M is a very large positive number.

Constraints (9)–(10) in the MORAP are handled using the modification strategy. The designs, in which the upper bound ($n_{\max}$) and the lower bound ($n_{\min}$) of the allowed component in each sub-system are not met, are modified. If the number of components in a sub-system is larger than $n_{\max}$, the surplus components will be randomly deleted in order to meet the constraints. If the number of components in a sub-system is smaller than $n_{\min}$, the shortage components will be randomly added in order to meet the constraints. In the latter case, the violation procedure (i.e., (16)–(19)) is immediately run in order to check whether the chromosome violates the cost and the weight objectives. The modification strategy should be held during the population initialization, crossover, and diversification.

3.3. Customized time-variant multi-objective particle swarm optimization method

In this section, we customize the approach proposed by Tripathi et al. [22] and propose the cTV-MOPSO method. The main properties of the proposed method by Tripathi et al. [22] are as follows:

- The TV-MOPSO is made adaptive in nature by allowing its vital parameters (viz., inertia weight and acceleration coefficients) to change with iterations. This adaption helps the algorithm to explore the search space more efficiently.
- A new diversity parameter is used to ensure sufficient diversity amongst the solutions of the non-dominated fronts, while retaining the convergence to the Pareto-optimal front concurrently.

The details of the algorithm can be found in Tripathi et al. [22]. Unfortunately, the proposed method by Tripathi et al. [22] cannot handle constraints. Therfore, it is incapable of solving real-world optimization problems. We revisited this algorithm and implemented it according to several constraints in the MORAP models (4)–(11). The main properties of the cTV-MOPSO method including constraint handling were not considered in the original TV-MOPSO method proposed by Tripathi et al. [22].

3.3.1. Handling the constraints in the cTV-MOPSO method

The constraint handling procedure defined for the NSGA-II method in Section 3.2.1 is also used in the cTV-MOPSO method.

3.4. Dynamic self-adaptive multi-objective particle swarm optimization method

In this section we discuss the fundamental principles of the proposed DSAMOPSO method. The DSAMOPSO method is designed to efficiently handle different constraints in the MORAP (i.e., model (4)–(11)). The main properties of the DSAMOPSO method are dynamic and self-adaptive constraint handling, dynamic parameter tuning, heuristic cost-benefit ratio, diversification, and global best selection procedure.

Dynamic penalty function. A dynamic constraint handling methodology is considered in the DSAMOPSO method. The penalty values are calculated dynamically considering the iteration number of the algorithm in a way that the penalty values will increase as the iteration of the algorithm continues.
Self-adaptive penalty values. The penalty values are calculated in a self-adaptive manner, considering both the violation value of each particle and the minimum violation value of the swarm.
Dynamic parameter tuning. The inertia weight, cognitive factor, and the social factor are determined using the linear functions of the iteration of the algorithm.
Heuristic cost-benefit ratio. Constraints (9) and (10) of the MORAP are handled using the modification strategy. A heuristic ratio is supplied to modify the structure of the violated swarms. The component which has a smaller cost-benefit ratio has a higher priority to be pruned.
Diversification. The PSO method suffers from the problem of premature convergence [22]. In order to overcome this problem, the DSAMOPSO method makes use of a diversification operator to maintain the level of diversity in the swarm population, thereby maintaining a good balance between the exploration and exploitation phenomena and preventing premature convergence. Diversification operators (also known as '*mutation operator*' in genetic algorithms) are used in the PSO methods to boost the exploration capability [5], to maintain the diversity of the swarm [24], and to prevent premature convergence [8].
Global best selection procedure. The global best selection is based on a binary tournament using a fast non-dominated sorting and crowding distance. Each particle of the swarm is assigned a rank based on its front, and then the global best particle is selected based on a binary tournament and using the crowding distance from top ranked fronts. The main advantages of the DSAMOPSO method in comparison with the cTV-MOPSO method and the NSGA-II method are summarized as follows:

- The DSAMOPSO method uses a dynamic self-adaptive penalty function which results in a better performance.
- A cost-benefit ratio is supplied in the DSAMOPSO method in order to modify the violated solutions in an effective manner
- The proposed DSAMOPSO method uses a modified genetic-based operator in order to search the complicated solution space of the MORAP more effectively.

3.4.1. Structure of a particle in the DSAMOPSO method for the MORAP

An integer-coded structure is supplied to depict a particle in the proposed DSAMOPSO method. The structure is presented in Fig. 2.where, each allele n_{ij} is the number of component type j in sub-system i, and $n_{\min} \le \sum_{j=1}^{|J|} n_{ij} \le n_{\max}$, $i=1,2,...,|I|$is preserved for all sub-systems. It is also notable that n_{ij} has the same definition as x_{ij} in the mathematical formulation of the MORAP in models (4)–(11).

3.4.2. Position and velocity vector of the DSAMOPSO method

Re-considering (3), the position and the velocity vector of a particle are respectively defined as follows:

$$\vec{V}_i(t+1) = W(t) \times \vec{V}_i(t) + C_1(t) r_1 \left(\vec{P}_i - \vec{X}_i(t) \right) + C_2(t) r_2 \left(\vec{P}_g - \vec{X}_i(t) \right), \quad i=1,2,...,m,$$

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1), \quad i=1,2,...,m \tag{20}$$

where, the parameters of (20) have the same definition as the parameters in Eq. (3). $W(t)$, $C_1(t)$, and $C_2(t)$ are determined using an iteration function. The random numbers r_1 and r_2 in Eq. (3) are generated independently for each particle.

3.4.3. Dynamic inertia weight, cognitive and social factors

The value of the inertia weight is determined dynamically by considering the algorithm iteration numbers between W_1 and $W_2 < W_1$ as follows [22]:

$$W(t) = (W_1 - W_2) \frac{\text{Max } t - t}{\text{Max } t} + W_2 \tag{21}$$

where, W_1 and W_2 are two parameters, *Max* t is the maximum allowed number of iterations and t is the current iteration number. The best values for W_1 and W_2 are experimentally determined as suggested by Tripathi et al. [22]. In order to improve the compromise between the exploration and exploitation phases in the PSO, time variant acceleration coefficients are supplied. The values of the cognitive and the social factors are also determined dynamically by considering the algorithm iteration number as follows [22]:

$$C_1(t) = (C_{1f} - C_{1i}) \frac{t}{\text{Max } t} + C_{1i} \tag{22}$$

$$C_2(t) = (C_{2f} - C_{2i}) \frac{t}{\text{Max } t} + C_{2i} \tag{23}$$

C_1 is allowed to decrease from an initial value, called C_{1i}, to a final value, called C_{1f}. C_2 is increased from an initial value, called C_{2i}, to a final value, called C_{2f}. The best values of C_{1i}, C_{1f}, C_{2i}, and C_{2f} are experimentally determined similar to Tripathi et al. [22]. We should note that*Max* t and t have the same definition as in Eq. (21).

3.4.4. Handling the constraints in the proposed DSAMOPSO method

A combination of the modification and penalty strategies are considered in the DSAMOPSO method to handle the constraints.

Penalty function strategy. The constraints (7) and (8) in the MORAP are handled using a dynamic self-adaptive penalty function. In this strategy, the iteration of the algorithm and the overall situation of the swarm are concurrently considered. The violation

Sub-System 1				Sub-System 2				...	Sub-System *i*			
Type 1	Type 2	...	Type *j*	Type 1	Type 2	...	Type *j*	...	Type 1	Type 2	...	Type *j*
n_{11}	n_{12}	...	n_{1j}	n_{21}	n_{22}	...	n_{2j}	...	n_{i1}	n_{i2}	...	n_{ij}

Fig. 2. Structure of a particle in swarm.

Table 1
Comparison between the mechanisms in the proposed meta-heuristics.

Method	Random initial Pop/ Swarm	Non-dominate ranking	Constraint handling			Fast sorting	Mutation	Tournament selection	Dynamic parameter tuning	Self-adaptive penalty	Global best selection
			Penalty	Modification							
				Random	Heuristic						
DSA-MOPSO	☑	☑	☑	□	☑	☑	☑	☑	☑	☑	☑
NSGA-II	☑	☑	☑	☑	□	☑	□	☑	☑	□	□
cTV-MOPSO	☑	☑	☑	☑	□	□	□	□	☑	□	☑

Table 2
Test problems.

Case	Component type	Number of sub-systems	Component Dimension			Number of Non-dominated solutions
			Reliability	Cost	Weight	
Small-size	U[2, 5]	U[2, 4]	U[0.5, 1]	U[2, 10]	U[2, 8]	U[3, 8]
Large-size	U[5, 15]	U[5, 10]	U[0.5, 1]	U[2, 10]	U[2, 8]	U[10, 18]

value of each particle in the swarm is calculated using Eqs. (16) and (17). Then, the dynamic self-adaptive penalty functions for the cost and the weight objectives are supplied as follows:

$$C'_{si} = C_{si} + \left[\left(\frac{o_{i1}}{o_{i1}^{\min}} \right)^{\alpha} \times t^{\beta} \right] \tag{24}$$

$$W'_{si} = W_{si} + \left[\left(\frac{o_{i2}}{o_{i2}^{\min}} \right)^{\alpha} \times t^{\beta} \right] \tag{25}$$

where, C'_{si} and W'_{si} are the modified values of the cost and the weight objectives for the violated particle i in the swarm; o_{i1} and o_{i2} have the same definition of Eqs. (16) and (17); $o_{i1}^{\min} = \min_i \{\varepsilon + o_{i1}\}$, and $o_{i2}^{\min} = \min_i \{\varepsilon + o_{i2}\}$ are the minimum violation values for each particle in the swarm; ε is a small positive value used to avoid division by zero; t is the iteration number; and α, β are the control parameters which are experimentally determined.

This type of dynamic self-adaptive penalty function allows for a more efficient search of the solution space and guarantees that the violations in the final steps of the algorithm are punished harder. Moreover, the penalty value has been calculated using the minimum violation of the swarm. The latter property allows for considering the best particles in the swarm.

Modification strategy. Constraints (9)–(10) in the MORAP are also handled using the modification strategy in the DSAMOPSO method. The designs in which $n_{\min} \le \sum_{j=1}^{|J|} n_{ij} \le n_{\max}, \quad i = 1,2,...,|I|$ are not met, are modified as follows:

$$\sum_{j=1}^{|J|} n_{ij} = \begin{cases} n_{\min}, & \text{if } \sum_{j=1}^{|J|} n_{ij} \le n_{\min}, \quad i = 1,2,...,|I| \\ \sum_{j=1}^{|J|} n_{ij} & \text{if } n_{\min} < \sum_{j=1}^{|J|} n_{ij} < n_{\max}, \quad i = 1,2,...,|I| \\ n_{\max} & \text{if } \sum_{j=1}^{|J|} n_{ij} \ge n_{\max}, \quad i = 1,2,...,|I| \end{cases} \tag{26}$$

These modifications should be in place during the swarm initialization, the swarm position updating, and the diversification of the swarm. The following may arise when $\sum_{j=1}^{|J|} n_{ij}$ is not in allowed an interval value: which component of the violated sub-systems should be pruned? In order to determine the proper

Table 3
Data for the benchmark case.

Component type j	Sub-system i								
	1			2			3		
	R	C	W	R	C	W	R	C	W
1	0.94	9	9	0.97	12	5	0.96	10	6
2	0.91	6	6	0.86	3	7	0.89	6	8
3	0.89	6	4	0.7	2	3	0.72	4	2
4	0.75	3	7	0.66	2	4	0.71	3	4
5	0.72	2	8	–	–	–	0.67	2	4

Table 4
Fitted parameters and user defined values of the algorithms for 50 non-dominated solutions in the archive.

AUGMECON		**cTV-MOPSO**	
Parameters:		*Parameters:*	
Archive size	50	Particle no.	20
Step size of cost	0.01	Archive size	50
Step size of weight	0.01	Maximum ieration no.	200
User defined values:		Initial C_1	2.5
Upper bound of cost	284	Final C_1	0.5
Upper bound of weight	192	Initial C_2	0.5
Lower bound of reliability	0.99	Final C_2	2.5
		W_1	0.7
		W_2	0.4
		Mutation rate	0.05
		User defined values:	
		Lower bound of reliability	0.9999
		Upper bound of reliability	1
		Upper bound of cost	284
		Upper bound of weight	192
NSGA-II parameters		**DSA-MOPSO parameters**	
Parameters:		*Parameters:*	
Chromosome no.	20	Particle no.	20
Archive size	50	Archive size	50
Maximum ieration no.	200	Maximum ieration no.	200
Cross rate	0.7	Initial C_1	2.5
Mutation rate	0.05	Final C_1	0.5
Alpha	1	Initial C_2	0.5
Beta	5	Final C_2	2.5
User defined values:		W_1	0.7
Lower bound of reliability	0.9999	W_2	0.4
Upper bound of reliability	1	Mutation rate	0.05
Upper bound of cost	284	Alpha	1
Upper bound of weight	192	Beta	5
		User defined values:	
		Lower bound of reliability	0.9999
		Upper bound of reliability	1
		Upper bound of cost	284
		Upper bound of weight	192

candidates for pruning, a cost-benefit heuristic is utilized in the DSAMOPSO method. A cost-benefit ratio is calculated for each component of the MORAP case. The reliability of the component is divided by the sum of the cost and the weight of the component. The ratio is calculated as follows:

$$Ratio_{ij} = \frac{r_{ij}}{c_{ij} + w_{ij}} \quad (27)$$

where,r_{ij}, c_{ij}, andw_{ij} are the reliability, the cost, and the weight of the component j in the sub-system i, respectively. The components with a lower $Ratio_{ij}$ have a higher pruning priority. Therefore, the values for the $Ratio_{ij}$ are ordered in an ascending order. The redundant components are pruned from the top of the sorted list. The shortage of the components is also supplied from the bottom of the sorted list. Table 1 compares the supplied mechanisms in the proposed meta-heuristic approaches.

3.4.5. Procedural algorithmic for the DSAMOPSO method

The proposed algorithm involves 21 steps grouped into three distinct processes as follows:

Initialization:
*Step*1. *Set the parameters of the DSAMOPSO method for the MORAP case.*
*Step*2. *Randomly generate an initial swarm considering constraints* (9) and (10).
*Step*3. *Modify the initial swarm usingEqs.* (26) and (27).
*Step*4. *Initialize the position, velocity, and the personal best values.*
*Step*5. *Calculate the violation values for each particle in the swarm using Eqs.* (16) and (17).
*Step*6. *Calculate the objective functions of each particle in the swarm using Eqs.* (4)–(6).
*Step*7. *Modify the objective functions using Eqs.* (24) and (25).
*Step*8. *Select the non-dominated solutions for the initial swarm and put them in the archive.*
*Step*9. *Select* $\vec{P}_g$ *from the current archive using a tournament selection procedure based on the crowding distance measure.*
Main Loop:
*Step*10. *Adjust the values of the parameters inertia coefficient, the local acceleration coefficient, and the global acceleration coefficient using Eqs.* (21)–(23), *respectively.*
*Step*11. *Update the position of each particle in the swarm using* (20) *and constraints* (9) and (10).
*Step*12. *Modify the swarm using Eqs.* (26) and (27).
*Step*13. *Calculate a violation value for each particle in the swarm using Eqs.* (16) and (17).
*Step*14. *Calculate the objective functions of each particle in the swarm using Eqs.* (4)–(6).
*Step*15. *Modify the objective functions using Eqs.* (24) and (25).
*Step*16. *Update the personal best for the i-th particle in the swarm.*
*Step*17. *Select* $\vec{P}_g$ *from the current archive and swarm using a tournament selection procedure based on a non-dominated rank and crowding distance measure.*
*Step*18. *Update the velocity vector of each particle in the swarm using the personal best and global best values.*
*Step*19. *Mutate some particles of the new born swarm.*

Table 5
Computational results of the accuracy metrics for the small-size test problems.

Problem	NNS				ER				GD			
	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	TV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	3	1	2	3	0.00	0.67	0.33	0.00	0.01	0.32	0.39	0.42
2	4	3	2	4	0.00	0.25	0.50	0.00	0.08	0.30	0.32	0.36
3	4	2	3	5	0.20	0.60	0.40	0.00	0.10	0.40	0.26	0.45
4	3	1	2	2	0.00	0.67	0.33	0.33	0.01	0.22	0.19	0.26
5	4	2	5	5	0.20	0.60	0.00	0.00	0.13	0.29	0.10	0.56
6	6	4	6	6	0.00	0.33	0.00	0.00	0.04	0.20	0.37	0.25
7	6	4	3	4	0.14	0.43	0.57	0.43	0.01	0.27	0.25	0.25
8	3	2	4	3	0.25	0.50	0.00	0.25	0.10	0.35	0.04	0.41
9	3	2	2	2	0.00	0.33	0.33	0.33	0.10	0.31	0.04	0.48
10	4	4	3	4	0.20	0.20	0.40	0.20	0.01	0.25	0.33	0.26
Ave.	4.00	2.50	3.20	3.80	0.10	0.46	0.29	0.15	0.06	0.29	0.23	0.37
Std. Dev.	1.15	1.18	1.40	1.32	0.11	0.17	0.21	0.17	0.05	0.06	0.13	0.11

Table 6
Computational results of the diversity metrics for the small-size test problems.

Problem	SM				DM			
	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	0.06	0.28	0.13	0.65	1.19	9.76	77.90	4.83
2	0.04	0.76	0.46	0.28	1.35	4.91	61.41	6.05
3	0.19	0.55	0.04	0.70	2.21	2.87	0.61	7.77
4	0.02	0.32	0.64	0.26	2.62	2.78	42.62	3.44
5	0.16	0.65	0.87	0.34	4.13	6.94	57.93	2.06
6	0.33	0.54	0.11	0.47	4.23	6.61	58.12	7.76
7	0.06	0.52	0.32	0.40	4.21	8.21	39.13	3.17
8	0.32	0.53	0.54	0.64	4.33	6.03	25.68	5.24
9	0.47	1.05	0.88	0.59	0.42	3.92	2.56	3.72
10	0.09	0.19	0.69	0.22	1.68	6.74	62.07	6.00
Ave.	0.17	0.54	0.47	0.45	2.64	5.88	42.80	5.00
Std. Dev.	0.15	0.25	0.31	0.18	1.49	2.27	26.07	1.93

Step20. If the termination conditions are not met go to step 10, otherwise go to Step 21.
Finalization:
Step21. Print the last Pareto front using the archived non-dominated solutions.

4. Test problems, parameter tuning, software implementation and comparison metrics

Two different sets of problems and a well-known benchmark case are considered to test and compare the performance of the DSAMOPSO, AUGMECON, cTV-MOPSO, and NSGA-II methods.

4.1. Test problems

A number of test problem sets each with 20 simulated cases are used here. The dimensions of these sets are presented in Table 2.

The properties of the test problems were simulated using uniform probability density functions. It is notable that $U[a, b]$ in Table 2 represent a uniform distribution function between a and b.

4.2. Well-known benchmark case

Next, a well-known benchmark case was selected from the literature [10,15,21]. This case consists of 3 subsystems with 5, 4 and 5 components in each subsystem. The problem requires

Table 7
CPU Times (seconds) of different approaches for the small-size test problems.

Problem	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	1.45	0.77	1.06	811.44
2	1.27	0.75	1.10	786.35
3	1.24	0.76	0.97	792.99
4	1.41	0.76	1.18	738.38
5	1.30	0.77	1.10	761.62
6	1.20	0.76	1.08	754.24
7	1.34	0.74	0.93	783.03
8	1.37	0.77	1.06	810.70
9	1.37	0.77	1.06	787.45
10	1.55	0.79	1.12	781.92
Ave.	1.35	0.76	1.07	780.81
Std. Dev.	0.10	0.01	0.07	23.42

Table 10
CPU Times (seconds) of different approaches for the large-size test problems.

Problem	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	78.42	41.40	56.91	43,765.62
2	68.46	40.40	59.40	42,412.25
3	66.87	41.00	52.56	42,770.50
4	75.83	41.00	63.44	39,824.92
5	70.06	41.60	59.40	41,078.78
6	64.68	41.00	58.15	40,680.73
7	72.45	39.81	50.38	42,233.13
8	74.04	41.40	57.22	43,725.82
9	74.04	41.40	57.22	42,471.96
10	83.39	42.39	60.33	42,173.42
Ave.	72.82	41.14	57.50	42,113.71
Std. Dev.	5.61	0.70	3.75	1263.04

Table 8
Computational results of the accuracy metrics for large-size test problems.

Problem	NNS				ER				GD			
	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	TV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	8	6	5	10	0.20	0.40	0.50	0.00	0.43	1.51	0.97	0.80
2	11	8	9	8	0.08	0.33	0.25	0.33	0.32	1.56	0.30	1.84
3	14	11	12	15	0.18	0.35	0.29	0.12	0.51	0.92	0.68	0.96
4	10	8	8	8	0.00	0.20	0.20	0.20	0.24	1.22	0.26	1.78
5	12	13	13	10	0.14	0.07	0.07	0.29	0.54	1.20	0.55	1.46
6	9	6	8	7	0.10	0.40	0.20	0.30	0.19	0.72	1.58	1.50
7	14	9	11	9	0.07	0.40	0.27	0.40	0.33	1.20	0.02	2.13
8	11	9	11	11	0.15	0.31	0.15	0.15	0.57	1.58	1.47	1.88
9	9	8	8	7	0.25	0.33	0.33	0.42	0.55	0.81	1.49	2.27
10	9	9	6	7	0.10	0.10	0.40	0.30	0.00	0.73	0.24	1.22
Ave.	10.70	8.70	9.10	9.20	0.13	0.29	0.27	0.25	0.37	1.15	0.76	1.58
Std. Dev.	2.11	2.11	2.60	2.49	0.07	0.12	0.12	0.13	0.19	0.34	0.58	0.49

Table 9
Computational results of the diversity metrics for the large-size test problems.

Problem	SM				DM			
	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	0.02	17.38	2.64	7.62	10.40	131.80	1460.20	90.80
2	7.88	9.76	17.32	9.70	14.40	122.20	1792.80	105.80
3	9.76	13.56	14.22	8.02	47.80	76.80	39.20	60.00
4	8.80	6.24	7.00	8.02	35.20	73.20	1332.20	106.00
5	8.22	20.22	9.12	7.06	67.60	74.80	392.20	46.60
6	0.66	3.70	6.34	8.60	83.00	140.80	66.60	37.60
7	8.68	9.22	6.78	14.18	90.40	89.80	958.00	93.80
8	5.82	18.86	6.24	7.30	56.40	131.20	1250.60	56.60
9	5.38	18.84	6.12	13.02	13.00	58.00	1502.00	62.80
10	3.88	16.52	14.06	7.90	89.20	117.80	1211.40	55.40
Ave.	5.91	13.43	8.98	9.14	50.74	101.64	1000.52	71.54
Std. Dev.	3.44	5.85	4.65	2.47	31.68	30.17	621.44	25.15

determining the optimum number of selected component types in each sub-system. The maximum number of components is 8 in each sub-system. Table 3 presents the information concerning the benchmark case [10,15,21].

4.3. Software-hardware implementation

The proposed AUGMECON method was coded using LINGO 12.0 and VBA for MS-Excel 12.0.The customized NSGA-II, cTV-MOPSO, and DSAMOPSO methods were coded using VBA for MS-Excel 12.0. All codes were run on a PIV Pentium portable PC with MS-Windows XP Professional, 1 GB of RAM, and 2.0 GHz Core 2 Due CPU.

4.4. Parameter tuning and estimation of the Pareto front

Although the implementation of the aforementioned approaches on the MORAP's cases revealed their ability to generate non-dominated solutions, generating a part of the Pareto front of the MORAP requires additional knowledge for comparing their performance. Therefore, experimental analysis was conducted to find the most fitted parameters in the algorithms. The parameters and the user defined values are presented in Table 4.

Considering the best known solution of the benchmark case, which was represented in Table 3, motivated us to set a lower bound of 0.9999 for reliability to illustrate the performance of the proposed approach in identifying the best known solutions.

4.5. Comparison metrics

We use several metrics proposed by Yu and Gen [23] to study the accuracy and the diversity of different procedures in re-generating the Pareto front of the MORAP. Since the real Pareto front of different test problems and benchmark cases cannot be achieved through an enumeration procedure, the Reference Set (RS) is retrieved in all runs. The RS contains the non-dominated solutions for all methods in all runs.

4.5.1. Accuracy measures

- *Number of non-dominated solutions* (*NNS*). This metric represents the NNS found by each method. Since the real Pareto front of the MORAP case is not attainable, a fast sorting procedure is used to determine whether a candidate solution for a given method is non-dominated. On the other hand, the fast sorting procedure checks each non-dominated solution for a given method with the RS. A non-dominated solution for a given algorithm may be dominated by a solution in the RS. The higher this metric, the more the method has converged towards the real Pareto front.
- *Error Ratio* (*ER*). The ER measures the non-convergence of the methods towards the real Pareto front of the MORAP case. The definition of the ER is as follows:

$$ER = \frac{\sum_{i=1}^{N} e_i}{N} \tag{28}$$

where, N is the number of non-dominated solutions found, and

$$e_i = \begin{cases} 0 & \text{if the solution } i \text{ belongs to Pareto front} \\ 1 & \text{otherwise} \end{cases}$$

The closer this metric is to 1, the less the solution has converged towards the Pareto front.

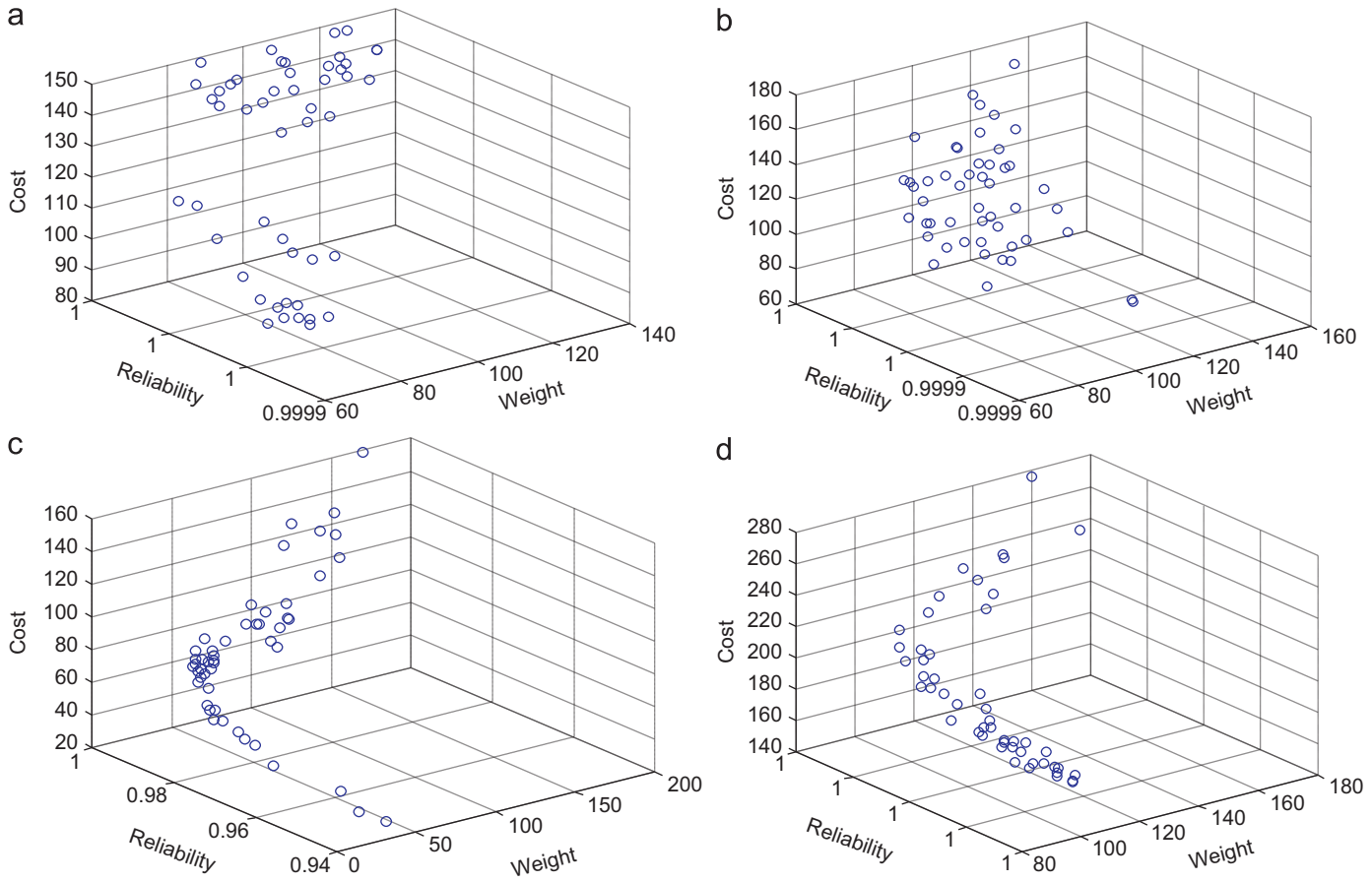

Fig. 3. Re-generated 3-D Pareto front of different approaches.

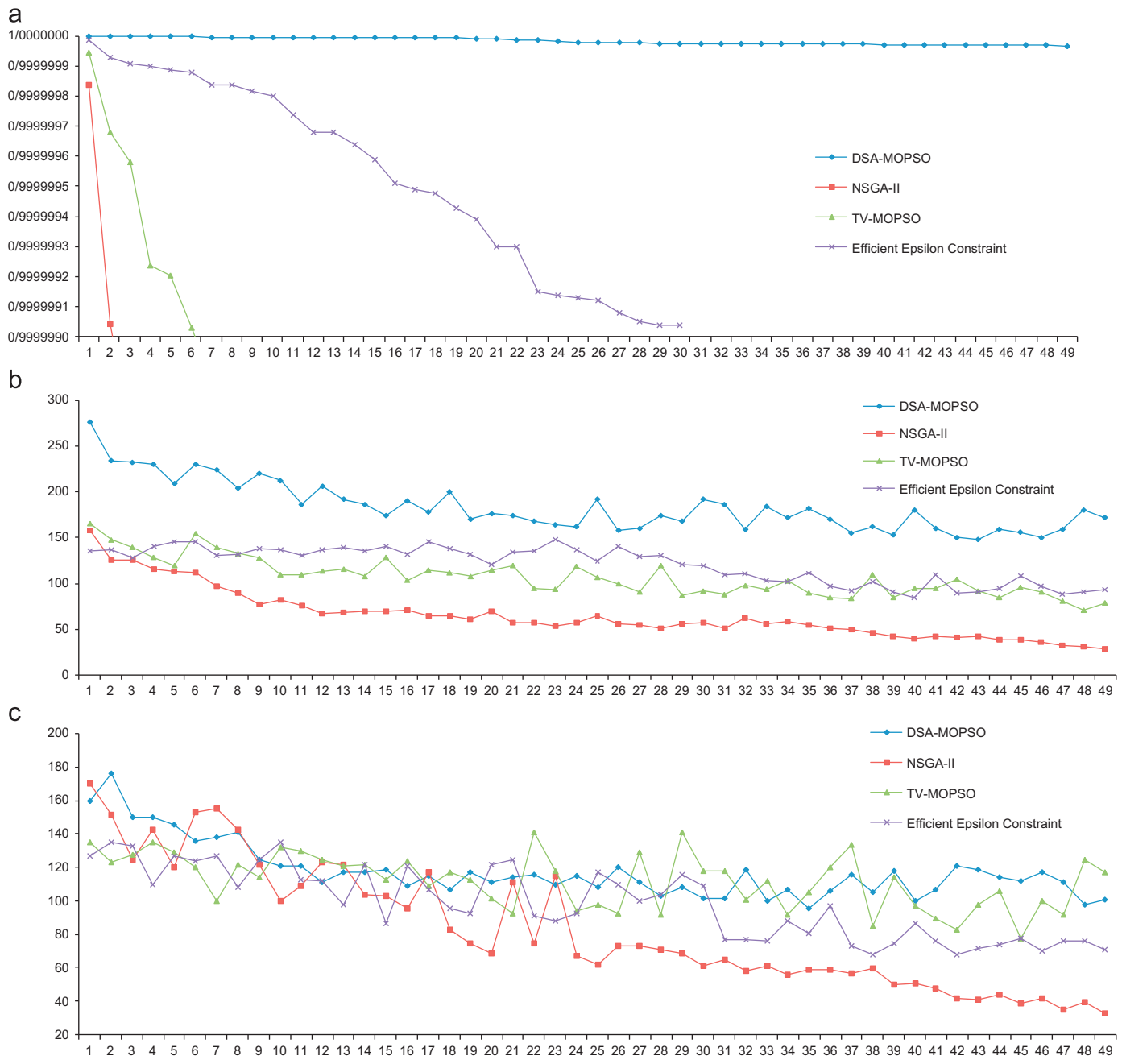

Fig. 4. Generated objective values in different methods for 50 non-dominated solutions in the archive. (a) Reliability Objective, (b) Cost Objective and (c) Weight Objective.

Generational distance (*GD*). This metric calculates the distance between the Pareto front/RS and the solution set. The definition of this metric as follows:

$$GD = \frac{\sum_{i=1}^{N} d_i}{N} \tag{29}$$

where $d_i = \min_{p \in PF} \left\{ \sqrt{\sum_{k=1}^{m} \left(z_k^i - z_k^p\right)^2} \right\}$ is the minimum Euclidean distance between the i-th solution and the Pareto front/RS, for which $|m|$ is the number of objective functions.

4.5.2. Diversity measures

Spacing metric (*SM*). The SM measures the uniformity of the spread of the points in the solution set. In order to calculate SM, $\overline{d}$, the mean value of all d_i, should be calculated first using the following equation:

$$\overline{d} = \frac{\sum_{i=1}^{N} d_i}{N} \tag{30}$$

Then, the SP which is the standard deviation of the closest distances is calculated using the following equation:

$$SP = \sqrt{\left(\sum_{i=1}^{N} \frac{\left(\overline{d} - d_i\right)^2}{N-1} \right)} \tag{31}$$

Diversification metric (*DiM*). The DiM measures the spread of the solution set and is calculated as follows:

$$DM = \left[\sum_{i=1}^{N} \max\left(\|x_i - y_i\| \right) \right]^{\frac{1}{2}} \tag{32}$$

where, $\|x_i - y_i\|$ is the Euclidean distance between of the non-dominated solution x_i and the non-dominated solution y_i.

5. Experimental results

In this section the results from the implementation of the aforementioned methods on the test problems and benchmark case are discussed.

5.1. Result of the small-size test problems

The accuracy measures for the small-size test problems are depicted in Table 5.

It can be concluded from Table 5 that the average NNS in the proposed DSAMOPSO method is higher compared with the other competing methods. This means that the DSAMOSPO method can generate more non-dominated solutions on average. The average of the ER metric in the proposed DSAMOPSO method is smaller in comparison with the other competing methods. This means that the DSAMOPSO method has fewer non-convergences towards the RS of the MORAP test problems. The average value of the GD measurement in the proposed DSAMOPSO method is smaller than the other methods. This means that the distance between the RS and the generated solution set in the DSAMOPSO method is low. The diversity measures obtained by the algorithms for small-size test problems are depicted in Table 6.

It can be concluded form Table 6 that the DSAMOPSO method provides non-dominated solutions that have lower average values for the SM measurement. Therefore, the non-dominated solutions obtained by the DSAMOPSO method are more uniformly distributed in comparison to those obtained by the other methods. The average of the DiM metric in the proposed DSAMOPSO method has a smaller value in comparison with the other methods. This means that the DSAMOPSO method has a narrower spread. Table 7 presents the computational time for all methods for the small-size test problems.

The average CPU time for the DSAMOPSO method obtained the third rank among the other competing methods. It is notable that the average CPU time for the NSGA-II and the cTV-MOPSO methods takes the first and second rank with a distance less than 1 s in comparison with the DSAMOPSO method. This can be neglected with respect to the several aforementioned possibilities of the DSAMOPSO method. It can be concluded from Table 7 that the AUGMECON method cannot be compared with the other methods. The AUGMECON method seeks local optimum solutions for a long time in each of the iteration of the algorithm.

5.2. Result of large-size test problems

The accuracy measures for the large-size test problems are depicted in Table 8.

Similar to the small-size test problems, it can be concluded from Table 8 that the average of NNS in the proposed DSAMOPSO method is higher than the other methods. This means that the number of non-dominated solutions for the DSAMOSPO method, which belongs to the RS, is higher than the other competing methods. The average of the ER metric in the proposed DSAMOPSO method has a smaller value in comparison with the other methods. This means that the DSAMOPSO method has fewer non-convergences towards the RS of the MORAP small-size test problems. The average value of the GD measurement in the proposed DSAMOPSO method is smaller than other methods. This means that the distance between the RS and the generated solution set in the DSAMOPSO method is short. The diversity measures obtained by the algorithms for large-size test problems are depicted in Table 9.

Similarly, it can be concluded form Table 9 that the DSAMOPSO method provides non-dominated solutions that have lower average values for the SM measurement. Therefore, the non-dominated solutions obtained by the DSAMOPSO method are more uniformly distributed in comparison with those obtained by the other competing methods. The average for the DiM metric in the proposed DSAMOPSO method has a smaller value in comparison with the other competing methods. This means that the DSAMOPSO method has a narrower spread.

Table 10 presents the computational time of all approaches for large-size test problems

Again, the average CPU time for the DSAMOPSO method was ranked 3rd among the other methods. It is notable that the average CPU time of the NSGA-II and the cTV-MOPSO methods took the first and second rank with a distance less than 1 s in comparison with the DSAMOPSO method. This can be neglected with respect to several aforementioned possibilities of the DSAMOPSO method. It can be concluded from Table 10 that the AUGMECON method represents a very long CPU time in comparison with the other competing methods. As the size of the test problem increases the computational time of the AUGMECON method grows increasingly.

Table 11
Upper and lower bounds of the generated non-dominated solutions.

	Reliability	Cost	Weight
Epsilon constraint method			
Upper bound	0.99999999	136	127
Lower bound	0.99995736	94	71
NSGA-II algorithm			
Upper bound	0.999999837753451	159	170
Lower bound	0.999947513898008	29	33
TV-MOPSO algorithm			
Upper bound	0.999999948164303	166	135
Lower bound	0.999938781193722	79	117
DSA-MOPSO algorithm			
Upper bound	0.999999999999211	276	160
Lower bound	0.999999968973551	172	101

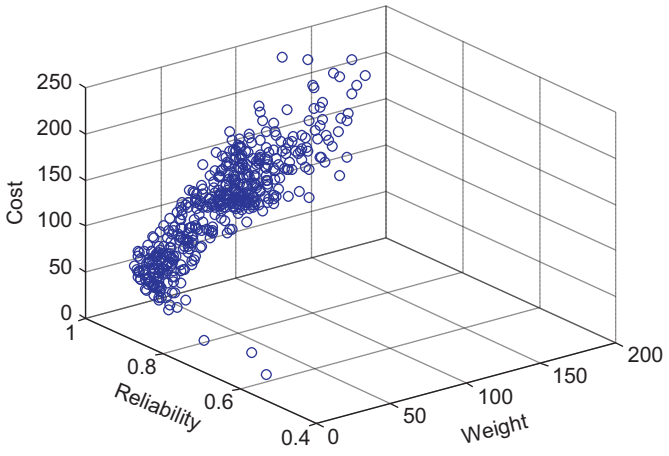

Fig. 5. Reference set for different approaches.

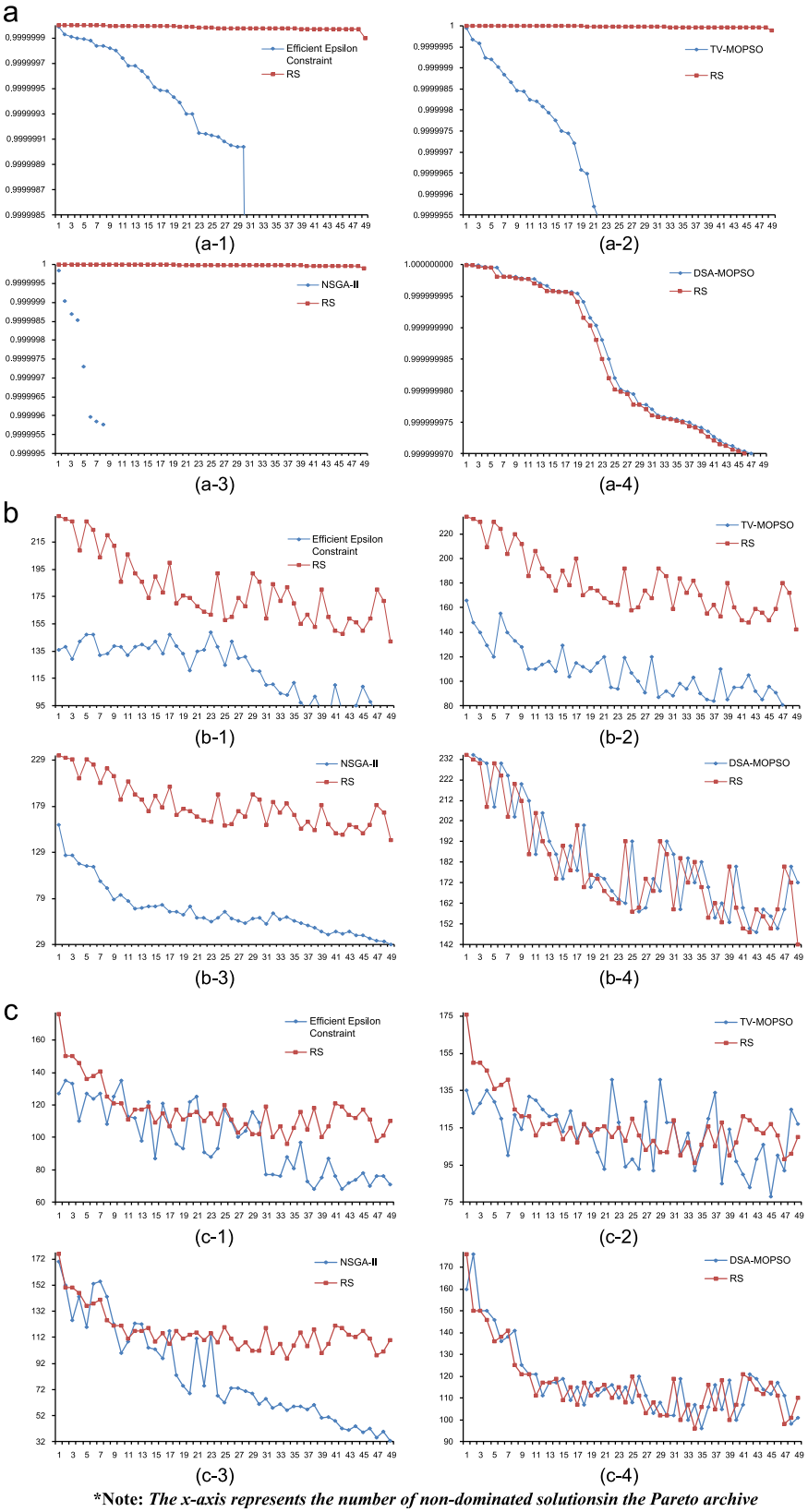

Fig. 6. Comparison of the RS with different approaches in each dimension. (a) Reliability Objective, (a-1) Epsilon-Constraint vs. RS, (a-2) Modified TV-MOPSO vs. RS, (a-3) Modified NSGA-II vs. RS and (a-4) Proposed DSA-MOPSO vs. RS, (b) Cost Objective, (b-1) Epsilon-Constraint vs. RS, (b-2) Modified TV-MOPSO vs. RS, (b-3) Modified NSGA-II vs. RS and (b-4) Proposed DSA-MOPSO vs. RS and (c) Weight Objective, (c-1) Epsilon Constraint VS RS, (c-2) Modified TV-MOPSO VS RS, (c-3) Modified-NSGA-IIVS RS and (c-4) Proposed DSA-MOPSO VS RS.

5.3. Result of well-known benchmark case

A more robust procedure is considered for the benchmark case. All algorithms are run 20 times to estimate the real Pareto front of benchmark case. This can acquire a proper infrastructure to compare their performances. The non-dominated solutions of each run formed regenerated Pareto fronts of each algorithm. The re-generated Pareto front of all procedures is presented in Fig. 3.

It can be concluded from Fig. 3 that all procedures are capable to re-generate some parts of the Pareto front. Moreover, it can be included that the re-generated Pareto fronts of the DSAMOPSO and the NSGA-II methods have better diversity. The latter case should be analyzed using statistical measures.

The performance metrics were selected and calculated for each algorithm. Fig. 4 represents the values of objective functions of different approaches.

Section (a) of Fig. 4 illustrates the reliability objective. It can be concluded that the achieved reliability values of the DSAMOPSO method outperform the other approaches.

Section (b) of Fig. 4 illustrates the cost objective. It can be concluded that the achieved cost values of the NSGA-II method are smaller than the other approaches.

It is clear that there is a trade-off between reliability and cost objectives. As the reliability of the designed system goes higher its cost becomes higher too. For example, the proposed design of the DSAMOPSO method has the highest reliability and cost values and

Table 12
Computational results of the accuracy metrics for the benchmark case MORAP.

Run	NNS				ER				GD			
	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	46	17	37	41	0.06	0.65	0.24	0.16	1.03	3.08	1.35	3.08
2	45	19	33	41	0.08	0.61	0.33	0.16	0.79	3.69	1.7	3.69
3	45	18	28	41	0.08	0.63	0.43	0.16	1.34	2.8	208	2.8
4	43	30	28	41	0.12	0.39	0.43	0.16	1.25	1.87	206	1.87
5	41	30	27	40	0.16	0.39	0.45	0.18	0.75	2.59	2.6	2.59
6	44	19	41	41	0.10	0.61	0.16	0.16	0.72	4.87	0.55	4.87
7	45	24	13	40	0.08	0.51	0.73	0.18	0.48	2.59	9.62	2.59
8	47	22	36	40	0.04	0.55	0.27	0.18	0.35	2.22	0.85	2.22
9	47	22	36	40	0.04	0.55	0.27	0.18	0.35	2.22	0.85	2.22
10	46	14	24	40	0.06	0.71	0.51	0.18	0.41	4.11	2.49	4.11
11	48	18	33	41	0.02	0.63	0.33	0.16	0.38	4.38	1.06	4.38
12	48	16	35	41	0.02	0.67	0.29	0.16	0.17	3.12	1.34	3.12
13	43	13	13	41	0.12	0.73	0.73	0.16	0.48	6.38	10.1	6.38
14	47	17	33	41	0.04	0.65	0.33	0.16	0.46	2.61	1.29	2.61
15	48	11	6	41	0.02	0.78	0.88	0.16	0.27	2.84	12.8	2.84
16	48	12	8	41	0.02	0.76	0.84	0.16	0.72	3.8	10.5	3.8
17	47	13	28	41	0.04	0.73	0.43	0.16	0.18	3.45	2.42	3.45
18	48	9	31	41	0.02	0.82	0.37	0.16	0.12	3.23	1.65	3.23
19	46	24	39	41	0.06	0.51	0.20	0.16	0.66	1.64	0.57	1.64
20	44	27	32	41	0.10	0.45	0.35	0.16	0.47	2.58	1.44	2.58
Ave.	45.8	18.8	28.1	40.8	0.07	0.62	0.43	0.17	0.57	3.2	23.9	3.2
Std. Dev.	2.02	6.08	10.3	0.44	0.04	0.12	0.21	0.01	0.34	1.12	62.8	1.12

Table 13
Computational results of the diversity metrics for the benchmark case of MORAP.

Run	SM				DM			
	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	5.92	4.29	3.06	4.29	54.1	84.9	8944	62.3
2	2.84	5.36	3.12	5.36	54.1	86.2	8944	62.3
3	6.61	5.07	1425	5.07	54.2	86.2	10,000	62.3
4	4.58	4.77	1426	4.77	54.7	86.2	10,205	62.3
5	2.17	5.08	4.92	5.08	55.1	86.2	19,343	62.3
6	3.09	9.36	1.53	9.36	55.2	86.2	19,343	62.3
7	2.4	4.24	9.27	4.24	55.4	86.5	19,343	62.3
8	1.8	3.07	1.68	3.07	55.5	86.5	19,343	62.3
9	1.8	3.07	1.68	3.07	55.5	86.5	19,343	62.3
10	1.75	7.62	3.95	7.62	55.6	86.5	19,343	62.3
11	2.63	9.02	2.23	9.02	55.6	86.5	19,343	62.3
12	1.18	6.16	2.55	6.16	56	86.6	19,343	62.3
13	1.4	10.1	9.48	10.1	56	86.6	19,343	62.3
14	2.85	3.66	2.53	3.66	57.1	86.6	19,343	62.3
15	1.86	2.82	9.79	2.82	57.1	86.6	19,343	62.3
16	5.03	4.88	9.05	4.88	57.4	86.6	19,343	62.3
17	0.91	4.59	4.06	4.59	57.4	86.6	19,343	62.3
18	0.86	4.58	3.4	4.58	57.4	86.6	19,343	62.3
19	2.85	2.48	1.26	2.48	57.4	86.8	19,343	62.3
20	2.23	5.59	2.44	5.59	57.4	86.8	19,343	62.3
Ave.	2.74	5.29	146	5.29	55.9	86.4	17,379	62.3
Std. Dev.	1.61	2.18	438	2.18	1.19	0.4	4039	0

the proposed design of the NSGA-II method has the lowest reliability and cost values.

Section (c) of Fig. 4 illustrates the weight objective. The same comparison can be made between the weight objective and the reliability, or the cost objective.

Table 11 shows the upper bound and lower bound of generated solutions for all the algorithms.

It can be concluded form Table 10 that the theoretical bounds of the well-known benchmark case [10,15,21] was improved by the proposed DSAMOPSO method.

As mentioned, all approaches were run 20 times. As the archive size of the approaches was set equal to 50, so 200 solutions were approximately generated (except for infeasibility conditions or situations that the archive was not fulfilled completely) in each run. Approximately, 4000 solutions were generated. These included dominated and similar solutions. The sum of all dissimilar and non-dominated solution in all 20 runs was equal to 2096. These 2096 solutions were sorted according to non-domination to form the RS which contained 460 final solutions. Fig. 5 represents the RS of aforementioned approaches for all runs.

The ranges of reliability, cost, and weight in the RS are [0.57415680, 0.999999999900315], [15, 234], and [11, 186], respectively. The range and the number of solutions in the RS are acceptable. Therefore, the RS can be used as a proper infrastructure in order to compare the performance of a single run of each algorithm on a relative basis.

Fig. 6 distinctively plots fifty non-dominated solutions of the RS in comparison with non-dominated solutions of a single run of each approach. This gives us a better sense of the closeness of each approach to the RS in different dimensions.

Section (a) of Fig. 6 plots the reliability objective of non-dominated solutions of each approach in comparison with reliability objective of solutions in the RS. Section (b) of Fig. 6 plots the cost objective of non-dominated solutions of each approach in comparison with the cost objective of solutions in the RS. Section (c) of Fig. 6 plots the weight objective of non-dominated solutions of each approach in comparison with weight objective of solutions in the RS.

It can be concluded that the generated non-dominated solutions of the proposed DSAMOPSO method are relatively close to the non-dominated solutions of the RS using the other approaches. This is an empirical proof of the relative preference of the proposed DSAMOPSO method.

In Fig. 6, each point on the RS curve should be compared vertically with its associated point on the DSAMOPSO curve. A horizontal comparison of the curves in this figure is meaningless since such a comparison shows that the DSAMOPSO method produces objective function values that are one lag ahead of the values provided by the RS. Therefore, we can conclude from Fig. 6 that the DSAMOPSO method can produce non-dominated solutions which are closer to the solutions produced with the RS in comparison with the other methods. The reason for the similarity of the solutions between the DSAMOPSO method and the RS is as follows. The RS has been established from the generated non-dominated solutions for all methods in 20 independent runs. The procedure for establishing the RS is as follows. All methods were run 20 times. As the archive size of the methods was set to 50, 200 solutions were generated (except for the infeasibility conditions or situations that the archive was not fulfilled completely) in each run. Approximately, 4000 solutions were generated. These included dominated and similar solutions. The sum of all dissimilar solutions for all 20 runs was equal to 2096. These 2096 solutions were sorted based on non-domination to form the RS which contained 460 final solutions. Fig. 5 represents the RS for the aforementioned methods for all runs.

Using the aforementioned procedure, since the RS contains the best solutions from several runs for all algorithms, it will contain more qualified solutions in comparison with a single run for each of the methods. Therefore, the RS has been formed during several runs for all methods among 4000 generated solutions. The generated solution for each method in Fig. 6 is the performance and the ability of a method in a single run. Naturally, the result from a single run of a method has less quality than the RS. Furthermore, the DSAMOPSO method produces solutions that are very close to the solutions provided by the RS in a single run among all other methods. This observation shows the relative dominance of the proposed DSAMOPSO method in regenerating non-dominated solutions which are closer to the RS in a single run.

The aforementioned accuracy and diversity metrics were calculated for all proposed methods in 20 different runs. The accuracy measures for the benchmark case are depicted in Table 12.

It can be concluded form Table 12 that the average of NNS in the proposed DSAMOPSO method is higher than the other algorithms. This means that the number of non-dominated solutions of DSAMOSPO which belongs to the RS is higher than the other approaches. The average of the ER metric in the proposed DSAMOPSO method has a smaller value in comparison with other approaches. This means that the DSAMOPSO method has fewer non-convergences towards the RS of the MORAP small-size test problems. The average value of GD measurement in the proposed DSAMOPSO method is smaller than other approaches. This means that the distance between the RS and the generated solution set in the DSAMOPSO method is low.

The diversity measures obtained by the algorithms for large-size test problems are depicted in Table 13.

Similarly, it can be concluded form Table 13 that the DSAMOPSO method provides non-dominated solutions that have lower average values for the SM measurement. Therefore, the non-dominated solutions obtained by the DSAMOPSO method are more uniformly distributed in comparison with those obtained by the other algorithms. The average of the DiM metric in the proposed DSAMOPSO method has a smaller value in comparison with the other methods. This means that the DSAMOPSO method has a narrower spread in comparison with the other competing methods. Table 14 presents the computational time of these methods for benchmark case.

Table 14
CPU times (seconds) of different approaches.

Run	DSA-MOPSO	NSGA-II	cTV-MOPSO	AUGMECON
1	3.94	2.08	2.85938	2199
2	3.44	2.03	2.98438	2131
3	3.36	2.06	2.64063	2149
4	3.81	2.06	3.1875	2001
5	3.52	2.09	2.98438	2064
6	3.25	2.06	2.92188	2044
7	3.64	2	2.53125	2122
8	3.72	2.08	2.875	2197
9	3.72	2.08	2.875	2134
10	4.19	2.13	3.03125	2119
11	3.48	2.13	2.75	2096
12	3.75	2.13	3	2186
13	4.09	2.09	2.42188	2188
14	3.66	2.13	2.73438	2013
15	3.48	2.08	2.40625	2174
16	3.17	1.98	2.45313	2001
17	3.73	2.11	2.85938	2061
18	3.75	2.14	2.84375	2141
19	3.14	2.16	2.82813	2135
20	4.14	2.14	2.95313	2027
Ave.	3.65	2.09	2.80703	2109
Std. Dev.	0.3	0.05	0.21676	66.8

Again, the average CPU time for the DSAMOPSO method obtained the third rank among the other competing methods. It is notable that the average CPU time of the NSGA-II and the cTV-MOPSO methods take the first and second rank with a distance less than 1 s in comparison with the DSAMOPSO method. This can be neglected with respect to several possibilities of the DSA-MOPSO method. It can be concluded from Table 14 that the AUGMECON method represents a very long CPU time in comparison with the other methods. As the size of the test problem increases the computational time of the AUGMECON method increases steadily.

5.4. Computational time of the algorithms

It can be concluded from Tables 7, 10, and 14 that there are significant differences between the computational times of the AUGMECON method and the other methods.

The epsilon constraint algorithm was implemented with an *unlimited version of LINGO* software. The MORAP is modeled as a non-linear programming problem which is hard to solve optimally. Moreover, the LINGO software uses non-linear solvers based on gradient and partial-derivatives in order to solve such mathematical programming problems.

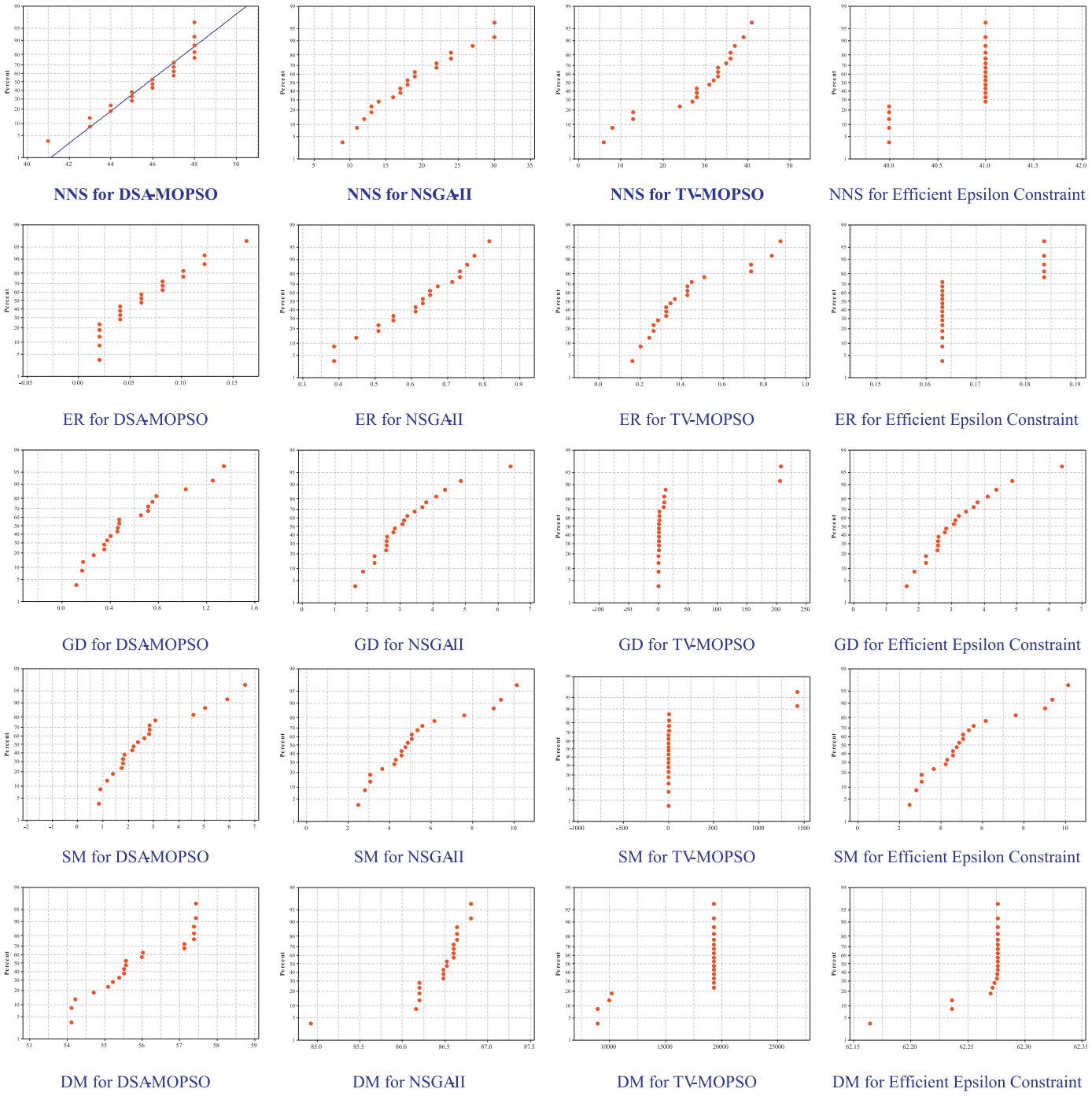

Fig. 7. Results of the Kolmogorov–Smirnov test for each metric for all procedures.

The main characteristics of the proposed DSAMOPSO method (i.e., dynamic constraint handling, self-adaptive constraint handling, dynamic parameter tuning, heuristic cost-benefit ratio, diversification or global best selection procedure) do not constrain the DSAMOPSO method but they impose simple computational efforts which can be easily are implemented using proper coding, software development, and hardware capabilities.

The MORAP is modeled as a mixed-integer non-linear multi-objective programming which is very hard to solve efficiently using mathematical approaches. Hence, the AUGMECON method should attempt to approach the neighborhood of a solution several times. Therefore, the algorithm may stop in a local optimum as the derivatives are close to zero.

5.5. Statistical analysis

Although the metrics in Tables 12 and 13 presented the relative dominance of the DSAMOPSO method, a meaningful comparison of the algorithms is discussed through statistical analysis. First, the normality test has been accomplished in order to check whether the normal distribution is fitted on the distribution of the aforementioned metrics over the solution space. The results of the Kolmogorov–Smirnov test are presented in Fig. 7.

It can be concluded from Fig. 7 that there is not enough evidence to reject the normality of the population. Therefore, we use parametric test to check the performance of the competing methods.

Then, the analysis of variance (ANOVA), in which different runs of the algorithms are assumed to be random samples, is applied to check whether there is a significant difference between the performances of different methods considering the aforementioned metrics. The results of the ANOVA are presented in Table 15.

Table 15 shows that there is enough evidence to reject the hypothesis of equal means for *NNS*, *ER* and *DiM* metrics. There is not enough evidence to reject the equal mean values for the *GD* and *SM* metrics. Finally, the Confidence Intervals (CI) has been calculated for the mean of the aforementioned metrics based on the pooled Standard Deviation (Std. Dev.). The results are presented in Table 16.

The confidence levels for all experiments were set to 95%. The tests have been accomplished using MINITAB 15.0 software. It can be concluded that the achieved mean values of *NNS*, and *ER*, in the DSAMOPSO method are more significant in comparison with the competing methods. The values of *GD* and SM are less significant compared with the competing methods.

The DSAMOPSO and the NSGA-II methods re-generate non-dominated solutions which have smaller values for the spacing metric. We can conclude that the generated solutions using the DSAMOPSO and the NSGA-II methods are more uniformly distributed throughout the RS in comparison with the competing methods. The average value of the diversification metric in the *c*TV-MOPSO is promising among with the competing methods. More formally, the NSGA-II method is able to find non-dominated solutions which are scattered. It is also notable that the best known solution of the MORAP has been improved using the DSAMOPSO method.

6. Conclusions and future research directions

In this paper, a DSAMOPSO method was proposed to solve the binary-state MORAPs. Different properties of the proposed DSAMOPSO method made it robust and competitive among the other existing methods in the literature. In order to supply a proper infrastructure for comparison, three well-known multi-objective procedures were also selected, modified, and customized.

Table 15
Analysis of variance.

I. First metric: NNS					
Source	Degree of freedom	Sum of square	Mean square	F	*P*-value
Factor	3	9020.2	3006.7	82.15	0.000
Error	76	2781.7	36.6	–	–
Total	79	11801.9	–	–	–
		$S=6.050$, $R\text{-}Sq=76.43\%$, $R\text{-}Sq(\text{adj.})=75.50\%$			
II. Second metric: ER					
Source	Degree of freedom	Sum of square	Mean square	F	*P*-value
Factor	3	3.7569	1.2523	82.15	0.000
Error	76	1.1585	0.0152	–	–
Total	79	4.9154	–	–	–
		$S=0.1235$, $R\text{-Sq}=76.43\%$, $R\text{-Sq(adj.)}=75.50\%$			
III. Third metric: GD					
Source	Degree of freedom	Sum of square	Mean square	F	*P*-value
Factor	3	7055	2352	2.38	0.076
Error	76	74937	986	–	–
Total	79	81992	–	–	–
		$S=31.40$, $R\text{-}Sq=8.60\%$, $R\text{-Sq(adj.)}=5.00\%$			
IV. Fourth Metric: SM					
Source	Degree of freedom	Sum of square	Mean square	F	*P*-value
Factor	3	302,323	100774	2.11	0.107
Error	76	3,638,303	47872	–	–
Total	79	3,940,626	–	–	–
		$S=218.8$, $R\text{-}Sq=7.67\%$, $R\text{-}Sq(\text{adj.})=4.03\%$			
V. Fifth Metric: DiM					
Source	Degree of freedom	Sum of square	Mean square	F	P-value
Factor	3	4,494,896,469	1,498,298,823	367.45	0.000
Error	76	309,896,727	4,077,589	–	–
Total	79	4,804,793,196	–	–	–
		$S=2019$, $R\text{-}Sq=93.55\%$, $R\text{-}Sq(\text{adj.})=93.30\%$			

Table 16
Calculated confidence interval for the accuracy and diversity metrics.

Individual 95% CIs for NNS mean based on pooled Std. Dev

Level	N	Mean	Std. Dev.
DSA-MOPSO20	**45.800**	**2.016**	
NSGA-II20 18.750	18.750	6.077	
TV-MOPSO20	28.050	10.257	
EPSILON20	40.750	0.444	

20 30 40 50

Individual 95% CIs for ER mean based on pooled Std. Dev.

Level N mean	N	Mean	Std. Dev.
DSA-MOPSO20	**0.0653**	**0.0411**	
NSGA-II20	0.6173	0.1240	
TV-MOPSO20	0.4276	0.2093	
EPSILON20	0.1684	0.0091	

0.20 0.40 0.60 0.80

Individual 95% CIs for GD mean based on pooled Std. Dev.

Level	N	Mean	Std. Dev.
DSA-MOPSO	**20**	**0.57**	**0.34**
NSGA-II	20	3.20	1.12
TV-MOPSO	23.87	62.78	
EPSILON 20	20	3.20	1.12

0 15 30 45

Individual 95% CIs for SM mean based on pooled Std. Dev.

Level N	N	Mean	Std. Dev.
DSA-MOPSO	20	2.7	1.6
NSGA-II	20	5.3	2.2
TV-MOPSO20	**146.4**	**437.6**	
EPSILON	20	5.3	2.2

0 100 200 300

Individual 95% CIs for DM mean based on pooled Std. Dev.

Level N	N	Mean	Std. Dev.
DSA – MOPSO	20	56	1
NSGA-II	20	86	0
TV-MOPSO20	**17379**	**4039**	
EPSILON	20	62	0

0 5000 10000 15000

First an AUGMECON method for generating qualified Pareto efficient solutions, in which the pitfalls of the conventional epsilon constraint method were addressed, was proposed. The efficient ε-constraint method can handle four problems in the conventional ε-constraint method with regards to the commensurable slack-based objective function, the lexicographic pay-off table, and the efficient searching procedure.

Second, a TV-MOPSO algorithm was customized, modified, and improved using dynamic penalty functions.

Third, a well-known multi-objective evolutionary algorithm, called the NSGA-II method, was also modified based on dynamic parameter tuning.

Afterwards, a binary-state MORAP, which was modeled as an NP-hard problem, was introduced. Different sets of test problems and a well-known benchmark case were used through the proposed procedures.

All the proposed algorithms were capable of solving small-size test problems. Several experiments were conducted to compare the performance of the proposed methods in re-generating the Pareto front of the MORAP.

Statistical analysis was supplied to compare the performance of the proposed algorithms. The proposed DSAMOPSO method showed relative preference in comparison with the other competing methods. The best known solutions of the MORAP were also improved using the DSAMOPSO method. This is a direct result of the special properties of the DSAMOPSO method (i.e., self-adaptive penalty functions, a heuristic cost-benefit ratio modification strategy, dynamic parameter tuning, fast ranking, evolutionary based operators, elitism, crowding distance, and rank-based tournament global best selection procedure). The MORAP studied in this paper was an NP-Hard mixed-integer nonlinear constrained multi-objective decision making problem, so the proposed DSAMOPSO method can efficiently be customized to solve other real life engineering and management decision problems.

Finally, in order to increase the goodness-of-fit of the proposed method, further analysis can be made through advanced statistical analysis such as the design of experiments, factorial design, and the Taguchi method on parameters tuning. Another improvement can be the hybridization of the proposed algorithm with an initial seeding procedure. The current algorithm initially uses random solutions. Generating and using systematic initial solutions through heuristic methods can improve the performance of the proposed algorithm.

Acknowledgment

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

References

[1] Aven T. On performance-measures for multistate monotone systems. Reliability Engineering & System Safety 1993;41:259–66.
[2] Chankong V, Haimes Y. Multi-objective decision making theory and methodology. New York: Elsevier Science; 1983.

[3] Chern MS. On the computational complexity of reliability redundancy allocation in a series system. Operations Research Letters 1992;11:309–15.
[4] Coelho LS. An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications. Reliability Engineering & System Safety 2009;94:830–7.
[5] Coello CAC, Pulido GT, Lechuga MS. Handling multiple objectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation 2004;8(3):256–79.
[6] Deb K, Pratap A, Agarwal S, Meyarivan TA. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6:182–97.
[7] Gen M, Yun Y. Soft computing approach for reliability optimization: state-of-the-art survey. Reliability Engineering & System Safety 2006;91:1008–26.
[8] Hu M, Wu T, Weir JD. An intelligent augmentation of particle swarm optimization with multiple adaptive methods. Information Sciences 2012;213: 68–83.
[9] Hwang CL, Masud ASM. Multiple objective decision making methods and applications. New York, USA: Springer-Verlag; 1979.
[10] Khalili-Damghani K, Amiri M. Solving binary-state multi-objective reliability redundancy allocation series-parallel problem using efficient epsilon-constraint, multi-start partial bound enumeration algorithm, and DEA. Reliability Engineering & System Safety 2012;103:35–44.
[11] Konak A, Coit DW, Smith AE. Multi-objective optimization using genetic algorithms: a tutorial. Reliability Engineering & System Safety 2006;91: 992–1007.
[12] Kennedy J, Eberhart, RC. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. New Jersey: Piscataway, IEEE Service Center; 1995, pp. 1942–1948.
[13] Kuo W, Zuo MJ. Optimal reliability modeling: principles and applications. Inc. Hoboken: John Wiley & Sons; 2003.
[14] Liang YC, Chen YC. Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm. Reliability Engineering & System Safety 2007;92:323–31.
[15] Li Z, Liao H, Coit DW. A two-stage approach for multi-objective decision making with applications to system reliability optimization. Reliability Engineering & System Safety 2009;94:1585–92.
[16] Mavrotas G. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. Applied Mathematics and Computation 2009;213:455–65.
[17] Misra KB, Ljubojevic MD. Optimal reliability design of a system: a new look. IEEE Transactions on Reliability 1973;22:255–8.
[18] Reyes-Sierra M, CoelloCoello CA. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. International Journal of Computational Intelligence Research 2006;2:287–308.
[19] Salazar D, Rocco CM, Galván BJ. Optimization of constrained multiple-objective reliability problems using evolutionary algorithms. Reliability Engineering & System Safety 2006;91:1057–70.
[20] Shi Y, Eberhart R. Empirical study of particle swarm optimization. In Congress on Evolutionary Computation (CEC'1999). Piscataway, NJ: IEEE Press; 1999 1945–1950.
[21] Taboada H, Baheranwala F, Coit DW, Wattanapongsakorn N. Practical solutions for multi-objective optimization: an application to system reliability design problems. Reliability Engineering & System Safety 2007;92:314–22.
[22] Tripathi PK, Bandyopadhyay S, Kumar Pal S. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. Information Sciences 2007;177:5033–49.
[23] Yu X, Gen M. Introduction to Evolutionary Algorithms. London Limited: Springer-Verlag; 2010 Chapter 6.
[24] Zhang Y, Gong DW, Ding Z. A bare-bones multi-objective particle swarm optimization algorithm for environmental/economic dispatch. Information Sciences 2012;192:213–27.
[25] Zhao JH, Liu Z, Dao MT. Reliability optimization using multi-objective ant colony system approaches. Reliability Engineering & System Safety 2007;92: 109–20.