Journal of the Brazilian Society of Mechanical Sciences and Engineering (2019) 41:293
https://doi.org/10.1007/s40430-019-1792-x

TECHNICAL PAPER

A swarm intelligence-based approach for reliability–redundancy allocation problems

Najeh Ben Guedria[1,2] · **Hichem Hassine**[1,3]

Received: 17 October 2017 / Accepted: 10 June 2019 / Published online: 19 June 2019

Abstract
This paper presents a new algorithm belonging to the class of swarm intelligence methods, called the adaptive simplified PSO (ASPSO)-based algorithm, for solving reliability–redundancy allocation problems. In this constrained nonlinear mixed-integer problem, both the number of redundant components and their reliability in each subsystem are to be decided simultaneously so as to maximize the reliability of the system. The proposed ASPSO operates with a new updating model to adjust the position of particles, without dealing with velocity. In addition, a randomization technique, based on the dispersion of particle bests through the search space, is used to speed up the convergence of the proposed approach and prevent it from being trapped within the local optimum. Moreover, to control the balance between exploration and exploitation, during the search process, two adaptive functions are utilized. The simulation results of four different benchmarks for the reliability–redundancy allocation problem are reported and compared. Accordingly, the solutions given by the new presented approach are all superior to those best known solutions provided by several methods in the literature.

Keywords Reliability optimization · Redundancy allocation · Particle swarm optimization · Particles dispersion · Adaptive function

1 Introduction

The main challenge for industries is to produce high-quality products at minimum cost. This goal is directly related to the reliability, maintainability and availability of the different components of production systems. However, with the technological development and industrial revolution, manufacturing systems are more and more complex. These systems should be operational and available over the production time. Nevertheless, their failure is inevitable. Therefore, in the recent years, improving system/product reliability has become an important issue to reach satisfactory performances in real-world engineering systems. Reliability can be defined as the probability that a component, device, system or process will perform its intended function without failure for a given time when operated correctly under specified environmental conditions. Hence, reliability studies aim to predict and estimate the probability of failure in order to optimize the operational management related to the provision of failure in maintenance policies.

The redundancy strategy is a helpful approach to assist the operational system design for reliability. Using redundant components is much more affordable to improve the system reliability instead of costing too much to enhance the component reliability. This approach has been applied in several engineering fields [1–3]. The optimal site of system parts is addressed as the redundancy allocation problem (RAP). System reliability can be maximized by adding suitable redundant parts to each subsystem, while fulfilling particular system-level constraints such as cost, weight and volume. Thus, the main goal of the RAP is to choose

Technical Editor: Fernando Antonio Forcellini, Dr.

✉ Najeh Ben Guedria
najeh.benguedria@istls.rnu.tn

Hichem Hassine
hassinehichem@yahoo.fr

[1] Higher Institute of Transport and Logistics of Sousse, University of Sousse, Cité Erriadh Sousse BP 247, 4023 Sousse, Tunisia

[2] Mechanical Laboratory of Sousse (LMS), National school of Engineering of Sousse, BP 264 Sousse Erriadh, 4023 Sousse, Tunisia

[3] Mechanics, Modelling and Manufacturing Laboratory (LA2MP), National School of Engineers of Sfax, BP 1173, 3038 Sfax, Tunisia

a set of available components and to determine the optimal redundancy levels of components [4]. These levels are treated as a set of discrete variables, and so, the RAP can be considered as a constrained integer programming problem [5]. With the growth of advanced technology, component reliability is more controllable even in the design stage. Consequently, the possibility to improve system reliability via the optimization of the reliability and redundancies of components has arisen a more interesting problem: the reliability–redundancy allocation problem (RRAP) [6, 7]. Formally, the RRAP is expressed as a nonlinear mixed-integer optimization problem. Decision variables are the number of redundancies (positive integers) as well as component reliabilities (floating-point variables fall between zero and one). In addition, from the complexity point of view, the RRAP has proven to be an NP-hard optimization problem [5, 8].

Over the past decades, several deterministic methods have been used to solve the RRAP, such as the heuristic methods [9–11], the branch and bound method [12, 13], the integer programming [14] and the dynamic programming method [15, 16]. However, these methods have shown some weaknesses. Indeed, heuristic techniques require derivatives for all nonlinear constraint functions that are not derived easily because of the high computational complexity. In the branch and bound method, effectiveness depends on the sharpness of the bound and is suitable only for small-sized problems, due to its highly required memory. The dynamic programming method is suitable only for special structures, such as parallel–series and series–parallel designs, of the objective function and constraints that are decomposable. To overcome this faintness, several meta-heuristics have been selected and successfully applied to solve various reliability optimization problems. These methods include the genetic algorithm (GA) [17–25], Tabu search (TS) [26–28], ant colony optimization (ACO) [29, 30], immune algorithm (IA) [31, 32], firefly algorithm (FA) [33], artificial bee colony (ABC) algorithm [34, 35], artificial fish swarm (AFS) algorithm [36], harmony search (HS) [37, 38], imperialist competitive algorithm (ICA) [39], Cuckoo search (CS) [40]. Furthermore, some novel meta-heuristic methods have been recently introduced to RRAP such as grey wolf optimizer algorithm [41] and biogeography-based optimization (BBO) [42] to name a few. Although meta-heuristic methods are able to find the optimal solution within a reasonable computation time, they have undergone several alterations, through embedding some innovative operators and strategies, to further enhance their global search abilities and quality solution for solving complex optimization problems, particularly RRAPs. For instance, in [23], GA is coupled with simulated annealing (SA) method for solving nonlinear integer reliability problems. Jang et al. [28] developed a TS-based algorithm with new mechanisms to tackle Multiple Multi-level RAP. Agarwal and Sharma [29] presented an ACO-based method coupled with an adaptive function and a local search technique for solving the constrained RAP for binary systems. Similarly, in [29, 30], ACO was combined with a neighborhood search procedure to improve the solution quality of large-scale reliability optimization problems. In [31], the author proposed a penalty-guided immune algorithm (IA), which effectively and efficiently explored the search space, to find a feasible optimal solution. Likewise, an immune two-phase approach was designed so as to solve the RRAP [32]. In the first phase, the allocation problem was solved using the IA. In the second phase, a procedure was applied to enhance the solutions by the IA. Coelho et al. [33] presented a chaotic FA approach coupled with (FAC) chaotic sequences. In pursuit of the same goal, Garg et al. [34] proposed a two-phase method based on ABC for solving RRAP. In the first phase, a ABC algorithm is created to solve the allocation problem. In the second phase, the number of component redundancy, obtained by the first phase, is fixed and a developed procedure is employed to enhance the component reliability allocation. A novel artificial fish swarm algorithm (NAFSA) was developed by He et al. [36] to tackle large-scale RRAPs. Two information sources, which were fish self-information and environment information, were used to regulate the balance of diversity and convergence. Zou et al. [37, 38] presented a modified global harmony search algorithm coupled with concepts from the PSO algorithm to solve several reliability problems. Afonso et al. [39] suggested the classical ICA combined with an attraction and repulsion concept. An efficient BBO algorithm, proposed by Garg [42], was utilized in order to solve RRAPs.

Particle swarm optimization (PSO), originally created by Kennedy and Eberhard [43, 44], is an interesting population-based stochastic optimization algorithm which mimics the behavior of a swarm of birds or a school of fish. In PSO, each particle is characterized by a position and a velocity. During the optimization process, a particle adjusts its velocity with respect to its personal experience and the experience of the whole swarm. The new position of a particle is then determined based on its previous position and its new velocity. Due to its simple concept, fast convergence, inexpensive computation and ease of implementation, it has been successfully applied to various engineering optimization problems including engineering design [45, 46], system identification [47], system control [48] and neural networks [49]. A comprehensive survey on PSO applications can be found in [50]. Despite all these merits, PSO has shown some faintness. The use of velocity updating and position updating is suitable to the convergence of PSO. Nevertheless, it is also harmful to head off premature convergence. Incoherently, its strong convergence is desirable. However, its poor performance to avoid falling into a local optimum is undesirable. Over the past decades, several attempts have been made to tackle these weaknesses effectively, resulting

in the development of a large number of PSO variants [51]. These new algorithms propose new updating rules, use various operators or combine PSO with other existing methods to produce new hybrid algorithms. The main goal of these variants is to enhance swarm diversity, to avoid premature convergence and improve convergence rate. For instance, to solve RRAPs, Coelho [52] put forward an efficient PSO algorithm simultaneously based on the Gaussian distribution and the chaotic sequence. Similarly, Yeh [53] designed a two-stage discrete PSO (2DPSO) approach so as to solve the multiple multi-level RAP in series systems. The propounded method utilized a simple mechanism to update particle's positions without velocity. Wu et al. [54] developed an improved PSO (IPSO) algorithm using two position-updating strategies. The former, used during early iterations, allowed each particle to benefit from its own best experience with a large probability. The latter, employed during the late iterations, lets each particle to profit from the experience of the best particle in the swarm. In addition, a mutation operator was used to prevent premature convergence. Huang [55] developed a simplified PSO algorithm with two updating mechanisms. The first mechanism adjusted the integer variables, and the second one updated the variables of components reliability, which were real numbers. In addition, an orthogonal array test was implemented to determine the best combination of the related parameters utilized in the updating mechanisms. Recently, Ouyang et al. [56] have suggested a stochastic perturbation PSO (SPPSO) algorithm to solve RRAPs with heterogeneous components. The authors embedded into the standard particle swarm method three operators: a perturbation operator, a convergence operator and a decentralization operator, to improve both global and local search abilities of the algorithm.

From the aforementioned literature overview, it is evident that PSO has undergone significant modifications to cope with the complexity of the RRAPs. However, the majority of the proposed variants are time-consuming, due to the implantation of complex procedures, and need a large number of iterations to reach the optimum or a near-optimum solution. Moreover, the simplicity of the basic PSO, which is the main cause of its attractiveness, is sacrificed. Motivated by the above observations, we introduce in this paper an adaptive simplified PSO (ASPSO)-based algorithm, which exhibits a competitive advantage over many state-of-the-art metaheuristic methods in terms of global search ability and convergence speed on four benchmark problems of RRAP tested in this paper. The ASPSO algorithm speeds up the search process toward converging to the global optimum by using a simple, yet efficient, position-updating rule without dealing with velocity. Each particle position is adaptively adjusted utilizing both global best and personal best positions. In addition, a new vector, created based on the dispersion of personal best positions through the search space, is combined with the updating mechanism to ensure the diversity of the swarm during the optimization process as well as to avoid premature convergence. Moreover, two adaptive functions are employed: The former adjusts the attraction of particles toward the global best and the personal best. The latter controls the degree of randomness added to the solution. To validate the performance of the ASPSO algorithm, four well-known benchmarks of RRAPs are used to test its solution accuracy and convergence speed. It is observed that the computational results provided by the ASPSO algorithm outperform the existing results of many other approaches in the literature.

The remaining content of this paper is structured as follows. In Sect. 2, the mathematical model of the RRAP is defined and four benchmark RRAPs (series, series–parallel, bridge and overspeed protection systems) are illustrated. The standard PSO algorithm is described briefly in Sect. 3. In Sect. 4, the suggested (ASPSO) algorithm is presented and described in detail. In Sect. 5, numerical experiments are carried out to test the performance of ASPSO for reliability optimization problems. Finally, conclusions are summarized in Sect. 6.

2 Problem formulation: RRAP

The RRAP of maximizing the system reliability can be formulated as a constrained mixed-integer programming problem:

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

subject to:

$$g(\mathbf{r}, \mathbf{n}) \le \mathbf{l} \tag{2}$$

$$0 \le r_i \le 1,\ r_i \in \mathbb{R},\ n_i \in \mathbb{Z}^+,\ 1 \le i \le m$$

where R_s is the system reliability; $f(.)$ is the objective function for the overall system reliability; $g(.)$ is the set of nonlinear constraint functions, which are usually associated with system's weight, volume and cost; $\mathbf{r} = (r_1, r_2, r_3, \ldots, r_m)$ is the vector of component reliability for the system; $\mathbf{n} = (n_1, n_2, n_3, \ldots, n_m)$ is the vector of the redundancy allocation for the system; r_i and n_i are the reliability and the number of redundant components in the ith subsystem, respectively; $\mathbf{l}$ is the vector of resource limitation and m is the number of subsystems in the system. As it can be noted, the RRAP belongs to the class of constrained nonlinear mixed-integer optimization problems, since the number of redundancies n_i represents the positive integer values, and the component reliability r_i denotes the real values between 0 and 1. The objective of the problem is to simultaneously determine the number of components n_i and components' reliability

r_i in each subsystem so as to maximize the overall system reliability.

In this study, four well-known benchmark problems of the RRAP are considered. The first three problems with nonlinear constraints, used by [16–18, 25, 31–33, 35–39, 52–54], are a series system, a series–parallel system and a complex (bridge) system. The fourth problem, investigated by [4, 17–25, 35–39, 52–54], is of an overspeed protection system. The four benchmarks are shown to maximize system reliability subject to multiple nonlinear constraints, and they can be stated as mixed-integer nonlinear programming problems. For each problem, both component reliabilities and redundancy allocations are to be decided simultaneously. The mathematical formulations of the four benchmark problems are outlined in the next subsections.

2.1 Notations and symbols

i, j	Indexes of subsystems and constraints, respectively; $1 \le i \le m,\ 1 \le j \le M$
m	Number of subsystems in the system
M	Number of constraints
$\mathbf{n}$	Vector of redundancy allocation of the system; $\mathbf{n} = (n_1, n_2, \ldots, n_m)$
n_i	Number of components in ith subsystem
$\mathbf{r}$	Vector of component reliabilities of the system; $\mathbf{r} = (r_1, r_2, \ldots, r_m)$
r_i	Reliability of each components in ith subsystem
g_j	jth constraint function
w_i	Weight of each component in ith subsystem
c_i	Cost of each component in ith subsystem
v_i	Volume of each component in ith subsystem
R_i	Reliability of the ith subsystem; $R_i = 1 - (1 - r_i)^{n_i}$
Q_i	Unreliability of the ith subsystem, $Q_i = 1 - R_i$
$n_{i,\max}$	Maximum number of components in the ith subsystem
R_s	System reliability
W	Upper bound on system weight
V	Upper bound on system volume
C	Upper bound on system cost
$\mathbb{Z}^+$	Set of nonnegative integers
N	Population size
$t_{\max}$	Maximum number of iterations
d	Number of design parameters
$\alpha,\ \beta$	Two adaptive functions
$a_1,\ a_2$	Two user-defined parameters; $0 \le a_1,\ a_2 \le 1$
σ	Vector of standard deviation of design parameters; $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_d)$
σ_k	Standard deviation of kth design parameter; $1 \le k \le d$
λ	Penalty coefficient, $\lambda \gg 0$

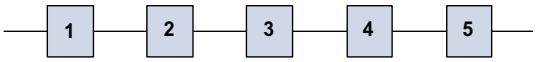

Fig. 1 Series system

2.2 *P*1: series system

The optimization problem of a series system (Fig. 1) is formulated as follows:

$$\text{Maximize } f(\mathbf{r}, \mathbf{n}) = \prod_{i=1}^{5} R_i(n_i) \tag{3}$$

subject to:

$$g_1(\mathbf{r}, \mathbf{n}) = \sum_{i=1}^{5} w_i . v_i^2 . n_i^2 \le V \tag{4}$$

$$g_2(\mathbf{r}, \mathbf{n}) = \sum_{i=1}^{5} \alpha_i . \left(-\frac{1000}{\ln r_i} \right)^{\beta_i} . \left(n_i + e^{0.25 n_i} \right) \le C \tag{5}$$

$$g_3(\mathbf{r}, \mathbf{n}) = \sum_{i=1}^{5} w_i . n_i . e^{0.25 n_i} \le W \tag{6}$$

$$0 \le r_i \le 1,\ r_i \in \mathbb{R},\ n_i \in \mathbb{Z}^+, \quad i = 1, 2, \ldots 5$$

where n_i is the number of components in the ith subsystem, $(1 \le i \le 5)$, r_i is the reliability of each component in subsystem i, $q_i = 1 - r_i^{n_i}$ is the failure probability of each component in subsystem i, $R_i(n_i) = 1 - q_i^{n_i}$ is the reliability of subsystem i, $f(\mathbf{r}, \mathbf{n})$ is system reliability, w_i is the weight of each component in subsystem i, v_i is the volume of each component in subsystem i, V is the upper limit of the sum of the subsystem's products of volume and weight, C is the upper limit of the cost of the system and W is the upper limit on the weight of the system. The parameters α_i and β_i are the physical features of the ith system component. The first constraint $g_1(\mathbf{r}, \mathbf{n})$, given in Eq. (4), is a combination of weight, redundancy allocation and volume. The second constraint $g_2(\mathbf{r}, \mathbf{n})$, presented in Eq. (5), is a cost constraint. The last constraint $g_3(\mathbf{r}, \mathbf{n})$, given in Eq. (6), is a weight constraint. It may be noted that the second constraint involves both integer and continuous variables. The input parameters of the series system are provided in Table 1.

Table 1 Data of problems (*P*1, *P*2, *P*3 and *P*4)

Parameters	*P*1, *P*3	*P*2	*P*4
$10^5.\alpha_i$	(2.330, 1.450, 0.541, 8.050, 1.950)	(2.500, 1.450, 0.541, 0.541, 2.100)	(1, 2.3, 0.3, 2.3)
β_i	(1.5, 1.5, 1.5, 1.5, 1.5)	(1.5, 1.5, 1.5, 1.5, 1.5)	(1.5, 1.5, 1.5, 1.5)
$w_i.v_i^2$	(1, 2, 3, 4, 2)	(2, 4, 5, 8, 4)	(1, 2, 3,2)
w_i	(7, 8, 8, 6, 9)	(3.5, 4, 4, 3.5, 4.5)	(6, 6, 8, 7)
V	110	180	250
C	175	175	400
W	200	200	500

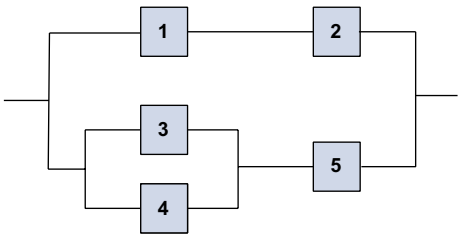

Fig. 2 Series–parallel system

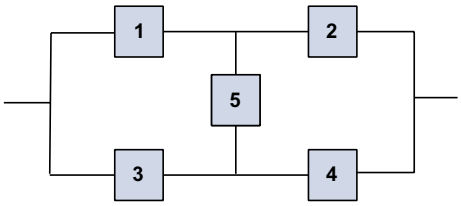

Fig. 3 Complex (bridge) system

2.3 *P*2: series–parallel system

The optimization problem of series–parallel systems (Fig. 2) is stated as follows:

$$\text{Maximize } f(\mathbf{r},\mathbf{n}) = 1-\left(1-R_1R_2\right)\left(1-\left(1-R_3\right)\left(1-R_4\right)R_5\right) \tag{7}$$

subject to the same constraints $g_1(\mathbf{r},\mathbf{n})$, $g_2(\mathbf{r},\mathbf{n})$ and $g_3(\mathbf{r},\mathbf{n})$ as for the series problem, and $0 \le r_i \le 1$, $r_i \in \mathbb{R}$, $n_i \in \mathbb{Z}^+$, $i = 1,2,\ldots 5$. Table 1 illustrates the input parameters of the series–parallel system.

2.4 *P*3: complex (bridge) system

The complex (bridge) system optimization problem is stated as follows:

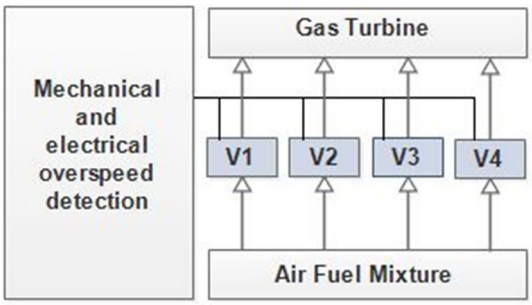

Fig. 4 Overspeed protection system

$$\begin{aligned}\text{Maximize } f(\mathbf{r},\mathbf{n}) = {} & R_1R_2 + R_3R_4 + R_1R_4R_5 + R_2R_3R_5 \\ & - R_1R_2R_3R_4 - R_1R_2R_3R_5 - R_1R_2R_4R_5 \\ & - R_1R_3R_4R_5 - R_2R_3R_4R_5 \\ & + 2R_1R_2R_3R_4R_5\end{aligned} \tag{8}$$

subject to the same constraints $g_1(\mathbf{r},\mathbf{n})$, $g_2(\mathbf{r},\mathbf{n})$ and $g_3(\mathbf{r},\mathbf{n})$ as for the series problem, and $0 \le r_i \le 1$, $r_i \in \mathbb{R}$, $n_i \in \mathbb{Z}^+$, $i = 1,2,\ldots 5$. The input parameters of the complex (bridge) system (Fig. 3) are shown in Table 1.

2.5 *P*4: overspeed protection system

The fourth considered problem is the overspeed protection for a gas turbine system (Fig. 4) [17, 31–40, 42, 52, 55]. This mixed-integer nonlinear problem is formulated as follows:

$$\text{Maximize } f(\mathbf{r},\mathbf{n}) = \prod_{i=1}^{4}\left(1-\left(1-r_i\right)^{n_i}\right) \tag{9}$$

subject to:

$$g_1(\mathbf{r},\mathbf{n}) = \sum_{i=1}^{4} v_i.n_i^2 \le V \tag{10}$$

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

$$g_3(\mathbf{r},\mathbf{n}) = \sum_{i=1}^{4} w_i . n_i . e^{0.25 n_i} \leq W \tag{12}$$

$$0.5 \leq r_i \leq 1 - 10^{-6},\ r_i \in \mathbb{R},\ 1 \leq n_i \leq 10,\ n_i \in \mathbb{Z}^+,\quad i = 1, \ldots, 4$$

The input parameters defining the overspeed protection system are presented in Table 1.

3 Overview of standard PSO

The proposed (ASPSO) algorithm is based on the PSO method. Before discussing the suggested ASPSO, the standard PSO is briefly overviewed in this section.

As a swarm intelligence method, PSO is based on social interactions and individual experiences. In PSO, each individual is called a "particle" which, in fact, represents a potential solution to a problem. Each particle is treated as a point in a d-dimensional space. The jth particle is represented as $\mathbf{x}_j = \{x_{j1}, x_{j2}, \ldots, x_{jd}\}$, with $j = 1, 2, \ldots, N$. The best previous position (the position giving the best fitness value) of any particle is recorded and represented as $\mathbf{p}_j = \{p_{j1}, p_{j2}, \ldots, p_{jd}\}$. The best particle among the whole swarm, known as the global best particle, is represented by $\mathbf{p}_g = \{p_{g1}, p_{g2}, \ldots, p_{gd}\}$. The particle current velocity, which represent the rate of the position change for particle j, is denoted as $\mathbf{v}_j = \{v_{j1}, v_{j2}, \ldots, v_{jd}\}$. The velocity and the position are updated according to the following equations:

$$\mathbf{v}_j^{t+1} = \mathbf{v}_j^t + c_1 r_1 \left(\mathbf{p}_g^t - \mathbf{x}_j^t \right) + c_2 r_2 \left(\mathbf{p}_j^t - \mathbf{x}_j^t \right) \tag{13}$$

$$\mathbf{x}_j^{t+1} = \mathbf{x}_j^t + \mathbf{v}_j^{t+1} \tag{14}$$

where $t = 1, 2, \ldots, t_{\max}$ is the iteration step, c_1 and c_2 are constants named cognitive and social learning factors, respectively, $c_1 + c_2$ is usually limited to 4 [57]. and r_1 and r_2 are two independent random numbers uniformly distributed in the range of [0,1]. Indeed, $\mathbf{v}_j \in \left[-\mathbf{v}_{\max}\ \ \mathbf{v}_{\max} \right]$, where $\mathbf{v}_{\max}$ is a problem-dependent constant defined in order to clamp the excessive roaming of particles. The velocity update equation, i.e., Eq. (13), of the PSO algorithm is a sum of three parts. The first part is that of exploration. The second part is the social part, which represents the collaboration among the particles. The third part is the cognition part, which represents the private thinking of the particle itself [57].

4 Proposed adaptive simplified PSO algorithm for RRAP

4.1 Adaptive simplified PSO algorithm

While empirical research has confirmed the usefulness of PSO as an optimization algorithm, it has shown also its limits to tackle some complex and intricate problems, including RRAPs. Sometimes, PSO can be trapped in local optima and can, in a later period of the optimization process, have a weak convergence rate. To improve the overall performance, many PSO variants have been proposed. Some of these algorithms have incorporated new operations and strategies, and others have combined some existing methods with PSO to produce new hybrid algorithms. Although they have been stated much better than the basic PSO, many of them have introduced additional mathematical or logical processes, which subsequently produced more complicated algorithm and spent more processing time. For the purpose of enhancing the performance of PSO while preserving its simplicity a novel adaptive simplified PSO algorithm (ASPSO) is put forward in this paper. The ASPSO algorithm mainly differs from the standard PSO algorithm in that the candidate solutions are directly represented by the best particle position $\mathbf{p}_j$ and that it uses only a simple updating rule without dealing with velocity.

In order to increase the convergence of PSO, we can write the update of the particle position in a single equation, by substituting Eq. (13) into (14), as:

$$\mathbf{x}_j^{t+1} = \mathbf{x}_j^t + \alpha \left(\mathbf{p}_g^t - \mathbf{x}_j^t \right) + \beta \left(\mathbf{p}_j^t - \mathbf{x}_j^t \right) \tag{15}$$

According to Gandomi et al. [58], this simplified updating model will give the same order of convergence. In addition, as can be noted, the velocity vanishes in the above equation, so avoiding velocity initialization as well as weaknesses related to the explosion phenomenon [57].

Similar to the updating rules of the classical PSO, the simplified updating model, Eq. (15), is based also on both the global best $\mathbf{p}_g$ and the personal best $\mathbf{p}_j$. The main reason of using the personal best is principally to increase diversity in the quality solution. As stated by Yang [59], this diversity can be simulated utilizing some randomness, and hence, there is no need for using personal best. Based on these interpretations, using the global best only will significantly improve the convergence of the algorithm. In contrast, it is well known that personal bests are responsible for maintaining good solutions found in the search space so far. Indeed, the personal bests are nothing but a memory which guides particles during the search process. Hence, depriving the particles of their own memories will decrease the diversity of the swarm and may lead particles

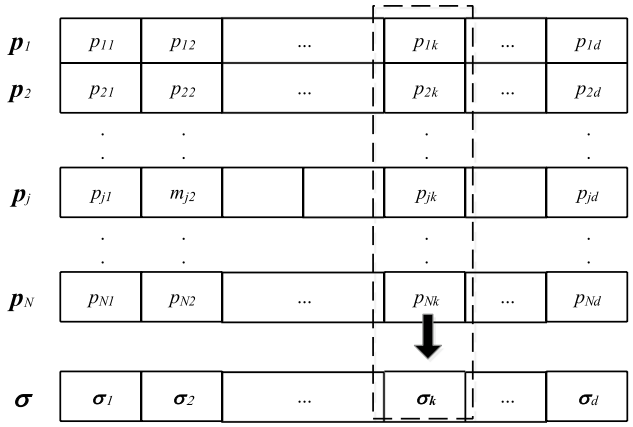

Fig. 5 Schematic representation of σ computation

to converge to a local optimum prematurely. With the intension to further accelerate the PSO convergence, taking advantage of the above-mentioned interpretations, we propose to replace the third term of Eq. (15) by some randomness, as suggested by Yang [59]. However, such randomness is produced using a new technique that conveniently exploits the memory of individuals (i.e., personal bests) gathered during the search process. Formally, the term $\left(\mathbf{p}_j^t - \mathbf{x}_j^t\right)$ in Eq. (15) is substituted by vector $\boldsymbol{\varepsilon}_j$, computed based on the dispersion of individuals best through the search space. The use of this new vector can improve the convergence of the algorithm and avoid premature convergence. Consequently, the particle position will be generated by a simpler formula expressed as follows:

$$\mathbf{x}_j^{t+1} = \mathbf{x}_j^t + \alpha\left(\mathbf{p}_g^t - \mathbf{x}_j^t\right) + \beta\ \boldsymbol{\varepsilon}_j^t \tag{16}$$

where $\boldsymbol{\varepsilon}_j^t$ is a row vector of a normally distributed $(N \times d)$ random matrix $\mathrm{E} = \left[\boldsymbol{\varepsilon}_1\ \boldsymbol{\varepsilon}_2\ \cdots\ \boldsymbol{\varepsilon}_N\right]^T$, computed at each iteration t. Each kth column $(k = 1, \ldots, d)$ of E is randomly generated with a mean zero and a standard deviation $\boldsymbol{\sigma} = \left[\sigma_1, \sigma_2, \ldots, \sigma_k, \ldots, \sigma_d\right]$, where σ_k is the standard deviation of the kth design parameter of all individual best positions, computed as depicted in Fig. 5. As it can be noticed, $\boldsymbol{\varepsilon}_j^t$ is nothing but a vector holding information made from a random combination of particles memory. During the optimization process, matrix $\mathbf{E}$ can be simply computed using the MATLAB command: $\mathrm{E} = normrnd(0, repmat(std(\mathbf{P}), Np, 1))$, where *std*, *repmat* and *normrnd* are three MATLAB functions. $std(\mathbf{P})$ returns a row vector containing the standard deviations corresponding to each column of matrix $\mathbf{P}$, whose rows are the individual bests. $repmat(\boldsymbol{\sigma}, Np, 1)$ returns a matrix containing Np copies of the row vector $\boldsymbol{\sigma}$, and $normrnd(0, \mathbf{A})$ generates a random matrix, with the same size as $\mathbf{A}$, from the normal distribution with mean parameter 0 and a standard deviation from $\mathbf{A}$.

Furthermore, in order to improve search intensification, we propose to focus search in a promising region. This is a region which has a high probability of containing the global optimum. The promising region is determined as a region surrounding the individual best positions. This is done by substituting the current particle position $\mathbf{x}_j^t$, in Eq. (16), by its associate individual best position $\mathbf{p}_j^t$, yielding to the following equation:

$$\mathbf{x}_j^{t+1} = \mathbf{p}_j^t + \alpha\left(\mathbf{p}_g^t - \mathbf{p}_j^t\right) + \beta\ \boldsymbol{\varepsilon}_j^t, \tag{17}$$

showing that the new position is located in the neighborhood of the best position found so far. We can rewrite the proposed updating model of the ASPSO, Eq. (17), as:

$$\mathbf{x}_j^{t+1} = \alpha\ \mathbf{p}_g^t + (1-\alpha)\mathbf{p}_j^t + \beta\ \boldsymbol{\varepsilon}_j^t \tag{18}$$

and schematically depicted in Fig. 6.

The parameters of the proposed model updating are α and β, and their values have a large influence on the ASPSO behavior as well as on its convergence speed. According to Eq. (18), it is clear that the particles will converge toward the global best when $\alpha = 1$ in any given step of the search process, even if the current global best is not the actual global best. In contrast, particles may explore search space without convergence toward optimum when $\alpha = 0$. Therefore, a well-tuned α, during optimization process, is highly mandatory. Indeed, a varying α can be helpful to well balance the exploration and exploitation abilities of the algorithm and to avoid convergence to local optima. In the same context, the randomization term $\beta\ \boldsymbol{\varepsilon}_j^t$ provides the ability, for ASPSO, to explore the search space conveniently and to escape local optima if the parameter β is chosen adequately. Moreover, reducing the randomness as iterations proceed, by varying β, can improve solutions quality. Based on the above analysis, it will be advantageous to consider α and β as two adaptive functions, defined as follows:

$$\alpha = a_1 \times \sin\left(\frac{\pi}{2}\frac{t}{t_{\max}}\right) \tag{19}$$

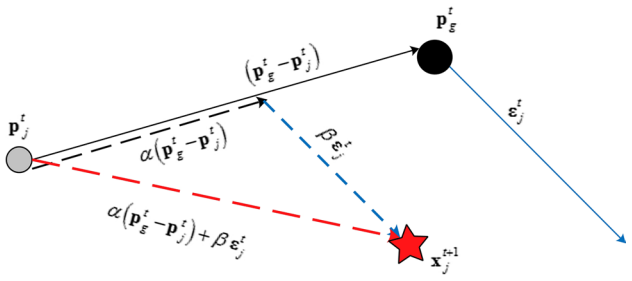

Fig. 6 Position updating strategy of ASPSO algorithm

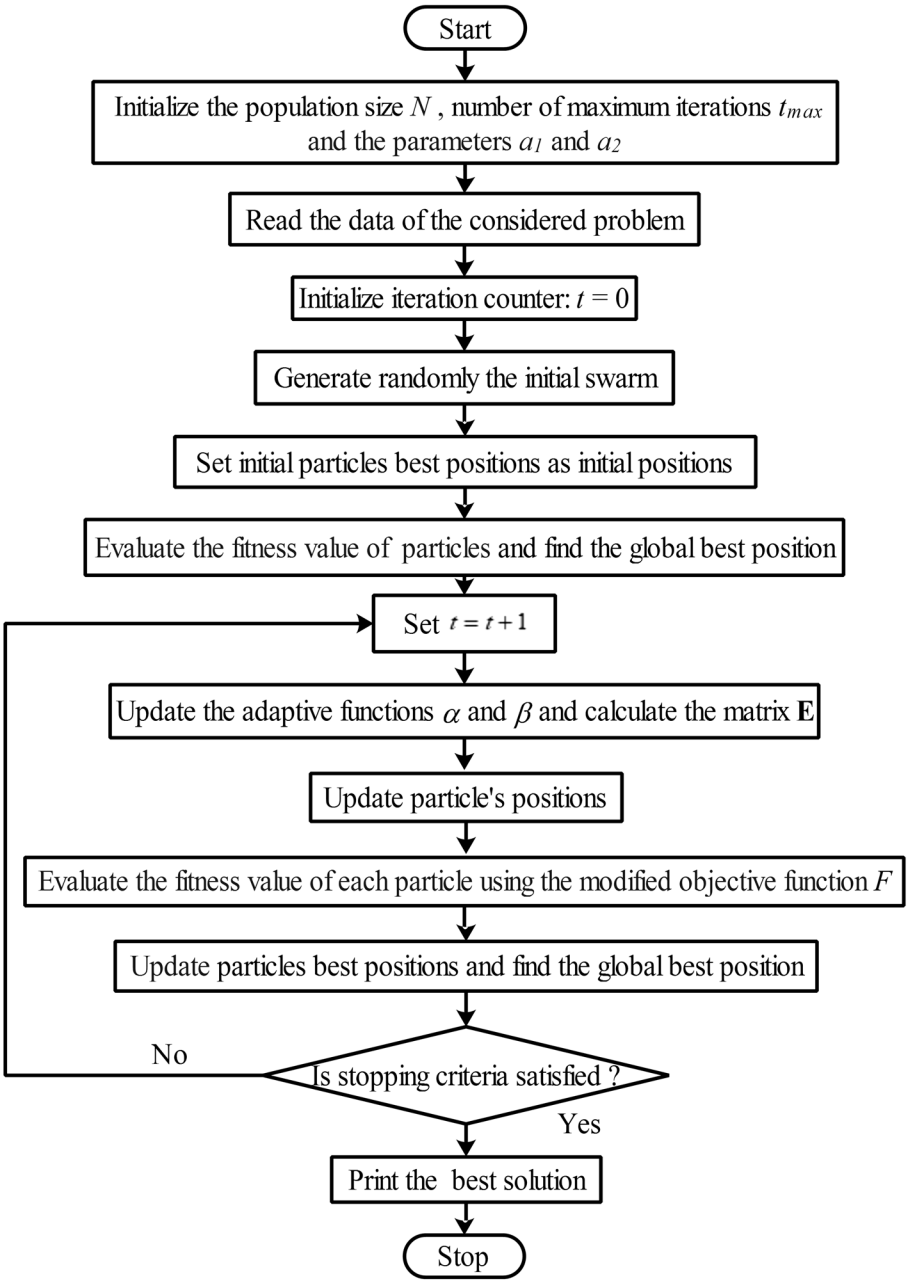

Fig. 7 Flowchart of proposed ASPSO approach

$$\beta = 1 - a_2 \times \sin\left(\frac{\pi}{2}\frac{t}{t_{\max}}\right) \tag{20}$$

where t is the current iteration, $t_{\max}$ is the maximum iteration number and a_1 and a_2 are two user-defined parameters in range [0,1]. As shown in Fig. 7, α is an increasing function which varies between 0 and a_1. According to Eq. (19), small values of α, at earlier iterations, give height control to the personal best position, $\mathbf{p}_j^t$, allowing the global exploration of a promising region, whereas, large values give a more influence of the global best position, $\mathbf{p}_g^t$, which favors solution refinement. In contrast, the second adaptive function β decreases from 1 to $(1 - a_2)$. Indeed, large values of β, at earlier iterations, favor global exploration, while small values, at later iterations, tend to enable exploitation. Accordingly, the fine-tuning of a_2 ensures that the exploration capability is stronger at initial iterations and it fades out during the search process to result in more exploitative capability. Based on a parametric study by varying a_1 from 0.0 to 1.0 and a_2 from 0.5 to 1.0 by a step of 0.1, we found that $a_1 = 0.8$ and $a_2 = 0.5$ are the best values for all considered RRAPs.

4.2 Constraint handling for ASPSO algorithm

Similar to other stochastic optimization methods, the proposed ASPSO algorithm is defined for unconstrained problems. However, during the optimization process, new positions may violate either the limits of the design variables or the problem specific constraints. For any position exceeding the boundary of the variable range, it is reset to the upper (or lower) boundary value. On the other hand, to accommodate the inclusion of constraints, the penalty function method [60] is adopted, as seen in Eq. (21):

$$F(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \text{ is feasible;} \\ f(\mathbf{x}) - \lambda \sum_{l=1}^{m} \max\left(0, g_l(\mathbf{x})\right) & \text{otherwise} \end{cases} \tag{21}$$

where $f(\mathbf{x})$ is the objective function to be maximized, m is the number of constraints, $g_l(\mathbf{x})$ is a specific constraint value and $\lambda > 0$ represents the penalty coefficient and is set to 10^5 in this paper.

4.3 Procedure of ASPSO algorithm

A detailed step-by-step procedure for the suggested ASPSO algorithm is described in the following, while the flowchart of the proposed approach is completely described in Fig. 7.

Step 1 Define the optimization problem and initialize the optimization parameters.

- Define the optimization problem as in Eqs. (1) and (2), the number of design variables (d) and the limits of design variables ($\mathbf{L}_b$, $\mathbf{U}_b$).
- Initialize the population size (N), the number of maximum iterations ($t_{\max}$) and parameters a_1 and a_2.

Step 2 Initialize the population.

- Generate the initial swarm according to the population size and the number of design variables. For ASPSO, the initial location of the jth particle is given as:

$$\mathbf{x}_j^0 = \mathbf{L}_b + \text{rand}\left(\mathbf{U}_b - \mathbf{L}_b\right) \tag{22}$$

where rand is a uniformly distributed random number between 0 and 1.

- Set $\mathbf{p}_j^0 = \mathbf{x}_j^0$, and find the global best position $\mathbf{p}_g^0$ such that $F\left(\mathbf{p}_g^0\right) = \max_j \left(F\left(\mathbf{p}_j^0\right)\right)$,

Step 3 Repeat until the stopping criteria is met: $\left(t = t_{\max}\right)$

- Update the iteration counter $(t = t + 1)$.
- Update the adaptive functions α and β using Eqs. (19) and (20), respectively.
- Calculate the random matrix $\mathbf{E}$, as defined in Sect. 4.1.
- Repeat for each particle, $(j = 1, \ldots, N)$.
  - Calculate a new position $\mathbf{x}_j^{t+1}$ using Eq. (18) and calculate its fitness $F\left(\mathbf{x}_j^{t+1}\right)$.
  - Update the jth best particle position $\mathbf{p}_j^t$: $\mathbf{p}_j^t = \begin{cases} \mathbf{x}_j^{t+1}, & \text{if } F\left(\mathbf{x}_j^{t+1}\right) \geq F\left(\mathbf{p}_j^t\right) \\ \mathbf{p}_j^t & \text{otherwise} \end{cases}$
- Find the current global best $\mathbf{p}_g^t$.

Step 4 Output the global best particle position $\mathbf{p}_g^t$ that holds the best-found solution.

5 Simulation results

Several reliability–redundancy optimization problems involve discrete variables, denoted as n_i which represents the number of redundant components in the ith subsystem. During the optimization process, the integer variables n_i are treated as real variables. In evaluating the objective functions, the real values are transformed to the nearest integer values. The presented ASPSO algorithm is coded in MATLAB programming software and simulation, and numerical solutions are run on an Intel Core i5-3337U 1.8-GHz personal computer with 6 GB of random-access memory (RAM) under a 64-bit Windows operating system. In order to eliminate stochastic discrepancy, in each case study, 50 independent runs are performed for each problem. The parameters of ASPSO are set as follows: the population size $N = 40$, the number of maximum iterations $t_{\max} = 5000$, and parameters a_1 and a_2 are set to 0.8 and 0.5, respectively. These parameters are empirically selected after numerous

Table 2 Statistical analysis for all the problems

Problem no.	Best	Worst	Mean	SD	Mean CPU (s)
*P*1	0.9316823875	0.9316820736	0.9316823618	4.94845×10^{-8}	2.4389
*P*2	0.9999766491	0.9999647634	0.9999760436	2.50091×10^{-6}	3.5551
*P*3	0.9998896375	0.9998893476	0.9998894750	1.42580×10^{-7}	3.5298
*P*4	0.9999546747	0.9999546680	0.9999546738	1.58139×10^{-9}	2.2830

experiments and fine-tuned based on their typical values, defined in Sect. 4.1.

To assess the performance of the proposed algorithm, the statistical features of the best results achieved by the presented ASPSO are shown in Table 2 for each problem. Actually, best, worst and mean, respectively, stand for the best, worst and mean obtained solutions. Standard deviation (SD) is stated as $\mathrm{SD} = \sqrt{\frac{1}{49}\sum_{k=1}^{50}\left(f_k - \bar{f}\right)^2}$, where f_k is the kth converged objective function value, and $\bar{f}$ represents the average objective function value. From these results, it can be concluded that the suggested approach performs quite well in terms of better solution on four RRAPs with a reasonable average computational time.

In addition, the maximum possible improvement (*MPI*) index [31] is used, and it is defined as: $\mathrm{MPI}(\%) = \left(f_{\mathrm{ASPSO}} - f_{\mathrm{other}}\right) / \left(1 - f_{\mathrm{other}}\right)$, where f_{ASPSO} indicates the best system reliability obtained by any other algorithms in the existing literature. Clearly, greater MPI implies greater improvement by the new proposed approach. The numerical results for the four test problems are presented in Tables 3, 4, 5 and 6, wherein the best solutions of each problem and the slack of each constraint are reported. Here, the slack of each constraint is the difference between the available and used resources for the best-found solution. Table 3 depicts the computation results corresponding to the series system (i.e., *P*1). Conspicuously, the best solution obtained by our approach, which is 0.9316823875, dominates those of several existing methods evaluated in [18, 31, 32, 35, 36, 39, 54, 61, 62] with an improvement factor of 2.7507%, 0.0079%, 0.4653%, 0.1526%, 0.0064%, 0.0035%, 0.0044%, 0.0002%, 0.0006%, respectively. It is worth pointing out that the result achieved using the ABC [16] is unfeasible as it violates the cost constraint. The results of the experiment for the second test problem (i.e., series–parallel system), provided in Table 4, show that the best solution by the proposed approach, which is 0.9999766491, is significantly better than the previous best known solutions, so far given by [16, 18, 31, 32, 35, 36, 39, 54]. The improvement attained by the suggested ASPSO over the existing approaches is 25.2771%, 9.5627%, 0.2950%, 0.0390%, 0.1672%, 0.0047% and 0.0004%, respectively, for Hikita et al. [16], Hsieh et al. [18], Chen [31], Wu et al. [54], Afonso et al. [39], He et al. [36] and Hsieh and You [32]. As it can be seen, the solution found by the proposed ASPSO is very close to that presented in [32]. However, even slight enhancements in reliability are critical and beneficial to system security and system efficiency. It may again be mentioned that the solution by the ABC algorithm [35] is unfeasible, since it violates the cost constraint function. Table 5 indicates that the solution of the complex (bridge) system problem (i.e., *P*3) found by the suggested method is comparatively better than the solutions achieved by other approaches [16, 18, 31, 32, 36, 38, 39, 52, 54]. Based on the MPI index, the improvement made by the presented algorithm over the existing approaches is 47.5962%, 8.6706%, 0.3859%, 0.0611%, 0.0159%, 0.0068%, 0.0068%, 0.00136%, and 0.2594%, respectively, for Hikita et al. [16], Hsieh et al. [18], Chen [31], Coelho [52], Zou et al. [38], Wu et al. [54], Afonso et al. [39], He et al. [36], and Hsieh and You [32]. From Table 6, it can be noted that our approach can find solutions significantly better than those provided in [17, 31–33, 36, 38, 39, 52, 54, 63] by an improvement factor 88.3781%, 91.4802%, 21.8529%, 3.5632%, 0.0104%, 0.0104%, 0.0038%, 0.0104%, 6.6188×10^{-5}, and 0.00044%, respectively, from Dhingra [63], Yokota et al. [17], Chen [31], Coelho [52], Zou et al. [38], Wu et al. [54], Afonso et al. [39], Coelho et al. [33], He et al. [36], and Hsieh and You [32]. It can be also observed that the optimal component redundancy by the proposed algorithm is (5,5,4,6), which is completely different from the other methods [17, 31, 36, 38, 39, 52, 54, 63]. The solutions given by the ABC algorithm [35] are unfeasible as well since they violate the cost constraint function. Based on the general observations of the computation results, we deduce that, overall, the presented ASPSO algorithm overcomes the other approaches in the literature in finding most effective solutions for the four reliability–redundancy allocation problems. Furthermore, the standard deviations of system reliabilities by our approach are quite low, given in Tables 3, 4, 5 and 6, and they also indicate that the ASPSO method appears to be efficient to solve the RRAPs. Moreover, the mean CPU time for evaluating the problems is relatively less as compared to other recent metaheuristic algorithms [32, 36] and is shown in their respective tables.

6 Conclusion

The main goal of this paper is to present an efficient approach to solve several reliability–redundancy allocation problems based on a new (ASPSO) algorithm. The proposed method was applied to four different types of RRAP problems including: series, series–parallel, complex (bridge) and overspeed protection systems. In these optimization problems, the redundancies and the corresponding reliability of each component in each subsystem are decided simultaneously under cost, weight and volume constraints. The suggested ASPSO operates with a simple, yet efficient, updating model to adjust the position of particles, without dealing with velocity. In addition, a new randomization technique, based on the dispersion of the particle best through the search space, has been used to speed up the convergence of the ASPSO algorithm and prevent it from being trapped into the local optimum. Moreover, two adaptive functions have been utilized in order to control the balance between exploration and exploitation during the search process. To

Table 3 Comparison of best result for series system ($P1$) using ASPSO with other results presented in the literature

Parameter	Kuo et al. [61]	Xu et al. [62]	Hikita et al. [16]	GA [18]	IA [31]	ABC [35]	IPSO [54]	AR-ICA [39]	NAFSA [36]	Hsieh and You [32]		Proposed approach
										Phase I	Phase II	
$\boldsymbol{n}$	(3,3,2,3,2)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)	(3,2,2,3,3)
r_1	0.77960	0.77939	0.777143	0.779427	0.779266	0.779399	0.78037307	0.779874	0.77938841387	0.780624390	0.779462304	0.7793972771
r_2	0.80065	0.87183	0.867514	0.869482	0.872513	0.871837	0.87178343	0.872057	0.87172098236	0.872299194	0.871883456	0.8718309034
r_3	0.90227	0.90288	0.896696	0.902674	0.902634	0.902885	0.90240890	0.903426	0.90303339184	0.904159546	0.902800879	0.9028815657
r_4	0.71044	0.71139	0.717739	0.714038	0.710648	0.711403	0.71147356	0.710960	0.71141836221	0.710647583	0.711350168	0.7114088472
r_5	0.85947	0.78779	0.793889	0.786896	0.788406	0.787800	0.78738760	0.786902	0.78778928811	0.785079956	0.787861587	0.7878064624
$f(\boldsymbol{r}, \boldsymbol{n})$	0.92975	0.931677	0.931363	0.931578	0.931678	0.931682	0.931680	0.93167939	0.93168226855	0.931662515	0.931682340	0.9316823875
MPI (%)	2.7507	0.0079	0.4653	0.1526	0.0064	[a]	0.0035	0.0044	0.0002	0.0291	0.0006	–
Slack (g_1)	27	27	27	27	27	27	27	27	27	27	27	27
Slack (g_2)	0.000010	0.013773	0.000000	0.121454	0.001559	−0.0002184[b]	0.121454	0.000099	6.7347×10^{-9}	0.00059	0.0000005284	1.49373×10^{-7}
Slack (g_3)	10.57248	7.518918	7.518918	7.518918	7.518918	7.518918	7.518918	7.518918	7.518918	7.518918	7.518918	7.5189182410
Mean	–	–	–	–	–	0.930580	–	0.92182324	0.92300041197	0.9263678	0.93168222	0.9316823618
SD	–	–	–	–	–	8.14×10^{-4}	5.2382×10^{-3}	1.86×10^{-2}	0.0067994	0.0000435	1.3×10^{-14}	4.94845×10^{-8}
Mean CPU	–	–	–	–	–	0.696	–	–	47.84	341.59	53.07813	2.4389

[a]Infeasible

[b]Violate constraint

Table 4 Comparison of best result for series–parallel system (*P*2) using ASPSO with other results presented in the literature

Parameter	Hikita et al. [16]	GA [18]	ABC [35]	IA [31]	IPSO [54]	AR-ICA [39]	NAFSA [36]	Hsieh and You [32]		Proposed approach
								Phase I	Phase II	
$\boldsymbol{n}$	(3,3,1,2,3)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)	(2,2,2,2,4)
r_1	0.838193	0.785452	0.8197457	0.812485	0.81918526	0.82201264	0.819787575273	0.826843262	0.819591561	0.8196618353
r_2	0.855065	0.842998	0.8450080	0.843155	0.84366421	0.84365640	0.845671943728	0.851425171	0.844951068	0.8449934986
r_3	0.878859	0.885333	0.8954581	0.897385	0.89472992	0.89129092	0.894868363315	0.907211304	0.895428548	0.8954738382
r_4	0.911402	0.917958	0.9009032	0.894516	0.89537628	0.89869886	0.895908268569	0.874832153	0.895522339	0.8955294635
r_5	0.850355	0.870318	0.8684069	0.870590	0.86912724	0.86824939	0.868295830551	0.865188599	0.868490229	0.8684462520
$f(\boldsymbol{r},\boldsymbol{n})$	0.99996875	0.99997418	0.99997731	0.99997658	0.99997664	0.99997661	0.999976648004	0.999976094	0.999976649	0.9999766491
MPI (%)	25.2771	9.5627	[a]	0.2950	0.0390	0.1672	0.0047	2.3220	0.0004	–
Slack (g_1)	53	40	40	40	40	40	40	40	40	40
Slack (g_2)	0.000000	1.194440	−1.469522[b]	0.002627	0.000561	0.000396	3.1248×10^{-8}	0.002385	0.000000	5.073315×10^{-8}
Slack (g_3)	7.110849	1.609289	1.609289	1.609289	1.609289	1.609289	1.609289	1.609289	1.609289	1.609289
Mean	–	–	0.99997517	–	–	0.99994991	0.999959785424	0.999951900	0.999976649	0.9999760436
SD	–	–	2.89×10^{-6}	–	1.3362×10^{-5}	2.58×10^{-6}	1.285×10^{-5}	7.6×10^{-10}	3.0×10^{-21}	2.5009×10^{-6}
Mean CPU	–	–	0.936	–	–	–	57.61	135.73	410.375	3.5551

[a]Unfeasible solution

[b]Violate constraint

Table 5 Comparison of best result for complex (bridge) system (*P*3) using ASPSO with other results presented in the literature

Parameter	Hikita et al. [16]	GA [18]	IA [31]	PSO-GC [52]	EGHS [38]	IPSO [54]	AR-ICA [39]	NAFSA [36]	Hsieh and You [32]		Proposed approach
									Phase I	Phase II	
$\boldsymbol{n}$	(3,3,2,3,2)	(3,3,3,3,1)	(3,3,3,3,1)	(3,3,2,4,1)	(3,3,2,4,1)	(3,3,2,4,1)	(3,3,2,4,1)	(3,3,2,4,1)	(3,3,3,3,1)	(3,3,3,3,1)	(3,3,2,4,1)
r_1	0.814483	0.814090	0.812485	0.826678	0.82883148	0.82868361	0.82764257	0.82832179189	0.814422607	0.816624176	0.8281268094
r_2	0.821383	0.864614	0.867661	0.857172	0.85836789	0.85802567	0.85747845	0.85797450730	0.867172241	0.868767396	0.8578229285
r_3	0.896151	0.890291	0.861221	0.914629	0.91334996	0.91364616	0.91419677	0.91422098825	0.859344482	0.858748781	0.9142292842
r_4	0.713091	0.701190	0.713852	0.648918	0.64779451	0.64803407	0.64927379	0.64775717018	0.715805054	0.710279379	0.6480631207
r_5	0.814091	0.734731	0.756699	0.715290	0.70178737	0.70227595	0.70409200	0.70300666185	0.741836548	0.753429200	0.7042949572
$f(\boldsymbol{r},\boldsymbol{n})$	0.9997894	0.99987916	0.99988921	0.99988957	0.99988962	0.99988963	0.99988963	0.99988963601	0.9998891120	0.9998893505	0.9998896375
MPI (%)	47.5962	8.6706	0.3859	0.0611	0.0159	0.0068	0.0068	0.00136	0.4739	0.2594	–
Slack (g_1)	18	18	19	5	5	5	5	5	18	18	5
Slack (g_2)	1.854075	0.376347	0.001494	0.000339	0.00004063	0.00000359	0.00004428	1.5485×10^{-5}	0.011392	0.000000	1.71957×10^{-6}
Slack (g_3)	4.264770	4.264770	4.264770	1.560466	1.56046629	1.56046629	1.56046629	1.56046629	4.264770	4.264770	1.560466288
Mean	–	–	–	0.99988594	0.99988263	–	0.99979532	0.99987756441	0.999824	0.9998893503	0.9998894750
SD	–	–	–	6.9×10^{-7}	1.60×10^{-5}	4.0163×10^{-5}	1.04×10^{-4}	2.1017×10^{-5}	5.6×10^{-9}	4.0×10^{-20}	1.4258×10^{-7}
Mean CPU	–	–	–	–	–	–	–	34.98	138.8469	234.2188	3.5298

Table 6 Comparison of best result for the over-speed protection system (*P*4) using ASPSO with other results presented in the literature

Parameter	Dhingra [63]	GA [17]	IA [31]	ABC [35]	PSO-GC [52]	EGHS [38]	IPSO [54]	AR-ICA [39]	FAC [33]	NAFSA [36]	Hsieh and You [32]		Proposed approach
											Phase I	Phase II	
n	(6,6,3,5)	(3,6,3,5)	(5,5,5,5)	(5,6,4,5)	(5,6,4,5)	(5,6,4,5)	(5,6,4,5)	(5,6,4,5)	(5,5,4,6)	(5,6,4,5)	(5,5,4,6)	(5,5,4,6)	(5,5,4,6)
r_1	0.81604	0.965593	0.903800	0.901614	0.902231	0.90186194	0.90163164	0.90148988	0.90165488	0.90160779120	0.900863647	0.901588628	0.9016126833
r_2	0.80309	0.760592	0.874992	0.849920	0.856325	0.84968407	0.84997020	0.85003526	0.88821801	0.84993077684	0.891220093	0.888192380	0.8882233820
r_3	0.98364	0.972646	0.919898	0.948143	0.948145	0.94842696	0.94821828	0.94812952	0.94807430	0.94814603278	0.949249268	0.948166022	0.9481408193
r_4	0.80373	0.804660	0.890609	0.888223	0.883156	0.88800590	0.88812885	0.88823833	0.84996263	0.88821809379	0.843612671	0.849969792	0.8499230934
$f(\boldsymbol{r},\boldsymbol{n})$	0.99961	0.999468	0.999942	0.999955	0.999953	0.99995467	0.99995467	0.999954673	0.99995467	0.99995467467	0.999953931	0.9999546745[c]	0.9999546747
MPI (%)	88.3781	91.4802	21.8529	[a]	3.5632	0.0104	0.0104	0.0038	0.0104	6.6188×10^{-5}	1.6143	0.00044	–
Slack (g_1)	65	92	50	55	55	55	55	55	55	55	55	55	55
Slack (g_2)	0.064	70.733576	0.002152	−0.0003364[b]	0.975465	0.00120356	0.000009	0.00213782	0.00934729	4.5195×10^{-5}	0.0761580	0.0001250	6.70473×10^{-7}
Slack (g_3)	4.348	127.583189	28.803701	24.80188272	24.801882	24.8018827	24.081883	24.8018827	15.36346308	24.802	15.3634631	15.3634631	15.36346309
Mean	–	–	–	0.9999487	0.999907	0.99993588	–	0.99993804	0.99993907	0.99995075542	0.9999062	0.999954673	0.9999546738
SD	–	–	–	9.244×10^{-6}	1.1×10^{-5}	2.20×10^{-5}	1.3895×10^{-5}	2.20×10^{-5}	1.45×10^{-5}	4.43×10^{-6}	5.7×10^{-9}	4.14×10^{-18}	1.581391576×10^{-9}
Mean CPU	–	–	–	–	–	–	–	–	–	17.96	124.7787	66.54688	2.2830

[a]Unfeasible solution

[b]Violate constraint

[c]Hsieh and You [32] it was reported 0.999954675

evaluate the performance of the ASPSO algorithm, numerical experiments have been conducted and compared with the previous studies for mixed-integer reliability problems. From the computational results, it is seen that the best solutions achieved by the proposed algorithm are all superior to the well-known best solutions in the literature for all test problems. The MPI index indicates that the improvements in our approach appear very small with respect to some previous heuristic techniques. However, even small enhancements in reliability are critical and beneficial to system security and efficiency. The robustness analysis reveals that ASPSO stably searches and explores the near-optimal solution with a very tiny standard deviation value. For further research, one can improve the suggested approach using self-adaptive techniques. Again, it will be interesting to investigate the application of the proposed algorithm to solve RRAP instances with more variables and/or larger-scale benchmarks. Finally, we propose to embed the suggested ASPSO as a local search technique into a global search metaheuristic such as GA or DE in order to solve RRAPs and other engineering problems such as damage detection.

References

1. Aslett LJM, Coolen FPA, Wilson SP (2015) Bayesian inference for reliability of systems and networks using the survival signature. Risk Anal 35:1640–1651. https://doi.org/10.1111/risa.12228
2. Saboori H, Hemmati R, Jirdehi MA (2015) Reliability improvement in radial electrical distribution network by optimal planning of energy storage systems. Energy 93:2299–2312. https://doi.org/10.1016/j.energy.2015.10.125
3. Ye Z-S, Xie M (2015) Stochastic modelling and analysis of degradation for highly reliable products. Appl Stoch Models Bus Ind 31:16–32. https://doi.org/10.1002/asmb.2063
4. Garg H, Rani M, Sharma SP, Vishwakarma Y (2014) Bi-objective optimization of the reliability–redundancy allocation problem for series–parallel system. J Manuf Syst 33:335–347. https://doi.org/10.1016/j.jmsy.2014.02.008
5. Chern M-S (1992) On the computational complexity of reliability redundancy allocation in a series system. Oper Res Lett 11:309–315. https://doi.org/10.1016/0167-6377(92)90008-Q
6. Misra KB, Ljubojevic MD (1973) Optimal reliability design of a system: a new look. IEEE Trans Reliab R-22:255–258. https://doi.org/10.1109/tr.1973.5215673
7. Tian Z, Zuo MJ, Huang H (2008) Reliability–redundancy allocation for multi-state series–parallel systems. IEEE Trans Reliab 57:303–310. https://doi.org/10.1109/TR.2008.920871
8. Ha C, Kuo W (2006) Reliability redundancy allocation: an improved realization for nonconvex nonlinear programming problems. Eur J Oper Res 171:24–38. https://doi.org/10.1016/j.ejor.2004.06.006
9. Agarwal M, Gupta R (2005) Penalty function approach in heuristic algorithms for constrained redundancy reliability optimization. IEEE Trans Reliab 54:549–558. https://doi.org/10.1109/TR.2005.853285
10. Ha C, Kuo W (2006) Multi-path heuristic for redundancy allocation: the tree heuristic. IEEE Trans Reliab 55:37–43. https://doi.org/10.1109/TR.2005.859227
11. Kim J-H, Yum B-J (1993) A heuristic method for solving redundancy optimization problems in complex systems. IEEE Trans Reliab 42:572–578. https://doi.org/10.1109/24.273585
12. Sun XL, Li D (2002) Optimality condition and branch and bound algorithm for constrained redundancy optimization in series systems. Optim Eng 3:53–65. https://doi.org/10.1023/A:1016541912439
13. Sup SC, Kwon CY (1999) Branch-and-bound redundancy optimization for a series system with multiple-choice constraints. IEEE Trans Reliab 48:108–117. https://doi.org/10.1109/24.784268
14. Misra KB, Sharma U (1991) An efficient algorithm to solve integer-programming problems arising in system-reliability design. IEEE Trans Reliab 40:81–91. https://doi.org/10.1109/24.75341
15. Nakagawa Y, Miyazaki S (1981) Surrogate constraints algorithm for reliability optimization problems with two constraints. IEEE Trans Reliab R-30:175–180. https://doi.org/10.1109/tr.1981.5221024
16. Hikita M, Nakagawa Y, Nakashima K, Narihisa H (1992) Reliability optimization of systems by a surrogate-constraints algorithm. IEEE Trans Reliab 41:473–480. https://doi.org/10.1109/24.159825
17. Yokota T, Gen M, Li Y-X (1996) Genetic algorithm for nonlinear mixed integer programming problems and its applications. Comput Ind Eng 30:905–917. https://doi.org/10.1016/0360-8352(96)00041-1
18. Hsieh Y-C, Chen T-C, Bricker DL (1998) Genetic algorithms for reliability design problems. Microelectron Reliab 38:1599–1605. https://doi.org/10.1016/S0026-2714(98)00028-6
19. Coit DW, Smith AE (1996) Reliability optimization of series–parallel systems using a genetic algorithm. IEEE Trans Reliab 45:254–260. https://doi.org/10.1109/24.510811
20. Gen M, Jong RK (1999) GA-based reliability design: state-of-the-art survey. Comput Ind Eng 37:151–155. https://doi.org/10.1016/S0360-8352(99)00043-1
21. Gen M, Yun Y (2006) Soft computing approach for reliability optimization: state-of-the-art survey. Reliab Eng Syst Saf 91:1008–1026. https://doi.org/10.1016/j.ress.2005.11.053
22. Daniel E, Salazar A, Claudio M, Rocco S (2007) Solving advanced multi-objective robust designs by means of multiple objective evolutionary algorithms (MOEA): a reliability application. Reliab Eng Syst Saf 92:697–706. https://doi.org/10.1016/j.ress.2006.03.003
23. Taguchi T, Yokota T, Gen M (1998) Reliability optimal design problem with interval coefficients using hybrid genetic algorithms. Comput Ind Eng 35:373–376. https://doi.org/10.1016/S0360-8352(98)00097-7
24. Tian Z, Zuo MJ (2006) Redundancy allocation for multi-state systems using physical programming and genetic algorithms. Reliab Eng Syst Saf 91:1049–1056. https://doi.org/10.1016/j.ress.2005.11.039
25. Wang L, Li LP (2012) A coevolutionary differential evolution with harmony search for reliability–redundancy optimization. Expert Syst Appl 39:5271–5278. https://doi.org/10.1016/j.eswa.2011.11.012
26. Kulturel-Konak S, Smith AE, Coit DW (2003) Efficiently solving the redundancy allocation problem using tabu search. IIE Trans 35:515–526. https://doi.org/10.1080/07408170304422
27. Ouzineb M, Nourelfath M, Gendreau M (2008) Tabu search for the redundancy allocation problem of homogenous series–parallel multi-state systems. Reliab Eng Syst Saf 93:1257–1272. https://doi.org/10.1016/j.ress.2007.06.004
28. Jang KW, Kim JH (2011) A tabu search for multiple multi-level redundancy allocation problem in series–parallel systems. Int J Ind Eng Theory Appl Pract 18:120–129

29. Agarwal M, Sharma VK (2010) Ant colony approach to constrained redundancy optimization in binary systems. Appl Math Model 34:992–1003. https://doi.org/10.1016/j.apm.2009.07.016
30. Ahmadizar F, Soltanpanah H (2011) Reliability optimization of a series system with multiple-choice and budget constraints using an efficient ant colony approach. Expert Syst Appl 38:3640–3646. https://doi.org/10.1016/j.eswa.2010.09.018
31. Chen T-C (2006) IAs based approach for reliability redundancy allocation problems. Appl Math Comput 182:1556–1567. https://doi.org/10.1016/j.amc.2006.05.044
32. Hsieh YC, You PS (2011) An effective immune based two-phase approach for the optimal reliability–redundancy allocation problem. Appl Math Comput 218:1297–1307. https://doi.org/10.1016/j.amc.2011.06.012
33. dos Santos Coelho L, de Andrade Bernert DL, Mariani VC (2011) A chaotic firefly algorithm applied to reliability–redundancy optimization. In: 2011 IEEE congress of evolutionary computation (CEC), pp 517–521. IEEE. https://doi.org/10.1109/cec.2011.5949662
34. Garg H, Rani M, Sharma SP (2013) An efficient two phase approach for solving reliability–redundancy allocation problem using artificial bee colony technique. Comput Oper Res 40:2961–2969. https://doi.org/10.1016/j.cor.2013.07.014
35. Yeh W-C, Hsieh T-J (2011) Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Comput Oper Res 38:1465–1473. https://doi.org/10.1016/j.cor.2010.10.028
36. He Q, Hu X, Ren H, Zhang H (2015) A novel artificial fish swarm algorithm for solving large-scale reliability–redundancy application problem. ISA Trans 59:105–113. https://doi.org/10.1016/j.isatra.2015.09.015
37. Zou D, Gao L, Wu J, Li S, Li Y (2010) A novel global harmony search algorithm for reliability problems. Comput Ind Eng 58:307–316. https://doi.org/10.1016/j.cie.2009.11.003
38. Zou D, Gao L, Li S, Wu J (2011) An effective global harmony search algorithm for reliability problems. Expert Syst Appl 38:4642–4648. https://doi.org/10.1016/j.eswa.2010.09.120
39. Afonso LD, Mariani VC, Dos Santos Coelho L (2013) Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability–redundancy optimization. Expert Syst Appl 40:3794–3802. https://doi.org/10.1016/j.eswa.2012.12.093
40. Garg H (2015) An approach for solving constrained reliability–redundancy allocation problems using cuckoo search algorithm. Beni-Suef Univ J Basic Appl Sci 4:14–25. https://doi.org/10.1016/j.bjbas.2015.02.003
41. Kumar A, Pant S, Ram M (2017) System reliability optimization using gray wolf optimizer algorithm. Qual Reliab Eng Int 33:1327–1335. https://doi.org/10.1002/qre.2107
42. Garg H (2015) An efficient biogeography based optimization algorithm for solving reliability optimization problems. Swarm Evol Comput 24:1–10. https://doi.org/10.1016/j.swevo.2015.05.001
43. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the sixth international symposium on micro machine and human science, pp 39–43. IEEE. https://doi.org/10.1109/mhs.1995.494215
44. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, pp 1942–1948. IEEE. https://doi.org/10.1109/icnn.1995.488968
45. Ben Guedria N (2016) Improved accelerated PSO algorithm for mechanical engineering optimization problems. Appl Soft Comput J 40:455–467. https://doi.org/10.1016/j.asoc.2015.10.048
46. Garg H (2016) A hybrid PSO-GA algorithm for constrained optimization problems. Appl Math Comput 274:292–305. https://doi.org/10.1016/J.AMC.2015.11.001
47. Chang HH, Lin LS, Chen N, Lee WJ (2013) Particle-swarm-optimization-based nonintrusive demand monitoring and load identification in smart meters. IEEE Trans Ind Appl 49:2229–2236. https://doi.org/10.1109/TIA.2013.2258875
48. Milner S, Davis C, Zhang H, Llorca J (2012) Nature-inspired self-organization, control, and optimization in heterogeneous wireless networks. IEEE Trans Mob Comput 11:1207–1222. https://doi.org/10.1109/TMC.2011.141
49. Bashir ZA, El-Hawary ME (2009) Applying wavelets to short-term load forecasting using PSO-based neural networks. IEEE Trans Power Syst 24:20–27. https://doi.org/10.1109/TPWRS.2008.2008606
50. Zhang Y, Wang S, Ji G (2014) A comprehensive survey on particle swarm optimization algorithm and its applications. Math Probl Eng. https://doi.org/10.1155/2015/931256
51. Nouaouria N, Boukadoum M, Proulx R (2013) Particle swarm classification: a survey and positioning. Pattern Recogn 46:2028–2044. https://doi.org/10.1016/j.patcog.2012.12.011
52. dos Santos Coelho L (2009) An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications. Reliab Eng Syst Saf 94:830–837. https://doi.org/10.1016/j.ress.2008.09.001
53. Yeh WC (2009) A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems. Expert Syst Appl 36:9192–9200. https://doi.org/10.1016/j.eswa.2008.12.024
54. Wu P, Gao L, Zou D, Li S (2011) An improved particle swarm optimization algorithm for reliability problems. ISA Trans 50:71–81. https://doi.org/10.1016/j.isatra.2010.08.005
55. Huang C-L (2015) A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems. Reliab Eng Syst Saf 142:221–230. https://doi.org/10.1016/j.ress.2015.06.002
56. Ouyang Z, Liu Y, Ruan SJ, Jiang T (2019) An improved particle swarm optimization algorithm for reliability–redundancy allocation problem with mixed redundancy strategy and heterogeneous components. Reliab Eng Syst Saf 181:62–74. https://doi.org/10.1016/j.ress.2018.09.005
57. Clerc M, Kennedy J (2002) The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6:58–73. https://doi.org/10.1109/4235.985692
58. Gandomi AH, Yun GJ, Yang X-S, Talatahari S (2013) Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simul 18:327–340. https://doi.org/10.1016/j.cnsns.2012.07.017
59. Yang XS (2010) Nature-inspired metaheuristic algorithms. Luniver Press, Beckington
60. Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191:1245–1287. https://doi.org/10.1016/s0045-7825(01)00323-1
61. Kuo W, Hwang C-L, Tillman FA (1978) A note on heuristic methods in optimal system reliability. IEEE Trans Reliab R-27:320–324. https://doi.org/10.1109/tr.1978.5220401
62. Xu Z, Kuo W, Lin H-H (1990) Optimization limits in improving system reliability. IEEE Trans Reliab 39:51–60. https://doi.org/10.1109/24.52612
63. Dhingra AK (1992) Optimal apportionment of reliability and redundancy in series systems under multiple objectives. IEEE Trans Reliab 41:576–582. https://doi.org/10.1109/24.249589