Neural Comput & Applic
DOI 10.1007/s00521-015-1978-9

ORIGINAL ARTICLE

A novel discrete bat algorithm for solving the travelling salesman problem

Yassine Saji[1] · **Mohammed Essaid Riffi**[1]

Received: 25 November 2014 / Accepted: 10 June 2015

Abstract The travelling salesman problem (TSP) is one of the well-known NP-hard combinatorial optimization and extensively studied problems in discrete optimization. The bat algorithm is a new nature-inspired metaheuristic optimization algorithm introduced by Yang in 2010, especially based on echolocation behavior of microbats when searching their prey. Firstly, this algorithm is used to solve various continuous optimization problems. In this paper we extend a discrete bat-inspired algorithm to solve the famous TSP. Although many algorithms have been used to solve TSP, the main objective of this research is to investigate this discrete version to achieve significant improvements, not only compared to traditional algorithms but also to another metaheuristics. Moreover, this study is based on a benchmark dataset of symmetric TSP from TSPLIB library.

Keywords Travelling salesman problem · NP-hard combinatorial optimization problem · Nature-inspired metaheuristic · Discrete bat-inspired algorithm

1 Introduction

Metaheuristics and especially nature-inspired ones have been the most applicable algorithms in combinatorial optimization problems over the last three decades; this is due to the importance of combinatorial optimization in the scientific world as well as the industrial world; to that end, several scientific studies are conducted about the most successful processes in nature, including biological systems, physical, and chemical processes such as natural evolution, swarm intelligence, annealing process, in a numerical algorithm [1–5]. Most combinatorial optimization problems are NP-hard [6–8], and the optimal or even suboptimal solution is typically difficult to be found. The metaheuristic method provides many advantages over an exact method. In some practical applications, the search of an optimal solution can be completely inappropriate due to many factors: the size of problem, the dynamic of work environment, the lack of precision in data collection, the difficulty in formulating some constraints or the presence of conflicting objectives, and sometimes a large number of real problems are not optimizable effectively by purely mathematical approaches. In this case, using an exact method has been often much slower than metaheuristic, which can cause additional costs in computation time. Nevertheless, nature-inspired algorithms still pose a real challenge in the sense to solve realistically large problem instances in reasonable computation time, such as ant colony optimization (ACO) [9], particle swarm optimization (PSO) [10, 11], bee colony optimization [12], firefly algorithm (FA) [13], cuckoo search (CS) [14], and very recently, bat algorithm (BA) [15]; it is, mainly, applied in continuous optimization problems [16], that is inspired by echolocation behavior of bats in searching for the optimum.

The travelling salesman problem (TSP) is one of the well-known and extensively studied problems in discrete or classical combinatorial optimization. The goal of this problem is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. TSP is known to be a NP-hard problem [17], it is easy to describe but very difficult to solve. For a symmetric TSP

✉ Yassine Saji
yassine.saji@gmail.com

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

Published online: 26 June 2015

Table 1 The computation time estimated if one tour requires 1 ns

Number of cities	Number of tours	Time in years
20	6.08E+16	2
25	3.1E+23	9.84E+6
30	4.4E+30	1.4E+14
35	1.48E+38	4.68E+21
40	1.02E+46	3.23E+29

with n cities, there are $(n-1)!/2$ possible tours, the easiest way to find a solution is to seek the possibility of all existing paths then choose the shortest one, as a result the computational time complexity of this algorithm will be exponential to the size of n given cities in factorial time $O(n!)$. Table 1 shows the time required to calculate tours if one tour evaluated in 1 ns (10^{-9} s).

The TSP has wide applicability in many industrial applications [18–21] such as analysis of the structure of crystals in X-ray crystallography, computer wiring and connecting components on a computer board, overhauling gas turbine engines, the order-picking problem in warehouses, the vehicle routing problem and scheduling problems, and many others. In real life, the process of finding a solution for this kind of problems is already in progress and attracts many researchers to investigate great efforts in developing novel methods.

Many studies have been devoted to solve TSP such as Tabu search (TS) [22], simulated annealing (SA) [23], genetic algorithms (GA) [24, 25], and greedy randomized adaptive search procedure (GRASP) [26]. Recently, in the last 15 years, several studies based on nature-inspired algorithms or swarm intelligence methods have been proposed for the solution of the TSP, such as ACO [27–30], PSO [31–33], CS [34], and many more metaheuristic algorithms. BA is a new metaheuristic developed by Yang in 2010: It is inspired from bats echolocation system used to find their prey. This novel algorithm was originally developed for solving continuous optimization problems, and typically, BA has proven to be effective in solving this field of optimization. The main aim of this paper is to introduce a new discrete variant of bat algorithm (DBA) to solve symmetric travelling salesman problem (STSP) and to test its efficiency compared to other nature-inspired and classic metaheuristic algorithms.

The remaining part of this paper is organized as follows: Sect. 2 provides a brief literature review of swarm intelligence algorithms and especially of BA. Section 3 introduces the standard BA. Section 4 describes briefly the TSP. Section 5 proposes a discrete bat algorithm (DBA) to solve STSP. Section 6 presents a set of experimental results of symmetric TSP benchmarks from TSPLIB library [35]. Finally, Sect. 7 concludes and gives some perspectives of research.

2 Literature review

In the last decade, the swarm intelligence algorithms have been emerged as a powerful method to solve various engineering optimization problems [6]. The vast majority of algorithms have been suggested depending on different intelligent behaviors of swarms or by mimicking the behaviors of biological systems in nature. Day after day, the number of researchers interested in swarm intelligence algorithms increases rapidly. This is a subject on which there is an ample literature. Dorigo et al. [27] proposed ACO by inspiration from the natural behavior of ant species. Karaboga and Basturk [36] developed artificial bee colony (ABC) by simulating the intelligent foraging behavior of honeybee swarm. Kennedy and Eberhart [37] proposed PSO by inspiration from the social behavior of bird flocking when searching for food. Yang proposed FA based on the flashing patterns of tropical fireflies. Passino [38] introduced bacterial foraging optimization algorithm (BFO) by observing the social foraging behaviors of *E. coli* bacterial swarm. Krishnan and Ghose [39] proposed glowworm swarm optimization algorithm (GSO) inspired by the luminescence capability of glowworms. Chu et al. [40] developed cat swarm optimization algorithm (CSO) by observing the seeking and tracing mode of cats. Gandomi and Alavi [41] proposed Krill Herd (KH) based on the simulation of the herding behavior of krill individuals. Gheraibia and Moussaoui [42] proposed penguins search optimization algorithm (PeSOA) based on collaborative hunting strategy of penguins. Meng et al. [43] inspired by the hierarchal order in the chicken swarm and the behaviors of the chicken swarm, introduced Chicken swarm optimization algorithm (CSOa). Bansal et al. [44] proposed a new swarm approach named spider monkey optimization algorithm (SMO) by modeling the foraging behavior of spider monkeys.

The BA is one of the newest swarm-intelligence-based algorithms proposed by Yang in 2010 [15], based on the intelligent echolocation behavior of bats when catching their prey. In recent years, the BA has become a popular bio-inspired algorithm and one can easily understand that the BA is applied to solve a large variety of optimization problems, while many researchers have provide a wide range of contributions to the literature. Gandomi et al. [45] focused on solving constrained optimization tasks by using BA. Khan et al. [46] proposed a fuzzy modification of BA for clustering of company workplaces. Tamiru and Hashim [47] used fuzzy systems for modeling energy destructions in the components of an industrial gas turbine. Yılmaz and

küçüksille [48] suggested three different methods to enhance the local and global search of BA, and they employed both standard test functions and constrained real-world problems to validate the performance of their approaches. Nguyen et al. [49] hybridized BA with ABC algorithm and four benchmark functions are used to test the convergence of the proposed method. Pan et al. [50] applied the concepts of parallel processing and communication strategy to hybrid PSO with BA and six benchmark functions are tested to validate the approached method. Wang et al. [51] employed a Gaussian walk with BA to improve the local search capability, and they modified the velocity equation to assure a good exploitation. Gandomi and Yang [52] introduced chaos mechanism into BA to improve the global search mobility of BA and optimized a set of benchmark problems with different chaotic maps. Mirjalili el al. [53] proposed a binary BA for unimodal, multimodal, and composite functions. Besides that, BA was applied to solve many other optimization problems which more detailed in [54]. However, the application of the standard BA algorithm to solve discrete problems remains limited until 2012, when Nakamura et al. [55] introduced the first binary version of BA to solve feature selection problems. A few years ago, some recent research focused on BA in combinatorial optimization problems, such as Xie et al. [56] introduced a differential Lévy-flights BA to minimize the permutation flow shop problem. Sabba and Chikhi [57] used the sigmoid function to propose a discrete binary BA for solving the multidimensional knapsack problem. Büyüksaatçı [58] used BA to solve the single-row facility layout problem. Fister et al. [59] proposed a modified BA for planning the sports training sessions, etc.

3 Bat-inspired algorithm

Recently, a new metaheuristic search algorithm is presented by Yang in 2010 [15], called the bat-inspired algorithm or BA [60]. The bat-inspired search is based on the echolocation behavior of bats to find the prey and discriminate different types of insects even in complete darkness with varying pulse rates of emission and loudness. They achieve this by emitting calls out to the environment and listening to the echoes that bounce back from them. They are able to identify location of other objects and to measure the distance from the targets by following delay of the returning sound. These emitting calls are very loud sound pulses that vary in properties according to their hunting strategies and depend on the species. The echolocation pulses are characterized by three features: pulse frequency, pulse emission rate, and loudness or intensity. The bats emit echolocation pulses with varying frequencies between 25 and 150 kHz depending on proximity of the target [15]. The pulse rate corresponds to the number of pulses emitted per second, and it can also be adapted by bats according to how far they are away from the target object and this pulse rate can be sped up to 200 pulses per second when approaching a target. Finally, bats decrease the intensity (loudness) of pulse from 120 (loudest) to 50 dB (quietest) as they come closer to their prey. Yang in [15] idealized echolocation behavior of bats and its associated parameters in a numerical optimization algorithm. The BA algorithm has been empirically tested and compared with other existing algorithms using some single and multi-objective standard functions of unconstrained optimization in [15, 60]. Furthermore, Yang and Gandomi [16, 45] validated the performance of BA in benchmark problems of constrained engineering optimization. These studies have clearly demonstrated a better efficiency of the bat-inspired algorithm over other methods.

The basic BA developed by Xin-She Yang [15] is described in following steps:

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. 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
3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

Algorithm 1: Pseudo-code of bat algorithm (BA)

1. Objective function $f(x)$, $x = (x_1, ..., x_d)^T$
2. Initialize the bat population x_i $(i = 1,2, ...,n)$ 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. Generate new solutions by adjusting frequency,
7. and updating velocities and locations/solutions [equations (1) to (3)]
8. **if** ($rand > r_i$)
9. Select a solution among the best solutions
10. Generate a local solution around the selected best solution
11. **end if**
12. Generate a new solution by flying randomly
13. **if** ($rand < A_i$ & $f(x_i) < f(x_*)$)
14. Accept the new solutions
15. Increase r_i and reduce A_i
16. **end if**
17. Rank the bats and find the current best x_*
18. **end while**
19. Post process results and visualization

First, initializing bat population: position x_i, velocity v_i and frequency f_i. The movement of the virtual bats is given by updating their velocities v_i^t and positions x_i^t at time step t using Eqs. 1–3, as follows:

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

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

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

where $\beta \in [0, 1]$ denotes a randomly generated vector from uniform distribution.

The variable x_* denotes the current global best location (solution), which is located after comparing all the solutions provided by the m bats.

Second, a random number is applied; if this random number verifies the condition in *Line 8* of *Algorithm 1* and after selection of one solution among the current best solutions of bat, a new solution will be accepted around the current best solutions; it can be represented by:

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

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

Third, the loudness A_i and the pulse emission rate r_i will be updated, and a solution will be accepted if a random number is less than loudness A_i and $f(x_i) < f(x_*)$. A_i and r_i are updated by:

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

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

where α, γ are constants and $f(\cdot)$ is fitness function. The algorithm repeats until maximal number of cycles is reached.

4 The travelling salesman problem

The travelling salesman problem was introduced in 1800s by the Irish mathematician W.R. Hamilton and the British mathematician Thomas Kirkman; it can be stated as a permutation problem with the objective of finding a shortest closed tour which visits all the cities in a given set; TSP can be modeled as completely connected graph in a D-dimensional Euclidean space (D is size of the problem), which cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's length. Mathematically, it can be defined as given a set of n cities, named $\{c_1, c_2, \ldots, c_n\}$, and permutations $\pi_1, \ldots, \pi_n$, the goal is to find a number of permutation $\pi_i = \{c_1, c_2, \ldots, c_n\}$, such that minimizes $f(\pi)$ the sum of all Euclidean distances d between each city from the same path π and it is given as follows:

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \tag{7}$$

The Euclidean distance d, between any two cities with coordinate (x_1, y_1) and (x_2, y_2) is calculated by:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{8}$$

Broadly, the TSP is classified as STSP. In this symmetric TSP variation, all of the edge costs are symmetric. This means that, for all vertices in the graph, the cost incurred, when moving from vertex i to vertex j, is the same as the cost incurred when moving from vertex j to vertex i, i.e., $d_{ij} = d_{ji}$.

5 Discrete bat algorithm for STSP

Generally, the optimization problems can be divided into two main classes, continuous and discrete [61, 62]. An optimization problem with real decision variables is known as a continuous optimization problem, whereas it is called discrete optimization problem if decision variables take discrete (usually integer) values. Moreover, in discrete problems, the set of feasible solutions is discrete or can be reduced to a discrete one by the discretization of the continuous space. Basically, the BA has been developed to optimize continuous nonlinear functions [15, 16] in which, each bat moves in search space toward continuous valued position, but many problems are, however, defined for discrete valued spaces where the domain of the variables is finite. Typically, many bio-inspired population-based

algorithms have been shown a good efficiency in solving continuous problems as well as to solve discrete problem like BA [15], CS [14], and PSO [37]. In the original versions of these algorithms, it is impossible to exploit them directly to solve discrete problems; So many researchers have modified these last algorithms to deal with discrete problems and we can cite some of them: binary bat algorithm (BBA) [55], discrete cuckoo search (DCS) [34], and discrete binary particle swarm optimization (DBPSO) [63].

In this paper, we propose a novel DBA to solve especially the symmetric TSP problem. The DBA saves the original concept and the same steps of the basic BA algorithm, but modifies some equations to shift to discrete space. So the following subsections describe briefly the various steps of extending version of BA to solve TSP.

5.1 Position and velocity representation

In the literature, there are many representations to define a TSP position (solution) in search space, i.e., path representation, adjacency representation, ordinal representation, matrix representation, and binary representation [64]. In basic BA, the position x_i of each virtual bat defines a potential solution to problem; to find the best solution, each bat adjusts its velocity by randomly selecting frequency f_i of sonic wave. On the other hand, each bat uses the pulse emission rate r_i and loudness A_i to control the intensive local search that is processed to generate a new individual around the current global best solution. So, in this study, the solution of n cities found by the ith bat is represented by a n-dimensional vector $x_i = (x_{i1}, x_{i2},\ldots, x_{in})$. The velocity v_i is viewed as a set of permutation $\pi_i = \{c_1, c_2, \ldots, c_n\}$, which allows being close to the global best solution $x_* = (x_{*1}, x_{*2},\ldots, x_{*n})$.

5.2 Position updating equation

In continuous BA, the bats' movement is calculated by the three previous Eqs. 1–3. By using Eqs. 1 and 2, each bat updates its velocity. The resulting value is applied in Eq. 3 to calculate the next position of each individual. However, in order to apply BA to solve TSP, these equations cannot be used directly and must be adapted to the problem.

In the same way as the standard BA, each bat selects a frequency f_i in the range of frequency $[f_{\min}, f_{\max}]$, where $f_{\min}$ and $f_{\max}$ are two integers in the range $[1, n]$, n denotes the number of cities. The frequency f_i denotes the number of cities of sub-tour saved from the current solution x_i^t when this last one crosses with the current global best solution x_*^t. The velocity v_i^t consists of a set of permutations that allow crossing two solutions x_i^t and x_*^t, while

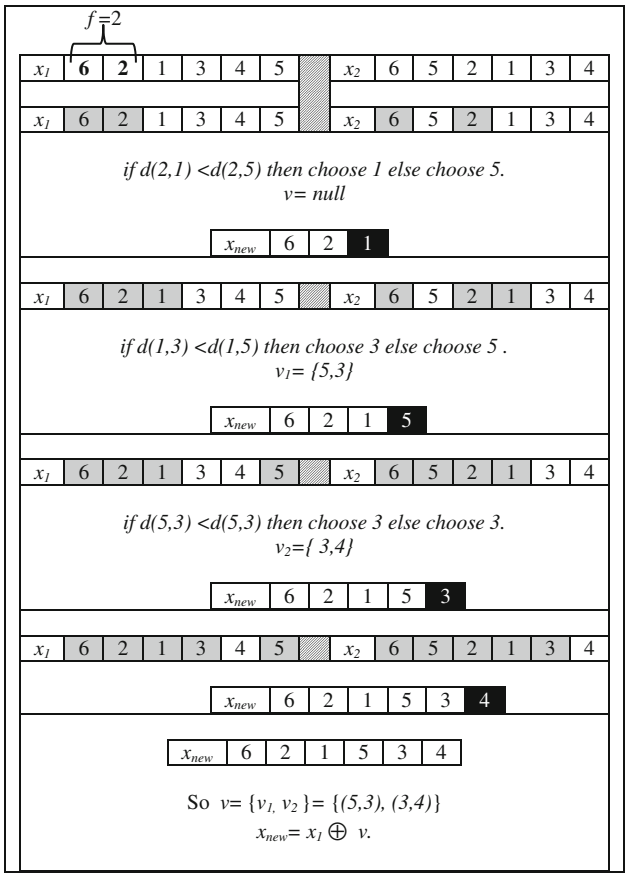

Fig. 1 An example of 2-exchange crossover heuristic for TSP

respecting 2-exchange crossover heuristic algorithm as described in [65].

The 2-exchange crossover mechanism can be illustrated by an example in Fig. 1. In this example, we take the frequency $f = 2$ and the distance matrix D defined as below. For this example, we consider two solutions x_1, x_2 and we want to cross them together to get a new solution. In the beginning, we start by saving the f first cities from x_1 and mark it as already assigned in x_2. Next, the new solution is initialized by the cities saving in x_1. After this, we continue to concatenate the last city of the new solution by the closest city either from x_1 or x_2. During the construction process of the new solution, it is necessary to ensure that the two candidate cities chosen from x_1 and x_2 not be already marked in the new solution.

$$D = \begin{pmatrix} 0 & . & 11 & . & 4 & . \\ 5 & 0 & . & . & 12 & 1 \\ . & 3 & 0 & 11 & . & . \\ 9 & 12 & . & 0 & 7 & . \\ . & . & 1 & . & 0 & 8 \\ 4 & . & 2 & 5 & . & 0 \end{pmatrix}$$

5.3 The neighborhood search

The definition of a neighborhood is important for combinatorial problems as well as continuous problems. Typically, neighborhood search or local search procedure is frequently used for iteratively improving a solution. In the context of the TSP, that means to search the better solution in the neighborhood of the existing solution by making the minimum changes on the last one. The most popularly local search procedure is *2-opt*, where two edges are exchanged iteratively until no further improvement is possible as showing in Fig. 2, in which (A) represents initial tour and (B) new tour. The tour (B) is created by taking two pairs of consecutive nodes, pairs (c, d) and (a, b) from the tour (A) and checking if the distance ($cd + ab$) is higher than ($cb + ad$), if that is the case, then exchange nodes a and c in pairs (c, d) and (a, b).

5.4 Discrete bat algorithm

This subsection describes briefly the various steps of extending version of BA to solve TSP and recalls the basic terminology used in the previous subsections. Steps of the proposed algorithm DBA are presented below, and the DBA algorithm is summarized in *Algorithm 2*.

Step 1: Initialize the size of bat population. Generate a random starting position x_i for each bat. Initialize the emission rates $r_i \in [0.0, 1.0]$ and the loudness $A_i \in [0.0, 1.0]$. Define pulse frequency $f_{\min}$ and $f_{\max}$ in range $[1, n]$.

Step 2: Calculate the global best solution.

Step 3: For each bat, generate new solutions by adjusting frequency Eq. 1, and updating velocities and locations/solutions, by using the following equations:

$$v_i^t = \chi\left(x_i^{t-1}, x_*, f_i\right), \tag{9}$$

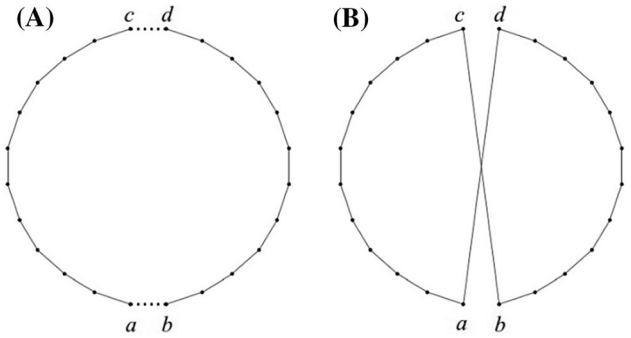

Fig. 2 *2-Opt* move: **a** original tour and **b** resulting tour

Table 2 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, $(f_{\min} \in \mathbb{N}/1 \leq f_{\min} \leq n)$
Maximal frequency $f_{\max}$	5, $(f_{\max} \in \mathbb{N}/f_{\min} < f_{\max} \leq n)$
Maximal number of iterations $t_{\max}$	200

$$x_i^t = \phi\left(x_i^{t-1}, v_i^t\right), \tag{10}$$

where the function χ(cross) takes three input arguments (two solutions and one integer) and returns a set of permutations reached by applying 2-exchange crossover mechanism described in Sect. 5.2. The function ϕ(sort) returns a new solution obtained by sorting the elements of x_i^{t-1} according to the permutations v_i^t.

Step 4: A random number is applied; if it is upper to r_i, then select a solution among the best solutions. Generate a local solution by exchanging two arcs from the selected best solution.

Step 5: Evaluate the new solution according to the Eqs. 7, 8.

Step 6: Generate a new solution by flying randomly.

Step 7: Generate a random number, if this number is less than loudness A_i and the last evaluated solution is better than the best solution so accept the new solutions.

Step 8: Stop the algorithm if the maximal number of iterations is reached. Return to Step 3 otherwise.

Algorithm 2: Pseudo-code of discrete bat algorithm (DBA)

1. Objective function $f(x)$, $x = (x_1, \ldots, x_d)^T$
2. Initialize the size of bat population.
3. Generate a random starting position 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 $f_{\max}$ $(f_{\max} > f_{\min})$ as integers in range $[1,n]$.
6. **while** ($t <$ *Max number of iterations*)
7. Generate new solutions by adjusting frequency,
8. and updating velocities and locations/solutions [Eqs (1) , (9) and (10)]
9. **if** ($rand > r_i$)
10. Select a solution among the best solutions
11. Generate a local solution by exchanging 2 arcs from the selected best solution.
12. **end if**
13. Evaluate the new solution ($fitness_i = f(x_i)$) according to the Eqs. (7, 8).
14. Generate a new solution by flying randomly.
15. **if** ($rand < A_i$ & $fitness_i < f(x_*)$)
16. Accept the new solutions
17. Increase r_i and reduce A_i
18. **end if**
19. Rank the bats and find the current best x_*
20. **end while**
21. Post process results and visualization

Table 3 Results of the DBA algorithm for symmetric instances from TSPLIB

Instance	Optimal	Best	Worst	Average	Std	PDav(%)	PDbest (%)	$C_{1\%}/C_{opt}$	Time(s)
eil51	426	**426**	**426**	**426**	**0.00**	**0.00**	**0.00**	**30/30**	0.20
berlin52	7542	**7542**	**7542**	**7542**	**0.00**	**0.00**	**0.00**	**30/30**	0.03
st70	675	**675**	**675**	**675**	**0.00**	**0.00**	**0.00**	**30/30**	0.43
pr76	108,159	**108,159**	**108,159**	**108,159**	**0.00**	**0.00**	**0.00**	**30/30**	0.57
eil76	538	**538**	542	538.76	1.16	0.14	**0.00**	30/19	1.54
kroA100	21,282	**21,282**	**21,282**	**21,282**	**0.00**	**0.00**	**0.00**	**30/30**	1.36
kroB100	22,141	**22,141**	**22,141**	**22,141**	**0.00**	**0.00**	**0.00**	**30/30**	3.35
kroC100	20,749	**20,749**	20,880	20,753.36	23.51	0.02	**0.00**	30/29	2.51
kroD100	21,294	**21,294**	21,374	21,303.50	23.92	0.04	**0.00**	30/24	7.55
kroE100	22,068	**22,068**	22,140	22,080.76	21.98	0.05	**0.00**	30/22	11.12
eil101	629	**629**	637	632.43	2.77	0.54	**0.00**	24/5	17.09
lin105	14,379	**14,379**	**14,379**	**14,379**	**0.00**	**0.00**	**0.00**	**30/30**	2.27
pr107	44,303	**44,303**	44,482	44,360.8	56.57	0.13	**0.00**	30/11	18.01
pr124	59,030	**59,030**	59,076	59,037.66	17.14	0.012	**0.00**	30/25	2.57
bier127	118,282	**118,282**	118,693	118,385.66	155.66	0.08	**0.00**	30/20	19.14
ch130	6110	**6110**	6155	6124.1	8.51	0.23	**0.00**	30/4	13.68
pr136	96,772	**96,772**	97,468	96,995	202.54	0.23	**0.00**	30/4	22.10
pr144	58,537	**58,537**	**58,537**	**58,537**	**0.00**	**0.00**	**0.00**	**30/30**	2.12
ch150	6528	**6528**	6584	6550.3	13.94	0.34	**0.00**	30/5	25.70
kroA150	26,524	**26,524**	26,649	26,560.2	40.55	0.13	**0.00**	30/3	21.75
kroB150	26,130	**26,130**	26,266	26,146.63	33.49	0.06	**0.00**	30/4	22.17
pr152	73,682	**73,682**	73,818	73,759.06	67.39	0.10	**0.00**	30/13	15.24
rat195	2323	2324	2360	2340.7	8.04	0.76	0.04	22/0	42.30
d198	15,780	**15,780**	15,870	15,802.83	21.31	0.14	**0.00**	30/3	38.75
kroA200	29,368	**29,368**	29,740	29,449.23	85.68	0.27	**0.00**	28/2	46.97
kroB200	29,437	29,439	29,703	29,527.4	74.26	0.30	0.00	30/0	53.10
ts225	126,643	**126,643**	**126,643**	**126,643**	**0.00**	**0.00**	**0.00**	**30/30**	18.24
tsp225	3916	**3916**	3990	3944.8	19.80	0.73	**0.00**	20/1	80.61
pr226	80,369	**80,369**	80,770	80,409.1	93.49	0.04	**0.00**	30/17	44.89
gil262	2378	2380	2410	2390.7	9.20	0.53	0.08	25/0	81.25
pr264	49,135	**49,135**	49,378	49,167.9	73.78	0.06	**0.00**	30/23	64.51
a280	2579	**2579**	2611	2, 586.83	8.94	0.30	**0.00**	28/6	98.09
pr299	48,191	**48,191**	48,552	48,311.7	82.73	0.25	**0.00**	30/1	102.64
lin318	42,029	42,154	42,713	42,462.16	137.66	1.03	0.29	12/0	120.14
rd400	15,281	15,336	15,574	15,465.3	51.78	1.20	0.35	7/0	194.11
fl417	11,861	11,865	11,921	11,884.1	10.41	0.19	0.03	30/0	112.36
pr439	107,217	107,291	108,775	107,683.33	351.44	0.43	0.06	27/0	223.09
rat575	6773	6862	6952	6903.83	22.73	1.93	1.31	0/0	423.56
rat783	8806	8948	9056	9010.4	27.72	2.32	1.61	0/0	758.49
pr1002	259,045	266,146	266,505	266,412.8	82.58	0.03	0.02	30/0	1195.20
nrw1379	56,638	58,188	58,404	58,299	76.97	2.93	2.73	0/0	1863.12

6 Numerical results and performance comparisons

To validate its performances, the proposed algorithm has been tested on 41 symmetric benchmarks of TSP taken from TSPLIB library [35], ranging from 51 to 1379 cities and each instance is run independently for 30 times. The DBA is coded in MATLAB R2010a, and the experiment has been made on a PC with processor Intel(R) Core(TM) 2 Duo CPU T4300@ 2.1 GHZ 800 MHz and 2.00 GB of RAM. Table 2 summarizes the simulation parameter values used for the experiments of DBA algorithm. Table 3 shows the numerical results of DBA algorithm (A number shown in bold in Table 3 indicates that DBA reaches the optimal solution of the tested instance). These results are obtained by calculating the Euclidean distances between all edges. The first column indicates the instance name ended by the number of cities, the second column stands for the best known from TSPLIB, the third column indicates the best results obtained by DBA, the fourth one represents the worst results obtained, the fifth column indicates the average solutions length found, the sixth column denotes the standard deviation of solutions obtained by the DBA algorithm over 30 independent runs, and the seventh and the eighth columns show, respectively, the percentage deviation of the average solution length over the optimal length of 30 runs "PDav(%)," and the percentage deviation of the best solution length found over the optimal solution length of 30 runs "PDbest(%)". In the ninth column, $C_{1\%}$ is the number of solutions (over 30 runs) for which the deviation from the optimal solution is less than or equal to 1 and "C_{opt}" is the number of the optimal solutions. The last column gives the average elapsed time in second during 30 runs.

The percentage deviation of a solution to the optimal solution is calculated by the following formula:

$$\text{PDsolution}(\%) = \frac{(\text{solution length} - \text{optimal solution length})}{\text{optimal solution length}} \times 100$$

Furthermore, the pulse emission rate and loudness are used to control the intensive local search, which allows generating new solutions around the current global best solution. The choice of these parameters values has a critical role in the performance of the algorithm, and the quality of solutions found. The preliminary experiments in Fig. 3 show that a larger emission rate value as well as small loudness value can provide a premature convergence toward undesirable solutions. Therefore, as shown from Fig. 4, it is preferred to use a small value of emission rate and loudness, typically equal to 0.5, to provide a trade-off between emission rate and loudness and give good solutions.

Additionally, in order to evaluate the performance and to give more credibility of our results, Tables 4, 5 and 6 make respectively a fair comparison of the DBA results with those of the both algorithms discrete particle swarm optimization (DPSO) [33], genetic simulated annealing ant colony system with particle swarm optimization techniques (GSA-ACS-PSOT) [66] and improved DCS [34] as presented in their original papers.

According to the values displayed in Table 3, we can see that the DBA can reach the optimal solutions of 73.17 % from all tested instances and 92.68 % of the values of PDbest(%) are less than 0.4 %, which means that the solutions found by the proposed DBA algorithm over 30 runs, are very close to the optimal solutions known. Furthermore, the average error rate PDav(%) of 0.00 % value indicates that all the solutions found over 30 runs are the same as the optimal solutions known. Indeed, the experimental results presented in Table 3 have shown that the DBA is very efficient for solving small-size as well as large-size instances of TSP problem in a reasonable time.

Tables 4, 5 and 6 present the test results for 41 instances over 30 runs of DBA compared, respectively, to DPSO [33], GSA-ACS-PSOT [66], and improved DCS [34] (the optimal solutions found by these algorithms are presented as bold). For the five compared instances in Table 4, the DBA algorithm can find the best solutions of all instances with a success rate of 100 % and it is clearly seen that DBA outperforms DPSO. Moreover, the Table 5 reports the best and the average solutions found by the DBA algorithm and the GSA-ACS-PSOT algorithms, respectively. For 18 tested instances, the DBA finds 16 optimal solutions in front of 11 optimal solutions found by GSA-ACS-PSOT and the average of all standard deviations of DBA/GSA-ACS-PSOT is equal to 34.61/161.55. Also, the DBA reaches the optimal solution over 30 trails for five times; however, GSA-ACS-PSOT does not exceed one instance. Figure 5 shows evidently that the curve associated with DBA is better in terms of solutions quality compared to GSA-ACS-PSOT. Furthermore, Table 6 indicates experimental results found by DBA and improved DCS for 21 instances of the TSPLIB ranging from 107 to 1379 cities. For these instances, the DBA algorithm finds the best solutions for 12 instances over the 21 tested instances, while improved DCS finds the best solutions only for nine instances and the average of all standard deviations of DBA/improved DCS is equal to 61.16/121.20. The displayed results in Fig. 6 show that the solutions of the DBA and the improved DCS are very close in the eleventh first instances; however, DBA outperforms clearly improved DCS for the rest large-size instances.

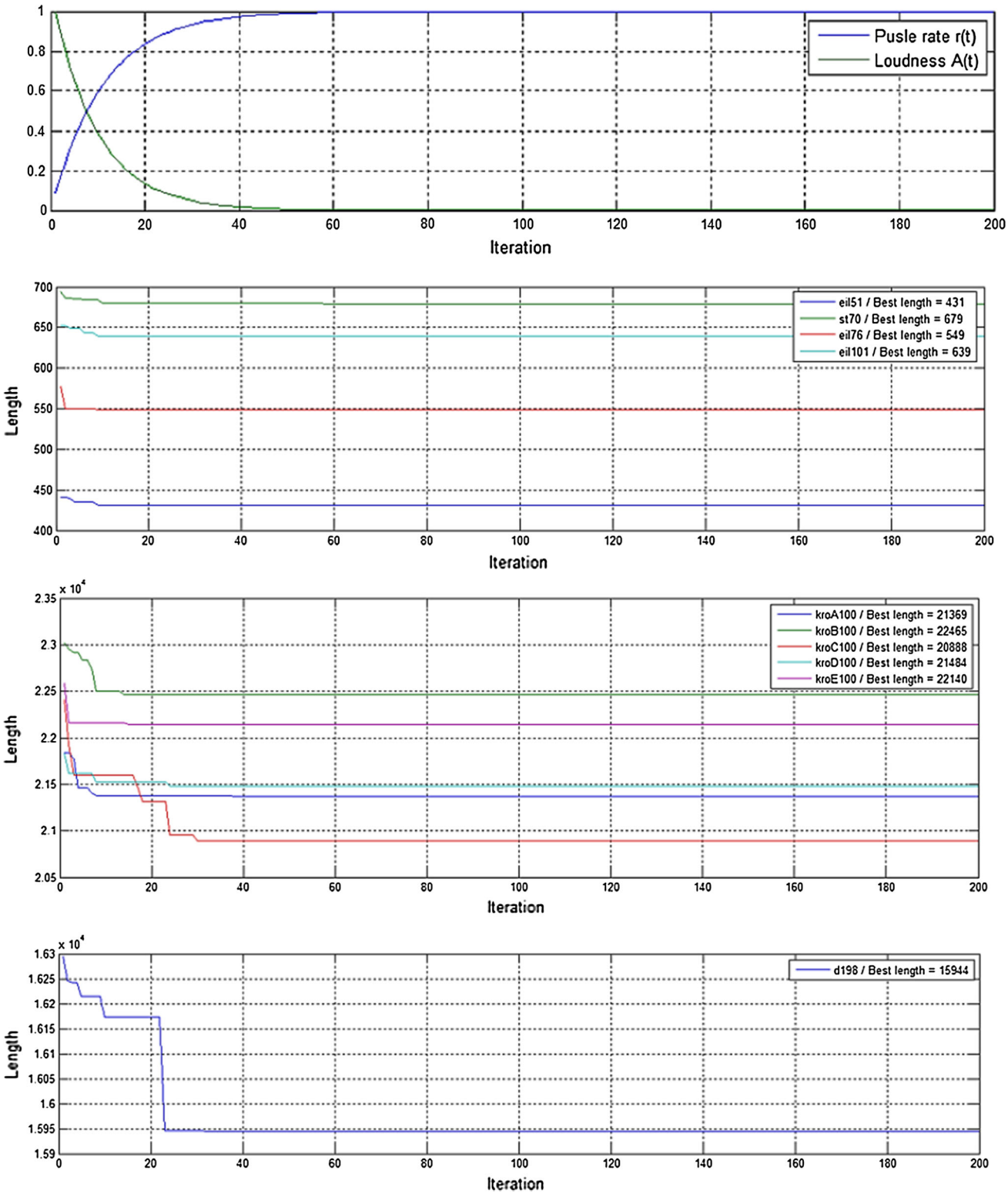

Fig. 3 The best lengths found of eil51, st70 eil76, eil101, kroA100, kroB100, kroC100, kroD100, kroE100 and d198 when r_i and A_i varying between 0 and 1

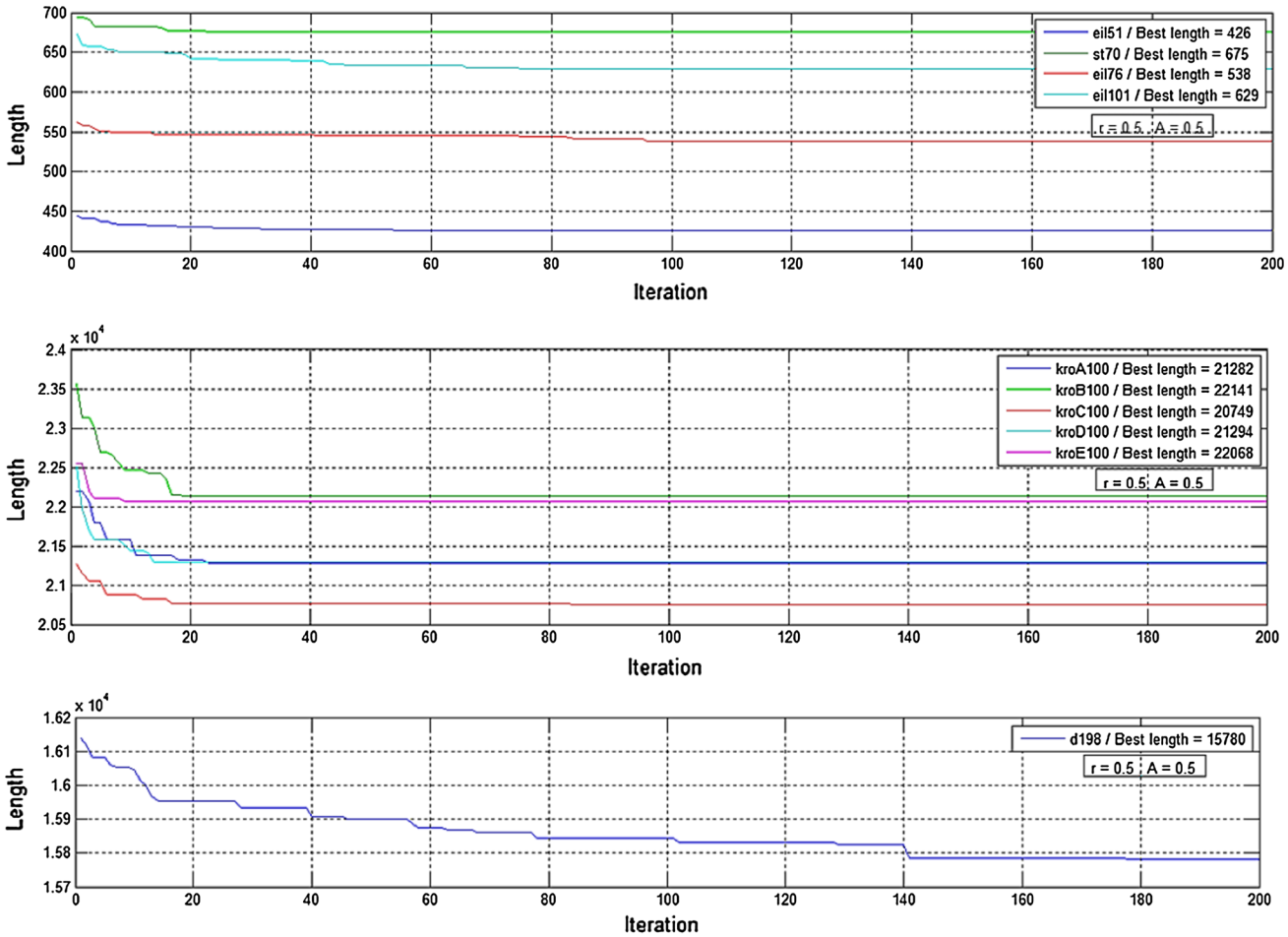

Fig. 4 The best lengths found of eil51, st70 eil76, eil101, kroA100, kroB100, kroC100, kroD100, kroE100 and d198 when $r_i = 0.5$ and $A_i = 0.5$

Table 4 Comparison of computational results of DBA algorithm and DPSO [33]

Instance	Optimal	DBA			DPSO		
		Best	Worst	PDav(%)	Best	Worst	PDav(%)
eil51	426	**426**	**426**	**0.00**	427	452	2.57
berlin52	7542	**7542**	**7542**	**0.00**	**7542**	8362	3.84
st70	675	**675**	**675**	**0.00**	**675**	742	3.34
pr76	108,159	**108,159**	**108,159**	**0.00**	108,280	124,365	3.81
eil76	538	**538**	542	0.14	546	579	4.16

7 Conclusion

This paper has proposed a DBA to solve the STSP. In this discrete version, we were based on basic bat algorithm as it is defined in inspiration sources, where we have extended some conventional BA operators to deal with a discrete optimization problem. The proposed discrete algorithm has been evaluated through its application to solve different instances of the STSP and its performance has been compared with DPSO [33], GSA-ACS-PSOT [66], and improved DCS [34]. The computational results revealed that our proposed DBA performs all compared algorithms to solve STSP. There are a number of directions that can be taken in our future research. The first one is to extend the DBA to solve other NP-hard combinatorial optimization problems, such as vehicle routing problem, scheduling problems, quadratic assignment problem, and many others. Another direction is to examine some discretization methods and encoding strategies used to switch between the continuous and the discrete search space, such as the

Table 5 Comparison of computational results of DBA algorithm and GSA-ACS-PSOT [66]

Instance	Optimal	DBA				GSA-ACS-PSOT			
		Best	Average	PDav(%)	Std	Best	Average	PDav(%)	Std
eil51	426	**426**	**426**	**0.00**	**0.00**	427	427.27	0.29	0.45
berlin52	7542	**7542**	**7542**	**0.00**	**0.00**	**7542**	**7542**	**0.00**	**0.00**
eil76	538	**538**	538.76	0.14	1.16	**538**	540.20	0.40	2.94
kroA100	21,282	**21,282**	**21,282**	**0.00**	**0.00**	**21,282**	21,370.47	0.41	123.36
kroB100	22,141	**22,141**	**22,141**	**0.00**	**0.00**	**22,141**	22,282.87	0.64	183.99
kroC100	20,749	**20,749**	20,753.36	0.02	23.51	**20,749**	20,878.97	0.62	158.64
kroD100	21,294	**21,294**	21,303.50	0.04	23.92	21,309	21,620.47	1.53	226.60
kroE100	22,068	**22,068**	22,080.76	0.05	21.98	**22,068**	22,183.47	0.52	103.32
eil101	629	**629**	632.43	0.54	2.77	630	635.23	0.99	3.59
lin105	14,379	**14,379**	**14,379**	**0.00**	**0.00**	**14,379**	14,406.37	0.19	37.28
bier127	118,282	**118,282**	118,385.66	0.08	155.66	**118,282**	119,442.83	0.98	580.83
ch130	6110	**6110**	6124.1	0.23	8.51	6141	6205.63	1.56	43.70
ch150	6528	**6528**	6550.3	0.34	13.94	**6528**	6563.70	0.54	22.45
kroA150	26,524	**26,524**	26,560.2	0.13	40.55	**26,524**	26,899.20	1.41	133.02
kroB150	26,130	**26,130**	26,146.63	0.06	33.49	**26,130**	26,448.33	1.21	266.76
kroA200	29,368	**29,368**	29,449.23	0.27	85.68	29,383	29,738.73	1.26	356.07
kroB200	29,437	29,439	29,527.4	0.30	74.26	29,541	30,035.23	2.03	357.48
lin318	42,029	42,154	42,462.16	1.03	137.66	42,487	43,002.09	2.31	307.51

Table 6 Comparison of computational results of DBA algorithm and improved DCS [34]

Instance	Optimal	DBA				Improved DCS			
		Best	Worst	PDav(%)	Std	Best	Worst	PDav(%)	Std
pr107	44,303	**44,303**	44,482	0.13	56.57	**44,303**	44,358	0.00	12.90
pr124	59,030	**59,030**	59,076	0.012	17.14	**59,030**	**59,030**	**0.00**	**0.00**
pr136	96,772	**96,772**	97,468	0.23	202.54	96,790	97,318	0.24	134.43
pr144	58,537	**58,537**	**58,537**	**0.00**	**0.00**	**58,537**	**58,537**	**0.00**	**0.00**
pr152	73,682	**73,682**	73,818	0.10	67.39	**73,682**	**73,682**	**0.00**	**0.00**
rat195	2323	2324	2360	0.76	8.04	2324	2357	0.81	8.49
d198	15,780	**15,780**	15,870	0.14	21.31	15,781	15,852	0.17	17.02
ts225	126,643	**126,643**	**126,643**	**0.00**	**0.00**	**126,643**	126,810	0.01	44.59
tsp225	3916	**3916**	3990	0.73	19.80	**3916**	3997	1.09	20.73
pr226	80,369	**80,369**	80,770	0.04	93.49	**80,369**	80,620	0.02	60.31
gil262	2378	2380	2410	0.53	9.20	2382	2418	0.68	9.56
pr264	49,135	**49,135**	49,378	0.06	73.78	**49,135**	49,692	0.24	159.98
a280	2579	**2579**	2611	0.30	8.94	**2579**	2623	0.51	11.86
pr299	48,191	**48,191**	48,552	0.25	82.73	48,207	48,753	0.58	131.79
rd400	15,281	15,336	15,574	1.20	51.78	15,447	15,704	1.65	60.56
fl417	11,861	11,865	11,921	0.19	10.41	11,873	11,975	0.41	20.45
pr439	107,217	107,291	108,775	0.43	351.44	107,447	109,013	0.69	438.15
rat575	6773	6862	6952	1.93	22.73	6896	7039	2.71	35.74
rat783	8806	8948	9056	2.32	27.72	9043	9171	3.44	38.09
pr1002	259,045	266,146	266,505	0.03	82.58	266,508	271,660	3.70	1126.86
nrw1379	56,638	58,188	58,404	2.93	76.97	58,951	59,837	4.78	213.89

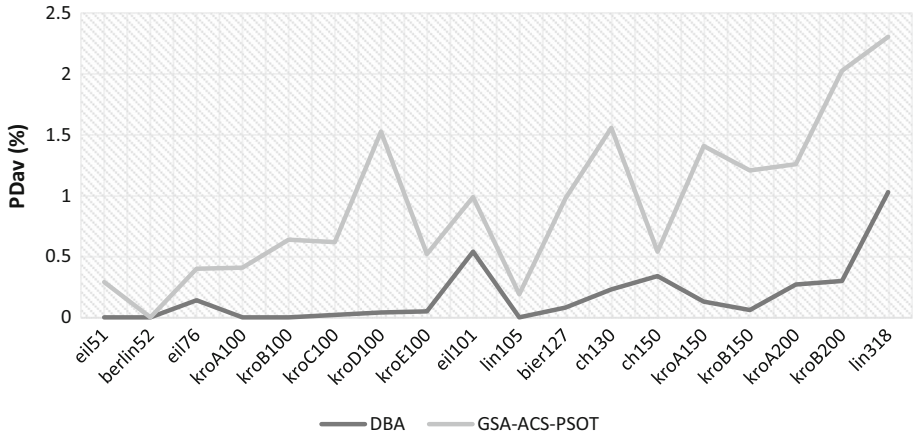

Fig. 5 PDav(%) (over 30 run) for 18 instances from TSPLIB

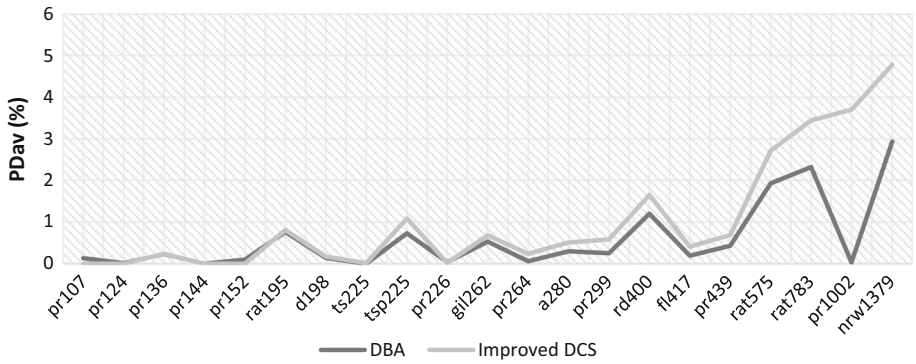

Fig. 6 PDav(%) (over 30 run) for 21 instances from TSPLIB

smallest position value method, the random-key encoding scheme, the great value priority method, and the sigmoid function. Finally, it may also be interesting to hybrid the DBA with other heuristics or metaheuristics and to analyze the performance of each algorithm in solving TSP problem.

References

1. Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor
2. Schwefel H-P (1981) Numerical optimization of computer models. Wiley, London
3. Kirkpatrick S (1984) Optimization by simulated annealing: quantitative studies. J Stat Phys 34(5–6):975–986
4. Kennedy J (2010) Particle swarm optimization. Encyclopedia of machine learning. Springer, Berlin, pp 760–766
5. Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life. Paris, France, pp 134–142
6. Wolsey LA, Nemhauser GL (2014) Integer and combinatorial optimization. Wiley, London
7. Papadimitriou CH, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity. Courier Dover Publications, Mineola
8. Wong W (1995) Matrix representation and gradient flows for NP-hard problems. J Optim Theory Appl 87(1):197–220
9. Dorigo M, Birattari M (2010) Ant colony optimization. Encyclopedia of machine learning. Springer, Berlin, pp 36–39
10. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1(1):33–57
11. Gandomi AH, Yun GJ, Yang X-S, Talatahari S (2013) Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simul 18(2):327–340
12. Lučić P, Teodorović D (2003) Computing with bees: attacking complex transportation engineering problems. Int J Artif Intell Tools 12(03):375–394
13. Yang X-S (2009) Firefly algorithms for multimodal optimization. Stochastic algorithms: foundations and applications. Springer, Berlin, pp 169–178
14. Yang X-S, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343
15. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, pp 65–74
16. Yang X-S, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483
17. Arora S (1998) Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J ACM (JACM) 45(5):753–782
18. Bland RG, Shallcross DF (1989) Large travelling salesman problems arising from experiments in X-ray crystallography: a preliminary report on computation. Oper Res Lett 8(3):125–128

19. Lenstra JK, Kan AR (1975) Some simple applications of the travelling salesman problem. Oper Res Q 717–733
20. Grötschel M, Jünger M, Reinelt G (1991) Optimal control of plotting and drilling machines: a case study. Math Methods Oper Res 35(1):61–84
21. Ratliff HD, Rosenthal AS (1983) Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Oper Res 31(3):507–521
22. Zachariasen M, Dam M (1996) Tabu search on the geometric traveling salesman problem. Meta-heuristics. Springer, Berlin, pp 571–587
23. Chen Y, Zhang P (2006) Optimized annealing of traveling salesman problem from the *n*th-nearest-neighbor distribution. Phys A 371(2):627–632
24. Potvin J-Y (1996) Genetic algorithms for the traveling salesman problem. Ann Oper Res 63(3):337–370
25. Qu L, Sun R (1999) A synergetic approach to genetic algorithms for solving traveling salesman problem. Inf Sci 117(3):267–283
26. Marinakis Y, Migdalas A, Pardalos PM (2005) Expanding neighborhood GRASP for the traveling salesman problem. Comput Optim Appl 32(3):231–257
27. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. Evol Comput IEEE Trans 1(1):53–66
28. Dorigo M, Gambardella LM (1997) Ant colonies for the travelling salesman problem. BioSystems 43(2):73–81
29. Liu A, Deng G, Shan S (2006) Mean-contribution ant system: an improved version of ant colony optimization for traveling salesman problem. Simulated evolution and learning. Springer, Berlin, pp 489–496
30. Manfrin M, Birattari M, Stützle T, Dorigo M (2006) Parallel ant colony optimization for the traveling salesman problem. Ant colony optimization and swarm intelligence. Springer, Berlin, pp 224–234
31. Clerc M (2004) Discrete particle swarm optimization, illustrated by the traveling salesman problem. New optimization techniques in engineering. Springer, Berlin, pp 219–239
32. Li X, Tian P, Hua J, Zhong N (2006) A hybrid discrete particle swarm optimization for the traveling salesman problem. Simulated evolution and learning. Springer, Berlin, pp 181–188
33. Shi XH, Liang YC, Lee HP, Lu C, Wang Q (2007) Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf Process Lett 103(5):169–176
34. Ouaarab A, Ahiod B, Yang X-S (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24(7–8):1659–1669
35. Reinelt G (1991) TSPLIB—a traveling salesman problem library. ORSA J Comput 3(4):376–384
36. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471
37. Kenndy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. pp 1942–1948
38. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. Control Syst IEEE 22(3):52–67. doi:10.1109/MCS.2002.1004010
39. Krishnanand K, Ghose D (2009) Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intell 3(2):87–124
40. Chu S-C, Tsai P-W, Pan J-S (2006) Cat swarm optimization. PRICAI 2006: trends in artificial intelligence. Springer, Berlin, pp 854–858
41. Gandomi AH, Alavi AH (2012) Krill Herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
42. Gheraibia Y, Moussaoui A (2013) Penguins search optimization algorithm (PeSOA). Recent trends in applied artificial intelligence. Springer, Berlin, pp 222–231
43. Meng X, Liu Y, Gao X, Zhang H (2014) A new bio-inspired algorithm: chicken swarm optimization. Advances in swarm intelligence. Springer, Berlin, pp 86–94
44. Bansal JC, Sharma H, Jadon SS, Clerc M (2014) Spider monkey optimization algorithm for numerical optimization. Memetic Comput 6(1):31–47
45. Gandomi AH, Yang X-S, Alavi AH, Talatahari S (2013) Bat algorithm for constrained optimization tasks. Neural Comput Appl 22(6):1239–1255
46. Khan K, Nikov A, Sahai A (2011) A fuzzy bat clustering method for ergonomic screening of office workplaces. In: Dicheva D, Markov Z, Stefanova E (eds) Third international conference on software, services and semantic technologies S3T 2011, vol 101., Advances in intelligent and soft computingSpringer, Berlin Heidelberg, pp 59–66. doi:10.1007/978-3-642-23163-6_9
47. Tamiru AL, Hashim FM (2013) Application of bat algorithm and fuzzy systems to model energy changes in a gas turbine. In: Yang X-S (ed) Artificial intelligence, evolutionary computing and metaheuristics, vol 427., Studies in computational intelligenceSpringer, Berlin Heidelberg, pp 685–719. doi:10.1007/978-3-642-29694-9_26
48. Yılmaz S, Küçüksille EU (2015) A new modification approach on bat algorithm for solving optimization problems. Appl Soft Comput 28:259–275
49. Nguyen T-T, Pan J-S, Dao T-K, Kuo M-Y, Horng M-F (2014) Hybrid bat algorithm with artificial bee colony. In: Pan J-S, Snasel V, Corchado ES, Abraham A, Wang S-L (eds) Intelligent data analysis and its applications, vol II-298., Advances in intelligent systems and computingSpringer, Berlin, pp 45–55. doi:10.1007/978-3-319-07773-4_5
50. Pan T-S, Dao T-K, Nguyen T-T, Chu S-C (2015) Hybrid particle swarm optimization with bat algorithm. In: Sun H, Yang C-Y, Lin C-W, Pan J-S, Snasel V, Abraham A (eds) Genetic and evolutionary computing, vol 329., Advances in intelligent systems and computingSpringer, Berlin, pp 37–47. doi:10.1007/978-3-319-12286-1_5
51. Cai X, Wang L, Kang Q, Wu Q (2014) Bat algorithm with Gaussian walk. Int J Bio-Inspired Comput 6(3):166–174
52. Gandomi AH, Yang X-S (2014) Chaotic bat algorithm. J Comput Sci 5(2):224–232
53. Mirjalili S, Mirjalili S, Yang X-S (2014) Binary bat algorithm. Neural Comput Appl 25(3–4):663–681. doi:10.1007/s00521-013-1525-5
54. Yang X-S, He X (2013) Bat algorithm: literature review and applications. Int J Bio-Inspired Comput 5(3):141–149
55. Nakamura RY, Pereira LA, Costa K, Rodrigues D, Papa JP, Yang X-S (2012) BBA: A binary bat algorithm for feature selection. In: Graphics, patterns and images (SIBGRAPI), 2012 25th SIBGRAPI Conference on IEEE, pp 291–297
56. Xie J, Zhou Y, Tang Z (2013) Differential Lévy-Flights bat algorithm for minimization makespan in permutation flow shops. In: Huang D-S, Jo K-H, Zhou Y-Q, Han K (eds) Intelligent computing theories and technology, vol 7996., Lecture Notes in Computer ScienceSpringer, Berlin Heidelberg, pp 179–188. doi:10.1007/978-3-642-39482-9_21
57. Sabba S, Chikhi S (2014) A discrete binary version of bat algorithm for multidimensional knapsack problem. Int J Bio-Inspired Comput 6(2):140–152
58. Büyüksaatçı S (2015) Bat algorithm application for the single row facility layout problem. In: Yang X-S (ed) Recent advances in swarm intelligence and evolutionary computation, vol 585., Studies in computational intelligenceSpringer, Berlin, pp 101–120. doi:10.1007/978-3-319-13826-8_6

59. Fister I, Rauter S, Yang X-S, Ljubič K (2015) Planning the sports training sessions with the bat algorithm. Neurocomputing 149:993–1002
60. Yang X-S (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3(5):267–274
61. Pappalardo E, Pardalos P, Stracquadanio G (2013) Mathematical optimization. Optimization approaches for solving string selection problems., Springer Briefs in OptimizationSpringer, New York, pp 13–25
62. Du D-Z, Pardalos PM (1999) Handbook of combinatorial optimization: supplement, vol 1. Springer, Berlin
63. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation. 1997 IEEE International Conference on, IEEE, pp 4104–4108
64. Larrañaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 13(2):129–170. doi:10.1023/A:1006529012972
65. Li L, Zhang Y (2007) An improved genetic algorithm for the traveling salesman problem. Advanced intelligent computing theories and applications. With aspects of contemporary intelligent computing techniques. Springer, Berlin, pp 208–216
66. Chen S-M, Chien C-Y (2011) Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Syst Appl 38(12):14439–14450