A novel discrete bat algorithm for heterogeneous redundancy allocation of multi-state systems subject to probabilistic common-cause failure

Yue Xu[a,*], Dechang Pi[a], Shengxiang Yang[b], Yang Chen[a]

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

[b] School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK

E-mail: ayue@nuaa.edu.cn

Abstract: This paper focuses on a heterogeneous redundancy allocation problem (RAP) for multi-state series-parallel systems subject to probabilistic common-cause failure and proposes a novel discrete bat algorithm to solve it. Although abundant research studies have been published for solving multi-state RAPs, few of them have studied probabilistic common cause failure, which motivates this paper. Due to the insufficient data of components, an interval-valued universal generating function is utilized to evaluate the availability of components and the whole system. The challenge of solving this kind of RAPs lies in not only the reliability estimation, but also the solution method. This paper presents a novel discrete bat algorithm (BA) for effectively dealing with the proposed RAP and alleviating the premature convergence of BA. Two main features of the adaptation are Hamming distance-based bat movement (HDBM) and Q learning-based local search (QLLS). HDBM transfers the Hamming distance between the current bat and the best bat in the swarm to the movement rate. Then, QLLS utilizes Q-learning to adjust the local search strategies dynamically during the iterations. The computational results from extensive experiments demonstrate that the proposed algorithm is powerful, which is more efficient than other state-of-the-arts on this sort of problems.

Keywords: heterogeneous redundancy allocation problem; multi-state system; probabilistic common-cause failure; bat algorithm; Hamming distance; Q learning; local search

1. Introduction

The redundancy allocation problem (RAP) as a combinatorial optimization problem has captured wide attention in the field of reliability engineering for different systems and under various assumptions [1]. According to the state of components and subsystems, RAPs can be further classified into two categories. The former is a binary-state RAP which only considers the perfectly functioning state and the completely failed state, whereas the latter, i.e., multi-state RAP, experiences more than two extreme states. It is obvious that the multi-state system (MSS) is more in accordance with the real situation than the binary-state system (BSS). Many systems can function even under states which are not functioning perfectly in many real world applications, i.e., communication networks, production, manufacturing, power generation, transportation of oil and gas, and so on [2, 3].

Typically, the RAP is formulated for the optimal structure of multi-state series-parallel systems (MSPSs) with the goal of minimizing the whole cost of the system whereas maintaining its availability above a predetermined threshold [4]. The series-parallel system has been considered mainly because of its universality in the real world. Due to the limitation of computational complexity, many early efforts have been devoted into the solution of binary-state RAPs [5-7]. However, the research of multi-state RAPs and their solution methods have become a trend in recent years since they are more practical. For example, Du and Li investigated designed inter-subsystem local search strategies in memetic algorithm for solving multi-state RAPs [4]. Zaretalab considered this sort of problems with reliable supplier selection [8]. Sun et al. took epistemic uncertainty into account [9]. Moreover, the solved problem is two-stage and multi-objective. Wang and Li combined particle swarm optimization and local search to solve it [10].

Although abundant research works have been presented for solving multi-state RAPs, very few of them have focused on common cause failure (CCF). Levitin firstly incorporated CCF into nonrepairable multi-state system analysis [11] by an implicit 2-stage approach. To be specific, the reliability function can be obtained numerically via this approach. CCFs are failures of multiple components due to a shared root cause or a common cause (CC) [12]. CCFs can be caused by external shocks or internal failures. Generally, it can be divided into two groups: deterministic CCF (DCCF) and probabilistic CCF

(PCCF). The main difference between them lies in the effect from a CCF on its common cause group (CCG). The former DCCF results in guaranteed failures of all components within the CCG while the latter PCCF results in failures of different components with different occurrence probability within the CCG [13]. A multi-state RAP considering DCCF was firstly introduced by Li et al. in [14] where the universal generating function (UGF) was used to evaluate the system availability and a genetic algorithm was utilized for optimizing the system structure.

On the other hand, the RAP for MSSs subject to PCCF has not been studied to the best of our knowledge. Compared with DCCF, PCCF is more practical and challenging. For calculating the availability in BSSs considering PCCF, there are several well-known methods such as the binomial failure rate model [15], explicit method [12, 16], and implicit method [12]. In the field of MSSs subject to PCCF, Huang et al. firstly proposed a reliability calculation method with two characteristics: fatal and non-fatal [17]. In most cases, however, the component data are insufficient. Invoked by [17] and to alleviate this problem, a new availability evaluation method is presented in this paper for MSSs considering PCCF.

Moreover, due to the huge and complex searching space, evolutionary algorithms (EAs) have captured much attention for solving this kind of problems. Bat algorithm (BA) which is a simple but efficient EA has been widely applied for solving many real-world optimization problems. It was firstly proposed by Yang inspired by the echolocation behavior of micro-bats [18]. In recent years, several published papers have verified the effectiveness of BAs for solving RAPs. Talafuse and Pohl firstly used a discrete BA to tackle the RAP[19]. Whereafter, Xu and Pi presented a discrete hybrid BA for the generalized RAP [20] where subsystems are connected with each other neither in series nor in parallel, but in some logical relationships [21]. Given by the wide application of BAs in RAPs, this paper proposes a novel version of BA for dealing with the new proposed RAP. According to decision values in RAPs, discretization is the most important concern among all improvements of the version. As reported in [22], premature convergence may occur under certain conditions. To solve this problem, the bat movement and local search should be modified for a better balance between exploration and exploitation.

To further advance the state-of-the-art of solving RAPs, this paper presents a new RAP and a novel BA version to solve it. The availability of MSSs subject to PCCF is evaluated. Considering the unknown or interval-valued state performance level, this paper makes use of the interval-valued UGF (IUGF) for the formulation of each component. In the solution method, Hamming distance-based bat movement and Q learning-based local search are proposed with the primary aim of discretization. To be specific, the first modification updates the movement equation with less parameters. The second one adjusts the local search strategies self-adaptively during each iteration. The proposed algorithm is tested on three typical systems under different predetermined thresholds with or without PCCF and compared with several well-known algorithms. Extensive experimental results indicate the new version outperforms the others on the created scenarios.

Besides, this paper controls the discrete local search strategies in BA with Q learning (QL). In recent years, the research on incorporating QL into EAs to control the operation has become a hot spot gradually. Rakshit et al. introduced QL into a memetic algorithm for selecting scaling factors of any variants of DE [23]. Samma et al. embedded QL into particle swarm optimization algorithm to control five possible operations [24]. Subsequently, they proposed a novel simulated annealing algorithm with QL with the aim of choosing its parameters adaptively [25]. A QL-based particle swarm optimization algorithm was developed to adjust the neighborhood structures in [26]. The experimental results of the above literatures all verified the effectiveness of embedding QL into EAs to control executions. It should be highlighted each individual has its own Q-table in the above literatures. Maintaining all Q-tables is time-consuming and takes up much memory. To overcome this limitation, a shared Q-table is used for the swarm which makes the algorithm simpler and more convenient.

The rest of the paper are arranged as follows. The related works are detailed in Section 2. Subsequently, the formulation of the RAP in MSPSs considering PCCF is shown in Section 3. In the following section, a novel discrete bat algorithm is proposed to solve the problem with Hamming distance-based bat movement and Q learning based-local search. Experimental results and analysis are supplied in Section 5. This paper is ended with conclusion and forecast in Section 6.

2. Related works

This section introduces the general formulation of RAP and how to calculate the system availability via IUGF. BA is as a solution method due to its successful application in RAPs. Since this paper makes use of QL to adjust the local search strategies, its description is also detailed in this section.

2.1 Formulation of the RAP

The structure of a typical multi-state series-parallel system with heterogeneous redundancy considering probabilistic common cause is shown in Fig. 1. For the system consists of Ns subsystems, assume there are V_i component levels available on the market for subsystem i, where $i = 1, 2, \ldots, Ns$. Generally, two different cases of the series parallel allocation problem can be considered [27]. In the first case, the components are binary-state but different kinds of components have different performance levels and probabilities. Thus, the system will have a range of different performance states and corresponding state probabilities depending on those of consisted components. In the second one, the components and the system are all multi-state, which makes the problem more challenging. In this study, the first case is considered for simplicity. The component of version j in subsystem i has two possible states: perfect function and complete failure, where $j = 1, 2, \ldots, V_i$. Each component is characterized by cost C_{ij}, performance level $\{0, g_{ij}\}$ and its corresponding state probability $\{1- r_{ij}, r_{ij}\}$.

For subsystem i, its structure is denoted by the number of components of each version, that is, $X_i = [X_{i1}, X_{i2}, \ldots, X_{iVi}]$; and the structure of the entire system can be represented as $X = [X_1, X_2, \ldots, X_{NS}]$. The cost of the entire system can be denoted as:

$$Cs = \sum_{i=1}^{Ns} \sum_{j=1}^{V_i} X_{ij} C_{ij} \tag{1}$$

In this study, the heterogeneous RAP for MSPSs with PCCF, RAP*MSPS*PCCF for short, is considered. Formulation of this problem which aims at minimizing the system cost subject to several constraints can be stated as the following pure integer nonlinear programming problem (INLP).

$$\begin{aligned} &Min \quad Cs \\ &s.t. \quad As \geq A_0 \end{aligned} \tag{2}$$

where As is the total availability of the system and A_0 represents the required availability of the whole system.

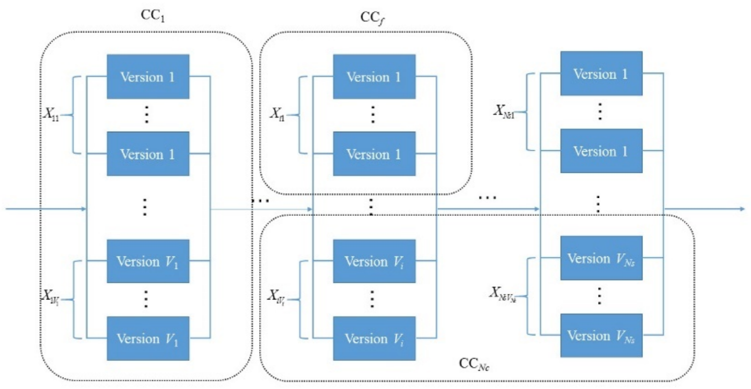

Fig. 1. Structure of a typical MSPS with heterogeneous redundancy considering PCCF

2.2 Calculating system availability by IUGF

Due to the insufficient data, the IUGF method is implemented in this paper to calculate the system availability. It was originally proposed by Li et al. for the cases where either the membership function of fuzzy values is unknown or the interval-valued state performance level and its corresponding probability are known [28, 29].

This method is briefly introduced as follows. Consider a variable Y that takes on a finite number S of possible values. Assume the interval-valued state performance level and its corresponding probability be $[y]$ and $[p]$ respectively, where $[y] = ([y_1], [y_2], \ldots, [y_S])$, $[p] = ([p_1], [p_2], \ldots, [p_S])$. Then, the u-function can be defined as:

$$U(z) = \sum_{s=1}^{S} [p_s] z^{[y_s]} \tag{3}$$

Similar with the operator of UGF, a general operator of IUGF can be expressed as:

$$\Omega\left(U_1(z), U_2(z)\right) = \sum_{s_1=1}^{S_1} \sum_{s_2=1}^{S_2} [p_{s_1}] \cdot [p_{s_1}] z^{f\left([y_{s_1}],[y_{s_2}]\right)} \tag{4}$$

Assume the $\underline{x}$ and $\overline{x}$ are lower and upper bound of x, respectively. More details about operators in interval form can be referred to [30]. When the system performance level is equal to the sum of that of components, the π operator is denoted as:

$$\pi\left(U_1(z), U_2(z)\right) = \sum_{s_1=1}^{S_1} \sum_{s_2=1}^{S_2} \left[\underline{p_{s_1}} \cdot \underline{p_{s_2}}, \overline{p_{s_1}} \cdot \overline{p_{s_2}}\right] z^{\left[\underline{y_{s_1}} + \underline{y_{s_2}}, \overline{y_{s_1}} + \overline{y_{s_2}}\right]} \tag{5}$$

When the system performance level is equal to the minimum of that of components, the σ operator is denoted as:

$$\sigma\left(U_1(z),U_2(z)\right)=\sum_{s_1=1}^{S_1}\sum_{s_2=1}^{S_2}\left[\underline{p_{s_1}}\cdot\underline{p_{s_2}},\overline{p_{s_1}}\cdot\overline{p_{s_2}}\right]z^{\left[\min\left(\underline{y_{s_1}},\underline{y_{s_2}}\right),\min\left(\overline{y_{s_1}},\overline{y_{s_2}}\right)\right]} \tag{6}$$

For the MSPS, the total capacity is the sum of the capacities of components in the parallel structure, while in the series structure, the performance is the minimum of that. Assume the specific demand level $w=\left[\underline{w},\overline{w}\right]$, the interval-valued availability of the system is given by:

$$[A]=\sum_{s=1}^{S}\frac{[p_s]\cdot\max\left(\overline{y_s}-\underline{w},0\right)}{\max\left(\overline{y_s}-\underline{y_s}+\overline{w}-\underline{w},\overline{y_s}-\underline{w}\right)} \tag{7}$$

2.3 Bat algorithm

Bat algorithm was developed to use the key idea of frequency tuning based on the echolocation of microbats [18]. In the standard bat algorithm, the echolocation characteristics of microbats can be idealized as the following three rules:

(1) All bats use echolocation to sense distance, and they also ‘know’ the difference between food/prey and background barriers in some magical way.

(2) Bat i flies randomly with velocity V_i at position X_i 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.

(3) 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 i), the new position $X_i(t)$, velocity $V_i(t)$, frequency $f_i(t)$ at iteration t can be updated as follows.

$$f_i = f_{\min}+(f_{\max}-f_{\min})\cdot rand \tag{8}$$

$$V_i(t+\mathbf{1})=V_i(t)+\left[X_i(t)-X_*\right]\cdot f_i \tag{9}$$

$$X_i(t+\mathbf{1})=X_i(t)+V_i(t+\mathbf{1}) \tag{10}$$

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

$$X_{new} = X_{old} + \varepsilon \cdot L(t) \tag{11}$$

where X_{old} is a high quality solution, $\varepsilon \in [-1,1]$ is a scaling factor which is a random number, while $L(t) = < L_i(t)>$ is 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 $rp_i(t)$ and loudness $L_i(t)$ are updated as follows.

$$L_i(t+\mathbf{1}) = \alpha \cdot L_i(t) \tag{12}$$

$$rp_i(t+\mathbf{1}) = rp_i(\mathbf{0}) \cdot \left[\mathbf{1} - \exp(-\gamma \cdot t)\right] \tag{13}$$

where α and γ are constants, $0<\alpha<1$, $\gamma>0$. In fact, α is similar to the cooling factor of a cooling schedule in the simulated annealing. Eventually, L_i will equal zero, while the final value of rp_i is $rp_i(0)$. The pseudo-code of the BA is given in Algorithm 1:

Algorithm 1: BA

```
1.    Objective function $F(X)$, $X=(X_1, \ldots, X_D)^T$
2.    Initialize the bat population $X_i$ and $V_i$ ($i$ = 1, 2, …, $N$), and related parameters
3.    Evaluate fitness of the bat population
4.    while ($t$ < Max number of iterations)
5.        Update frequency, velocity, and position using (8)-(10), respectively
6.        if $rand > rp_i$
7.            Select a solution among the best solutions
8.            Generate a local solution around the selected best solution using (11)
9.        end if
10.       Generate a new solution by flying randomly
11.       if ($rand < L_i$ & $F(X_i(t+1)) < F(X_*)$)
12.           Accept the new solution
13.           Reduce loudness and pulse emission using (12) and (13), respectively
14.       end if
15.   Rank the bats and find the current best $X_*$
16.   end while
17.   Postprocess results and visualization
```

2.4 Q learning

Q learning is a model-free reinforcement learning algorithm for agents to learn how to act optimally in controlled Markovian domains [31]. In QL, agent usually performs an action through state transition in the environment and receives a reward or penalty by executing the action to reach the goal state [32], as illustrated in Fig. 2.

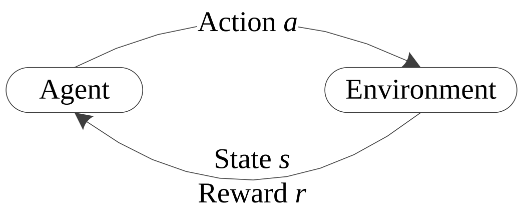

Fig. 2. Reinforcement learning

As detailed in Fig. 2, QL includes five basic components, i.e., agent, environment, action, state, and reward. Considering that *S* is a set of possible states of the learning agent in a given environment, and *A* is a set of possible actions that agent can choose and execute, these components of agent *i* at time *t* can be represented as (14).

$$s_t^i \in S, a_t^i \in A, r_t^i \in S \times A \to \mathbb{R}. \tag{14}$$

Hence, the total cumulative reward, referred to as Q-value, can be updated as (15).

$$Q_{t+1}(s_t, a_t) = (1-lr)Q(s_t, a_t) + lr\left[r_{t+1} + df \max_{lr} Q(s_{t+1}, a_t) \right], \tag{15}$$

where *lr* and *df* are learning rate and discount factor, respectively, and both within [0,1]. As can be seen from (15), the learning rate is a balance between exploration and exploitation. That is, a low value of the learning rate makes the algorithm learn more about the existing information, while the larger the learning rate *lr*, the less the retention of the previous Q-value. Hence, *lr* normally is set to a high value at the beginning of the search process, and is decreased during the iterations [23, 24].

$$lr(t) = 1 - \left(0.9 \times \frac{t}{iter_{max}} \right), \tag{16}$$

where $iter_{max}$ is the maximum number of iterations. Besides, the discount factor is a tradeoff between previous experience and immediate reward. The greater the value of *df*, the algorithm attaches more importance to previous information; the smaller, the algorithm is more concerned about the reward. Usually, it is set to 0.8, as also suggested in [23, 24]. According to the above definitions, the pseudo code of QL is described in Algorithm 2.

Algorithm 2. The pseudo code of QL

```
1.  Initialize Q($s$, $a$) = $zero$ in Q-table;
2.  Randomly select an initial state $s$;
3.  Repeat
4.    Select the best action $a_t$ for $s_t$ from the Q-table;
5.    Execute action $a_t$ and get a new solution;
6.    Get the maximum Q value for the next state $s_{t+1}$;
7.    Obtain the immediate reward $r_t$;
8.    Update Q-table entry using (15);
9.    $s_t = s_{t+1}$;
10. Until the $Max\_FEs$ is met.
```

3 Problem formulation

3.1 IUGF of the component considering PCCF

In previous studies, independent failure or common cause failure was mostly assumed for the reliability analysis of multi-state systems. Huang et al. firstly proposed a multi-state availability calculation method considering PCCFs with two characteristics: fatal and non-fatal [17]. If a component failure event is caused by a fatal PCCF, then the component completely fails with an occurrence probability; if it is caused by a non-fatal PCCF, then the component partially fails with an occurrence probability. In their assumption, the performance level of the component caused by a non-fatal PCCF is a precise value. In most cases, however, the component data are insufficient. To deal with this issue, an interval value is utilized to represent the performance level caused by a non-fatal PCCF and a novel availability calculation method is detailed in this paper.

In the PCCF analysis [33], suppose a system is subjected to PCCF from Nc external and s-independent CCs. The occurrence probability of CC_c is denoted as p_c, where c = 1, 2, …, Nc. Each CC_c causes a component affected by it, for example, component A fails with probability q_{cA}, where $q_{cA} = Pr$(Component A fails | CC_c occurs).

The IUGF of each component is presented as:

$$\begin{aligned} U_{ij}(z) &= (1-p_c)\cdot r_{ij} z^{g_{ij}} + p_c\cdot(1-q_{cij})\cdot r_{ij} z^{g_{ij}} + p_c\cdot q_{cij}\cdot r_{ij} z^{[0,g_{ij}]} + (1-r_{ij}) z^0 \\ &= (1-p_c\cdot q_{cij}) r_{ij} z^{g_{ij}} + p_c\cdot q_{cij}\cdot r_{ij} z^{[0,g_{ij}]} + (1-r_{ij}) z^0 \end{aligned} \tag{17}$$

In (17), the first term is the case that the CC does not occur or the component is not affected even though the CC occurs. In this case, the component runs perfectly. The second term means a non-fatal PCCF occurs and the component partially fails with the performance level x. Two boundary situations are considered: If $x = 0$, the component completely fails caused by a fatal PCCF; if $x = g_{ij}$, the component fails independently without considering CCF. This study utilizes an interval-valued performance level $[0, g_{ij}]$ because the failure part is unknown. In the third one, the component fails locally and the performance level is zero.

3.2 An illustrative example

Fig. 3 illustrates a series-parallel system consisting of two subsystems subjected to two external and s-independent CCs: CC_1 and CC_2. The occurrence of CC_1 affects component A and B, the occurrence of CC_2 affects component C and E. Thus, component A and B form a probabilistic common-cause group $PCCG_1$, component C and E form another group $PCCG_2$, component D fails independently.

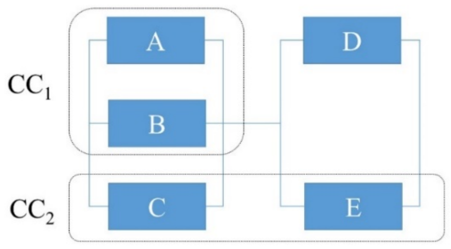

Fig. 3. Structure of a series-parallel system subjected to CC_1 and CC_2

The following parameter values are recommended in the subsequent analysis, which were also used in [33].

- The component is binary-state: perfect function with performance level 100% and complete failure with performance level 0%. For simplicity, all component local failure probability is set as 0.1, i.e., $r_A = r_B = r_C = r_D = r_E = 0.9$.
- The occurrence probability of common causes is set as: p_1=0.01, p_2=0.02.
- Given that the common cause occurs, the conditional probability that each component fails is shown as: $q_{1A} = 0.3$, $q_{1B} = 0.6$, $q_{2C} = 0.6$, $q_{2E} = 1$.

According to (17), the IUGF of each component can be calculated as:

$$\begin{aligned}
&U_{\mathrm{A}}(z)=0.8973z^{1}+0.0027z^{[0,1]}+0.1000z^{0};\\
&U_{\mathrm{B}}(z)=0.8946z^{1}+0.0054z^{[0,1]}+0.1000z^{0};\\
&U_{\mathrm{C}}(z)=0.8892z^{1}+0.0108z^{[0,1]}+0.1000z^{0};\\
&U_{\mathrm{D}}(z)=0.9000z^{1}+0.1000z^{0};\\
&U_{\mathrm{E}}(z)=0.8820z^{1}+0.0180z^{[0,1]}+0.1000z^{0};
\end{aligned}\tag{18}$$

Then, the IUGF of each subsystem using π operator is shown as:

$$\begin{aligned}
U_1(z)=\pi\left(U_{\mathrm{A}},U_B,U_C\right)&=0.7138z^{3}+0.0151z^{[2,3]}+0.0001z^{[1,3]}+0.2396z^{2}\\
&\quad+0.0034z^{[1,2]}+0.0268z^{1}+0.0002z^{[0,1]}+0.0010z^{0};\\
U_2(z)=\pi\left(U_{\mathrm{D}},U_E\right)&=0.7938z^{2}+0.0162z^{[1,2]}+0.1782z^{1}+0.0018z^{[0,1]}+0.0100z^{0};
\end{aligned}\tag{19}$$

Subsequently, the IUGF of the entire MSPS using σ operator is computed as:

$$U_S(z)=\sigma\left(U_1,U_2\right)=0.7688z^{2}+0.0185z^{[1,2]}+0.1997z^{1}+0.0020z^{[0,1]}+0.0110z^{0};\tag{20}$$

Depending on the availability calculation equation (7) in IUGF analysis, the system availability As = 0.9872 is equal to for a given demand level 90%.

4 Solution method

Before getting into the details of the solution method, it should be highlighted that the original BA was developed essentially for addressing continuous optimization problems. Thus, some modifications are needed to enable it for addressing a discrete problem as our proposed RAP*MSPS*PCCF.

4.1 Hamming distance-based bat movement (HDBM)

First of all, analyzing the movement functions (8)-(10), it can be deduced that the position of a bat i at iteration t relies on its position X_i in the previous iteration and velocity V_i in the current iteration. As can be easily found, the update of velocity cannot be directly used for solving the presented discrete problem. Besides that, the factor f_i is randomly generated within the specified range for continuous optimization problems. Hence, both (8) for frequency and (9) for velocity have not been taken into account in the proposed algorithm.

Instead, Hamming distance which is the number of different elements in the sequence [34] is utilized to describe the distance between bat *i* and the best bat X_* in the swarm according to (21).

$$HD = \text{HammingDistance}(X_i,\ X_*) \tag{21}$$

It is similar to the concept of the original BA but is a discrete number. Then, a transfer function [35] is introduced to map the Hamming distance into a probability value in [0, 1]. The results of our previous study [20] indicated that BA achieves the best performance via (22) compared with other 7 transfer functions.

$$T(HD) = \frac{2}{\pi}\arctan(\frac{2}{\pi}HD) \tag{22}$$

The greater the Hamming distance, the greater the transfer probability. In the early stage of the algorithm, if the current bat is far away from the best bat, it should be adjusted to the direction of X_* to advance the convergence ability of the algorithm. As the iterations progress, diversity is more important for the algorithm to explore new search spaces. Hence, the current bat should be adjusted to improve the diversity capability when it is close to X_*. In this paper, differential operator is the way to adjust the current bat at different stages of the algorithm in the following way.

```
if $rand > t/t_{\max}$                                                  // At the early stage of the algorithm
    $X_{ij}(t+1) = \begin{cases} X_{*j}(t) + (X_{r_1 j}(t) - X_{r_2 j}(t)) & rand \le T(HD) \\ X_{ij}(t) & \text{otherwise} \end{cases}$    // Improve the convergence rate
else                                                                    // At the later stage of the algorithm        (23)
    $X_{ij}(t+1) = \begin{cases} X_{ij}(t) + (X_{r_1 j}(t) - X_{r_2 j}(t)) & rand > T(HD) \\ X_{ij}(t) & \text{otherwise} \end{cases}$    // Maintain the population diversity
end
```

where *rand* is a random value from a uniform distribution, X_{r_1} and X_{r_2} are solutions randomly selected from the whole population, $r_1 \neq r_2$. The whole discrete procedure is illustrated in Fig. 4. In this example, the Hamming distance is 5 according to (21), and the transfer probability is calculated 0.92 according to (22). The differential operator is used to adjust the position of bat *i* with a great transfer probability. The aim of HDBM at this stage is to adjust the position when the bat is far away from the best bat in the swarm to accelerate the population convergence. After this procedure, the Hamming distance is improved to 4.

X_i	3	0	2	5	4	4	4
X_*	4	0	1	0	0	5	4
$X_{r_1} - X_{r_2}$	-1	+0	-1	-4	-1	+1	+3
rand	0.25	0.89	0.93	0.34	0.66	0.82	0.97

$t = t+1$

X_i	2	0	2	1	3	5	4

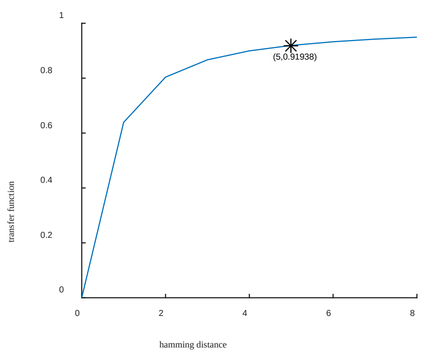

Fig. 4. The whole discrete procedure

4.2 Q learning based-local search (QLLS)

In this paper, LS is integrated with QL to select the optimal local search strategy self-adaptively according to the reward or penalty from the search region. Let the swarm act as an agent, and the environment be the search space. One thing to notice: there is only one Q-table and all bats share it. It is different from existing studies [26, 36-38] embedding QL into optimization algorithm, where each individual has its own Q-table independently. By contrast, our proposed method is simpler and more convenient with less memory and time consumption.

4.2.1 State

Before describing the state in QLLS, the following concepts are introduced. Evolution progress is divided into three states in this paper: evolution, struggle, and stagnation. The definitions are given below for minimization problems.

- Evolution: The best solution is improved, i.e., $F(X_*(t+1)) < F(X_*(t))$. The algorithm converges in this state.
- Struggle: The best solution is not improved within a certain number of iterations i.e., $F(X_*(t+1)) = F(X_*(t))$ for count iterations ($count \leq tr \times t_{max}$), where *count* is the number of iterations that the best solution has not been changed and *tr* represents the probability of the threshold in the whole iterations. In the struggle progress, the algorithm has more chance to improve the solution.
- Stagnation: The best solution is not improved after a certain number of iterations, i.e., $F(X_*(t+1)) = F(X_*(t))$ for count iterations ($count \geq tr \times t_{max}$). In the stagnation progress, the algorithm is trapped in the local optimum with a high probability.

Due to the ability of describing the progress of the current iteration, evolution, struggle, and stagnation are regarded as the states of the swarm.

4.2.2 Action

Three main issues should be concerned for the action: what is the action in QLLS, who is been performed action, and how to perform action.

The classic BA itself contains a local search mechanism to prevent the swarm from being trapped into local optima. However, the original LS mechanism in BA cannot be directly applied for the discrete optimization problems. For this reason, searching neighborhoods around the best solution and the current solution is performed with the aim of preparing it for facing the designed RAP*MSPS*PCCF. Reducing the search into a properly-designed neighborhood space efficiently explores the multimodal landscape with less computational consumption [4]. In [10], five LS neighborhood strategies were proposed by Wang and Li, including "+1" strategy, "-1" strategy, "-1, +1" strategy, "-1, +β" strategy, and "-all, +β" strategy. From a pre-experiment analysis, it can be known that different strategies and their combinations have different impacts on the performance of search for different problems. In other words, it is uncertain which strategy of combination is the best to be selected for different problems. To tackle this problem, performing five different LS strategies forms the set of action.

In the original LS mechanism, the high quality solution X_{old} in (11) is ambiguity in definition. For representing it, the literature [39] adopted the best solution X_* in the swarm while in [22], the bat i in the current iteration $X_i(t)$ was used to update the local movement. According to some preliminary experiments, we found that (1) local search around the best solution accelerates the population convergence but it is easy to get the algorithm into local optimization; (2) local search around the current bat enhances the population diversity and improves the global searching capability of the algorithm. Therefore, the best solution X_* and the current solution X_i are all objects of local search in this paper.

Besides, it should be noted that the proposed algorithm performs local search on the best solution or the current solution depending on different states. In the first two states (evolution and struggle), the algorithm should perform local search on the best solution to explore the global search region with the goal of accelerating the convergence and improving the global search ability. When the best solution gets into local optimum corresponding to the third state, local search around the best solution does not help the swarm out of dilemma. In this case (stagnation), local search around the current solution explores more unknown solutions, which can enhance the diversity and improve the capability of local search.

4.2.3 Reward/penalty and epsilon-greedy method

Reward or penalty relies on whether the best solution is improved, as (24).

$$r(t+1)=\begin{cases}+1 & \text{if } F(X_*(t+1))<F(X_*(t))\\ \text{-}1 & \text{otherwise}\end{cases} \tag{24}$$

Besides introducing the five basic components in QL, an epsilon-greedy method is also used to choose the action from the shared Q-table for the balance between exploitation and exploration.

$$\varepsilon\text{=0.7-0.05}t \tag{25}$$

$$a_t=\begin{cases}\max\limits_a Q(s_t,:) & \text{if } rand \le 1-\varepsilon\\ \text{select randomly from five LS strategies} & \text{otherwise}\end{cases} \tag{26}$$

Assume the current swarm is in the stagnation state, a simple example is presented to clarify QLLS, as follows.

Step 1. Obtain the state for the swarm: s_t = Stagnation.

Step 2. Select the optimal action for the current state from the shared Q-table such as a_t = +1.

Step 3. According to the current state, obtain the object of local search: $X_i(t)$.

Step 4. Perform the selected local search strategy around the object: adding one component into the current bat $X_i(t)$. This added component can be any of the versions available for the subsystem.

Step 5. Get the reward/penalty from the whole swarm, for example, r_t = +1.

Step 6. Obtain the next state for the swarm such as s_{t+1} = Evolution.

Step 7. Update Q-table entry Q(Stagnation, a_t).

Step 8. Update the current state: s_t = s_{t+1}.

4.3 Complete framework of the proposed algorithm

A novel discrete bat algorithm, NDBA for short, is proposed incorporating with Hamming distance-based bat movement

(HDBM) and Q learning-based local search (QLLS) in this paper. Completed procedure of NDBA is described in Algorithm 3. Discretization is the main concern for both the modification of bat movement and that of local search. In HDBM (line 7-9 in Algorithm 3), the transfer probability is based on the Hamming distance between the best solution and the current solution. With the high transfer probability, the differential operator is embedded directly into the movement function. In QLLS (line 11 to line 20 and line 25 to line 28), the optimal local search strategies is performed on different objects according to the state of the swarm. Specially, a duplicate checking strategy is used in line 14, where ismember() is a function whether the current bat exists in the previous generation.

Algorithm 3: NDBA

```
1.  Initialize the bat population and related parameters
2.  Initialize the shared Q-table as a zero matrix with three rows and five columns
3.  Define the fitness function $F$ according to the penalty function [40]
4.  Evaluate fitness of the bat population
5.  while ($t \leq$ Max number of iterations $t_{max}$)
6.      // HDBM
7.      Calculate the Hamming distance between bat $i$ and the best solution according to (21)
8.      Compute the transfer probability according to (22)
9.      Generate the solution according to (23)
10.     // QLLS
11.     Update the iteration number that the best solution has not been improved
12.     Obtain the state $s_t$ for the swarm
13.     Select the optimal action $a_t$ for $s_t$ from Q-table using the epsilon-greedy method (26)
14.     if ($rand > rp_i$ || ismember($X_i(t)$, $X(t$-1$)$))
15.         if $s_t$ == Stagnation
16.             Perform the selected local search strategy around the current bat $X_i$
17.         else
18.             Perform the selected local search strategy around the best solution $X_*$
19.         end
20.     end
```

```
21.         if ($rand < L_i$ & $F(X_i(t+1)) < F(X_*)$)
22.             Accept the new solution
23.             Reduce loudness and pulse emission according to (27) and (28), respectively
24.         end
25.         Get the reward/penalty from the whole swarm according to (24)
26.         Obtain the next state $s_{t+1}$ for the swarm
27.         Update Q-table entry according to (15)
28.         Update the current state $s_t = s_{t+1}$
29.         $t=t+1$
30.     end while
31.     Post-process results and visualization
```

As illustrated in [20], the computational complexity of the standard BA is $O(t_{max} \times N \times F)$, where t_{max} is the maximum number of iterations, N is the population size, and $O(F)$ is the computational complexity of its fitness evaluation. In this paper, the proposed algorithm does not increase extra loop operations, compared with the standard BA. Hence, both of them have the same computational complexity.

5 Experimental study

5.1 Description of benchmark problems

Three well-known benchmark problems are used to validate the effectiveness of our proposed NDBA method in solving RAPs for MSPSs considering PCCF. The information of the test problems found in previous publications is collected and presented in Table 1. For each subsystem, [40-42] describe several functionally equivalent components of various versions in detail. For all test cases, we randomly set the upper limit of each version MAX_{ij} = [5, 5, 3].

Table 1 Tested benchmark problems.

Pro.	Ns	V	A_0		
P1 [41]	4	[5, 4, 6, 5]	0.900	0.960	0.990
P2 [42]	5	[7, 5, 4, 9, 4]	0.975	0.980	0.990
P3 [43]	4	[11, 7, 9, 7]	0.940	0.960	0.990

For simplicity, assume that all systems are subjected to two external and s-independent CCs (CC_1 and CC_2) with the occurrence probability 0.01 and 0.02. More details about the test problems considering PCCF are recorded in Table 2, which are randomly generated. Taking P1 as an example, subsystems 1 and 2 form a PCCG, denoted $PCCG_1$, subsystems 3 and 4 form another group $PCCG_2$. Given that CC_1 occurs, the subsystem 1 fails with rate 0.6.

Table 2 Tested benchmark problems considering PCCF.

Pro.	PCCG$_c$		*Pr*(Component *i* fails | *CC$_c$* occurs)	
	1	2	*i*	q_{ci}
P1	{1, 2}	{3, 4}	1	0.6
			2	0.1
			3	0.3
			4	0.4
P2	{1, 4}	{2, 3, 5}	1	0.3
			2	0.8
			3	0.9
			4	0.1
			5	0.9
P3	{1, 3}	{2, 4}	1	0.9
			2	0.6
			3	0.2
			4	0.5

5.2 Calibration of parameters

To calibrate the parameters of NDBA, the Design of Experiments (DOE) methodology [44] is employed in this section. The proposed NDBA has the following key controlled factors: loudness *L*, pulse rate *pr*, and threshold rate *tr*. The loudness represents the acceptance or rejection of a new solution, the pulse rate decides the probability of local search, and the threshold rate controls the state of the best solution in the swarm. According to some preliminary experiments and related literatures, the potential values (levels) chosen for these parameters are listed in Table 3, which results in a total of $3\times3\times4=36$ configurations. Each parameter configuration is executed 5 times on all instances. In summary, there are a total of

36×9×5=1620 independent runs needed for tuning parameters of NDBA.

Table 3 The levels for tuning parameters.

Level	*L*	Level	*pr*	Level	*tr*
1	$L_i(t+\mathbf{1})=\alpha\cdot L_i(t)$ $L_i(\mathbf{0})=[\mathbf{1},\mathbf{2}],\alpha=\mathbf{0.9}$ [18, 39]	1	$rp_i(t+\mathbf{1})=rp_i(\mathbf{0})\times\left[\mathbf{1}\text{-}e^{-\gamma\times t}\right]$ $rp_i(\mathbf{0})=[\mathbf{0},\mathbf{1}],\gamma=\mathbf{0.9}$ [18]	1	0.05
2	$L_i(t+\mathbf{1})=\alpha\cdot L_i(t)$ $L_i(\mathbf{0})=\mathbf{0.95},\alpha=\mathbf{0.9}$ [45]	2	$rp_i(t+\mathbf{1})=rp_i(\mathbf{0})\times\left[\mathbf{1}\text{-}e^{-\gamma\times t}\right]$ $rp_i(\mathbf{0})=\mathbf{0.85},\gamma=\mathbf{0.9}$ [45]	2	0.10
3	$L(t)=\left(\frac{L_{\mathbf{0}}\text{-}L_\infty}{\mathbf{1}-t_{max}}\right)(t-t_{max})+L_\infty$ [22] $L_{\mathbf{0}}=\mathbf{0.9},L_\infty=\mathbf{0.6}$	3	$rp(t)=\left(\frac{rp_{\mathbf{0}}\text{-}rp_\infty}{\mathbf{1}-t_{max}}\right)(t-t_{max})+rp_\infty$ [22] $rp_{\mathbf{0}}=\mathbf{0.1},rp_\infty=\mathbf{0.7}$	3	0.15
				4	0.20

First of all, a statistical procedure Shapiro-Wilk's (SW) test [46] is performed for normality on the normalized results obtained by different parameter configurations. The result shows that the *p*-value is less than 0.05, so the null hypothesis is rejected: the sample does not follow a normal distribution. Furthermore, a histogram is illustrated to graphically demonstrate distribution of the normalized results, as shown in Fig. 5. From this picture, it can be observed that a majority of results are near zero and few results are concentrated between 0.9 and 1. A possible reason is that the algorithm achieves satisfactory solutions under most parameter configurations.

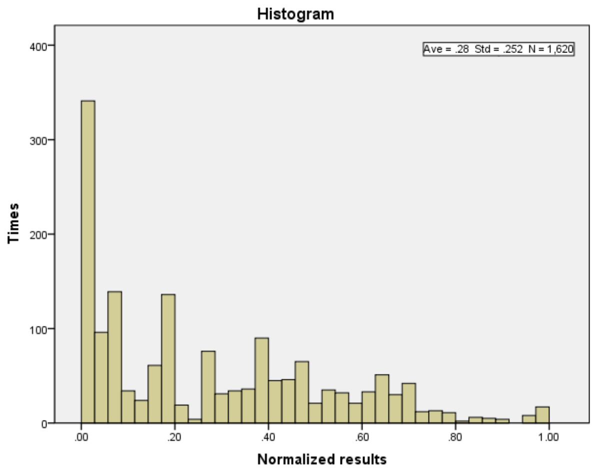

Fig. 5. The histogram of the normalized results obtained by different parameter configurations.

Since the results are not normally distributed, Friedman test [47] is carried out to display the average ranking of all configurations, listed in Table 4 (The besting ranking is in bold and the worst is in italics). The lower the mean rank, the better the algorithm. From this table, the 7th configuration (27)-(29) obtains the best ranking, which is suggested in the following experiments.

$$L_i(t+\mathbf{1}) = \alpha \cdot L_i(t), L_i\left(\mathbf{0}\right) = \left[\mathbf{1},\mathbf{2}\right], \alpha = \mathbf{0.9} \tag{27}$$

$$rp_i(t+\mathbf{1}) = rp_i(\mathbf{0}) \times \left[\mathbf{1}\text{-}e^{-\gamma \times t}\right], rp_i\left(\mathbf{0}\right) = \mathbf{0.85}, \gamma = \mathbf{0.9} \tag{28}$$

$$tr = 0.15 \tag{29}$$

Table 4 The results of Friedman test for parameter configurations

No.	level			ranking	No.	level			ranking
	L	*pr*	*tr*			*L*	*pr*	*tr*	
1	1	1	1	19.21	19	2	2	3	21.82
2	1	1	2	19.27	20	2	2	4	*22.53*
3	1	1	3	17.30	21	2	3	1	16.58
4	1	1	4	18.21	22	2	3	2	19.42
5	1	2	1	18.24	23	2	3	3	20.17
6	1	2	2	17.28	24	2	3	4	20.24
7	1	2	3	**14.94**	25	3	1	1	19.51
8	1	2	4	21.80	26	3	1	2	18.63
9	1	3	1	18.49	27	3	1	3	17.61
10	1	3	2	17.71	28	3	1	4	18.57
11	1	3	3	17.97	29	3	2	1	16.61
12	1	3	4	17.02	30	3	2	2	18.20
13	2	1	1	16.59	31	3	2	3	17.80
14	2	1	2	20.14	32	3	2	4	18.81
15	2	1	3	19.21	33	3	3	1	19.14
16	2	1	4	19.61	34	3	3	2	16.16
17	2	2	1	18.57	35	3	3	3	18.31
18	2	2	2	18.57	36	3	3	4	15.74

For a pairwise comparison, Nemenyi test [48] is further performed to report any significant differences between parameter configurations. The critical value $q_{0.05}$ is 3.84, based on the Studentized range statistic divided by $\sqrt{2}$. According to (30), the corresponding critical difference *CD* is 8.52, where *k* is the number of configurations, *NR* is the number of independent runs of each configuration.

$$CD = q_\alpha \sqrt{k\left(k+1\right)/6NR} \tag{30}$$

As shown in Table 4, the 7th configuration obtains the best ranking 14.94, and the 20th gets the worst ranking 22.53. Since even the difference between the best and the worst is less than 8.52, it can be concluded that there is no significant difference among all parameter configurations and the proposed algorithm is actually quite insensitive to parameters.

5.3 Performance analysis of improvement structures of NDBA

In this section, the performance of two improvement structures (HDBM and QLLS) is investigated to demonstrate their contributions. To achieve this goal, the following experiments are carried out 5 times independently for each instance. Firstly, HDBM is compared with HDBM1 and HDBM2 to verify which is the better approach for discretization. In the latter two methods, the movement of bat is not changed as iterations progress. HDBM1 only focuses on improving the convergence rate while HDBM2 only considers maintaining the population diversity. Then, QLLS is compared with five different local search strategies.

Table 5 illustrates the statistical comparisons between all improvement structures, including the mean values and the mean rankings (The best results are in bold and the worst are in italics). These improvement structures are carried out on 9 cases under the same number of fitness evaluation. From this table, HDBM achieves the best ranking among all improvements for the bat movement. It verifies improving convergence capability at the early stage and maintaining diversity at the latter stage are more effective than considering only one of them. Moreover, HDBM2 performs better than HDBM1 on P1 while worse on P3. From this result, it can be implied that maintaining diversity is more important than improving convergence on relatively simple questions. Among all local search strategies, LS1 yields the worst ranking. A possible reason is that LS1 adding one component increases time on the fitness calculation. Although it improves the system reliability, it increases the system cost at the same time. In addition, QLLS outperforms compared with other local search strategies, which demonstrates adjusting the local search strategies dynamically are the most effective. From the above, HDBM and QLLS are meaningful for the proposed NDBA. Furthermore, Nemenyi test is also carried out to report any significant differences between these improvement structures. The critical value $q_{0.05}$ is 3.10 and the corresponding critical difference *CD* is 1.79 according to (30). As the ranking listed in Table 5, the differences between the best and the others are all more than 1.79. That is to say, the interval which is combined by HDBM and QLLS does not overlap with any other improvements which indicates the combination of HDBM and QLLS is significantly different from the others.

Table 5 The comparisons between improvement structures of NDBA

Algorithm	P1			P2			P3			Ranking
	0.900	0.960	0.990	0.975	0.980	0.990	0.940	0.960	0.990	
HDBM	9.519	10.675	12.062	19.238	19.309	19.970	31.848	32.451	36.603	5.40
HDBM1	10.410	11.471	12.682	20.155	20.249	21.263	32.849	34.560	38.701	6.62
HDBM2	10.005	11.272	12.159	20.155	20.218	21.263	33.002	34.749	38.844	6.31
HDBM_LS1	*13.718*	*15.523*	15.577	*26.651*	*26.879*	*34.969*	*56.130*	*47.392*	*51.137*	*8.53*
HDBM_LS2	8.226	9.364	11.305	17.853	17.441	18.555	27.270	29.512	33.411	3.96
HDBM_LS3	7.892	8.413	10.401	19.430	18.767	20.823	25.709	27.546	32.759	3.42
HDBM_LS4	7.819	9.055	10.628	21.354	18.902	24.056	23.632	26.489	31.237	3.51
HDBM_LS5	13.159	14.049	*16.522*	21.048	21.494	31.890	27.695	33.779	34.430	6.11
HDBM_QLLS	**6.977**	**8.105**	**9.409**	**16.004**	**16.036**	**16.577**	**22.349**	**23.642**	**28.516**	**1.13**

5.4 Comparison with other state-of-the-arts

To further verify the performance of NDBA, it has been compared with several state-of-the-art variants of BA. These well-known algorithms are as follows: the origin BA proposed in 2010 [18], a novel BA with habitat selection and Doppler effect (NBA) in 2015 [49], a directional BA (DBA) in 2017 [22], a discrete hybrid BA with constriction coefficient and estimation of distribution algorithm (DCBA-dEDA) in 2019 [20], a BA with double mutation operators (TMBA) in 2020 [45]. The compared algorithms are strictly re-implemented with the same programming in the same environment and follow the parameters suggested in their literature. For a fair comparison, each algorithm is run 20 times for each instance. The common parameters of these algorithms are the same, i.e., the population size *N* is set to 20 and the max number of iterations t_{max} is set to 200.

The results are analyzed from two aspects: accuracy and convergence. Table 6 shows the statistical comparisons between all BA versions, including the mean values and the mean rankings (The best results are in bold and the worst are in italics). These versions are carried out on 9 cases under the same number of fitness evaluation. Most of the improved versions outperforms the standard BA which means the improvements are meaningful. As is obvious, the mean ranking of NDBA is lower than that of all compared algorithms. It can be concluded that NDBA owes the better search ability. Besides, Nemenyi test is also carried out. The critical value $q_{0.05}$ is 2.85 and the corresponding critical difference *CD* is 0.56 according to (30). As the ranking listed in Table 6, the differences between the best and the others are all more than 0.56. It implies the proposed algorithm is significantly better than other BA versions.

Table 6 The comparisons between BA versions

Algorithm	P1			P2			P3			Ranking
	0.900	0.960	0.990	0.975	0.980	0.990	0.940	0.960	0.990	
BA [18]	*9.274*	*10.821*	*11.932*	*21.044*	*20.821*	*21.578*	34.831	*35.204*	*39.607*	*5.51*
NBA [49]	8.199	9.564	10.717	17.546	17.536	17.999	26.088	27.153	31.739	3.31
DBA [22]	7.555	9.909	11.359	19.585	18.960	20.329	*35.552*	34.517	40.128	4.98
DCBA-dEDA [20]	7.555	8.568	9.778	16.787	16.841	17.328	24.629	25.858	29.980	2.24
TMBA [45]	8.305	9.859	11.045	18.348	18.573	19.239	25.858	26.540	30.843	3.78
NDBA	**7.105**	**8.102**	**9.305**	**16.007**	**16.040**	**16.584**	**22.520**	**23.964**	**28.520**	**1.18**

In addition to measuring the performance by statistical results, the convergence rate *cr* which is depended on the number of function evaluations is also utilized for comparing the proposed NDBA with other state-of-the-arts. This paper strictly follows a general scheme [50] proposed by Gandomi to evaluate the convergence rate. Besides, curve plots are also given in Fig. 7 for better illustrating the convergence process.

As detailed in Fig. 6, all modified versions are superior to the standard BA regarding the convergence, especially NBA. A main reason for the best convergence performance of NBA is the habitat selection from quantum behavior and mechanical behavior. Indeed, it is an effective mechanism which can accelerate the convergence significantly. However, NBA yields the third average ranking as statistical results listed in Table 6. From these results, it can be concluded that NBA does not effectively eliminate the prematurity of BA. In contrast to NBA, the proposed NDBA achieves a good balance between exploration and exploitation by HDBM and QLLS. In the former one, the algorithm speeds up the convergence and owes a good global ability at the early stage. As iterations go on, the algorithm focuses on the diversity to avoid local optimum. The latter one selects the optimal local search strategies via Q-learning to improve its performance.

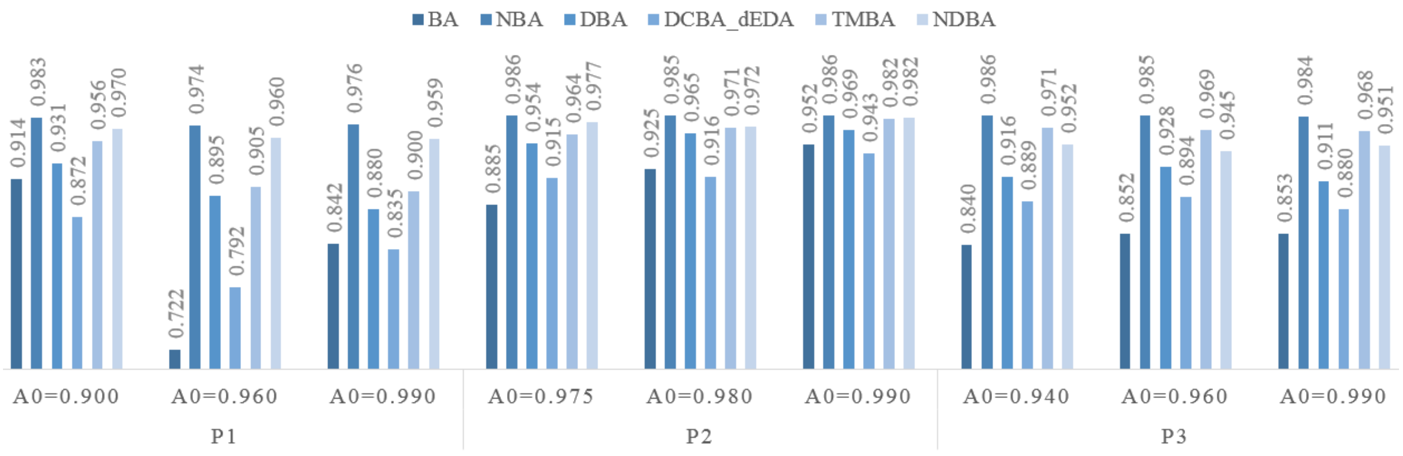

Fig. 6. The convergence rate of all BA versions.

This section also compares the proposed NDBA on standard RAP benchmarks (P1[41], P2[42], and P3[43]) without considering PCCF to check whether NDBA is comparable to the results that have been reported in existing literatures, e.g. GA [41-43], TS [51], SP/TG [52], PSO/LS [10], and MQEA [4]. These algorithms are performed on 9 cases with the same required system availabilities and available component versions, as shown in Section 5.1. However, they have different numbers of fitness evaluation *NFE*, given in Table 7. In this paper, the upper limit of each version MAX_{ij} is set to 8, suggested by [10]. The population size N is set to 50 and the max number of iterations t_{max} is set to 200. Due to no extra loop operations in NDBA, the number of fitness evaluation *NFE* is 1.0e4. Table 7 also lists the statistical comparisons between these powerful algorithms, including the best values C (The best results are in bold and the worst are in italics). From Table 7, SP/TG, PSO/LS, MQEA, and NDBA obtain better results than GA and TS. It implies the proposed NDBA is competitive compared with the state-of-the-arts. Moreover, NDBA spends the least *NFE*s among all algorithms. That is to say, NDBA captures the satisfactory solutions with fewer numbers of fitness evaluation, which also clearly highlights the efficiency of NDBA.

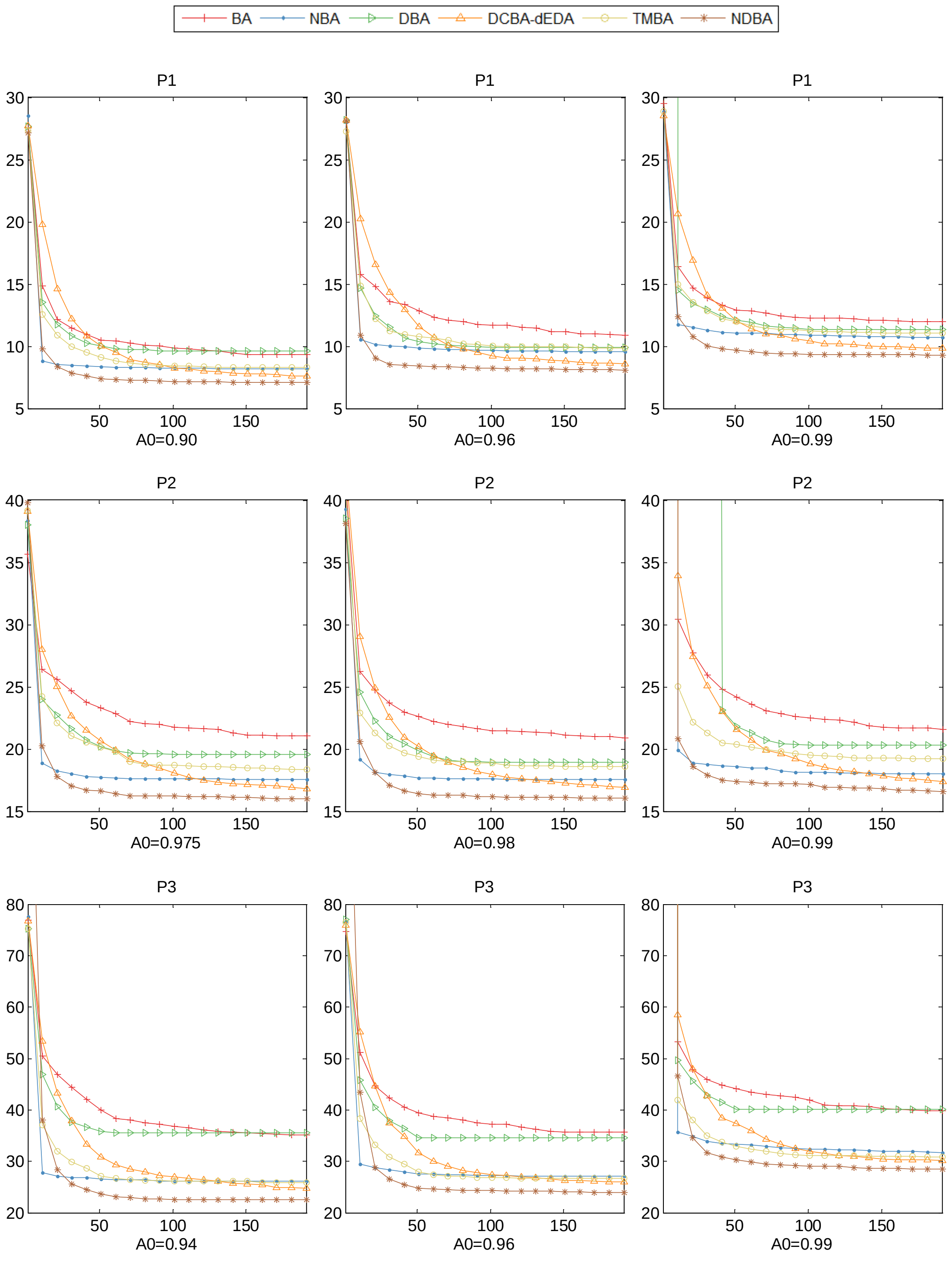

Fig. 7. The convergence plot of all BA versions.

Table 7 The comparisons between the state-of-the-arts without considering PCCF

Algorithm		P1			P2			P3		
		0.900	0.960	0.990	0.975	0.980	0.990	0.940	0.960	0.990
GA [41-43]	*C*	5.846	6.924	8.175	*16.450*	*16.520*	*17.050*	15.315	17.900	24.304
	NFEs	≈5.0e4	≈5.0e4	≈5.0e4	≈1.0e5	≈1.0e5	≈1.0e5	--	--	--
TS [51]	*C*	*5.986*	*7.303*	*8.328*	*16.450*	*16.520*	*17.050*	*17.805*	*21.155*	*24.305*

	NFEs	--	--	--	--	--	--	--	--	--
SP/TG [52]	*C*	**5.423**	**7.009**	**8.180**	**12.855**	**14.770**	**15.870**	**17.419**	**20.570**	**23.779**
	NFEs	≈1.2e6	≈1.2e6	≈1.0e6	≈3.0e6	≈2.5e6	≈3.0e6	≈5.0e6	≈5.0e6	≈5.0e6
PSO/LS [10]	*C*	**5.423**	**7.009**	**8.180**	**12.855**	**14.770**	**15.870**	**17.419**	**20.570**	**23.779**
	NFEs	>1.0e5	>1.0e5	>1.0e5	>2.0e5	>2.0e5	>2.0e5	>2.0e5	>2.0e5	>2.0e5
MQEA [4]	*C*	**5.423**	**7.009**	**8.180**	**12.855**	**14.770**	**15.870**	**17.419**	**20.570**	**23.779**
	NFEs	1.2e4	1.2e4	1.2e4	1.2e4	1.2e4	1.2e4	1.5e4	1.5e4	1.5e4
NDBA	*C*	**5.423**	**7.009**	**8.180**	**12.855**	**14.770**	**15.870**	**17.419**	**20.570**	**23.779**
	NFEs	**1.0e4**	**1.0e4**	**1.0e4**	**1.0e4**	**1.0e4**	**1.0e4**	**1.0e4**	**1.0e4**	**1.0e4**

"--" means the value is not available (The algorithm TS is terminated by the running time 2000s)

6 Conclusions

This paper aims to determine the optimal structure of multi-state series-parallel systems considering PCCF via availability evaluation and solution method. IUGF is utilized for the insufficient data caused by a non-fatal PCCF. To solve this problem, a novel discrete bat algorithm NDBA is proposed in this paper with two effective modifications: HDBM and QLLS. The former modification relies on the Hamming distance between the current bat and the best bat in the swarm to adjust its position dynamically, owing the following benefits: (1) enabling BA for addressing discrete problems; (2) enhancing the convergence ability at the early stage of algorithm; (3) maintaining the population diversity as the iterations progress; (4) having less parameters by decreasing *f*. The superiority of the latter one mainly owes to the following aspects: (1) enabling BA for addressing discrete problems too; (2) problem-independent by adjusting the local search strategies self-adaptively; (3) making a good balance between exploitation and exploration through local search around different objects. (4) tackling the shortcomings such as time-consuming and taking up much memory via the shared Q-table. To be specific, problem-independence means that NDBA has an excellent and stable performance no matter of the problem landscape. Besides, a comprehensive statistical analysis including the Friedman test and Nemenyi post-hoc test has been implemented in this paper. Experiment results exhibit that the proposed NDBA achieves significantly better performance compared with other state-of-the-art algorithms in solving the RAP*MSPS*PCCF.

The future work with both practical and theoretical benefits can be summarized as follows: multiple objectives, reliability-redundancy allocation, fuzzy valued data, and running time of evaluating the fitness function.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (U1433116), Fundamental Research Funds for Central Universities (NP2017208), and Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX19_0202). We thank the editor and the referees for handling and reviewing our submitted paper.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Reference

[1] K. Sabri-Laghaie and M. Karimi-Nasab, "Random search algorithms for redundancy allocation problem of a queuing system with maintenance considerations," *Reliability Engineering & System Safety,* vol. 185, pp. 144-162, 2019.

[2] C.-M. Lai and W.-C. Yeh, "Two-stage simplified swarm optimization for the redundancy allocation problem in a multi-state bridge system," *Reliability Engineering & System Safety,* vol. 156, pp. 148-158, Dec 2016.

[3] O. Chryssaphinou, N. Limnios, and S. Malefaki, "Multi-State Reliability Systems Under Discrete Time Semi-Markovian Hypothesis," *IEEE Transactions on Reliability,* vol. 60, pp. 80-87, Mar 2011.

[4] M. Du and Y.-F. Li, "An investigation of new local search strategies in memetic algorithm for redundancy allocation in multi-state series-parallel systems," *Reliability Engineering & System Safety,* vol. 195, p. 106703, 2020.

[5] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Transactions on reliability,* vol. 45, pp. 254-260, 1996.

[6] S. Kulturel-Konak, A. E. Smith, and D. W. Coit, "Efficiently solving the redundancy allocation problem using tabu search," *IIE transactions,* vol. 35, pp. 515-526, 2003.

[7] T.-C. Chen and P.-S. You, "Immune algorithms-based approach for redundant reliability problems with multiple component choices," *Computers in Industry,* vol. 56, pp. 195-205, 2005.

[8] A. Zaretalab, V. Hajipour, and M. Tavana, "Redundancy allocation problem with multi-state component systems and reliable supplier selection," *Reliability Engineering & System Safety,* vol. 193, p. 106629, 2020.

[9] M.-X. Sun, Y.-F. Li, and E. Zio, "On the optimal redundancy allocation for multi-state series–parallel systems under epistemic uncertainty," *Reliability Engineering & System Safety,* vol. 192, p. 106019, 2019.

[10] Y. Wang and L. Li, "Heterogeneous redundancy allocation for series-parallel multi-state systems using hybrid particle swarm optimization and local search," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 42, pp. 464-474, 2011.

[11] G. Levitin, "Incorporating common-cause failures into nonrepairable multistate series-parallel system analysis," *IEEE Transactions on Reliability,* vol. 50, pp. 380-388, Dec 2001.

[12] C. Wang, L. Xing, and G. Levitin, "Explicit and implicit methods for probabilistic common-cause failure analysis," *Reliability Engineering & System Safety,* vol. 131, pp. 175-184, 2014.

[13] L. Xing and W. Wang, "Probabilistic common-cause failures analysis," in *Reliability & Maintainability Symposium*, 2009.

[14] C.-y. Li, X. Chen, X.-s. Yi, and J.-y. Tao, "Heterogeneous redundancy optimization for multi-state series–parallel systems subject to common cause failures," *Reliability Engineering & System Safety,* vol. 95, pp. 202-207, 2010.

[15] K. C. Chae, "System reliability using binomial failure rate," in *1988. Proceedings., Annual Reliability and Maintainability Symposium*, 1988, pp. 136-138.

[16] C. Wang, L. Xing, and G. Levitin, "Probabilistic common cause failures in phased-mission systems," *Reliability Engineering & System Safety,* vol. 144, pp. 53-60, 2015.

[17] C. G. B. Huang T J, Yang Z C, "Muti-state system reliability calculation considering probabilistic common cause failure," *Chinese Journal of Ship Research,* pp. 1-6, 2019 (in chinese).

[18] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, ed: Springer, 2010, pp. 65-74.

[19] T. Talafuse and E. Pohl, "A bat algorithm for the redundancy allocation problem," *Engineering Optimization*, vol. 48, pp. 900-910, 2016.

[20] Y. Xu and D. Pi, "A hybrid enhanced bat algorithm for the generalized redundancy allocation problem," *Swarm and Evolutionary Computation*, vol. 50, p. 100562, 2019.

[21] K.-H. Chang and P.-Y. Kuo, "An efficient simulation optimization method for the generalized redundancy allocation problem," *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, vol. 265, pp. 1094-1101, 2018.

[22] A. Chakri, R. Khelif, M. Benouaret, and X.-S. Yang, "New directional bat algorithm for continuous optimization problems," *Expert Systems with Applications*, vol. 69, pp. 159-175, 2017.

[23] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, *et al.*, "Realization of an Adaptive Memetic Algorithm Using Differential Evolution and Q-Learning: A Case Study in Multirobot Path Planning," *IEEE Transactions on Systems Man & Cybernetics Systems*, vol. 43, pp. 814-831, 2013.

[24] H. Samma, C. P. Lim, and J. M. Saleh, "A new Reinforcement Learning-based Memetic Particle Swarm Optimizer," *Applied Soft Computing*, vol. 43, pp. 276-297, 2016.

[25] H. Samma, J. Mohamad-Saleh, S. A. Suandi, and B. Lahasan, "Q-learning-based simulated annealing algorithm for constrained engineering design problems," *Neural Computing and Applications*, pp. 1-15, 2019.

[26] Y. Xu and D. Pi, "A reinforcement learning-based communication topology in particle swarm optimization," *Neural Computing and Applications*, pp. 1-26, 2019.

[27] J. E. Ramirez-Marquez and D. W. Coit, "A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems," *Reliability Engineering & System Safety*, vol. 83, pp. 341-349.

[28] C. Y. Li, X. Chen, X.-s. Yi, and J.-y. Tao, "Interval-Valued Reliability Analysis of Multi-State Systems," *IEEE Transactions on Reliability*, vol. 60, pp. 323-330.

[29] A. Lisnianski, "Estimation of boundary points for continuum-state system reliability measures," *Reliability Engineering & System Safety,* vol. 74, pp. 81-88.

[30] H. L. Sun and W.-x. Yao, "The basic properties of some typical systems' reliability in interval form," vol. 30, pp. 364-373.

[31] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning,* vol. 8, pp. 279-292, 1992.

[32] P. K. Das, H. S. Behera, and B. K. Panigrahi, "Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity," *Engineering Science & Technology An International Journal,* vol. 19, pp. 651-669, 2016.

[33] L. Xing and W. Wang, "Probabilistic Common-Cause Failures Analysis," in *Annual Reliability and Maintainability Symposium, 2008 Proceedings,* ed, 2008, pp. 355-+.

[34] E. Osaba, X.-S. Yang, I. Fister Jr, J. Del Ser, P. Lopez-Garcia, and A. J. Vazquez-Pardavila, "A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection," *Swarm and evolutionary computation,* vol. 44, pp. 273-286, 2019.

[35] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation,* vol. 9, pp. 1-14, 2013.

[36] H. Samma, C. P. Lim, and J. M. Saleh, "A new reinforcement learning-based memetic particle swarm optimizer," *Applied Soft Computing,* vol. 43, pp. 276-297, 2016.

[37] H. Samma, J. Mohamad-Saleh, S. A. Suandi, and B. Lahasan, "Q-learning-based simulated annealing algorithm for constrained engineering design problems," *Neural Computing and Applications,* pp. 1-15, 2019.

[38] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, *et al.*, "Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multirobot path planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 43, pp. 814-831, 2013.

[39] J. Xie, Y. Zhou, and H. Chen, "A novel bat algorithm based on differential operator and Lévy flights trajectory,"

Computational intelligence and neuroscience, vol. 2013, 2013.

[40] M. A. Mellal and E. Zio, "A penalty guided stochastic fractal search approach for system reliability optimization," *Reliability Engineering & System Safety,* vol. 152, pp. 213-227, 2016.

[41] G. Levitin, A. Lisnianski, H. Ben-Haim, and D. Elmakis, "Redundancy optimization for series-parallel multi-state systems," *IEEE Transactions on Reliability,* vol. 47, pp. 165-172, 1998.

[42] G. Levitin, A. Lisnianski, and D. Elmakis, "Structure optimization of power system with different redundant elements," *Electric Power Systems Research,* vol. 43, pp. 19-27, 1997.

[43] A. Lisnianski, G. Levitin, H. Ben-Haim, and D. Elmakis, "Power system structure optimization subject to reliability constraints," *Electric Power Systems Research,* vol. 39, pp. 145-152.

[44] D. C. Montgomery, *Design and analysis of experiments,* 1976.

[45] Q. Liu, J. Li, L. Wu, F. Wang, and W. Xiao, "A novel bat algorithm with double mutation operators and its application to low-velocity impact localization problem," *Engineering Applications of Artificial Intelligence,* vol. 90, p. 103505, 2020.

[46] W. M. B. Shapiro S S "An analysis of variance test for normality (complete samples)," *Biometrika,* vol. 67, pp. 215-216, 1976.

[47] M. Friedman, "A Comparison of Alternative Tests of Significance for the Problem of m Rankings," *Annals of Mathematical Statistics,* vol. 11, pp. 86-92, 1940.

[48] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research,* vol. 7, pp. 1-30, 2006.

[49] X.-B. Meng, X. Z. Gao, Y. Liu, and H. Zhang, "A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization," *Expert Systems with Applications,* vol. 42, pp. 6350-6364, 2015.

[50] A. H. Gandomi, "Interior search algorithm (ISA): A novel approach for global optimization," *Isa Transactions,* vol. 53, pp. 1168-1183, 2014.

[51] M. Ouzineb, M. Nourelfath, and M. Gendreau, "Tabu search for the redundancy allocation problem of homogenous series–parallel multi-state systems," *Reliability Engineering & System Safety*, vol. 93, pp. 1257-1272, 2008.

[52] Mohamed, Ouzineb, Mustapha, NourelfathMichel, and Gendreau, "A heuristic method for non-homogeneous redundancy optimization of series-parallel multi-state systems," *Journal of Heuristics,* 2011.