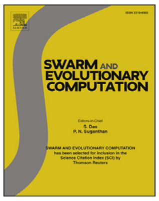
Swarm and Evolutionary Computation 50 (2019) 100562

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



A hybrid enhanced bat algorithm for the generalized redundancy allocation problem

Yue Xu, Dechang Pi [*]

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

A R T I C L E I N F O

Keywords:
Generalized redundancy allocation problem
Cellular automata
Monte Carlo simulation
Constriction coefficient
Transfer function
Discrete bat algorithm
Estimation of distribution algorithm with differential perturbation

A B S T R A C T

A majority of existing works dealing with redundancy allocation problems are based on traditional series-parallel structures. While in many real-life scenarios, the way of connecting subsystems is not limited to a series-only configuration. This paper considers a generalized redundancy allocation problem (GRAP), where the system structure is a more general network. Since the reliability evaluation in GRAPs is a NP-hard problem and the traditional exact symbolic reliability calculation is not suitable, a cellular automata based monte carlo simulation method is implemented in this paper to estimate the system reliability. It is a relatively simple but effective method without knowing the MPs/MCs. Moreover, to deal with GRAPs, a novel discrete bat algorithm is proposed in this paper with a goal of determining an optimal system structure that achieves the minimum cost under several constraints by using redundant components in parallel. Computational complexity of the proposed algorithm is also calculated in this paper. In the end, three experiments are carried out based on ten networks to set parameters, measure the effectiveness of the modifications, and compare with other state-of-the-art algorithms, separately. The reported computational results show that the proposed algorithm is powerful, which is more superior on this sort of problems.

1. Introduction

Redundancy allocation problem (RAP) has been an active research area attracting a great deal of attention in recent decades due to its wide and valuable application [1]. Usually, exchanging the existing components with more reliable components or/and using redundant components in parallel can greatly improve the system reliability [2]. Hence, a core issue of solving RAPs is to configure an optimal structure to strike a balance between system reliability and extra expense. The RAPs have been proved NP-hard by Chern [3]. With increasing system size, RAPs become more and more complicated. Hence, many optimization algorithms have been proposed to tackle these problems. Liang and Smith firstly applied an ant colony meta-heuristic optimization method in reliability design to solve the traditional RAP [4]. The traditional RAP is devoted to the binary-state reliability optimization, where the system and the components experience only two states of perfectly operating or complete failure. Compared to a binary-state system, a multi-state system (MSS), which can experience more than two extreme states, is more practical in the real world [5,6]. Based on the multi-state series-parallel system, Ramirez-Marquez and Coit proposed a heuristic approach for optimizing the RAP [7]. In many real world systems, the entire system, components and subsystems are at different levels. Wang et al. investigated the RAP on multi-level systems and presented a novel memetic algorithm to deal with it [8]. Furthermore, an efficient simulated annealing algorithm was introduced for the RAP, taking a choice of redundancy strategies into consideration [9].

The literatures mentioned above are devoted to redundancy allocation problems with a series-parallel or k-out-of-n type structure. Subsystems are in series, and the components of each subsystem are in parallel. In practice, however, many redundant systems, i.e., telecommunications systems, might have a more complex network topology where subsystems are connected with each other neither in series nor in parallel, but in some logical relationships [10]. Such RAPs are called generalized redundancy allocation problems (GRAP for briefly), firstly discussed in Refs. [10,11]. In GRAPs, subsystems are not limited to a series–only configuration, the components of each subsystem are in parallel. Due to their complex structures, the reliability functions used in the traditional RAPs are not applicable. In addition, optimization algorithms should be improved to solve this kind of problem. Hence, two major challenges of dealing with GRAPs are to calculate the network reliability and to advance optimization algorithms.

Considering GRAPs, network reliability is a common measurement.

* Corresponding author.
E-mail addresses: ayue@nuaa.edu.cn (Y. Xu), dc.pi@nuaa.edu.cn (D. Pi).

https://doi.org/10.1016/j.swevo.2019.100562
Received 3 December 2018; Received in revised form 27 June 2019; Accepted 4 August 2019
Available online 7 August 2019

Most methods, used to calculate the network reliability, are based either on the minimal path (MP) or on the minimal cut (MC) [11–13]. The symbolic reliability in GRAPs was obtained by the inclusion-exclusion method after MPs/MCs was known [11]. There are two drawbacks of this approach. (1) The problems to find all MPs/MCs and to obtain the network reliability of a complex system are both NP-hard [14,15]. (2) It is difficult to determine the exact symbolic reliability functions even for smaller sized networks. A monte carlo simulation (MCS) procedure for a complicated system without the task of knowing MP/MC was proposed in Refs. [16,17]. The MCS methods have the following advantages [18]. (1) MCS methods can be applied to different kinds of network configuration such as series, parallel, and complex networks. (2) MCS can be applied to analyze systems where components follow various distributions. (3) Faster computers are produced as technology progresses, so the manipulating time of MCS decreases. Thus, researches on network reliability measurement by monte carlo simulation (MCS) have become a focus [17, 19–23]. Because some reliability related problems in networks or graphs can be mapped onto cellular automata (CA), a CA-based MCS was presented in Ref. [22] with the following benefits: allowing a straightforward parallel implementation and enhancing the performance of classical algorithms. The literature [23] extended the CA-based MCS methodology for all-terminal and k-terminal connection problems. In this paper, a CA-based MCS methodology containing the advantages of both is utilized to evaluate the reliability of GRAPs.

This paper also aims to propose a novel heuristic algorithm to determine an optimal system structure that achieves the minimum cost under reliability constraint. Bat algorithm (BA) is a bio-inspired algorithm and carries the search process using artificial bats as search agents mimicking the natural pulse loudness and emission rate of real bats [24, 25]. Since the original bat algorithm has been developed by Xin-She Yang in 2010 [24], BAs have been applied in almost every area of optimization [1,26–29]. To our best knowledge, the BA has not been yet used to solve GRAPs. In this paper, a novel discrete bat algorithm is proposed to deal with GRAPs.

The contribution of this paper can be summarized as follows. (1) A CA-based MCS methodology is used to evaluate the reliability of GRAPs. Different from the current studies on GRAPs considering small-sized networks, this method, without knowing MP/MC, is more appropriate regardless of system scale. (2) A constriction coefficient, which is used to update the velocity of bats, is introduced into BA in this paper to improve the global search ability. (3) Considering discrete coding, 8 transform functions, containing s-shaped family and v-shaped family, are utilized to calculate the transform probabilities. By comparing 8 transfer functions, this paper finds the best transfer function for discrete bat algorithms. (4) Estimation of distribution algorithm with differential perturbation is hybridized with the origin BA to enhance the local search capability. The computational complexity is analyzed based on the overview of the proposed algorithm. Three computational experiments, Ex1, Ex2, and Ex3, are carried out to aid in evaluating the performance of the proposed algorithm. In Ex1, the objective is to set parameters and determine which transfer function used in discrete algorithms is the best. In Ex2, the goal is to demonstrate the effectiveness of each modification. The aim of EX3 is to compare the results with other notable methods to further exhibit the performance. Experimental results demonstrate the proposed algorithm outperforms the other existing powerful methods on the created scenarios.

The rest of the paper is arranged as follows. A detail description of the generalized redundancy allocation problem is presented in Section 2, including problem description and CA-based MCS. Related works consisting of BA and EDA are shown in Section 3. Section 4 elaborates on the proposed algorithm with solution representation, fitness function, movement of virtual bats, local search, and the overview of DCBA-dEDA. Parameter setting, experimental results and analysis are supplied in Section 5. This paper is ended with conclusion and forecast in Section 6.

2. Generalized redundancy allocation problem

2.1. Problem description

In the traditional RAPs, subsystems are in series, and the components of each subsystem are in parallel. Consider a simple series-parallel system (shown as Fig. 1) consisting of n subsystems, for any subsystem i, $1 \le i \le n$, there are m_i component types available. Let $x_{i,j}$ be the number of redundant components of type j used in subsystem i, $1 \le j \le m(i)$. Each component can be characterized by its reliability $r_{i,j}$ and cost $c_{i,j}$.

The reliability of subsystem i R_i and the reliability of the whole system R can be calculated as (1) and (2).

$$R_i = 1 - \prod_{j=1}^{m(i)} \left(1 - r_{i,j}\right)^{x_{i,j}} \tag{1}$$

$$R = \prod_{i=1}^{n} R_i \tag{2}$$

The total cost of system C can be computed as (3).

$$C = \sum_{i=1}^{n} \sum_{j=1}^{m(i)} c_{i,j} \times x_{i,j} \tag{3}$$

For both RAPs and GRAPs, the components used in the same subsystem are connected in parallel. Hence, the reliability calculation for parallel connection using (1) is applicable in GRAPs. The main difference between the RAPs and the new GRAPs is the system configuration, in other words, the way of connecting subsystems. In RAPs, all subsystems are connected in a series-only configuration. While in GRAPs, the connection of subsystems is not limited to a series-only configuration. The systems might have a complex network topology because the subsystems are connected with each other neither in series nor in parallel, but in some logical relationships.

Hence, the reliability evaluation in GRAPs is more difficult than that in a series-parallel system. However, whether in the traditional RAPs or the generalized RAPs, exchanging the existing components with more reliable elements or/and using redundant components in parallel can greatly improve the system reliability [2]. A core issue is to configure an optimal structure to strike a balance between system reliability and extra expense. Let $X = [x_{1,1}, x_{1,2}, \ldots, x_{1,m(1)}, \ldots, x_{i,j}, \ldots x_{n,m(n)}]$ denote the system structure (vector); $R(X)$ and R_0 represent the reliability function and system-level constraint limit for the reliability, separately; $n_{\max,i}$ be the maximum number of components for each subsystem. Traditionally, the RAPs can be formulated as the objective of maximizing the system reliability or minimizing the system cost under some constraints. In this paper, the GRAPs are considered with the goal of minimizing system cost subject to the non-linear constraint of reliability using (4).

$$\begin{aligned} &Minimize \quad C = \sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{i,j} \times x_{i,j} \\ &s.t. \qquad R(X) \ge R_0 \\ &\qquad\qquad \sum_{j=1}^{m_i} x_{i,j} \le n_{max,i} \end{aligned} \tag{4}$$

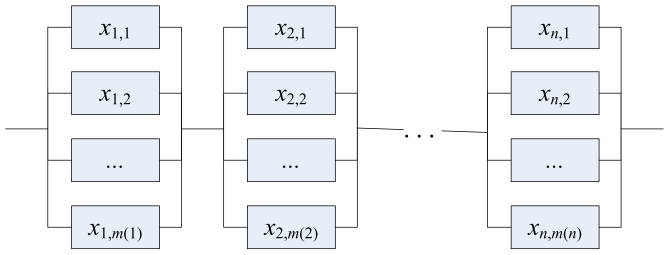


Fig. 1. The traditional RAP (a simple series-parallel system).

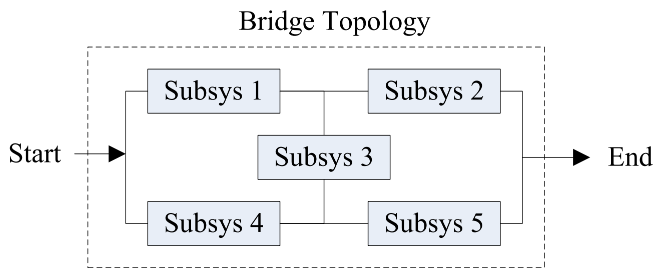


Fig. 2. A simple GRAP configuration (a bridge system).

2.2. CA-based MCS

Based on the above analysis, the reliability function used in RAPs is not applicable in GRAPs. The current network reliability calculation for GRAPs in Ref. [11] is NP-hard and prone to error with the increasing size of the system. In this paper, a CA-based MCS method [22] is implemented to evaluate the reliability of GRAP without the task of knowing MP/MC.

A bridge system, one of the simplest GRAP configurations, is illustrated in Fig. 2. It is common in electronics and coal conveyor systems [30,31], but the RAPs cannot be applied to such a simple example.

To better elucidate the network reliability estimation, the remainder of this paper presents it as an activity on arc (AOA) binary-state network where each arc represents a subsystem that is either operational or nonoperational and where all nodes represent joints that are perfectly and never fail. The AOA network for Fig. 2 is shown in Fig. 3.

CA was originally conceived by Ulam and Von Neumann in the 1950s to provide a framework for studying the behavior of complex systems. Define that G(N,A) is a network with the set of nodes N and arcs A, the main concepts of CA [22,32] are listed as follows.

(1) Nodes and their states: A cell is a node and its state $w(x)$ is coded in a binary digit (0 or 1), which represents the quiescent state or the activated state, where x is the index of node.

$$w(x) = \begin{cases} 0 & quiescent \\ 1 & activated \end{cases} \tag{5}$$

(2) Arcs and their states: Each arc, between node x and y, denotes a subsystem with an associated reliability R_{xy}. The state of each arc s_{xy} can be evaluated using its reliability and a random number *rand*, distributed uniformly between [0,1], as follows.

$$s_{xy} = \begin{cases} 1 & rand \geq R_{xy} \\ 0 & otherwise \end{cases} \tag{6}$$

(3) Neighborhoods: The neighborhood of node x is the set of nodes connected to it by its input arcs, defined as $NE_x = \{y \in N \text{ s.t. } (y,x) \in A\}$.

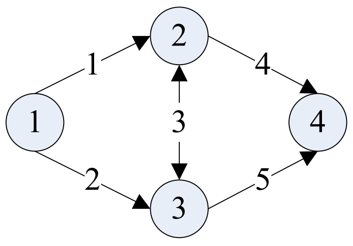


Fig. 3. The AOA binary-state network reduced from the GRAP shown in Fig. 2.

(4) Transition rules: To take into account the probabilistic nature of each arc, the Boolean state for each node by the transition rule is written as:

$$w(x) = OR\left[AND(w(y), x_{yx}), \ldots, AND(w(z), x_{zx})\right],\quad y, \ldots, z \in NE_x \tag{7}$$

All initial nodes are in the quiescent state and the source node is activated. Then each node executes its transition rule, corresponding to the state of its neighborhood node and state of the arc between them.

The basic CA method is shown as Algorithm 1. After evaluating whether there is a path from the source to the end by CA, MCS is used to obtain the approximate system reliability. A CA-based MCS is shown as Algorithm 2.

Algorithm 1
CA.

```
Input: A binary-state work G(N,A) with the source node sn, the sink node tn.
Output: The state of node tn.
Start
1.  Set all nodes in the quiescent state.
2.  Activate the input node sn.
3.  Set $iteration = 1$.
4.  Update each node state by (7).
5.  $iteration = iteration + 1$.
6.  If node t is activated, then stop: there is a path between sn and tn.
7.  If $iteration < |N|-1$ go to 4 else
8.  If node tn is in the quiescent state, then sn-tn path does not exist.
End
```

Algorithm 2
CA-based MCS.

```
Input: A binary-state work G(N,A) with the source node sn, the sink node tn, the
number of simulation replications M.
Output: The estimator $R^*$.
Start
1.  Set $con = 0$, $iteration = 1$.
2.  Generate a random state S for all arcs using (6).
3.  Using CA to evaluate if there is a path between sn and tn. (see Algorithm 1)
4.  If there is a path, then update: $con = con + 1$.
5.  $iteration = iteration + 1$.
6.  If $iteration < M$ go to 2 else stop.
7.  The system reliability $R^* = con/M$.
End
```

Example 1. Take the bridge system shown in Fig. 2 as an example, the information for the components is listed in Table 1. Suppose that the system vector X = [1,1,2,2,0,1,0,1,2,2,2,1,0,1,2], and its structure are shown in Fig. 4.

The total cost of the system corresponding to X using (3) is:

Table 1
Component Types (choices) for the bridge system shown in Fig. 2

i/j	$r_{i,j}$			$c_{i,j}$		
	1	2	3	1	2	3
1	0.55	0.65	0.75	20	60	100
2	0.65	0.85	0.88	30	50	60
3	0.75	0.76	0.77	40	45	60
4	0.55	0.60	0.75	20	30	70
5	0.55	0.65	0.70	20	60	100

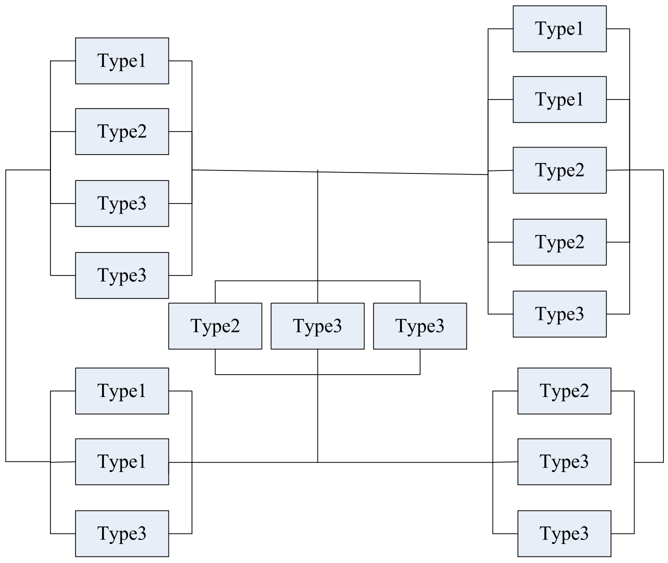


Fig. 4. The system structure corresponding to X.

$$C = \sum_{i=1}^{5}\sum_{j=1}^{3} c_{i,j} \times x_{i,j} \\ = (c_{1,1} \times 1 + c_{1,2} \times 1 + c_{1,3} \times 2) + \ldots + (c_{5,1} \times 0 + c_{5,2} \times 1 + c_{5,3} \times 2) \\ = 995 \tag{8}$$

According to the above analysis, the reliability of each subsystem should be computed in advance using (1).

$$\begin{aligned} R_1 &= 1-(1-0.55)^1\cdot(1-0.65)^1\cdot(1-0.75)^2 \approx 0.9902 \\ R_2 &= 1-(1-0.65)^2\cdot(1-0.85)^0\cdot(1-0.88)^1 = 0.9853 \\ R_3 &= 1-(1-0.75)^0\cdot(1-0.76)^1\cdot(1-0.77)^2 \approx 0.9873 \\ R_4 &= 1-(1-0.55)^2\cdot(1-0.60)^2\cdot(1-0.75)^1 = 0.9919 \\ R_5 &= 1-(1-0.55)^0\cdot(1-0.65)^1\cdot(1-0.70)^2 = 0.9685 \end{aligned} \tag{9}$$

After the reliability of each subsystem is known, a CA-based MCS (see Algorithm 2) is implemented in this paper to estimate the system reliability. Using CA-MCS, after 10,000 replications, the approximate reliability R^* is 0.9996. According to the following symbolic reliability function (10), which was suggested in Ref. [11], the exact system reliability is 0.999594886651725.

$$R = R_1R_4 + (1-R_1R_4)R_2R_5 + (1-R_2)(1-R_4)R_1R_3R_5 \\ + (1-R_1)(1-R_5)R_2R_3R_4 \tag{10}$$

Fig. 5 illustrates the reliability convergence for the Example 1 and provides a graphical view of how the approximations change at each iteration. The experiment starts the number of replications at 1000 and conducts an independent simulation (with 1000 runs) until 100,000th run is reached. As can be observed in Fig. 5, the CA-MCS method can provide a high-quality estimate.

3. Related works

For completeness purpose, a brief presentation of the required background information about BA and EDA is given in the following section.

3.1. BA

Bat algorithm was developed to use the key idea of frequency tuning based on the echolocation of microbats [24]. In the standard bat

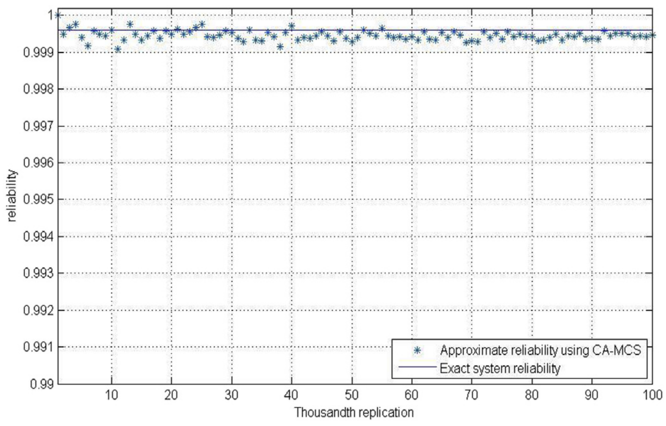


Fig. 5. The reliability convergence for the Example 1.

algorithm, the echolocation characteristics of microbats can be idealized as the following three rules:

- All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way.
- Bat *np* fly randomly with velocity V_{np} at position X_{np} with a fixed frequency $f_{\min}$, varying wavelength λ and loudness L_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $rp \in [0,1]$, depending on the proximity of their target.
- Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) L_0 to a minimum constant value $L_{\min}$.

For each bat (say *np*), the new position X_{np} (*t*), velocity V_{np} (*t*), frequency f_{np} (*t*) at iteration *t* can be updated as follows.

$$f_{np} = f_{\min} + (f_{\max} - f_{\min}) \times rand \tag{11}$$

$$V_{np}(t+1) = V_{np}(t) + \left[X_{np}(t) - X_*\right] \times f_{np} \tag{12}$$

$$X_{np}(t+1) = X_{np}(t) + V_{np}(t+1) \tag{13}$$

where $rand \in [0,1]$ is a random vector drawn from a uniform distribution, two parameters $f_{\min}$ and $f_{\max}$ are the domains of frequency, X_* is the current best solution. In order to improve local search capability, a new solution for each bat is generated locally using a random walk:

$$X_{new} = X_{old} + \varepsilon \times L(t) \tag{14}$$

where X_{old} is a high quality solution, $\varepsilon \in$ [-1,1] is a scaling factor which is a random number, while $L(t) = < L_{np}(t) >$ denotes the average loudness of all the bats at time *t*. Bats tend to decrease the loudness and increase the rate of emitted ultrasonic sound when they chase prey. The pulse rate rp_{np} (*t*) and loudness L_{np} (*t*) are updated as (15) and (16), respectively.

$$L_{np}(t+1) = \alpha \times L_{np}(t) \tag{15}$$

$$rp_{np}(t+1) = rp_{np}(0) \times [1 - \exp(-\gamma \times t)] \tag{16}$$

where α and γ are constants, 0<α<1, γ>0. In fact, α is similar to the cooling factor of a cooling schedule in the simulated annealing. Eventually, L_{np} will equal zero, while the final value of rp_{np} is $rp(0)$. The pseudo-code of the BA is given in Algorithm 3:

Algorithm 3
BA.

```
01. Initialize the bat population and related parameters.
02. Define the fitness function $F$.
03. Evaluate fitness of the bat population.
04. for $t$=1 to $t_{max}$ do
05.     for each bat $X_{np}$ do
06.         Update frequency, velocity, and position using (11)-(13), respectively;
07.         if rand > $rp_{np}$
08.             Generate a local solution around the best solution using (14);
09.         end if
10.         if ( rand < $L_{np}$ ) && ( $f(X_{np}(t+1)) < f(X_*)$ )
11.             Accept the new solution;
12.             Reduce loudness and pulse emission using (15) and (16), respectively;
13.         end if
14.     end for
15.     Rank the bats and find the current best $X_*$
16. end for
```

3.2. EDA

EDA is a stochastic optimization technique that explores the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions [33]. EDA works in the following iterative way.

Algorithm 4
EDA.

```
Start
1. Selection: Select promising solutions from the current population to form the
   parent set by a selection method (e.g., truncation selection).
2. Modelling: Build a probabilistic model $PM(X)$ based on the statistical information
   extracted from the parent set.
3. Sampling: Sample new solutions according to the constructed probabilistic model
   $PM(X)$.
4. Replacement.
End
```

First, the population is sorted according to the fitness function and selects *trun*NP* elitist individuals by step 1 in Algorithm 4, *trun* is the rate of selecting elitist individuals. The aim is to make sure the promising solutions have more chances to enter the next generation by building a probabilistic model.

Considering discrete coding, this paper uses discrete probabilistic model. In this model, the probability of value *v* appearing in position *d* in a *D*-th dimension solution vector *X*, in the population at generation *t*, can be computed as follows.

$$PM\left(X^d(t)=v\right)=\frac{\sum_{np=1}^{NP} equal\left(X_{np}^d(t),v\right)}{NP} \tag{17}$$

$$equal\left(X_{np}^d(t),v\right)=\begin{cases}1 & if\ X_{np}^d(t)=v\\ 0 & otherwise\end{cases} \tag{18}$$

The probabilistic matrix of elitist populations by truncation selection is shown as follows.

$$PE\left(X^d(t)=v\right)=\frac{\sum_{np=1}^{trun\cdot NP} equal\left(X_{np}^d(t),v\right)}{trun\times NP} \tag{19}$$

After building the probabilistic model, these newly generated vectors can be represented as a probability matrix by simply counting the number of occurrences of each value in each bit position. A competitive learning method is used to update the probabilistic model as shown in (20). The probability update rule can not only retain enough historical information but also collect current elite individual information.

$$PM\left(X^d(t+1)=v\right)=(1-LR)\cdot PM\left(X^d(t)=v\right)+LR\cdot PE\left(X^d(t)=v\right) \tag{20}$$

where, *LR* is learning rate parameter. To further explain this procedure, a simple example of minimum sum function (dimension $D=3$, population size $NP=10$, $np=2$, selected rate $trun=0.4$, $v\in\{0,\ 1,\ 2\}$, $LR=0.1$) is given below.

(1) Initial population (the last column is its fitness values) *X*(*t*) according to its probabilistic matrix *PM*(*t*) at iteration *t*:

$$PM\left(X^d(t)=v\right)=\begin{bmatrix}0.3&0.2&0.4&0.6&0.2\\0.5&0.5&0.5&0.4&0.4\\0.2&0.3&0.1&0&0.4\end{bmatrix}_{v\times d},X(t)=\begin{bmatrix}0&2&0&1&1&4\\1&1&1&0&1&4\\0&0&0&1&2&3\\2&2&1&1&1&7\\1&2&1&0&0&4\\0&1&1&0&0&2\\2&1&0&0&2&5\\1&1&1&1&2&6\\1&0&0&0&1&2\\1&1&2&0&2&6\end{bmatrix}$$

(2) Selection: sort the population *X*(*t*) and select 4 elitist individuals to form the parent set *XE*(*t*).

$$X(t)=\begin{bmatrix}0&1&1&0&0&2\\1&0&0&0&1&2\\0&0&0&1&2&3\\0&2&0&1&1&4\\1&1&1&0&1&4\\1&2&1&0&0&4\\2&1&0&0&2&5\\1&1&1&1&2&6\\1&1&2&0&2&6\\2&2&1&1&1&7\end{bmatrix},XE(t)=\begin{bmatrix}0&1&1&0&0&2\\1&0&0&0&1&2\\0&0&0&1&2&3\\0&2&0&1&1&4\end{bmatrix}$$

(3) Modelling: Build a probabilistic model *PE*(*t*) for elite population *XE*(*t*).

$$PE\left(X^d(t)=v\right)=\begin{bmatrix}0.75&0.5&0.75&0.5&0.25\\0.25&0.25&0.25&0.5&0.5\\0&0.25&0&0&0.25\end{bmatrix}_{v\times d}$$

(4) Sampling: Generate a new solution X_{EDA} according to the new probabilistic update rule *P*(*t*+1).

$$\begin{aligned}P\left(X^d(t+1)=v\right)&=0.9\cdot PM\left(X^d(t)=v\right)+0.1\cdot PE\left(X^d(t)=v\right)\\&=\begin{bmatrix}0.345&0.23&0.435&0.59&0.205\\0.475&0.475&0.475&0.41&0.41\\0.18&0.295&0.09&0&0.385\end{bmatrix}_{v\times d}\end{aligned}$$

$$X_{EDA}=\begin{bmatrix}1&1&0&0&1&3\end{bmatrix}$$

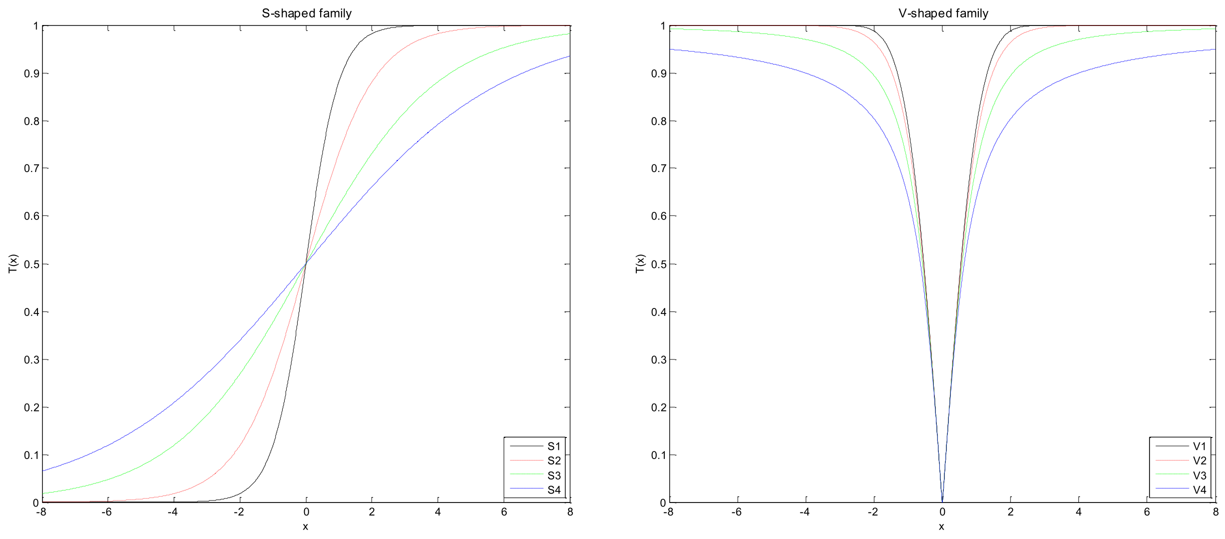


Fig. 6. S-shaped and v-shaped family of transfer functions.

4. Solution method: a novel enhanced discrete bat algorithm

This paper proposes a novel discrete bat algorithm to deal with the generalized redundancy allocation problem, including constriction coefficient, transfer function, EDA with differential perturbation. Details are elaborated below.

4.1. Solution representation

Same as a majority of swarm-intelligence algorithms, the proposed algorithm has a fixed number of solutions, and the length of each solution is determined by the problem. For the generalized redundancy allocation problem, assume the number of subsystems is n, for any subsystem i, $1 \le i \le n$, there are m_i component types available. Let $x_{i,j}$ be the number of redundant components of type j used in subsystem i, $1 \le j \le m(i)$. Due to the unknown positions of the food resources, the initial bat population X is randomly generated from a discrete-value vector, encoded as $[x_{1,1}, x_{1,2}, \ldots, x_{1,m(1)}, \ldots, x_{i,j}, \ldots x_{n,m(n)}]$.

4.2. Fitness function

This paper considers the GRAP with a goal of minimizing system cost subject to the non-linear constraints of reliability. As shown in Section 2, the total cost and reliability of the system are computed using (3) and Algorithm 2 (CA-based MCS), respectively. Since the number of simulation replications is M, the computational complexity of the fitness calculation is $O(M)$.

Given a demand reliability R_0, the population may encounter infeasible solutions of which the reliability is less than the given value (i.e., $R(X) < R_0$). For guiding a search toward feasible regions, it is common to use a penalty function [30,31,34,35]. Thus, the fitness F can be obtained as (21).

$$F = C + \max(0, pen \times (R_0 - R)) \tag{21}$$

where pen is a penalty coefficient set to 10000 in this study.

4.3. Movement of virtual bats

The movement of virtual bats means the global search ability, which makes the algorithm explore every region of the feasible search space. During the past few years, many modifications have been proposed to update the velocity and position of bats, such as inertial weight [26], directional echolocation [27], habitat selection [36], different triangle-flipping strategy [37], random disturbance strategy [38], centroid strategy [39], and so on. In this paper, the update of velocity equation is enhanced by constriction coefficient, firstly proposed by Ref. [40]. Then, considering discrete coding for GRAPs, different transfer functions are introduced to map velocity values to probability values and the position is updated.

4.3.1. Constriction coefficient

The velocity update Eq. (12) consists two terms. Only if using the first term affects the solutions may it be observed that these solutions overflow the space by keeping their velocities and directions, thus reducing their convergence speeds rapidly; While only if the second term affects the solutions may it be observed that the solutions converge to a region somewhere around the global best solution ($X*$), thus facing the premature convergence problem [26]. To tackle this issue, a tradeoff between exploration and exploitation is very important. In this paper, the following modification structure is proposed as follows, inspired by the study [40].

Define that $X_{np}^d(t)$ and $V_{np}^d(t)$ denote the position and velocity of bat np in d-th dimension at iteration t separately; Each bat np contains a memory of its previous best solution, represented as P_{np}, $P_{np}^d(t)$ is the d-th position of its previous best solution at iteration t; P_g is the current best solution found in the population. A new velocity update equation using constriction coefficient is shown as (22).

$$V_{np}^d(t+1) = \chi\left[V_{np}^d(t) + \left(X_{np}^d(t) - P_g^d(t)\right) \cdot f_{np}^d \cdot c_1 + \left(X_{np}^d(t) - P_{np}^d(t)\right) \cdot f_{np}^d \cdot c_2\right] \tag{22}$$

In Eq. (22), the parameter χ is constriction coefficient. The difference between the current solution and the current best solution indicates the global search ability and makes the algorithm explore every region of the feasible search space; the difference between the current solution and the previous best solution means the local search ability and accelerates the algorithm converging to the near-optimal solutions. Acceleration constants c_1 and c_2, called social and cognitive parameter respectively, are used to balance between exploration and exploitation.

4.3.2. Transfer function

For solving GRAPs, one of the key issues is to transform bat position to allow only discrete values. Transfer functions are considered as the simplest and cheapest operators in designing discrete heuristic algorithms [41]. According to Rashedi et al. [42], some concepts should be taken into account as follows.

- The range of a transfer function should be bounded in the interval [0,1], as they represent the probability that a particle should change its position.
- A transfer function should provide a high probability of changing the position for a large absolute value of the velocity. Particles having large absolute values for their velocities are probably far from the best solution, so they should switch their positions in the next iteration.
- A transfer function should also present a small probability of changing the position for a small absolute value of the velocity.
- The return value of a transfer function should increase as the velocity rises. Particles that are moving away from the best solution should have a higher probability of changing their position vectors in order to return to their previous positions.
- The return value of a transfer function should decrease as the velocity reduces.

Based on these concepts, eight transfer functions were proposed to map velocity values to probability values in Refs. [43,44]. Due to the shapes of these curves depicted in Fig. 6, these functions can be grossly divided into two categories: s-shaped family and v-shaped family. The mathematical formulation is also available in Table 2.

After calculating the probabilities using transfer functions, the position updating rule used in binary PSO is presented as:

$$X_{np}^{d}(t+1)=\begin{cases}\left(X_{np}^{d}(t)\right)^{-1} & \textit{if rand} < T\left(V_{np}^{d}(t+1)\right)\\ X_{np}^{d}(t) & \textit{otherwise}\end{cases} \tag{23}$$

where $(X_{np}^{d}(t))^{-1}$ is the complement of $X_{np}^{d}(t)$. Considering discrete coding, a position updating rule for Discrete BA (DBA) presented in this paper is as (24).

$$X_{np}^{d}(t+1)=\begin{cases}X_{np}^{d}(t)+K & \textit{if rand} < T\left(V_{np}^{d}(t+1)\right)\\ X_{np}^{d}(t) & \textit{otherwise}\end{cases} \tag{24}$$

$K\in$[-1,0,1] is a discrete random variable.

4.4. Local search

In order to improve the local search capability, a local random walk was used to generate a new position in the origin BA using (14). The high quality solution is chosen among the best solutions according to the Algorithm 3. To tackle the randomness of selection, BA is hybridized by EDA with differential perturbation, dEDA for briefly. Hybridization is a growing area of intelligent systems research, which aims to combine the desirable properties of different approaches to mitigate their individual weaknesses [45]. To our knowledge, BA has been combined with many other state-of-the-art algorithms to complement on each other, e.g., simulated annealing (SA) [46], invasive weed optimization (IWO) [26], harmony search (HS) [47], and so on.

As we known, EDA utilizes the probabilistic model to guide the exploitation of the search space. Hence, the high quality solution X_{old} is generated by EDA rather than selecting the best solution, denoted as X_{EDA}. This benefits enhancing the local search ability and preventing solutions from getting trapped in local optimum. Inspired by the random walk around the high quality solution, this paper perturbs the solution generated by EDA using a random differential operator, shown as (25).

$$X_{new}=\begin{cases}X_{EDA}+\left(X_{ran\_i}-X_{ran\_j}\right) & \textit{if } (rand\cdot L<p)\\ X_{EDA} & \textit{otherwise}\end{cases} \tag{25}$$

where *ran_i* and *ran_j* are two random values from a uniform distribution; *p* is a differential perturbation operator. If a random value multiplied by the loudness is less than *p*, this paper perturbs X_{EDA}. The advantage of introducing the differential perturbation into EDA is that the difference between two random solutions adjusts the search area around X_{EDA}. At the beginning of the iteration, the difference is large and the algorithm searches more widely round X_{EDA}; as the iteration progresses, the difference becomes smaller and the search region around X_{EDA} reduces, so as to enhance the exploitation ability of BA.

4.5. Overview of DCBA-dEDA

This paper proposes a discrete BA with constriction coefficient and dEDA, denoted as DCBA-dEDA. According to the above description, the completed procedure of DCBA-dEDA is described in Algorithm 5.

Algorithm 5
DCBA-dEDA.

```
01. Initialize the bat population and related parameters.
02. Define the fitness function $F$ using (21).
03. Evaluate fitness of the bat population.
04. for $t$=1 to $t_{max}$ do
05.     for each bat $X_{np}$ do
06.         Update frequency, velocity, and position by (11), (22), and (24), respectively;
07.         if rand > $rp_{np}$
08.             Obtain a new solution by EDA using Algorithm 4;
09.             Generate a local solution by dEDA using (25);
10.         end if
11.         if ( rand < $L_{np}$ ) && ( $F(X_{np}(t+1)) < F(X_*)$ )
12.             Accept the new solution;
13.             Reduce loudness $L$ and pulse emission $rp$ using (28) and (16), respectively;
14.         end if
15.     end for
16.     Rank the bats and find the current best $X_*$
17. end for
```

Line 6 represents the movement of virtual bats with the following enhancements: updating the velocity using constriction coefficient and generating the discrete position by transfer function. Line 7 to 10 are improvements for the local search ability: the hybridization with EDA guides the exploitation of the search space and makes BA not easy to get into local optimum than a local random walk; introducing the random differential operator into EDA makes a tradeoff between the exploitation and exploration capability during the iteration. In addition, the different values of loudness, which controls the acceptance or rejection of a new solution, have a great influence on the results of the algorithm, in the experiment section. The loudness is updated by (28) from the results in parameter tuning. According to the above discussions, the flowchart of the proposed algorithm is illustrated in Fig. 7.

As shown in Ref. [37], the computational complexity of the standard BA is $O(t_{\max}\times NP\times f)$, where $t_{\max}$ is the maximum number of

Table 2
The mathematical formulation of transfer functions.

S-shaped family		V-shaped family	
Name	Transfer function	Name	Transfer function
S1	$T(x)=\frac{1}{1+e^{-2x}}$	V1	$T(x)=\left\lvert erf\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert=\left\lvert\frac{2}{\sqrt{\pi}}\int_0^{(\sqrt{\pi}/2)x}e^{-t^2}dt\right\rvert$
S2	$T(x)=\frac{1}{1+e^{-x}}$	V2	$T(x)=\lvert\tanh(x)\rvert$
S3	$T(x)=\frac{1}{1+e^{(-x/2)}}$	V3	$T(x)=\left\lvert(x)/\sqrt{1+x^2}\right\rvert$
S4	$T(x)=\frac{1}{1+e^{(-x/3)}}$	V4	$T(x)=\left\lvert\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\rvert$

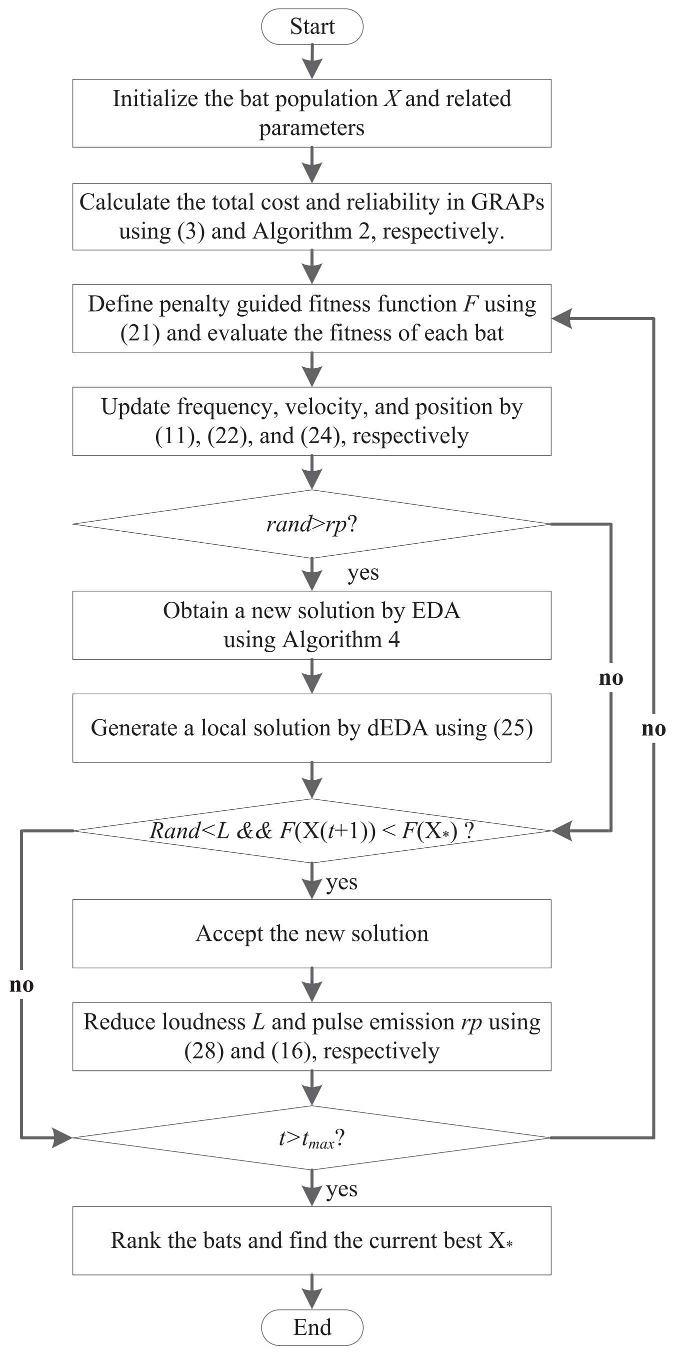

Start
Initialize the bat population X and related parameters
Calculate the total cost and reliability in GRAPs using (3) and Algorithm 2, respectively.
Define penalty guided fitness function F using (21) and evaluate the fitness of each bat
Update frequency, velocity, and position by (11), (22), and (24), respectively
$rand>rp$?
yes
Obtain a new solution by EDA using Algorithm 4
Generate a local solution by dEDA using (25)
no
no
$Rand<L$ && $F(X(t+1)) < F(X_*)$?
yes
Accept the new solution
no
Reduce loudness L and pulse emission rp using (28) and (16), respectively
$t>t_{max}$?
yes
Rank the bats and find the current best X_*
End

Fig. 7. The flowchart of the proposed algorithm DCBA-dEDA.

generations, NP is the population size, and $O(f)$ is the computational complexity of its fitness evaluation. In this paper, the computational complexity of fitness evaluation is mainly decided by the Algorithm 2 (CA-based MCS). Since the number of simulation replications is M, the computational complexity of the fitness calculation is $O(M)$. For the movement of virtual bats (line 6), it does not increase extra loop operations. While for the part of local search, it performs dEDA including sorting the population $O(NP \times \log(NP))$, building a $n_{\max} \times D$ probabilistic model $O(n_{\max} \times D)$, generating a new EDA solution from a $n_{\max} \times D$ probabilistic model $O(n_{\max} \times D)$, and differential perturbation $O(1)$, where $n_{\max}$ is the maximum number of components for each subsystem and D is the dimension of the solution. Hence, the computational complexity of local search is $O(NP \times \log(NP)+2\, n_{\max} \times D)$. Above all, the total computation complexity of the proposed algorithm when dealing with GRAPs is $O(t_{\max} \times NP \times M \times (NP \times \log(NP)+n_{\max} \times D))$.

Table 3
Quantitative time complexity (all results in seconds).

D	$T0$	$T1$	$<T2>$	$(<T2> - T1)/T0$
10	0.1390	0.1537	0.8990	5.3605
30		0.4476	1.4618	7.2958
50		0.9098	2.2310	9.5040
100		3.1218	5.0501	13.8714

Furthermore, the quantitative time complexity is also measured according to the guidelines in Ref. [48], and the results on 10, 30, 50, and 100 dimensions of the benchmark function f_{18} [48] are shown in Table 3. $T0$ is the time calculated by running the following test program (Algorithm 6); $T1$ denotes the time to execute 200,000 evaluations of f_{18}; $T2$ represents the time to execute DCBA-dEDA with 200,000 evaluations of f_{18} and $<T2>$ is an average of $T2$ obtained in five independent runs.

Algorithm 6
Test algorithm.

```
for i=1: 1000000
    x=0.55 + (double)i; x= x + x; x= x + x/2; x= x^2;
    x= sqrt(x); x= log(x); x= exp(x); x= x/(x+2);
end for
```

5. Experimental results and discussion

In this paper, ten networks are adopted to test algorithms, which are more complex than the existing GARPs. Three computational experiments, Ex1, Ex2, and Ex3, are implemented to aid in evaluating the performance of the proposed algorithm. In Ex1, the objective is to set parameters. In Ex2, the goal is to demonstrate the effectiveness of modifications. In Ex3, the aim is to compare the results with other notable methods.

5.1. Benchmark problems and experimental setup

Ten benchmark problems [11,19] in the field of network reliability, shown as Fig. 8, are adopted in this study to test all algorithms. The number of nodes and edges are listed in Table 4, denoted as *n_node* and *n_edge*. As it can be seen from Table 4, network 9 and network 10 are relatively large. It may require excessive simulation time to estimate the network reliability and determine the optimal redundancy allocation. The structure of GRAPs solved in Ref. [11] are more simple than that of network 9 and network 10. Hence, it challenges the CA-based MCS method.

The component data including type, reliability, and cost is shown in Table 5. Here, i is the index of subsystem and j is the index of component type. For example, case 1 represents there are three choices for subsystem 1: reliability 0.55 and cost 20, reliability 0.65 and cost 60, or reliability 0.75 and cost 100.

The number of simulation runs is set to 10,000 when calculating the fitness function, the reliability constraint is 0.99, and the maximum number of components for each subsystem is 2. Besides, the population size of is 50 and the max iteration is 200. In order to make a fair comparison, these common parameters are identical for all test algorithms. Each algorithm is executed 10 times independently for each instance.

5.2. Ex1: parameter setting

The setting of parameters has a significant influence on the efficiency of stochastic algorithms [49]. Generally, methods for changing the value of a parameter can be classified into three categories [50], deterministic parameter control, adaptive parameter control, and self-adaptive parameter control. In the first way, the parameter is changed by some deterministic in rule predetermined manner without using any feedback from the search, i.e., a time-varying schedule. The second way takes

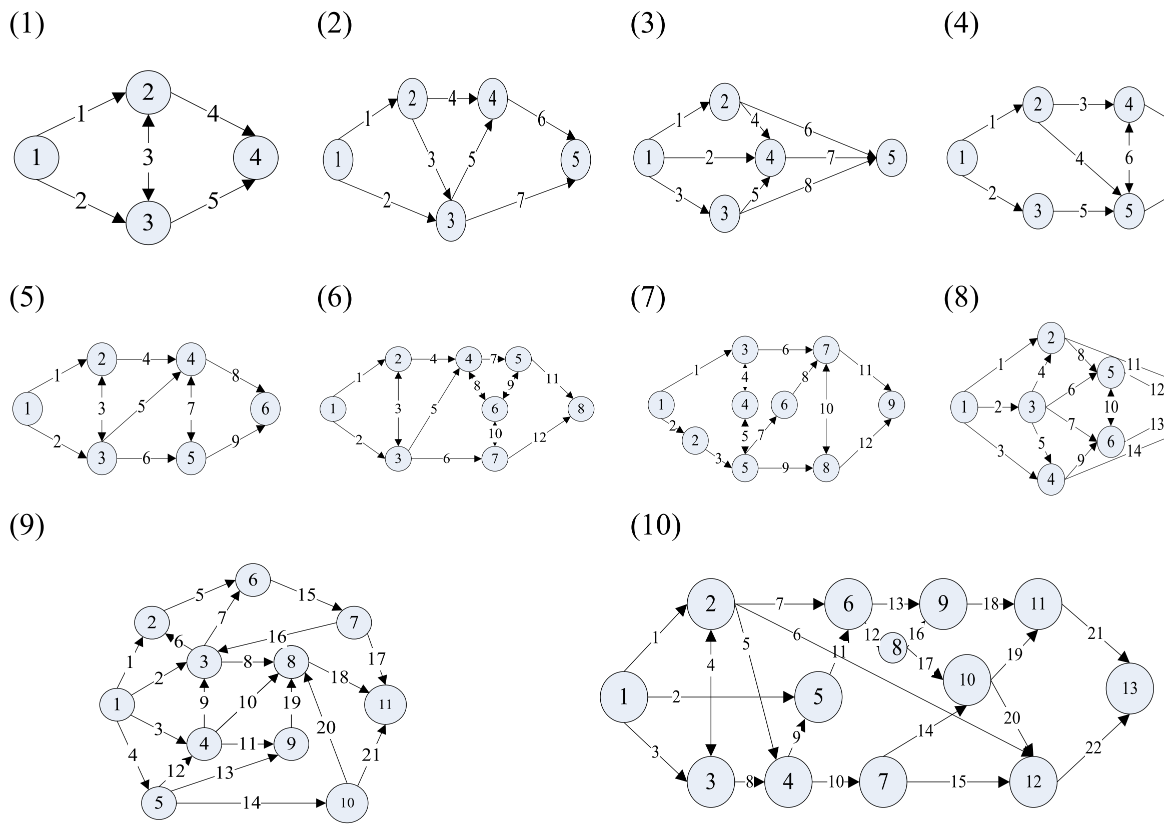


Fig. 8. Benchmark networks 1-10.

Table 4
The number of edges and nodes for network 1-10.

No.	n_edge	n_node	No.	n_edge	n_node
1	5	4	6	12	8
2	5	7	7	12	9
3	8	5	8	14	7
4	8	6	9	21	11
5	9	6	10	22	13

feedback form the search as an input to alter the parameter. In the last method, the parameter is determined by encoding into the chromosomes and undergoing the operators. In the parameter setting in BA, Iztok et al. adopted the third method to control two strategy parameters (the pulse rate and the loudness) and proposed a novel hybrid self-adaptive bat algorithm [51]. In this paper, parameter tuning is carried out before the run and deterministic parameter control is adopted during the search process.

The comparison phase of parameter tuning is evaluated by the mean value obtained by the algorithm at the end of iteration, and the best mean value is in bold.

(1) Constriction coefficient χ and Acceleration constants c_1 and c_2: As shown in Ref. [40], type 1 constriction coefficient is a function of ϕ, where $\phi = c_1 + c_2$, which can be calculated using the following Eq. (26).

$$\chi = \begin{cases} \dfrac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} & \text{if } \phi > 4 \\ 1 & \text{otherwise} \end{cases} \tag{26}$$

As reported in Ref. [40], quick almost linear convergence is

Table 5
The component data for GRAP test problems.

i/j	$r_{i,j}$			$c_{i,j}$			i/j	$r_{i,j}$			$c_{i,j}$		
	1	2	3	1	2	3		1	2	3	1	2	3
1	0.55	0.65	0.75	20	60	100	12	0.59	0.60	0.62	15	20	25
2	0.65	0.85	0.88	30	50	60	13	0.55	0.60	0.70	65	70	90
3	0.75	0.76	0.77	40	45	60	14	0.66	0.69	0.73	65	72	99
4	0.55	0.60	0.75	20	30	70	15	0.65	0.66	0.69	30	31	35
5	0.55	0.65	0.70	20	60	100	16	0.54	0.56	0.59	10	15	20
6	0.60	0.69	0.70	20	30	35	17	0.68	0.69	0.75	60	69	90
7	0.66	0.78	0.80	50	70	80	18	0.50	0.52	0.53	25	30	40
8	0.58	0.59	0.63	30	55	60	19	0.65	0.67	0.69	50	60	80
9	0.68	0.73	0.77	15	25	35	20	0.68	0.70	0.75	75	100	120
10	0.52	0.54	0.62	45	46	50	21	0.54	0.59	0.63	45	49	59
11	0.62	0.68	0.70	35	40	45	22	0.60	0.70	0.80	90	120	200

obtained only when $\phi > 4$. In this paper, the factor ϕ is trained with different settings, including $\phi > 4$ (3.9), $\phi = 4$, and $\phi > 4$ (4.1, 4.2, 4.3, 4.4, 4.5), which is demonstrated in Table 6.

As seen from Table 6, the case of $\phi > 4$ obtains better results than the rest cases, which is consistent with [40]. Clerc recommended a satisfactory setting $\chi = 0.729$, $c_1 = c_2 = 2.05$, which was also utilized in many PSOs papers [52–54]. While combining BA with constriction coefficient, the setting of $\phi = 4.1$ outperforms only on network 1. As the networks becomes more complicated, the factor with higher value seems a suitable setting. When ϕ is set to 4.3, the algorithm achieves better results on 5 out of 10 problems compared with other values. Hence, Eq. (27) is considered as a satisfactory setting in the proposed algorithm.

$$\chi = 0.582,\ c_1 = c_2 = 2.15 \tag{27}$$

(2) Transfer function T: Transfer functions are utilized in the DCBA-dEDA to map velocity values to probability values for updating the positions. This paper independently carries out the proposed algorithm with eight different transfer functions, as shown in Table 7.

As it can be seen from Table 7, the following results are found: (1) 6 out of 10 mean values obtained by V4-based DCBA-EDA are the lowest and the solution to Network 2 achieved by V4-based DCBA-EDA is the lowest as that obtained by V3-based DCBA-EDA; (2) the means obtained by v-shaped family based algorithms are better than that obtained by s-shaped family based algorithms; (3) When using s-shaped family transfer functions, the means decrease from S1 to S4 at most cases. Note that the mathematical formulations of s-shaped transfer functions can be unitize as $T(x) = 1/(1 + \exp(-s^*x))$, where $s \in \{2, 1, 1/2, 1/3\}$. Hence, it could also be concluded that when using s-shaped transfer functions T(x), the mean value declines with the decrease of value s.

(3) Loudness L: The decreasing factor L controls the acceptance or rejection of a new solution. However, according to the origin BA, the probability of accepting new solution is very low, or even zero, in the later stage of evolution. Hence, this paper carries out different strategies to update the loudness, such as (a) random, (b) $L = 0.95$, $\alpha = 0.9$, which was suggested in Ref. [26], (c) monotonically decreasing, proposed in Ref. [27], (d) nonlinear decreasing $n = 2$, (e) nonlinear decreasing $n = 3$, (f) nonlinear decreasing $n = 4$. The results of different trails are shown in Table 8.

It is clear that the 5 out of 10 best values are obtained when using nonlinear decreasing strategy, shown in (28). Where, L_{init} and L_{final} are the initial and final value of the factor, respectively. The following settings were recommended in Ref. [27]: $L_{init} = 0.9$ and $L_{final} = 0.6$.

$$L = \left(\frac{t_{max} - t}{t_{max}}\right)^n (L_{init} - L_{final}) + L_{final},\ n = 2 \tag{28}$$

(4) Perturbation factor p: In the proposed dEDA, the key controlled factor is perturbation operator p, which decides the proportion of differential perturbation in EDA. This parameter is trained with different values ranging from 0 to 1, listed in Table 9. It is clear from the result indicated in Table 9 that the proposed algorithm with low differential perturbation rate produces better solution. Hence, the perturbation factor is suggested 0.2.

As recommended by the above experiments, the optimum values used for DCBA-dEDA in this study have been summarized as below: constriction coefficient $\chi = 0.582$, acceleration constants $c_1 = c_2 = 2.15$; transfer function V4; loudness L using (28), $L_{init} = 0.9$ and $L_{final} = 0.6$; perturbation factor $p = 0.2$.

5.3. Ex2: effectiveness analysis

This paper combines the origin BA with constriction coefficient and dEDA. To confirm the contribution of these enhancement structures in BA, the following experiments are carried out. In the first trail, the original BA without any modification is executed. In the second trail, the BA is executed with constriction coefficient (cBA, proposed in this paper). The following trails run the BA with EDA and dEDA, respectively. In the last trail, the BA is run with constriction coefficient and dEDA. Note that each algorithm is discrete and applies transfer function V4. The common parameters for these trails are all same, as suggested in parameter tuning.

Table 10 reports the mean results **of** these trails. It can be seen from Table 10 that the BA with improvement structures (constriction coefficient or/and dEDA) achieves better result than itself, which means that these improvement structures are effective. In addition, the BA with dEDA performs better than BA-EDA on 7 out of 10 problems, which implies the differential perturbation enhances the performance of BA with EDA.

5.4. Ex4: comparison of existing notable algorithms

To measure the efficiency of the proposed method, it has been compared with the results of several well-performing algorithms. The description and parameters of these algorithms are listed as below:

(1) DCBA-dEDA (this paper): A discrete BA with constriction coefficient and dEDA, the limitation of frequency $f_{\min} = 0$ and $f_{\max} = 1$, constriction coefficient $\chi = 0.582$, acceleration constants $c_1 = c_2 = 2.15$, transfer function V4, loudness L using (28), $L_{init} = 0.9$ and $L_{final} = 0.6$, pulse emission $rp = 0.85$, updating factors $\gamma = 0.9$, perturbation factor $p = 0.2$.

Table 6
Mean value of ϕ on different benchmark functions.

No.	ϕ						
	3.9	4	4.1	4.2	4.3	4.4	4.5
1	4.13E+02	4.09E+02	**3.85E+02**	3.98E+02	4.10E+02	4.07E+02	4.09E+02
2	4.60E+02	4.67E+02	4.19E+02	**4.16E+02**	4.21E+02	4.37E+02	4.23E+02
3	7.05E+02	7.11E+02	6.32E+02	6.48E+02	**6.29E+02**	6.44E+02	6.55E+02
4	5.57E+02	5.42E+02	4.83E+02	4.66E+02	**4.66E+02**	4.84E+02	4.84E+02
5	5.76E+02	5.72E+02	4.92E+02	**4.68E+02**	4.72E+02	4.86E+02	4.83E+02
6	7.56E+02	7.11E+02	5.58E+02	5.24E+02	**5.14E+02**	5.15E+02	5.50E+02
7	7.15E+02	6.91E+02	5.23E+02	5.12E+02	**4.91E+02**	4.94E+02	5.20E+02
8	1.28E+03	1.29E+03	1.13E+03	1.02E+03	**9.85E+02**	1.01E+03	1.03E+03
9	1.84E+03	1.88E+03	1.62E+03	1.47E+03	1.42E+03	**1.31E+03**	1.32E+03
10	2.00E+03	2.03E+03	1.62E+03	1.45E+03	1.35E+03	1.33E+03	**1.29E+03**

Table 7
Mean value of T on different benchmark functions.

No.	T							
	S1	S2	S3	S4	V1	V2	V3	V4
1	4.01E+02	4.03E+02	3.96E+02	3.92E+02	3.82E+02	**3.79E+02**	3.80E+02	3.80E+02
2	4.83E+02	4.67E+02	4.56E+02	4.45E+02	4.24E+02	4.26E+02	**4.20E+02**	**4.20E+02**
3	7.22E+02	6.99E+02	6.97E+02	6.83E+02	6.56E+02	6.44E+02	**6.31E+02**	6.40E+02
4	5.74E+02	5.62E+02	5.39E+02	5.33E+02	4.84E+02	**4.80E+02**	4.91E+02	4.86E+02
5	5.97E+02	5.97E+02	5.57E+02	5.44E+02	5.08E+02	5.00E+02	4.87E+02	**4.80E+02**
6	8.10E+02	8.04E+02	7.44E+02	6.75E+02	6.19E+02	5.98E+02	5.84E+02	**5.62E+02**
7	7.96E+02	7.84E+02	7.16E+02	7.11E+02	5.66E+02	5.47E+02	5.37E+02	**5.26E+02**
8	1.37E+03	1.33E+03	1.31E+03	1.29E+03	1.16E+03	1.16E+03	1.14E+03	**1.12E+03**
9	2.06E+03	2.01E+03	2.00E+03	1.92E+03	1.70E+03	1.67E+03	1.67E+03	**1.64E+03**
10	2.23E+03	2.25E+03	2.19E+03	2.05E+03	1.78E+03	1.76E+03	1.67E+03	**1.66E+03**

Table 8
Mean value of L on different benchmark functions.

No.	L					
	a	*b*	*c*	*d*	*e*	*f*
1	4.08E+02	4.34E+02	3.87E+02	3.83E+02	**3.77E+02**	3.87E+02
2	4.62E+02	4.75E+02	4.35E+02	**4.31E+02**	4.40E+02	4.35E+02
3	6.89E+02	7.23E+02	6.60E+02	**6.58E+02**	6.61E+02	6.65E+02
4	5.26E+02	5.64E+02	5.05E+02	5.13E+02	**5.03E+02**	5.09E+02
5	5.56E+02	6.02E+02	5.17E+02	**5.12E+02**	5.17E+02	5.29E+02
6	7.12E+02	7.55E+02	6.21E+02	**6.19E+02**	6.25E+02	6.45E+02
7	6.92E+02	7.60E+02	6.00E+02	**5.87E+02**	6.15E+02	5.96E+02
8	1.25E+03	1.32E+03	**1.18E+03**	1.19E+03	1.20E+03	1.22E+03
9	1.88E+03	1.95E+03	**1.75E+03**	1.78E+03	1.76E+03	1.76E+03
10	1.97E+03	2.10E+03	1.84E+03	1.84E+03	1.85E+03	**1.79E+03**

Table 9
Mean value of p on different benchmark functions.

No.	p										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	3.82E+02	3.85E+02	3.81E+02	3.83E+02	3.84E+02	**3.73E+02**	3.76E+02	3.82E+02	3.84E+02	3.80E+02	3.80E+02
2	4.16E+02	4.17E+02	4.15E+02	**4.07E+02**	4.14E+02	4.08E+02	4.13E+02	4.22E+02	4.29E+02	4.36E+02	4.37E+02
3	6.14E+02	6.18E+02	**6.06E+02**	6.14E+02	6.16E+02	6.12E+02	6.27E+02	6.54E+02	6.62E+02	6.60E+02	6.66E+02
4	4.61E+02	4.59E+02	4.69E+02	4.60E+02	**4.58E+02**	4.60E+02	4.68E+02	4.81E+02	5.02E+02	5.07E+02	5.05E+02
5	4.57E+02	4.50E+02	**4.48E+02**	4.59E+02	4.50E+02	4.51E+02	4.61E+02	4.90E+02	5.10E+02	5.05E+02	5.11E+02
6	**5.03E+02**	5.07E+02	5.09E+02	5.06E+02	5.06E+02	5.06E+02	5.15E+02	5.57E+02	6.08E+02	6.12E+02	6.01E+02
7	4.93E+02	4.82E+02	**4.81E+02**	4.84E+02	4.81E+02	4.86E+02	4.94E+02	5.22E+02	5.44E+02	5.77E+02	5.80E+02
8	9.91E+02	9.83E+02	9.86E+02	**9.80E+02**	9.95E+02	9.95E+02	1.02E+03	1.15E+03	1.21E+03	1.17E+03	1.20E+03
9	1.21E+03	**1.21E+03**	1.21E+03	1.21E+03	1.23E+03	1.24E+03	1.38E+03	1.49E+03	1.67E+03	1.70E+03	1.70E+03
10	1.22E+03	1.24E+03	**1.22E+03**	1.23E+03	1.23E+03	1.26E+03	1.32E+03	1.52E+03	1.70E+03	1.60E+03	1.68E+03

(2) wBA [26]: BA with inertia weight, the limitation of frequency $f_{\min} = 0$ and $f_{\max} = 1$, loudness $L = 0.95$, pulse emission $rp = 0.85$, updating factors $\alpha = \gamma = 0.9$, initial and final values of inertia weight: $w_{init} = 0.9$ and $w_{final} = 0.2$, modulation index of inertia weight $n = 2$, coefficient factor $\xi_{init} = 0.6$, transfer function V4.
(3) DBA [27]: Directional BA, the limitation of frequency $f_{\min} = 0$ and $f_{\max} = 2$, the limitation of pulse emission $r_{init} = 0.1$ and $r_{final} = 0.7$, loudness L using monotonically decreasing function, $L_{init} = 0.9$ and $L_{final} = 0.6$, transfer function V4.
(4) BA_OR [38]: BA with optimal forage strategy and random disturbance strategy, frequency f is selected randomly from [0,5], loudness $L = 0.9$, pulse emission $rp = 0.9$, the updating factors $\alpha = 0.99$ and $\gamma = 0.9$, switch parameter $\eta = 0.8$, transfer function V4.
(5) EDA: Estimation of distribution algorithm, learning rate $LR = 0.1$, truncation selection rate $trun = 0.1$.
(6) SSO [11,55]: Simplified swarm optimization, threshold value $C_g = 0.55$, $C_p = 0.75$, and $C_w = 0.95$.
(7) HSO [11]: Hybrid swarm optimization, HSO with Level 2 in Factor 1, level 3 in Factor 2, and level 1 in Factor 3, threshold value $C_g = 0.55$, $C_p = 0.75$, and $C_w = 0.95$.

In these comparison algorithm, wBA, DBA, and BA_OR are three notable variants of BA, EDA is one of the hybrid algorithms of the proposed algorithm, SSO and HSO are current state of the art methods for solving the generalized redundancy allocation problem. The results of all compared algorithms carried out on different cases are presented in Table 11, including the mean value (Mean), the stand deviation (Std), the min value (Min), and the max value (Max). The best results are in bold.

From the results shown in Table 11, the proposed algorithm DCBA-

Table 10
Mean results for effectiveness analysis.

No.	BA	cBA	BA-EDA	BA-dEDA	DCBA-dEDA
1	4.17E+02	4.07E+02	4.12E+02	4.10E+02	3.72E+02
2	4.73E+02	4.37E+02	4.65E+02	4.63E+02	4.14E+02
3	7.08E+02	6.44E+02	6.90E+02	6.85E+02	6.17E+02
4	5.59E+02	4.84E+02	5.55E+02	5.57E+02	4.64E+02
5	5.73E+02	4.86E+02	5.65E+02	5.63E+02	4.51E+02
6	7.28E+02	5.15E+02	7.21E+02	7.22E+02	5.10E+02
7	6.96E+02	4.94E+02	6.06E+02	6.10E+02	4.78E+02
8	1.27E+03	1.01E+03	1.16E+03	1.15E+03	9.74E+02
9	1.88E+03	1.31E+03	1.86E+03	1.85E+03	1.19E+03
10	2.02E+03	1.33E+03	1.93E+03	1.90E+03	1.25E+03

Table 11
The mean results of all compared algorithms.

No.	Statistics	DCBA-dEDA	wBA	DBA	BA_OR	EDA	SSO	HSO
1	Mean	**3.72E+02**	4.03E+02	3.86E+02	3.77E+02	3.95E+02	4.10E+02	4.04E+02
	Std	1.08E+01	1.34E+01	1.20E+01	**5.30E+00**	1.98E+01	1.98E+01	1.63E+01
	Min	**3.60E+02**	3.80E+02	3.60E+02	3.65E+02	3.60E+02	3.90E+02	3.90E+02
	Max	3.90E+02	4.20E+02	4.05E+02	**3.80E+02**	4.30E+02	4.60E+02	4.40E+02
2	Mean	**4.14E+02**	4.49E+02	4.27E+02	4.20E+02	4.44E+02	4.24E+02	4.25E+02
	Std	1.06E+01	1.20E+01	1.49E+01	**7.25E+00**	2.02E+01	1.71E+01	2.14E+01
	Min	**3.95E+02**	4.25E+02	4.10E+02	4.10E+02	4.00E+02	4.00E+02	4.10E+02
	Max	**4.30E+02**	4.60E+02	4.65E+02	**4.30E+02**	4.70E+02	4.50E+02	4.80E+02
3	Mean	**6.17E+02**	6.85E+02	6.28E+02	6.39E+02	6.51E+02	6.46E+02	6.50E+02
	Std	1.58E+01	2.28E+01	**1.40E+01**	1.49E+01	3.54E+01	2.66E+01	4.28E+01
	Min	**5.90E+02**	6.60E+02	6.10E+02	6.10E+02	6.10E+02	6.15E+02	6.10E+02
	Max	**6.40E+02**	7.20E+02	6.50E+02	6.65E+02	7.15E+02	6.85E+02	7.35E+02
4	Mean	**4.64E+02**	5.37E+02	4.82E+02	4.74E+02	4.87E+02	4.77E+02	4.91E+02
	Std	1.41E+01	2.58E+01	1.21E+01	**6.99E+00**	1.81E+01	1.23E+01	1.98E+01
	Min	**4.45E+02**	5.05E+02	4.60E+02	4.65E+02	4.50E+02	4.55E+02	4.55E+02
	Max	**4.90E+02**	5.85E+02	5.00E+02	4.90E+02	5.05E+02	4.95E+02	5.20E+02
5	Mean	**4.51E+02**	5.56E+02	4.69E+02	4.65E+02	4.90E+02	4.84E+02	4.82E+02
	Std	**6.99E+00**	3.40E+01	1.39E+01	8.82E+00	3.28E+01	2.17E+01	2.31E+01
	Min	**4.40E+02**	5.20E+02	4.45E+02	4.50E+02	4.55E+02	4.50E+02	4.50E+02
	Max	**4.60E+02**	6.10E+02	4.90E+02	4.75E+02	5.75E+02	5.10E+02	5.10E+02
6	Mean	**5.10E+02**	6.79E+02	5.33E+02	5.39E+02	5.31E+02	5.16E+02	5.29E+02
	Std	2.05E+01	5.18E+01	2.73E+01	**1.36E+01**	1.81E+01	1.58E+01	2.35E+01
	Min	**4.85E+02**	5.95E+02	5.00E+02	5.20E+02	5.01E+02	4.95E+02	4.95E+02
	Max	5.50E+02	7.65E+02	5.86E+02	5.65E+02	5.55E+02	**5.40E+02**	5.70E+02
7	Mean	**4.78E+02**	6.56E+02	5.04E+02	4.97E+02	5.16E+02	4.99E+02	4.94E+02
	Std	**1.33E+01**	5.59E+01	2.13E+01	1.40E+01	2.54E+01	2.91E+01	1.89E+01
	Min	**4.56E+02**	5.65E+02	4.75E+02	4.70E+02	4.85E+02	4.60E+02	4.70E+02
	Max	**5.01E+02**	7.21E+02	5.51E+02	5.20E+02	5.60E+02	5.50E+02	5.30E+02
8	Mean	**9.74E+02**	1.20E+03	1.03E+03	1.08E+03	1.02E+03	1.00E+03	1.00E+03
	Std	3.23E+01	4.17E+01	4.46E+01	3.29E+01	4.59E+01	**2.83E+01**	4.54E+01
	Min	**9.26E+02**	1.13E+03	9.82E+02	1.02E+03	9.65E+02	9.60E+02	9.31E+02
	Max	**1.01E+03**	1.27E+03	1.13E+03	1.12E+03	1.10E+03	1.05E+03	1.10E+03
9	Mean	**1.19E+03**	1.79E+03	1.32E+03	1.48E+03	1.30E+03	1.23E+03	1.24E+03
	Std	**3.59E+01**	5.95E+01	4.26E+01	5.26E+01	3.66E+01	4.26E+01	5.99E+01
	Min	**1.15E+03**	1.70E+03	1.27E+03	1.34E+03	1.24E+03	1.18E+03	1.15E+03
	Max	**1.25E+03**	1.89E+03	1.39E+03	1.53E+03	1.34E+03	1.31E+03	1.33E+03
10	Mean	**1.25E+03**	1.84E+03	1.34E+03	1.46E+03	1.27E+03	1.28E+03	1.26E+03
	Std	3.99E+01	5.38E+01	5.08E+01	3.99E+01	5.23E+01	4.77E+01	**3.12E+01**
	Min	**1.18E+03**	1.76E+03	1.29E+03	1.40E+03	1.20E+03	1.22E+03	1.21E+03
	Max	**1.31E+03**	1.92E+03	1.41E+03	1.51E+03	1.36E+03	1.36E+03	1.32E+03

dEDA achieves the best mean value at all cases. Considering SD, BA_OR obtains the best SD for 4 problems (Network 1, Network 2, Network 4, and Network 6); DCBA-dEDA performs the best for 3 problems; DBA, SSO, and HSO get the best SD for one network, respectively. It implies the proposed algorithm is more stable than any compared algorithm except BA_OR.

For a more comprehensive statistical analysis, Friedman test [56] is performed to show the average ranking of all algorithms. Furthermore, two post-hoc tests (Wilcoxon test and Nemenyi test) are conducted to overcome the drawback of Freidman test, as suggested in Ref. [57]. In the Wilcoxon test [58], a pairwise comparison between the algorithm with the lowest rank and other algorithms is carried out. The Nemenyi test is applied to report any significant differences between individual classifiers [59] and displays the graphical presentation of the results. Both Nemenyi and Wilcoxon test are conducted with a significance level 0.05.

Table 12
The results of Friedman test and Nemenyi test.

Algorithm	Friedman	Nemenyi	
	Mean rank	*CD*	S.
DCBA-dEDA	1.93	[1.48, 2.38]	‡
wBA	6.74	[6.29, 7.19]	†
DBA	3.89	[3.44, 4.34]	†
BA_OR	4.09	[3.64, 4.54]	†
EDA	4.26	[3.81, 4.71]	†
SSO	3.50	[3.05, 3.95]	†
HSO	3.60	[3.15, 4.05]	†

The results of non-parametric test are listed in Table 12, including the mean rank of Friedman test together with critical difference *CD* of Nemenyi test. It should be noted that Symbol ‡ represents the best algorithm in Nemenyi test, Symbol † denotes the significant difference between the best algorithm and the corresponding algorithm. The

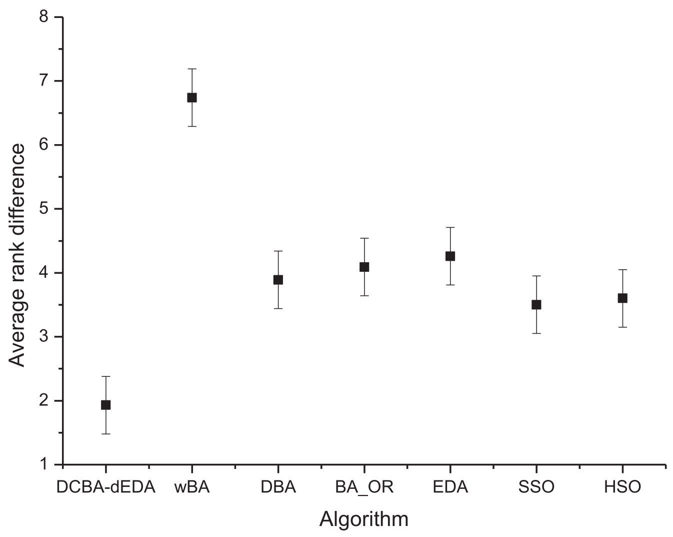


Fig. 9. The graphical presentation of Nemenyi test.

Table 13
Wilcoxon test at different cases.

No.	DCBA-dEDA vs.					
	wBA	DBA	BA_OR	EDA	SSO	HSO
1	**4.68E-04**	**1.56E-02**	2.14E-01	**9.63E-03**	**1.96E-04**	**2.58E-04**
2	**2.66E-04**	**3.55E-02**	1.92E-01	**2.44E-03**	1.96E-01	2.16E-01
3	**1.69E-04**	1.81E-01	**7.75E-03**	**2.30E-02**	**1.67E-02**	6.79E-02
4	**1.73E-04**	**1.18E-02**	**3.54E-02**	**1.09E-02**	6.69E-02	**9.77E-03**
5	**1.68E-04**	**4.21E-03**	**4.05E-03**	**4.59E-04**	**2.05E-03**	**2.05E-03**
6	**1.80E-04**	**6.27E-02**	**5.59E-03**	**2.31E-02**	4.04E-01	8.84E-02
7	**1.63E-04**	**5.38E-03**	**8.44E-03**	**9.22E-04**	6.56E-02	7.54E-02
8	**1.83E-04**	**5.14E-03**	**1.82E-04**	**2.83E-02**	1.62E-01	1.99E-01
9	**1.83E-04**	**1.83E-04**	**1.83E-04**	**2.45E-04**	**3.76E-02**	8.20E-02
10	**1.83E-04**	**1.70E-03**	**1.83E-04**	1.00E+00	2.41E-01	3.45E-01

graphical presentation of the results is depicted in Fig. 9, where the points are the mean rank computed by Friedman test and the lines represent the confidence interval *CD*.

From the mean rank shown in Table 12, DCBA-dEDA has the best mean rank among all algorithms while the wBA obtains the worst rank. As is obvious depicted in Fig. 9, the average of the proposed algorithm DCBA-dEDA is lower than any other algorithms. The lower the mean rank, the better the algorithm. Besides, the interval of DCBA-dEDA do not overlap with any other algorithms which indicates the proposed algorithm are significantly different from other algorithms.

The algorithm with the lowest rank from the Friedman test is chosen as the control method in the Wilcoxon test. Table 13 presents the results of Wilcoxon test (a method of pairwise comparison) between DCBA-dEDA and other compared algorithms. The result in bold means that the DCBA-dEDA significantly outperforms the specific algorithm. The p-values for wBA, DBA, BA_OR, and EDA are less than the significant level 0.05 at most cases. It means the performance of DCBA-dEDA is significantly better than these algorithms when dealing with a majority of GRAPs.

6. Conclusions

In this study, a generalized redundancy allocation problem is solved by CA-based MCS and DCBA-dEDA. The reliability is evaluated without the task of knowing the MPs/MCs. Compared with the existing studies of reliability calculation in GRAPs, it is more effective when considering relatively large networks. Furthermore, a hybrid discrete bat algorithm is presented in this paper. Several modifications are embedded to the origin BA to enhance the exploitation and exploration capabilities and as a result, these improvement structures are proved to be effective and the performance of BA is significantly enhanced. Extensive experiments have been performed to set the optimal parameters and measure the efficiency of the proposed algorithm. Experimental results indicate that V4 is the optimal transfer function for discrete BAs. It is worth noting that the value of loudness, which controls the acceptance or rejection of a new solution, has a great influence on the results of BAs. This paper uses a nonlinear decreasing strategy to update loudness. Besides, a comprehensive statistical analysis including Friedman test, Nemenyi post-hoc test, and Wilcoxon post-hoc test has been implemented in this paper. Results exhibit that the proposed DCBA-dEDA achieves significantly better performance compared with other state-of-the-art algorithms.

The future work can be summarized as follows. Multi-objective GRAPs and multi-state systems would have both practical and theoretical benefits.

Acknowledgement

This work was partially supported by National Natural Science Foundation of China (U1433116), the Fundamental Research Funds for the Central Universities (NP2017208), and the Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX19_0202).

References

[1] W. Kuo, V.R. Prasad, An annotated overview of system-reliability optimization, IEEE Trans. Reliab. 49 (2000) 176–187.
[2] T.J. Hsieh, W.C. Yeh, Penalty guided bees search for redundancy allocation problems with a mix of components in series–parallel systems, Comput. Oper. Res. 39 (2012) 2688–2704.
[3] M.-S. Chern, On the computational complexity of reliability redundancy allocation in a series system, Oper. Res. Lett. 11 (1992) 309–315.
[4] Y.C. Liang, A.E. Smith, An ant colony optimization algorithm for the redundancy allocation problem (RAP), IEEE Trans. Reliab. 53 (Sep 2004) 417–423.
[5] C.-M. Lai, W.-C. Yeh, Two-stage simplified swarm optimization for the redundancy allocation problem in a multi-state bridge system, Reliab. Eng. Syst. Saf. 156 (Dec 2016) 148–158.
[6] O. Chryssaphinou, N. Limnios, S. Malefaki, Multi-state reliability systems under discrete time semi-Markovian hypothesis, IEEE Trans. Reliab. 60 (Mar 2011) 80–87.
[7] J.E. Ramirez-Marquez, D.W. Coit, A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems, Reliab. Eng. Syst. Saf. 83 (2004) 341–349.
[8] Z. Wang, K. Tang, X. Yao, A memetic algorithm for MultiLevel redundancy allocation, IEEE Trans. Reliab. 59 (2010) 754–765.
[9] Amirhossain Chambari, A.A. Najafi, H.A. Rahmati, Karimi Seyed, et al., An efficient simulated annealing algorithm for the redundancy allocation problem with a choice of redundancy strategies, Reliab. Eng. Syst. Saf. 119 (2013) 158–164.
[10] K.-H. Chang, P.-Y. Kuo, An efficient simulation optimization method for the generalized redundancy allocation problem, Eur. J. Oper. Res. 265 (2018) 1094–1101.
[11] W.C. Yeh, A new exact solution algorithm for a novel generalized redundancy allocation problem, Inf. Sci. 408 (2017).
[12] T. Aven, Availability evaluation of oil/gas production and transportation systems, Reliab. Eng. 18 (1987) 35–44.
[13] W.C. Yeh, S.C. Wei, Economic-based resource allocation for reliable Grid-computing service based on grid bank, Future Gener. Comput. Syst. 28 (2012) 989–1002.
[14] C.J. Colbourn, The Combinatorics of Network Reliability, Oxford University Press, Inc., 1987.
[15] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.
[16] W.C. Yeh, A MCS-RSM approach for network reliability to minimise the total cost, Int. J. Adv. Manuf. Technol. 22 (2003) 681–688.
[17] W.C. Yeh, Y.C. Lin, Y.Y. Chung, M. Chih, A particle swarm optimization approach based on Monte Carlo simulation for solving the complex network reliability problem, IEEE Trans. Reliab. 59 (2010) 212–221.
[18] H. Wang, H. Pham, Survey of reliability and availability evaluation of complex networks using Monte Carlo techniques, Microelectron. Reliab. 37 (1997) 187–209.
[19] W.C. Yeh, Y.C. Lin, Y.Y. Chung, Performance analysis of cellular automata Monte Carlo Simulation for estimating network reliability, Expert Syst. Appl. 37 (2010) 3537–3544.
[20] E. Zio, L. Podofillini, V. Zille, A combination of Monte Carlo simulation and cellular automata for computing the availability of complex network systems, Reliab. Eng. Syst. Saf. 91 (2006) 181–190.
[21] J.E. Ramirez-Marquez, D.W. Coit, A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability, Reliab. Eng. Syst. Saf. 87 (2005) 253–264.
[22] C.M. Rocco S, Network reliability assessment using a cellular automata approach, Reliab. Eng. Syst. Saf. 78 (2002) 289–295.
[23] M.R.S. Claudio, E. Zio, Solving advanced network reliability problems by means of cellular automata and Monte Carlo sampling, Reliab. Eng. Syst. Saf. 89 (2005) 219–226.
[24] X.S. Yang, A new metaheuristic bat-inspired algorithm, Comput. Knowl. Technol. 284 (2010) 65–74.
[25] X.S. Yang, X. He, Bat algorithm: literature review and applications, Int. J. Bio-Inspired Comput. 5 (2013) 141–149.
[26] S. Yılmaz, E.U. Küçüksille, A new modification approach on bat algorithm for solving optimization problems, Appl. Soft Comput. 28 (2015) 259–275.
[27] A. Chakri, R. Khelif, M. Benouaret, X.S. Yang, New directional bat algorithm for continuous optimization problems, Expert Syst. Appl. 69 (2017) 159–175.
[28] D. Rodrigues, L.A.M. Pereira, R.Y.M. Nakamura, K.A.P. Costa, X.S. Yang, A.N. Souza, et al., A wrapper approach for feature selection based on bat algorithm and optimum-path forest, Expert Syst. Appl. 41 (2014) 2250–2258.
[29] P. Suarez, A. Iglesias, A. Galvez, Make robots be bats: specializing robotic swarms to the Bat algorithm, Swarm Evol. Comput. 44 (Feb 2019) 113–129.
[30] C.M. Lai, W.C. Yeh, Two-stage simplified swarm optimization for the redundancy allocation problem in a multi-state bridge system, Reliab. Eng. Syst. Saf. 156 (2016) 148–158.
[31] Y. Wang, L. Li, A PSO algorithm for constrained redundancy allocation in multi-state systems with bridge topology, Comput. Ind. Eng. 68 (2014) 13–22.
[32] P. Kubat, Estimation of reliability for communication/computer networks - simulation/analytic approach, IEEE Trans. Commun. 37 (1989) 927–933.
[33] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm Evol. Comput. 1 (2011) 111–128.
[34] J. Liu, K.L. Teo, X. Wang, C. Wu, An exact penalty function-based differential search algorithm for constrained global optimization, Soft Comput. 20 (2016) 1305–1313.

[35] G. Levitin, A. Lisnianski, Structure optimization of power system with bridge topology, Electr. Power Syst. Res. 43 (1998) 19–27.
[36] X.B. Meng, X.Z. Gao, Y. Liu, H. Zhang, A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization, Expert Syst. Appl. 42 (2015) 6350–6364.
[37] X. Cai, W. Hui, Z. Cui, J. Cai, X. Yu, W. Lei, Bat algorithm with triangle-flipping strategy for numerical optimization, Int. J. Mach. Learn. Cybern. 9 (2017) 1–17.
[38] X. Cai, X.Z. Gao, Y. Xue, Improved bat algorithm with optimal forage strategy and random disturbance strategy, Int. J. Bio-Inspired Comput. 8 (2016) 205–214.
[39] Z. Cui, C. Yang, X. Cai, et al., Optimal LEACH protocol with modified bat algorithm for big data sensing systems in Internet of Things, J. Parallel Distrib. Comput. 132 (2019) 217–229.
[40] K.J. Clerc, The particle swarm - explosion, stability, and convergence in a multidimensional complex space[, IEEE Trans. Evol. Comput. 6 (2002) 58–73.
[41] S. Saremi, S. Mirjalili, A. Lewis, How important is a transfer function in discrete heuristic algorithms, Neural Comput. Appl. 26 (2015) 625–640.
[42] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, BGSA: binary gravitational search algorithm, Nat. Comput. 9 (2010) 727–745.
[43] S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer functions for binary particle swarm optimization, Swarm Evol. Comput. 9 (2013) 1–14.
[44] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, Computational Cybernetics and Simulation, 1997, pp. 4104–4108, 1997.
[45] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications, Nat. Comput. 7 (2008) 109–124.
[46] X.S. He, W.J. Ding, X.S. Yang, Bat algorithm based on simulated annealing and Gaussian perturbations, Neural Comput. Appl. 25 (2014) 459–468.
[47] G.L. Wang G, A novel hybrid bat algorithm with harmony search for global numerical optimization, J. Appl. Math. (2013) 233–256, 2013.
[48] J.J. Liang, B.-Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University, Singapore, December 2013.
[49] B. Zhang, Q.K. Pan, L. Gao, X.L. Zhang, H.Y. Sang, J.Q. Li, An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming, Appl. Soft Comput. 52 (2017) 14–27.
[50] A.E. Eiben, J.E. Smith, Introduction to evolutionary computing, Evol. Comput. 12 (2014) 269–271.
[51] F.I. Jr, S. Fong, J. Brest, I. Fister, A novel hybrid self-adaptive bat algorithm, Sci. World J. 2014 (2014) 709738.
[52] C.A. Voglis, K.E. Parsopoulos, I.E. Lagaris, Particle swarm optimization with deliberate loss of information, Soft Comput. 16 (2012) 1373–1392.
[53] M. Marinaki, Y. Marinakis, G.E. Stavroulakis, Vibration control of beams with piezoelectric sensors and actuators using particle swarm optimization, Expert Syst. Appl. 38 (2011) 6872–6883.
[54] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Congress on Evolutionary Computation, 2002.
[55] W.C. Yeh, Orthogonal simplified swarm optimization for the series–parallel redundancy allocation problem with a mix of components, Knowl. Based Syst. 64 (2014) 1–12.
[56] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1940) 86–92.
[57] U. Mlakar, I.F. Jr, I. Fister, Hybrid self-adaptive cuckoo search for global optimization, Swarm Evol. Comput. 29 (2016) 47–72.
[58] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks? J. Mach. Learn. Res. 17 (10) (2016).
[59] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.