Egyptian Informatics Journal 18 (2017) 221–232

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

Incorporating a modified uniform crossover and 2-exchange neighborhood mechanism in a discrete bat algorithm to solve the quadratic assignment problem

Mohammed Essaid Riffi [a], Yassine Saji [a,*], Mohammed Barkatou [b]

[a] LAROSERI Laboratory, Department of Computer Science, Faculty of Science, Chouaïb Doukkali University, Route Ben Maachou, 24000 El Jadida, Morocco
[b] Department of Mathematics, Faculty of Science, Chouaïb Doukkali University, Route Ben Maachou, 24000 El Jadida, Morocco

ARTICLE INFO

Article history:
Received 21 January 2016
Revised 13 November 2016
Accepted 22 February 2017
Available online 7 March 2017

Keywords:
Bat algorithm
Quadratic assignment problem
NP-hard problem
Combinatorial optimization problem

ABSTRACT

The bat algorithm is one of the recent nature-inspired algorithms, which has been emerged as a powerful search method for solving continuous as well as discrete problems. The quadratic assignment problem is a well-known NP-hard problem in combinatorial optimization. The goal of this problem is to assign n facilities to n locations in such a way as to minimize the assignment cost. For that purpose, this paper introduces a novel discrete variant of bat algorithm to deal with this combinatorial optimization problem. The proposed algorithm was evaluated on a set of benchmark instances from the QAPLIB library and the performance was compared to other algorithms. The empirical results of exhaustive experiments were promising and illustrated the efficacy of the suggested approach.

1. Introduction

Nowadays metaheuristics pose a great challenge for solving a numerous difficult combinatorial problems appearing in various industrial, economic, and scientific domains. Most of these problems are Non-deterministic Polynomial-time hard NP-hard [1,2], i.e. there is no polynomial time algorithm to solve them. Solving such combinatorial optimization problems require to involve the search for an optimal solution among a collection of solutions in finite search space. The nature has long been a rich source of inspiration for many scientists. Drawing their inspiration from the most successful process in nature, researchers develop a class of metaheuristics namely nature-inspired algorithms. The nature-inspired algorithms offer additional advantages over classical algorithms [3] and they also seek to find acceptable results within a reasonable time, rather than an ability to guarantee the optimal or sub-optimal solution. Moreover, the most metaheuristic algorithms are based on swarm intelligence, biological systems, physical and chemical systems [4]. In particular, the swarm-intelligence algorithms have shown their promising performance and they have been gaining much popularity in solving many engineering optimization problems; such as particle swarm optimization [5], spider monkey [6], cuckoo search [7], bee colony optimization [8], firefly algorithms [9] and in the last years the bat-inspired algorithm [10,11].

The Quadratic Assignment Problem (QAP) is proven to be among the hardest combinatorial optimization problems [12] and no polynomial time algorithm is expected to exactly solve the problem for large instances and sometime even small instances may require considerable computation time. In literature of combinatorial optimizations, researchers have applied different approaches ranging from heuristics or meta-heuristics until hybrid approaches, to find optimal or near optimal solutions for the QAP problem. The most widely used heuristic algorithms for QAP problem are simulated annealing [13], tabu search [14], neural network [15], ant colonies [16], memetic algorithms [17], genetic algorithms [18], iterated local search [19], hybrid heuristics [20], very large-scale neighborhood search [21], particle swarm optimization [22], bat algorithm [23], and bees algorithm [24].

The Bat-inspired Algorithm (BA) is a population-based stochastic optimization technique, which has recently been applied in

* Corresponding author.
E-mail address: yassine.saji@gmail.com (Y. Saji).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2017.02.003
1110-8665/

many applications. This algorithm was developed to deal with continuous optimization and especially, BA has also been proven effective in solving several discrete optimization problems. Owing to the complexness of the QAP for both exact and heuristic approaches, this problem is a suitable testing platform for novel optimization techniques. Therefore, the design of the improved methods for the QAP still poses a challenge to many researchers. This study investigates the applicability of BA to provide a very good compromise between solution quality and execution time. To that end, this work introduces a novel discrete version of BA named Discrete Bat Algorithm (DBA) by involving some modifications of the original algorithm. Firstly, these modifications concern position representation and its update equation, and velocity representation and its update equation. Secondly, we incorporate a modified uniform crossover operator in the movement equation. This idea aims to enhance the search strategy of BA. Thirdly, we use the 2-exchange neighborhood mechanism to improve the local search of BA. Finally, we adjust some suitable parameters values to balance between intensification and diversification capabilities of BA. We compare the performance of our proposed approach to five algorithms from the current literature: Bees Algorithm (BeA) [24], Modified Bat Algorithm (MBA) [23], a genetic algorithm with Sequential Constructive Crossover (SCX) [25], Hybrid Ant colony System coupled with the QAP (HAS-QAP) [16] and Hierarchical Particle Swarm Optimization (HPSO) [22].

The rest of this paper is organized as follows: the second section presents the literature review of BA. Section 3 introduces the original BA. Section 4 describes the quadratic assignment problem. Section 5 discusses our discrete approach and details its main components. The results of computational experiments and discussions are presented in Section 6. Finally, the conclusions and some perspectives of research are given in Section 7.

2. Related work

The BA is a swarm-intelligence algorithm based on the echolocation behaviour of microbats when searching their prey. Initially, this algorithm has been developed to optimize continuous nonlinear functions [10,26]. Actually, the algorithm shows a good efficiency when it has been applied to solve various optimization problems [27]. In the current literature, many variants of BA have been introduced to enhance or to implement the original algorithm to solve different sorts of problems. Gandomi and Yang integrated chaos mechanism in BA to improve the global search mobility of the basic algorithm and to optimize effectively a set of benchmark problems [28]. Cai et al. introduced a Gaussian walk with BA to enhance the local search capability and they modified the velocity equation to assure a good exploitation of the search space [29]. Khan et al. introduced a fuzzy modification of BA for clustering of company workplaces [30]. Yılmaz and Küçüksille embedded two modifications in structure to enhance the local and global search and they also used both standard test functions and constrained real-world problems to verify the effectiveness of proposed approach [11]. Nguyen et al. hybridized BA and artificial bee colony algorithm based on communication strategies in parallel processing for swarm intelligent algorithms with the aim to

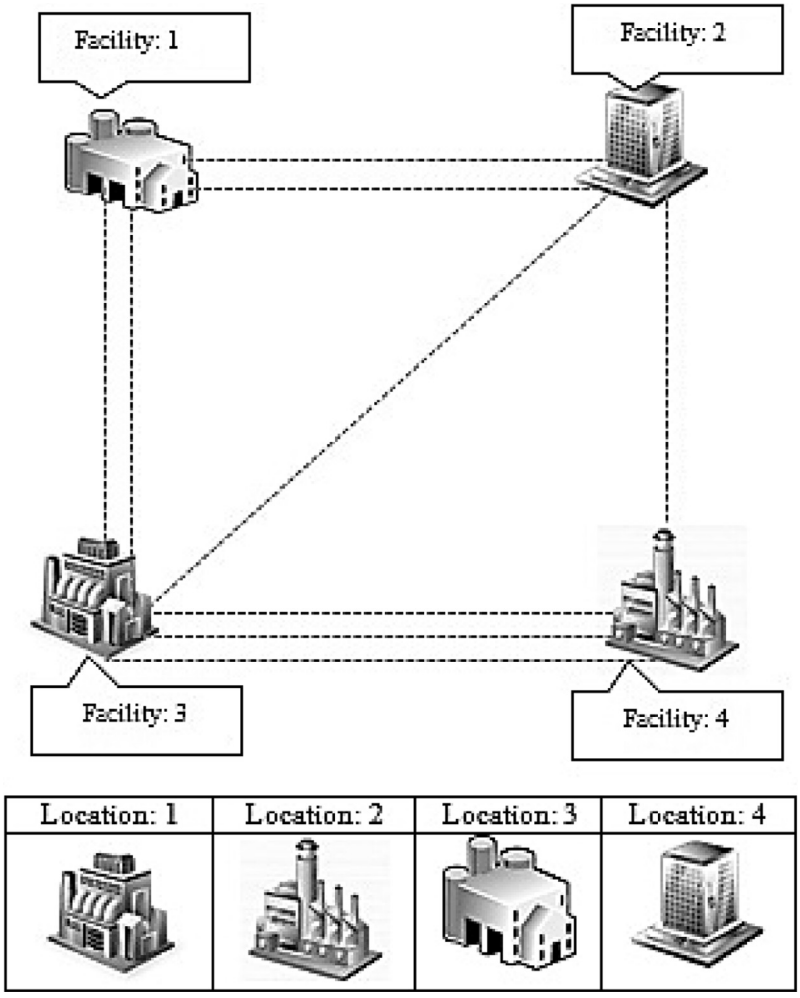

Fig. 1. Illustrative example of quadratic assignment problem with the permutation $\pi = (2, 4, 3, 1)$ corresponds to optimal solution.

improve the convergence and accuracy of both algorithms [31]. Fister et al. incorporated two mechanisms within the original BA algorithm in order to obtain the hybrid self-adaptive BA, the first one is self-adaptation of control parameters, and the second is the hybridization of the self-adaptive BA using four DE strategies as a local search heuristics [32]. Mirjalili et al. used the sigmoid function to propose a binary version of BA to solve unimodal, multimodal, and composite functions [33]. Chen et al. employed the Doppler Effect to enhance the efficiency of BA in solving optimization problems [34]. Although BA was widely used to solve continuous problems, nevertheless, the application of the basic BA had not been exploited effectively to solve discrete problems before 2012. The first binary version proposed by Nakamura et al. [35] in 2012 to solve classifications and feature selection problems. They have maintained the same structure of the original algorithm, whilst they have transformed both global and local position update by using the sigmoid function. Recently, many studies have been focused on using BA to solve some discrete problems. For example, Xie et al. proposed a differential Lévy-flights BA to optimize the permutation flow shop problem [36]. Luo et al. integrated an intensive virtual population neighborhood search into proposed DBA for optimal permutation flow shop scheduling problem [37]. Sabba and Chikhi suggested a discrete binary BA based on sigmoid function for solving the multidimensional knapsack problem [38]. Bü yüksaatçı used BA to solve the single row facility layout problem [39]. Fister et al. introduced a modified bat algorithm for planning the sports training sessions [40]. Dekhici and Belkadi hybridized a discrete version of BA with generalized evolutionary walk algorithm to solve the mono-processors two stages flexible flow shop [41]. Hasançebi and Carbas introduced a BA for structural optimization of steel frames designed for minimum weight, and they used three real-size large steel frames to validate the performance of their approach [42,43]. Eslam et al. used a discrete BA for finding community structure in social network [44]. Saji et al. introduced two discrete variants of BA to solve the Euclidean travelling salesman problem [45,46], and so on. As we have mentioned previously, the BA has been applied to many discrete optimization problems, anyway, it has been rarely applied to any assignment problem [23]. In this paper, we propose a novel discrete version of BA to solve the quadratic assignment problem as NP-hard problem [12] in combinatorial optimization.

3. Bat algorithm

Bat-inspired algorithm is a meta-heuristic search optimization firstly proposed by Xin-She Yang in 2010 [10]. This algorithm is derived by simulating the echolocation behaviours of micro-bats when seeking their prey and discriminating different types of insects even at complete darkness by varying pulse rates of emission and loudness. The echolocation guides the hunting strategies of bats as these last tend to decrease the loudness and increase the rate of emitted ultrasonic depending on the proximity of their target.

Based on the echolocation behaviour of bats, Yang in [10] describes the basic steps of BA using the following three idealized rules:

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers.
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r in the range of [0,1], depending on the proximity of their target where 0 denotes no pulses at all, and 1 denotes the maximum rate of pulse emission.
3. The loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

The basic pseudo code of BA as presented in its original paper is summarized in Algorithm 1.

Algorithm 1: Pseudo-code of bat algorithm

1. Objective function $f(x)$, $x = (x_1, \ldots, x_d)^T$
2. Initialize the bat population x_i $(i = 1,2, \ldots, n)$ and v_i
3. Define pulse frequency f_i at x_i
4. Initialize pulse rates r_i and the loudness A_i
5. **while** ($t <$ *Max number of iterations*)
6. **for** $i = 1$ to m
7. Generate new solution by adjusting frequency,
8. and updating velocity and location/solution // Equations (1), (2) and (3)
9. **if** ($rand > r_i$)
10. Select a solution among the best solutions of current iteration
11. Generate a local solution around the selected best solution
12. **end if**
13. Generate a new solution by flying randomly
14. **if** ($rand < A_i$ & $f(x_i) < f(x_*)$) // x_i is current solution and x_* is global best solution
15. Accept the new solutions
16. Increase r_i and reduce A_i // Equations (5) and (6)
17. **end if**
18. **end for**
19. Rank the bats and find the current global best x_*
20. **end while**
21. Post process results and visualization

In the BA, at iteration t, an artificial bat x_i is determined by a position vector x_i^t, a velocity vector v_i^t, a frequency vector f_i a pulse rate r_i^t and a loudness A_i^t, which are updated during the research process to find the best solution as Eqs. (1)(3):

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i, \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{3}$$

where $\beta \in [0,1]$ is a random vector generated from uniform distribution and where x_* is the current global best solution which is located after comparing all the solutions among all the m bats.

For the local search, a new solution will be accepted around the current best solutions by using a random walk:

$$x_{new} = x_{old} + \varepsilon A^t, \tag{4}$$

where $\boldsymbol{\varepsilon} \in [-1, 1]$ is a random number while $A^t = \langle A_i^t \rangle$ is the average loudness of all the bats at current time step.

During the flying process, the loudness A_i and the pulse emission rate r_i will be updated, whenever a new best solution will be accepted. The loudness and the pulse emission rate are updated as follow:

$$A_i^{t+1} = \alpha A_i^t, \tag{5}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)], \tag{6}$$

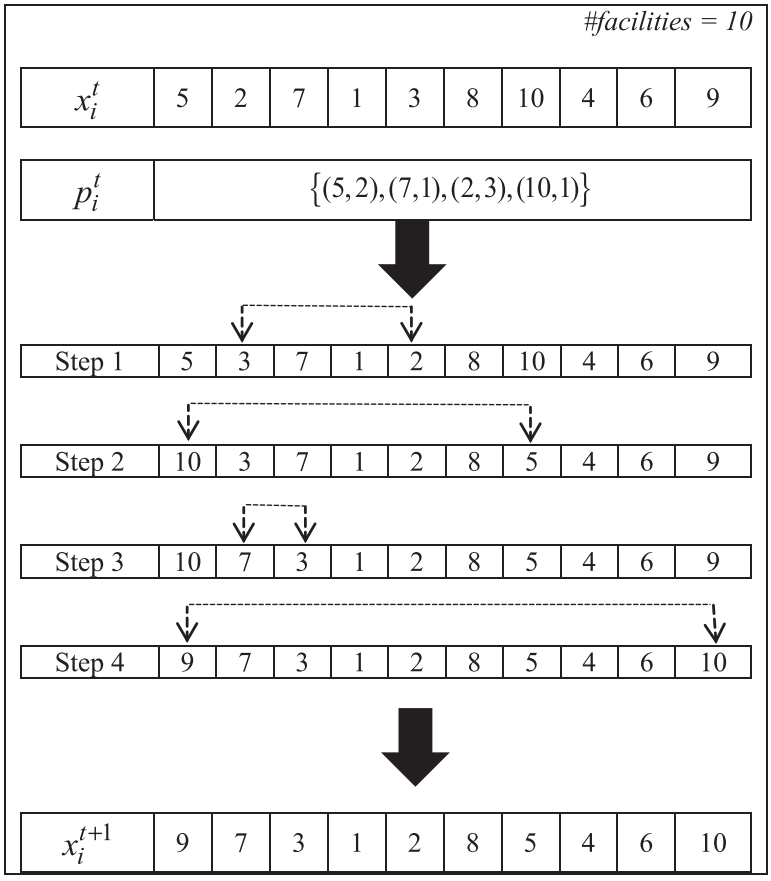

Fig. 2. Illustration of sort function.

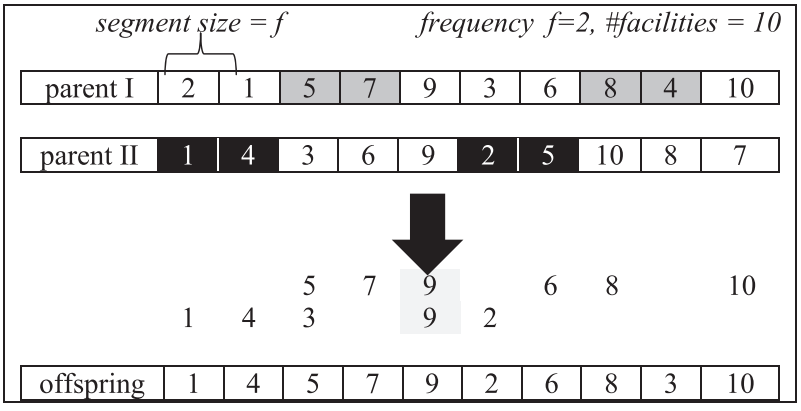

Fig. 3. Example of modified uniform crossover (UX).

where α, γ are two constants, generally $\alpha = \gamma$ and the initial values A_i^0 and r_i^0 are chosen randomly. The choice of these parameters requires some experimenting and depends the problem tackled.

4. Quadratic assignment problem

The QAP is an important issue in combinatorial optimization problem. This problem was originally introduced by Koopmans and Beckmann in 1957 [47] and classified as a NP-hard problem [12]. The QAP is described as a problem to allocating a set of n facilities to a set of n locations in such a way that each location is designated to exactly one facility and vice versa. For each pair of facilities versus locations assignment costs are known. The objective is to minimize cost allocation of facilities into locations, while taking into account the sum of distances multiplied by the corresponding flows between all facilities.

Table 1
The parameters of the problem.

Parameter	Value
Population size: m	15
Emission rates r_i	0.5, $r_i \in [0.0, 1.0]$
Loudness A_i	0.5, $A_i \in [0.0, 1.0]$
Minimal frequency f_{min}	1
Maximal frequency f_{max}	2
Maximal number of iterations t_{max}	200

In a formal way, QAP can be stated as given n facilities, n locations, and two matrices $A = (a_{ij})_{n\times n}$ and $B = (b_{kl})_{n\times n}$, where a_{ij} is the flow between facilities i and j for all $i, j \in \{1,2,\ldots,n\}$ and b_{kl} is the distance between locations k and l for all $k, l \in \{1,2,\ldots,n\}$. The goal of this problem is to find the permutation $\pi_* = (\pi(1), \pi(2), \ldots, \pi(n)) \in \prod$ (Π set of all possible permutations) that minimizes cost function C:

$$C(\pi_*) = \min_{\pi\in\prod} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \tag{7}$$

Let us consider the problem of assigning four facilities to four locations and two matrices A and B. In Fig. 1, the dotted line between a pair of facilities indicates that there is required flow between the facilities, and to calculate the assignment cost of the permutation, the distances between locations are needed. One of possible assignment is to place facility 1 to location 3, facility 2 to location 4, and facility 3 to location 1, and facility 4 to location 2. This assignment can be written as the permutation $\pi = (3,4,1,2)$ but it is not necessarily the optimal one.

$$A = \begin{pmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 3 & 0 \end{pmatrix}, \; A \text{ is the flow Matrix}$$

$$B = \begin{pmatrix} 0 & 2 & 1 & 3 \\ 2 & 0 & 1 & 2 \\ 1 & 1 & 0 & 3 \\ 3 & 2 & 3 & 0 \end{pmatrix}, \; B \text{ is the distance Matrix}$$

Besides the facility location problem, QAP has a wide applicability to formulate a great variety of scientific and industrial problems such as design of type-writer keyboards, backboard wiring in electronics, balancing turbine runners, layout design, data allocation, scheduling, and many others. The QAP can be transformed into many classic combinatorial optimization problems such as the travelling salesman [48], graph partitioning problems [48], maximum clique [48], data allocation problem [49], weapon target assignment [50].

5. Discrete bat algorithm for QAP

Basically, the standard BA is a continuous optimization algorithm developed to optimize continuous non-linear functions [10,26]; however this algorithm cannot be used to solve discrete problems directly. In this study, a DBA is designed based on original BA with the aim to solve discrete combinatorial problems as QAP typical case. In order to apply DBA to QAP, there are some modifications of fundamental equations: the first modification concerns position and velocity representation; the second modification involves position-updating equation.

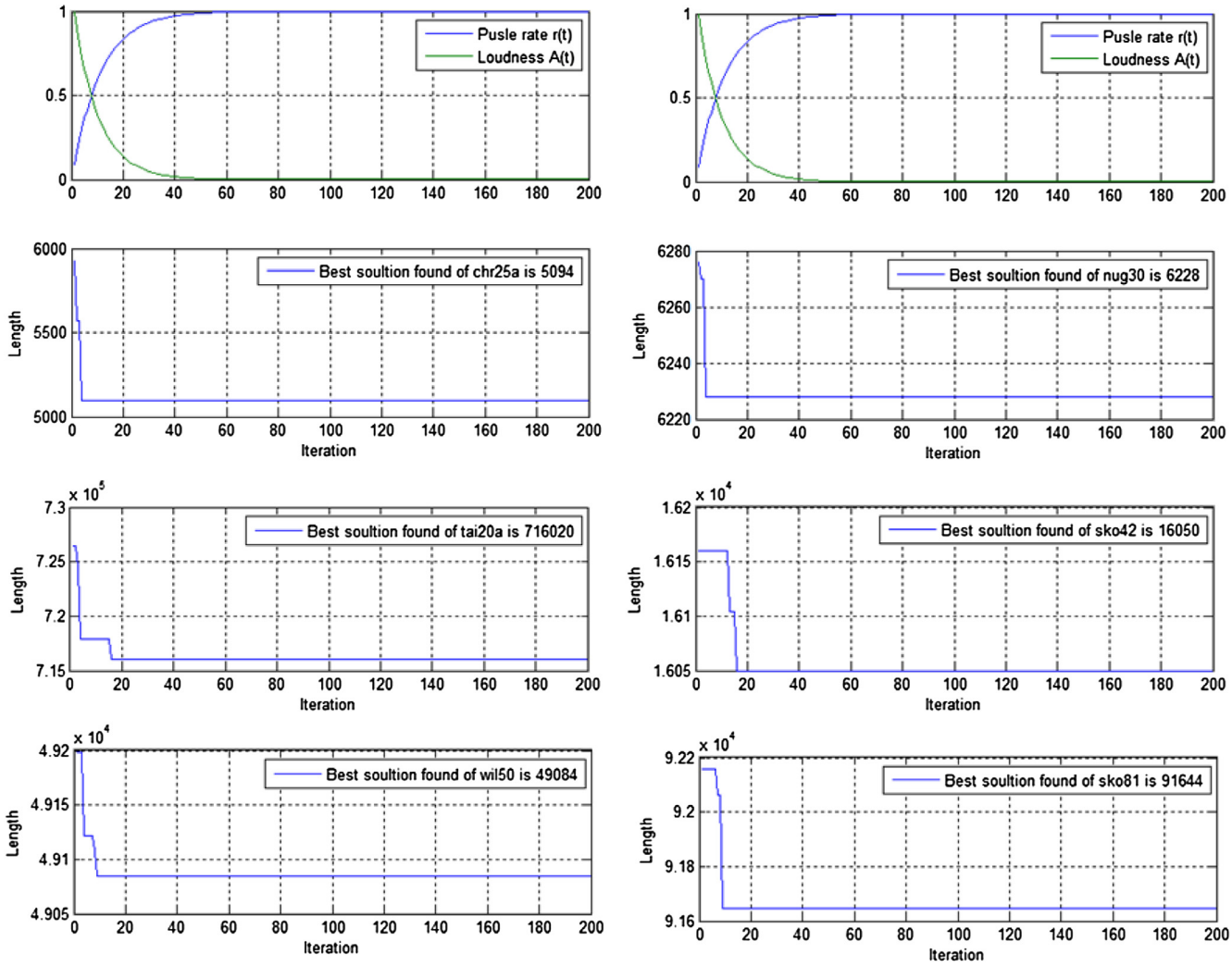

Fig. 4. The best solution found of char25a, nug30, tai20a, sko42, wil50 and sko81 when r_i and A_i are varying between 0 and 1.

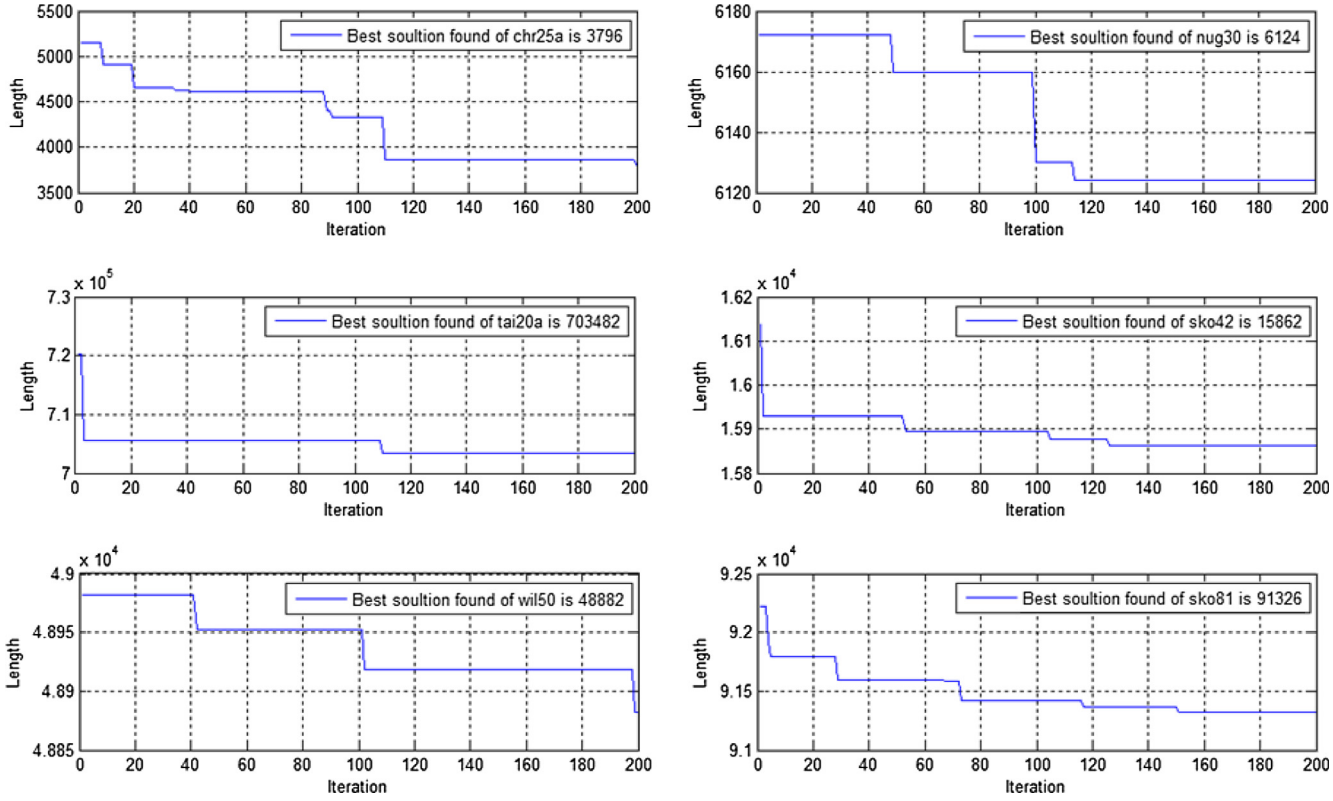

Fig. 5. The best solution found of chr25a, nug30, tai20a, sko42, wil50 and sko81 when r_i and A_i are equal to 0.5.

In addition, for combinatorial problems, some neighborhood search methods are always used to enhance the quality of the solution. In this paper, the 2-exchange neighborhood function is a suitable neighborhood search method for QAPs. The details of this function will be presented in the following subsection.

5.1. The 2-exchange neighborhood

In the context of QAP, we recall that the aim is to find a permutation π of n elements that minimizes Eq. (7), where $\pi \in \Pi$. The 2-exchange neighborhood π' is obtained by the set of permutations which can be reached by applying the following pairwise exchanging neighborhood function $N_2 : \prod \to 2^{\prod}$ that defines for each $\pi \in \prod$ a set of neighboring solutions of π, $N_2(\pi) = \{\pi' | \pi' \in \prod, \phi(\pi, \pi') = 2\}$ where $\phi(\pi, \pi')$ is the Hamming distance between permutations π and π'. The neighboring permutation (solution) $\pi' \in N_2(\pi)$ is obtained from π by applying the swapping move $\chi(\pi, i, j) : \prod \times N \times N \to \prod$, between the ith and jth element in the current solution π, that means $\pi'(i) = \pi(j)$, $\pi'(j) = \pi(i)$, $\pi'(k) = \pi(k)$, $k = 1, 2 \ldots, n$, $k \neq i, k \neq j$, and we note $\pi' = \pi \oplus \chi(\pi, i, j)$. The criterion to select a pair of exchanging elements from all possible pairs $\left(\text{equals } \frac{n(n-1)}{2}\right)$ is calculated by the following formula:

$$\begin{aligned}\Delta_c(\pi, i, j) &= (a_{ii} - a_{jj})(b_{\pi(j)\pi(j)} - b_{\pi(i)\pi(j)}) + (a_{ij} - a_{ji})(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)}) \\ &+ \sum_{k=1, k\neq i,j}^{n} (a_{ik} - a_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) \\ &+ (a_{ki} - a_{kj})(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)}) \end{aligned} \quad (8)$$

where $a_{ii}, b_{ii} = 0, i = 1, 2, \ldots, n$.

In the symmetric case, the formula (8) is simpler and can rewrite as follow:

$$\Delta_c(\pi, i, j) = \sum_{k=1, k\neq i,j}^{n} (a'_{ik} - a'_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) \quad (9)$$

where $A\prime = (a'_{ij})_{n\times n} / a'_{ij} = a_{ij} + a_{ji}, i, j = 1, 2, \ldots, n, i \neq j, a'_{ii} = a_{ii}, i = 1, 2, \ldots n$ is the new symmetric flow matrix if A is asymmetric flow matrix (for more details see [51]).

The outline of the local search algorithm used in this paper is shown in Algorithm 2.

Algorithm 2: Pseudo-code of local search algorithm

```
1. Procedure local_search
2. repeat
3.     Generate a neighboring solution π' ∈ N2(π);
4.   if (C(π') < C(π))
5.       π = π';
6.   end if
7. until (∀π' ∈ N2(π), C(π') ⩾ C(π))
8. end Procedure
```

5.2. Position and velocity equations

In the basic BA developed by Yang [10], the movement of each virtual bat in n-dimensional search space (n is the size of problem) is characterized by two vectors; position and velocity. During the search process, the bats tend to fly towards the best position (solution) found since the first iteration by updating their velocities v_i^t and positions x_i^t at time step $\boldsymbol{t}$. So, in the case of QAP, the solution of n-assignments found by the ith bat is represented by n-

dimensional vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The velocity v_i is viewed as a set of permutations $p_i = \{p_1, p_2, \ldots, p_n\}$ that allows being close to the global best solution $x_* = (x_{*1}, x_{*2}, \ldots, x_{*n})$. The standard position update and velocity equations will be redefined as follow:

$$v_i^{t+1} = \psi(x_i^t, x_*, f_i) \quad (10)$$

$$x_i^{t+1} = \sigma(x_i^t, v_i^{t+1}) \quad (11)$$

In this case, the function ψ in Eq. (10) is a crossover function with three arguments and returns a set of permutations p_i^t. The frequency f_i is an integer (between $f_{\min}$ and $f_{\max}$) that denotes the size of segments copied from either x_* or x_i^t to the offspring (the proposed function is more detailed in Fig. 3). The function σ in Eq. (11) is used to sort the locations assigning in x_i^t while respecting p_i^t and the template of this function is given in Fig. 2.

Table 2
Results of the DBA algorithm for some instances from QAPLIB [59].

Instance	BKV	Best	Worst	SD	$\%\delta_{avg}$	$\%\delta_{best}$	$t_{avg}(s)$
bur26a	5,426,670	5,426,670	5,426,670	**0.00**	**0.00**	**0.00**	1.10
bur26b	3,817,852	3,817,852	3,817,852	**0.00**	**0.00**	**0.00**	0.80
bur26c	5,426,795	5,426,795	5,426,795	**0.00**	**0.00**	**0.00**	0.75
bur26d	3,821,225	3,821,225	3,821,225	**0.00**	**0.00**	**0.00**	0.90
bur26e	5,386,879	5,386,879	5,386,879	**0.00**	**0.00**	**0.00**	0.82
bur26f	3,782,044	3,782,044	3,782,044	**0.00**	**0.00**	**0.00**	0.43
bur26g	10,117,172	10,117,172	10,117,172	**0.00**	**0.00**	**0.00**	0.21
bur26h	7,098,658	7,098,658	7,098,658	**0.00**	**0.00**	**0.00**	0.35
chr12a	9552	9552	9552	**0.00**	**0.00**	**0.00**	0.09
chr15b	7990	7990	7990	**0.00**	**0.00**	**0.00**	0.86
chr18a	11,098	11,098	11,682	227.20	1.06	**0.00**	0.05
chr20c	14,142	14,142	14,142	**0.00**	**0.00**	**0.00**	0.30
chr25a	3796	3796	4410	190.95	8.32	**0.00**	5.96
els19	17,212,548	17,212,548	17,212,548	**0.00**	**0.00**	**0.00**	0.08
esc16a	68	68	68	**0.00**	**0.00**	**0.00**	0.00
esc16b	292	292	292	**0.00**	**0.00**	**0.00**	0.00
esc16c	160	160	160	**0.00**	**0.00**	**0.00**	0.00
esc16d	16	16	16	**0.00**	**0.00**	**0.00**	0.00
esc16e	28	28	28	**0.00**	**0.00**	**0.00**	0.00
esc16f	0	0	0	**0.00**	**0.00**	**0.00**	0.00
esc32a	130	130	130	**0.00**	**0.00**	**0.00**	0.02
esc32e	2	2	2	**0.00**	**0.00**	**0.00**	0.03
esc32g	6	6	6	**0.00**	**0.00**	**0.00**	0.02
esc64a	116	116	166	**0.00**	**0.00**	**0.00**	0.18
wil50	48,816	48,844	48,954	27.77	0.14	0.05	115.67
nug20	2570	2570	2570	**0.00**	**0.00**	**0.00**	0.88
nug21	2438	2438	2438	**0.00**	**0.00**	**0.00**	0.12
kra30a	88,900	88,900	90,670	678.82	0.38	**0.00**	0.26
kra30b	91,420	91,420	91,690	97.57	0.11	**0.00**	2.12
nug30	6124	6124	6176	19.05	0.21	**0.00**	4.9
rou20	725,522	725,662	731,798	1904.8	0.36	0.01	7
tai12a	224,416	224,416	224,416	**0.00**	**0.00**	**0.00**	0.00
tai12b	39,464,925	39,464,925	39,464,925	**0.00**	**0.00**	**0.00**	0.01
tai15a	388,214	388,214	388,214	**0.00**	**0.00**	**0.00**	0.50
tai15b	51,765,268	51,765,268	51,765,268	**0.00**	**0.00**	**0.00**	0.09
tai17a	491,812	491,812	491,812	**0.00**	**0.00**	**0.00**	0.39
tai20a	703,482	703,482	712,600	2891.9	0.85	**0.00**	0.18
tai20b	122,455,319	122,455,319	122,455,319	**0.00**	**0.00**	**0.00**	0.11
tai25a	1,167,256	1,172,754	1,192,030	6642.3	1.51	0.47	12.10
tai25b	344,355,646	344,355,646	344,355,646	**0.00**	**0.00**	**0.00**	1.72
tai30a	1,818,146	1,831,272	1,854,640	8066.6	1.34	0.72	20.25
tai30b	637,117,113	637,117,113	637,917,440	291418.9	0.02	**0.00**	3.30
tai35a	2,422,002	2,438,440	2,477,554	10798.69	1.79	0.67	35.43
tai35b	283,315,445	283,315,445	284,457,830	290275.7	0.19	**0.00**	20.01
tai40a	3,139,370	3,139,370	3,217,018	10679.3	2.02	1.3	51.12
tai40b	637,250,948	637,250,948	637,376,825	38744.1	0.00	**0.00**	14.86
tai50a	4,938,796	5,042,654	5,080,430	10982.3	2.69	2.10	100
tai50b	458,821,517	458,830,119	461,082,283	663488.4	0.11	**0.00**	126.34
tai60a	7,205,962	7,387,482	7,411,000	6163.05	2.73	2.5	166.23
tai60b	608,215,054	608,414,385	609,014,871	155656.2	0.09	0.03	221.65
tai64c	1,855,928	1,855,928	1,855,928	**0.00**	**0.00**	**0.00**	0.26
tai80a	13,499,184	13,810,130	13,891,054	21496.4	2.67	2.30	420.62
tai80b	818,415,043	819,367,807	825,968,525	2386128.5	0.56	0.11	532.41
tai100a	21,052,466	21,541,326	21,632,428	24712.6	2.5	2.3	1045.27
tai100b	1185996137	1188168753	1190873912	906873.1	0.29	0.18	1126.25
sko42	15,812	15,812	15,916	28.17	0.30	**0.00**	42.87
sko49	23,386	23,421	23,570	37.97	0.34	0.14	97.71
sko56	34,458	34,524	34,700	56.16	0.46	0.19	159.8
sko64	48,498	48,656	48,802	42.94	0.44	0.32	265.63
sko72	66,256	66,422	66,658	77.79	0.46	0.25	361.59
sko81	90,998	91,252	91,828	151.65	0.45	0.27	512.73
sko90	115,534	115,874	116,378	121.30	0.48	0.29	653.61

The values shown in bold indicate that the optimal solution is reached for each instance.

5.3. The crossover operation

The crossover operation was firstly introduced in genetic algorithms to produce a new population. The idea behind the crossover is to benefit the best characteristics from each of the parents when producing the new generation. In the literature, there is a variety of crossover operators proposed to solve the quadratic assignment problem such as the ordered crossover (OX) [52], the cycle crossover (CX) [17], the multiple parent crossover (MPX) [53], the swap path crossover (SPX) [54], the partially mapped crossover (PMX) [55], the cohesive crossover (COHX) [56], the one point crossover (OPX) [57], and the uniform crossover (UX) [58]. In order to increase the diversity of the population and to avoid the premature convergence toward undesirable regions in the search space, we propose some modification of the uniform crossover (UX) [58]. In this paper, the modified uniform crossover (UX) is used to enhance the search strategy of BA. Like the standard crossover (UX) process, the order of scanning each parent is from left to right, and all items assigned to the same position in both parent individuals are copied to this position in the offspring. The unassigned items are grouped in segments of size f (frequency). The first segment is copied from one of parents to child; the remaining elements (second segment) are copied from the other parent if they are not yet included in the offspring. Otherwise, if there still any unassigned positions in the child, the missing items can be taken from either the first or second parent in such a way that it does not appear twice in the offspring. An example of this crossover process is illustrated in Fig. 3.

5.4. Discrete bat algorithm

In DBA, the controlling parameters are initialized as described in Table 1, expected the starting solution x_i which is randomly generated for each bat. During the search process, the bats use a memory to communicate the best global solution x_* between them. To generate a new solution, every bat adjusts its frequency and updates its current velocity and position (Eqs. (1), (10) and (11)). Then, one solution is selected among the best solutions according to the pulse emission rate r_i. Afterwards, the selected solution is locally changed by using Algorithm 2, in a manner that improves the current solution and the best global solution found so far. Then,

Table 3
Comparative results between BeA [24] and DBA for solving some instances from QAPLIB [59].

Instance	BKV	BeA				DBA			
		Best	I_{best}	t (ms)	$\%\delta_{best}$	Best	I_{best}	t (ms)	$\%\delta_{best}$
chr12a	9552	9552	289	375	**0.00**	9552	8	16.20	**0.00**
chr15b	7990	7990	222	452	**0.00**	7990	20	49.37	**0.00**
esc16a	68	68	2	514	**0.00**	68	1	0.00	**0.00**
had16	3720	3720	24	468	**0.00**	3720	8	71.29	**0.00**
chr18a	11,098	11,118	170	530	0.18	11,098	155	19.73	**0.00**
chr20c	14,142	14,142	543	609	**0.00**	14,142	97	74.91	**0.00**
rou20	725,522	727,322	700	590	0.25	725,522	88	116.60	**0.00**
tai20a	703,482	724,472	446	671	2.98	703,482	173	51.80	**0.00**
nug21	2438	2442	603	747	0.16	2438	97	98.82	**0.00**
chr25a	3796	3796	974	1156	**0.00**	3796	126	108.72	**0.00**
bur26a	5,426,670	5,431,640	832	826	0.09	5,426,670	65	130.58	**0.00**
bur26b	3,817,852	3,817,948	335	796	**0.00**	3,817,852	39	168.93	**0.00**
esc32a	130	144	902	1092	10.77	130	192	164.46	**0.00**
wil50	48,816	50,216	991	2371	2.87	48,844	320	507.36	0.05
esc64a	116	116	340	3027	**0.00**	116	1	0.00	**0.00**
tai80a	13,499,184	14,784,380	737	4087	9.52	13,810,130	345	3208.33	2.30

The values shown in bold indicate that the optimal solution is reached for each instance.

Table 4
Comparative results between MBA [23] and DBA for solving some instances from QAPLIB [59].

Instance	BKV	MBA				DBA			
		Best	Worst	Average	$\%\delta_{avg}$	Best	Worst	Average	$\%\delta_{avg}$
bur26a	5,426,670	5,519,108	5,537,973	5,530,825	1.90	5,426,670	5,426,670	5,426,670	**0.00**
bur26b	3,817,852	3,895,507	3,905,048	3,898,402	2.10	3,817,852	3,817,852	3,817,852	**0.00**
bur26c	5,426,795	5,520,289	5,526,200	5,523,970	1.79	5,426,795	5,426,795	5,426,795	**0.00**
bur26d	3,821,225	3,879,814	3,897,039	3,887,216	1.72	3,821,225	3,821,225	3,821,225	**0.00**
bur26e	5,386,879	5,491,088	550,633	5,500,460	2.10	5,386,879	5,386,879	5,386,879	**0.00**
bur26f	3,782,044	3,830,614	3,861,240	3,850,692	1.80	3,782,044	3,782,044	3,782,044	**0.00**
bur26g	10,117,172	10,320,120	10,342,560	10,328,699	2.09	10,117,172	10,117,172	10,117,172	**0.00**
bur26h	7,098,658	7,251,046	7,278,182	7,267,393	2.37	7,098,658	7,098,658	7,098,658	**0.00**
esc32e	2	2	2	2	**0.00**	2	2	2	**0.00**
esc32g	6	6	6	6	**0.00**	6	6	6	**0.00**
esc16a	68	68	68	68	**0.00**	68	68	68	**0.00**
esc16b	292	292	292	292	**0.00**	292	292	292	**0.00**
esc16c	160	160	160	160	**0.00**	160	160	160	**0.00**
esc16d	16	16	16	16	**0.00**	16	16	16	**0.00**
esc16e	28	28	28	28	**0.00**	28	28	28	**0.00**
esc16f	0	0	0	0	**0.00**	0	0	0	**0.00**
lipa20a	3683	3821	3829	3824	3.8	3683	3683	3683	**0.00**
lipa30a	13,178	13,614	13,625	13,618	3.3	13,178	13,178	13,178	**0.00**

The values shown in bold indicate that the optimal solution is reached for each instance.

the fitness of the new solution is evaluated according to Eq. (7) and a random solution is generated. Basing on the acceptance rate calculated by the loudness value A_i, the last evaluated solution is accepted, if improves the best global solution. Eventually, the best global solution is communicated to the next generation and the search process is progressed until the maximal number of iterations is reached.

The whole DBA in pseudo-code for optimizing the QAP is summarized in Algorithm 3.

Algorithm 3: Pseudo-code of discrete bat algorithm

1. Objective function $f(x)$, $x = (x_1, \ldots, x_d)^T$
2. Initialize the size of bat population.
3. Generate a random starting solution x_i for each bat.
4. Initialize the emission rates $r_i \in [0.0, 1.0]$ and the loudness $A_i \in [0.0, 1.0]$.
5. Define Pulse frequency $f_{\min}$ and $f_{\max}$ as integers in range $[1, 2]$.
6. **while** ($t <$ *Max number of iterations*)
7. **for** $i = 1$ to m
8. Generate new solution by adjusting frequency,
9. and updating velocity and location/solution // Equations (1), (10) and (11)
10. **if** ($rand > r_i$)
11. Select a solution among the best solutions of current iteration
12. Generate a local solution around the selected best solution by using the local search algorithm.
13. **end if**
14. Evaluate the new solution ($fitness_i = C(x_i)$) according to Equation (7) // x_i is current solution
15. Generate a new solution by flying randomly
16. **if** ($rand < A_i$ & $fitness_i < C(x_*)$) // x_* is global best solution
17. Accept the new solution
18. Increase r_i and reduce A_i // Equations (5) and (6)
19. **end if**
20. **end for**
21. Rank the bats and find the current global best x_*
22. **end while**
23. Post process results and visualization

6. Computational results and discussion

6.1. Results

The whole algorithmic approach was implemented in MATLAB R2010a and simulated with Intel(R) Core(TM) 2 Duo CPU T4300@ 2.1GHZ 800 MHz and 2.00 GB of RAM. The test instances were taken from the standard QAPLIB [59]. The size of the tested instances ranges from 12 to 100 and it is mentioned in the name of instance. For example, in the instance named nug30, the number 30 represents the number of provided facilities. Table 1 summarizes the parameters values of DBA algorithm. The choice of these parameters values is based on preliminary experiments by using a subset of QAPLIB benchmarks.

Furthermore, the pulse emission rate and loudness are two important parameters of DBA as well as standard BA and adjusting suitable values have a crucial effect on the quality of solutions. Fig. 4 indicates the best solution found of chr25a, nug30, tai20a, sko42, wil50 and sko81 when r_i and A_i are varying between 0 and 1. Fig. 5 presents the best solution found of chr25a, nug30, tai20a, sko42, wil50 and sko81 when r_i and A_i are equal to 0.5. The pre-experiments in Fig. 4 show that a small loudness value likewise a large emission rate value gives rise to quick convergence toward unsuccessful solutions. Based on Fig. 5, the experiments show that DBA can provide good results when the values of pulse emission rate and loudness are between 0.4 and 0.6. In order to fine-tune these two parameters and to achieve the desired performance, these values are fixed on 0.5.

Further, Table 2 summarizes the numerical results of DBA algorithm for 62 instances over 50 independent runs. The first column "Instance" shows the name of instance. The BKVs indicated in the column two present the best-known solution values reported from QAPLIB library and the values "Best" described in the third column are the optimal values found by DBA over 50 runs. The column "Worst" denotes the worst solutions found by DBA. The column "SD" indicates the standard deviation. The remaining columns indicate respectively the following performance measures:

- The percentage deviation of the best value found from the best-known value over 50 runs is calculated by the following formula:

$$\%\delta_{best} = \frac{C_{best} - C_*}{C_*} \times 100 \tag{12}$$

where C_{best} is the best objective value found over 50 runs and C_* is the best-known value taken from QAPLIB.

- The percentage deviation of the average solutions found from the best-known solution value over 50 runs is calculated by the next formula:

$$\%\delta_{avg} = \frac{C_{avg} - C_*}{C_*} \times 100 \tag{13}$$

where C_{avg} is the average objective function value over 50 runs.

The last column "$t_{avg}(s)$" is the average elapsed time in second during 50 runs.

Furthermore, Tables 3–6 report the results of the comparison between DBA and five other algorithms as extracted from their original papers. Besides the previous mentioned performance measures, in Table 3, the columns "I_{best}" indicate the number of iterations required to find the best solutions. The columns "*t(ms)*" present the computational time to reach the best solution expressed in milliseconds (ms). In Table 4, the columns "Average" denote the average cost of solutions found. In Table 6, the columns "Mean" represent the mean of solution cost (expressed as the average percentage deviation, which is the percentage by which the cost exceeds the BKV). Table 7 shows the average rank of each over the common problem instances for each pairing (16 problem instances for DBA vs SCX, 18 problem instances for DBA vs MBA, 23 problem instances for DBA vs HAS-QAP and 44 problem instances for DBA vs HPSO).

To determine significant differences between DBA and other algorithms, we applied the Wilcoxon signed rank test [60] (at the $\alpha = 0.05$ level) as a popular nonparametric statistical test. In this comparison, we used DBA as the control method and we considered the Holm-Bonferroni step-down method [61] to control the family-wise error rate [62], at the $\alpha = 0.05$ level. The Holm-Bonferroni step-down procedure is similar to the classical Bonferroni correction for multiple hypothesis testing. To describe Holm-Bonferroni step-down procedure, let $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_m$ be the ordered p values of the m individual tests and $H_1, H_2, \ldots, H_m$ the corresponding ordered hypotheses. Let k the smallest index that verifies

$$p_k > \frac{\alpha}{m + 1 - k} \tag{14}$$

If no such k exists then reject all the null hypotheses, otherwise the null hypotheses $H_1, \ldots, H_{k-1}$ are rejected and the null hypotheses $H_k, \ldots, H_m$ are not rejected. This method retains control the family-wise error rate at α [62]. These results are presented in Table 8 and denote that there is a significant difference in all three cases.

Table 5
Comparative results between SCX [25], HAS-QAP [16] and DBA for solving some instances from QAPLIB [59].

Instance	BKV	SCX		HAS-QAP		DBA	
		$\%\delta_{avg}$	t_{avg} (s)	$\%\delta_{avg}$	t_{avg} (s)	$\%\delta_{avg}$	t_{avg} (s)
tai20a	703,482	4.50	6	0.675	26	0.85	0.18
tai20b	122,455,319	6.56	6	0.0905	27	**0.00**	0.11
tai25a	1,167,256	4.58	9	1.189	50	1.51	12.10
tai25b	344,355,646	5.24	9	**0.00**	50	**0.00**	1.72
tai30a	1,818,146	4.29	13	1.311	87	0.72	20.25
tai30b	637,117,113	8.78	13	**0.00**	90	**0.00**	3.30
tai35a	2,422,002	4.95	18	1.762	145	1.79	35.43
tai35b	283,315,445	5.00	18	0.0256	147	0.19	20.01
tai40a	3,139,370	4.53	24	1.989	224	2.02	51.12
tai40b	637,250,948	7.30	24	**0.00**	240	**0.00**	14.86
tai50a	4,938,796	4.51	37	2.8	467	2.69	100
tai50b	458,821,517	5.60	37	0.1916	480	0.11	126.34
tai60a	7,205,962	4.54	54	3.070	820	2.73	166.23
tai60b	608,215,054	5.12	54	0.0483	855	0.09	221.65
tai80a	13,499,184	4.35	95	2.689	2045	2.67	420.62
tai80b	818,415,043	6.63	95	0.6670	2073	0.56	532.41
sko42	15,812	–	–	0.076	248	0.30	42.87
sko49	23,386	–	–	0.141	415	0.34	97.71
sko56	34,458	–	–	0.101	639	0.46	159.8
sko64	48,498	–	–	0.129	974	0.44	265.63
sko72	66,256	–	–	0.277	1415	0.46	361.56
sko81	90,998	–	–	0.144	2041	0.45	512.73
sko90	115,534	–	–	0.231	2825	0.48	653.61

The values shown in bold indicate that the optimal solution is reached for each instance.

Table 6
Comparative results between HPSO [22] and DBA for solving some instances from QAPLIB [59].

Instance	BKV	HPSO		DBA		Instance	BKV	HPSO		DBA	
		Mean	$\%\delta_{avg}$	Mean	$\%\delta_{avg}$			Mean	$\%\delta_{avg}$	Mean	$\%\delta_{avg}$
bur26a	5,426,670	1.9	0.22	**0.00**	**0.00**	tai12a	224,416	8.11	1.42	**0.00**	**0.00**
bur26b	3,817,852	1.96	0.41	**0.00**	**0.00**	tai12b	39,464,925	4.82	1.79	**0.00**	**0.00**
bur26c	5,426,795	2.3	0.24	**0.00**	**0.00**	tai15a	388,214	7.28	0.89	**0.00**	**0.00**
bur26d	3,821,225	2.18	0.42	**0.00**	**0.00**	tai15b	51,765,268	1.05	0.15	**0.00**	**0.00**
bur26e	5,386,879	2.14	0.42	**0.00**	**0.00**	tai20a	703,482	12.04	0.73	1.82	0.85
bur26f	3,782,044	2.39	0.59	**0.00**	**0.00**	tai20b	122,455,319	9.67	1.04	**0.00**	**0.00**
bur26g	10,117,172	2.07	0.22	**0.00**	**0.00**	tai25a	1,167,256	11.62	0.45	2.43	1.51
bur26h	7,098,658	2.27	0.4	**0.00**	**0.00**	tai25b	344,355,646	27.12	4.34	**0.00**	**0.00**
chr25a	3796	168.41	11.79	12.35	8.32	tai30a	1,818,146	12.25	0.41	2.24	1.34
els19	17,212,548	27.04	4.91	**0.00**	**0.00**	tai30b	637,117,113	25.18	3.28	0.04	0.02
kra30a	88,900	26.85	1.09	0.42	0.38	tai35a	2,422,002	13.19	0.38	3.02	1.79
kra30b	91,420	24.98	1.23	0.19	0.11	tai35b	283,315,445	27.86	2.42	0.22	0.19
wil50	48,816	8.69	0.26	1.53	0.14	tai40a	3,139,370	13.52	0.32	3.85	2.02
nug30	6124	16.85	0.79	0.21	0.21	tai40b	637,250,948	35.94	2.3	**0.00**	**0.00**
nug20	2570	11.65	0.95	**0.00**	**0.00**	tai50a	4,938,796	13.87	0.27	4.32	2.69
sko42	15,812	16.54	0.46	0.54	0.30	tai50b	458,821,517	34.88	1.9	0.15	0.11
sko49	23,386	15.52	0.42	0.70	0.34	tai60a	7,205,962	13.54	0.3	4.41	2.73
sko56	34,458	15.88	0.43	0.90	0.46	tai60b	608,215,054	36.50	1.28	0.09	0.09
sko64	48,498	14.68	0.42	0.68	0.44	tai80a	13,499,184	12.46	0.18	5.3	2.67
sko72	66,256	14.52	0.26	0.58	0.46	tai80b	818,415,043	34.48	0.94	0.83	0.56
sko81	90,998	13.98	0.28	0.47	0.45	tai100a	21,052,466	11.68	0.14	3.41	2.50
sko90	115,534	13.79	0.23	0.93	0.48	tai100b	1185996137	33.03	0.96	0.35	0.29

The values shown in bold indicate that the optimal solution is reached for each instance.

The *p*-values smaller than, or equal to, the corresponding Holm step-down threshold (last column) are statistically significant and interpreted accordingly. The statistical significance results are shown in boldface in ascending order according to *p*-value. These results reveal that the null hypothesis should be rejected in all three cases.

6.2. Discussion of results

As it can be observed from Table 2, the best-known solution has been found about 70% from all tested instances and nearly 89% of the values $\%\delta_{best}$ are less than 0.5%. In addition, the average deviation rate from best solutions of 0.00% value shows that the solutions found over 50 runs are similar to the best-known solutions. Therefore, we can discern from these experimental results that DBA is an efficient algorithm for solving easy instances likewise hard instances of QAP problem in a reasonable time.

For the compared instances in Table 3, DBA can reach the BKV for 14 out of the 16 cases with an average deviation of the best value found of 0.14; however, BeA only achieves the BKV for 8 instances over 16 tested instances with an average deviation of the best value found of 1.67. Another interesting result is that the number of iterations spend to find the best solution in BeA algorithm is significantly more than DBA, which increases the com-

Table 7
Comparison of average ranking and average computing time of five algorithms.

Comparison	# Instances	Rank	Time (s)
DBA vs SCX	16	**0.99**/5.40	107.89/**32**
DBA vs MBA	18	**0.00**/1.27	–/–
DBA vs HAS-QAP	23	0.82/**0.76**	**166.09**/712.30
DBA vs HPSO	44	**0.71**/1.18	–/–

The lowest value is shown in boldface.

Table 8
Results of Wilcoxon signed-rank tests (at the $\alpha = 0.05$ level), with the Holm Bonferroni step-down procedure used to control the family-wise error rate.

Comparison	# Instances	p-values	Holm
DBA vs HAS-QAP	23	**1.502E−15**	0.0167
DBA vs HPSO	44	**3.984E−05**	0.025
DBA vs SCX	16	**0.013**	0.05

puting time for BeA. It is clear from comparison that the proposed DBA is superior to BeA in terms of solutions quality and computational complexity. According to the results displayed in Table 4, we can observe that DBA achieves the optimal solution for all instances with higher success rate of 100%. From Table 7, we observe that DBA has an average rank of 0.00, whereas MBA has an average rank of 1.27. These results show the superiority of the proposed DBA in terms of the solution quality compared to MBA. From Tables 5, 7, and 8, we can make the following observations.

For the first sixteen problems, DBA finds the best-known solution for four instances with averages rank of 0.99 and computing time of 107.89 versus averages rank of 5.40 and computing time of 32 for SCX. Comparing the two algorithms, DBA statistically outperforms SCX (with p-value < 0.05) over 16 instances. Comparing DBA to HAS-QAP, we observe that HAS-QAP shows best results than both DBA, but it requires higher computing budgets. However, the difference between the two methods is statistically significant. From these results, it can be inferred that DBA offers a good compromise between the execution time and the solution quality and it does remain very competitive with SCX and HAS-QAP.

From Tables 6, 7, we observe that DBA is able to attain the BKV for 17 out of 44 instances with an average rank of 0.71, whereas HPSO has an average rank of 1.18 with no BKV reached. In addition, Table 8 indicates that the difference between DBA and HPSO is statistically significant. From these results, we find that DBA is most effective approaches to solve QAP. We conclude that DBA competes favorably with HPSO on the most instances.

7. Conclusions

In this paper, a swarm-intelligence algorithm, so-called Bat Algorithm (BA), was introduced to solve the Quadratic Assignment Problem (QAP). More specifically, a discrete version of BA was proposed, namely Discrete Bat Algorithm (DBA) for QAP. One of the main contributions of this paper was to introduce some new modifications in the original version of BA. These modifications regards position representation and its update equation, and velocity representation and its update equation. Besides that, a modified uniform crossover was incorporated with BA to enhance the search strategy and the 2-exchange neighborhood mechanism was used to improve local search capability. The proposed algorithm was applied on a set of benchmark instances taken from QAPLIB library. It has been shown that the effectiveness as well as performance of DBA has reached that of the other techniques such as BeA, MBA, SCX, HAS-QAP and HPSO in terms of the quality of solutions and the computing time. Our future research will be mainly focused on the application of DBA algorithm in other combinatorial optimization problems such as scheduling models, logistic network models, vehicle routing models, etc. We will also try to enhance the current algorithm in order to obtain even better results by involving hybrid approaches and by using other discretization methods to switch between the continuous and the discrete search space.

References

[1] Papadimitriou CH, Steiglitz K. Combinatorial optimization: algorithms and complexity. Courier Dover Publications; 1998.
[2] Wong W. Matrix representation and gradient flows for NP-hard problems. J Optim Theory Appl 1995;87:197–220.
[3] Koziel S, Yang X-S. Computational optimization, methods and algorithms. Springer; 2011.
[4] Fister Jr. I, Yang X-S, Fister I, Brest J, Fister D. A brief review of nature-inspired algorithms for optimization, arXiv preprint arXiv:1307.4186; 2013.
[5] Cagnina LC, Esquivel SC, Coello CAC. Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 2008;32.
[6] Kumar S, Kumari R, Sharma VK. Fitness based position update in spider monkey optimization algorithm. Proc Comput Sci 2015;62:442–9.
[7] Gandomi AH, Yang X-S, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 2013;29:17–35.
[8] Lučić P, Teodorović D. Computing with bees: attacking complex transportation engineering problems. Int J Artif Intell Tools 2003;12:375–94.
[9] Huang M, Yuan J, Xiao J. An adapted firefly algorithm for product development project scheduling with fuzzy activity duration. Math Problems Eng 2015;2015.
[10] Yang X-S. A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer; 2010. p. 65–74.
[11] Yılmaz S, Küçüksille EU. A new modification approach on bat algorithm for solving optimization problems. Appl Soft Comput 2015;28:259–75.
[12] Sahni S, Gonzalez T. P-complete approximation problems. J ACM (JACM) 1976;23:555–65.
[13] Wilhelm MR, Ward TL. Solving quadratic assignment problems by 'simulated annealing'. IIE Trans 1987;19:107–19.
[14] Taillard E. Robust taboo search for the quadratic assignment problem. Parallel Comput 1991;17:443–55.
[15] Bousono-Calzon C, Manning M. The hopfield neural network applied to the quadratic assignment problem. Neural Comput Appl 1995;3:64–72.
[16] Gambardella LM, Taillard E, Dorigo M. Ant colonies for the quadratic assignment problem. J Operational Res Soc 1999:167–76.
[17] Merz P, Freisleben B. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. Evolutionary Comput, IEEE Trans 2000;4:337–52.
[18] Misevicius A. An improved hybrid genetic algorithm: new results for the quadratic assignment problem. Knowl-Based Syst 2004;17:65–73.
[19] Stützle T. Iterated local search for the quadratic assignment problem. Eur J Oper Res 2006;174:1519–39.
[20] Tseng L-Y, Liang S-C. A hybrid metaheuristic for the quadratic assignment problem. Comput Optim Appl 2006;34:85–113.
[21] Ahuja RK, Jha KC, Orlin JB, Sharma D. Very large-scale neighborhood search for the quadratic assignment problem. Informs J Comput 2007;19:646–57.
[22] Helal AM, Abdelbar AM. Incorporating domain-specific heuristics in a particle swarm optimization approach to the quadratic assignment problem. Memetic Comput 2014;6:241–54.
[23] Shukla A. A modified bat algorithm for the quadratic assignment problem. In: Evolutionary computation (CEC), 2015 IEEE congress on. IEEE; 2015. p. 486–90.
[24] Chmiel W, Szwed P. Bees algorithm for the quadratic assignment problem on CUDA platform. Man-machine interactions, vol. 4. Springer; 2016. p. 615–25.
[25] Ahmed Z. A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem. J Sci Ind Res 2014;73:763–6.
[26] Yang X-S, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. Eng Comput 2012;29:464–83.
[27] Yang X-S, He X. Bat algorithm: literature review and applications. Int J Bio-Inspired Comput 2013;5:141–9.
[28] Gandomi AH, Yang X-S. Chaotic bat algorithm. J Comput Sci 2014;5:224–32.
[29] Cai X, Wang L, Kang Q, Wu Q. Bat algorithm with Gaussian walk. Int J Bio-inspired Comput 2014;6:166–74.
[30] Khan K, Nikov A, Sahai A. A fuzzy bat clustering method for ergonomic screening of office workplaces. In: Dicheva D, Markov Z, Stefanova E, editors. Third international conference on software, services and semantic technologies S3T 2011. Berlin Heidelberg: Springer; 2011. p. 59–66.
[31] Nguyen T-T, Pan J-S, Dao T-K, Kuo M-Y, Horng M-F. Hybrid bat algorithm with artificial bee colony. In: Pan J-S, Snasel V, Corchado ES, Abraham A, Wang S-L, editors. Intelligent data analysis and its applications, vol. II. Springer International Publishing; 2014. p. 45–55.

[32] Fister I, Fong S, Brest J, Fister I. A novel hybrid self-adaptive bat algorithm. Sci World J 2014;2014.
[33] Mirjalili S, Mirjalili S, Yang X-S. Binary bat algorithm. Neural Comput Appl 2014;25:663–81.
[34] Chen Y-T, Shieh C-S, Horng M-F, Liao B-Y, Pan J-S, Tsai M-T. A guidable bat algorithm based on doppler effect to improve solving efficiency for optimization problems. In: Hwang D, Jung J, Nguyen N-T, editors. Computational collective intelligence. Technologies and Applications, (Springer International Publishing; 2014. p. 373–83.
[35] Nakamura RY, Pereira LA, Costa K, Rodrigues D, Papa JP, Yang X-S, BBA: a binary bat algorithm for feature selection, graphics, patterns and images (SIBGRAPI), 2012 25th SIBGRAPI conference on, (IEEE2012), p. 291–7.
[36] Xie J, Zhou Y, Tang Z. Differential Lévy-flights bat algorithm for minimization makespan in permutation flow shops. In: Huang D-S, Jo K-H, Zhou Y-Q, Han K, editors. Intelligent computing theories and technology. Berlin Heidelberg: Springer; 2013. p. 179–88.
[37] Luo Q, Zhou Y, Xie J, Ma M, Li L. Discrete bat algorithm for optimal problem of permutation flow shop scheduling. Sci World J 2014;2014.
[38] Sabba S, Chikhi S. A discrete binary version of bat algorithm for multidimensional knapsack problem. Int J Bio-Inspired Comput 2014;6:140–52.
[39] Büyüksaatçı S. Bat algorithm application for the single row facility layout problem. In: Yang X-S, editor. Recent advances in swarm intelligence and evolutionary computation. Springer International Publishing; 2015. p. 101–20.
[40] Fister I, Rauter S, Yang X-S, Ljubič K. Planning the sports training sessions with the bat algorithm. Neurocomputing 2015;149:993–1002.
[41] Dekhici L, Belkadi K. A bat algorithm with generalized walk for the two-stage hybrid flow shop problem. Int J Decision Support Syst Technol (IJDSST) 2015;7:1–16.
[42] Hasançebi O, Carbas S. Bat inspired algorithm for discrete size optimization of steel frames. Adv Eng Softw 2014;67:173–85.
[43] Carbas S, Hasancebi O. Optimum design of steel space frames via bat inspired algorithm. In: 10th World congress on structural and multidisciplinary optimization; 2013. p. 1–10.
[44] Hassan EA, Hafez AI, Hassanien AE, Fahmy AA. A discrete bat algorithm for the community detection problem. In: Hybrid artificial intelligent systems. Springer; 2015. p. 188–99.
[45] Saji Y, Riffi ME, Ahiod B. Discrete bat-inspired algorithm for travelling salesman problem, Complex Systems (WCCS). In: 2014 second world conference on; 2014. p. 28–31.
[46] Saji Y, Riffi M. A novel discrete bat algorithm for solving the travelling salesman problem. Neural Comput Appl 2015:1–14.
[47] Koopmans TC, Beckmann M. Assignment problems and the location of economic activities. Econ: J Econ Soc 1957:53–76.
[48] Pardalos PM, Wolkowicz H. Quadratic assignment and related problems: dimacs workshop. American Mathematical Soc.; 1994.
[49] Tosun U, Dokeroglu T, Cosar A. Heuristic algorithms for fragment allocation in a distributed database system. In: Computer and information sciences III. Springer; 2013. p. 401–8.
[50] Chen J, Xin B, Peng Z, Dou L, Zhang J. Evolutionary decision-makings for the dynamic weapon-target assignment problem. Sci China Ser F: Inf Sci 2009;52:2006–18.
[51] Misevicius A. An implementation of the iterated tabu search algorithm for the quadratic assignment problem. OR Spectrum 2012;34:665–90.
[52] Vazquez M, Whitley LD. A hybrid genetic algorithm for the quadratic assignment problem, GECCO, (Citeseer 2000), p. 135–42.
[53] Misevičius A, Rubliauskas D. Performance of hybrid genetic algorithm for the grey pattern problem. Inf Technol Control 2005;34:15–24.
[54] Ahuja RK, Orlin JB, Tiwari A. A greedy genetic algorithm for the quadratic assignment problem. Comput Oper Res 2000;27:917–34.
[55] Migkikh VV, Topchy AP, Kureichik VM, Tetelbaum AY, Combined genetic and local search algorithm for the quadratic assignment problem. In: Proceedings of IC on evolutionary computation and its applications, EvCA, vol. 96; 1996. p. 335–41.
[56] Drezner Z. A new genetic algorithm for the quadratic assignment problem. INFORMS J Comput 2003;15:320–30.
[57] Lim M, Yuan Y, Omatu S. Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. Comput Optim Appl 2000;15:249–68.
[58] Tate DM, Smith AE. A genetic approach to the quadratic assignment problem. Comput Oper Res 1995;22:73–83.
[59] Burkard RE, Karisch SE, Rendl F. QAPLIB – a quadratic assignment problem library. J Global Optim 1997;10:391–403.
[60] Wilcoxon F. Individual comparisons by ranking methods. Biometrics Bull 1945;1:80–3.
[61] Holm S. A simple sequentially rejective multiple test procedure. Scand J Stat 1979:65–70.
[62] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolutionary Comput 2011;1:3–18.