Engineering Science and Technology, an International Journal 22 (2019) 673–691

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

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

Full Length Article

Tournament selection based antlion optimization algorithm for solving quadratic assignment problem☆

Haydar Kılıç, Uğur Yüzgeç *

Department of Computer Engineering, Bilecik Seyh Edebali University, 11210 Bilecik, Turkey

ARTICLE INFO

Article history:
Received 28 June 2018
Revised 16 October 2018
Accepted 29 November 2018
Available online 8 December 2018

Keywords:
Tournament selection
Antlion
Quadratic assignment problem

ABSTRACT

We propose and develop an improved version of antlion optimizer (ALO), namely tournament selection based antlion optimization algorithm for quadratic assignment problem (QAP). ALO algorithm has some handicaps, such as long run time, local optima stagnation and premature convergence for some problems. The literature describes different methods that improve the performance of antlion optimizer, but most of these are about specific optimization problems. In this paper, we introduce the tournament selection method instead of the roulette wheel method on random walking mechanism of ALO and we update some equations used in ALO algorithm. To compare the proposed tournament selection based ALO (TALO) algorithm with classic ALO, we deal with ten benchmark functions from literature. The comparison results are evaluated according to the different metrics, such as mean best, standard deviation, optimality, accuracy, CPU time, number of function evaluations (NFE). The detailed analyzes of the proposed TALO algorithm are performed. These are the convergence analysis, statistical analysis, search history analysis, trajectory analysis, average distance analysis, computational complexity analysis. The proposed TALO algorithm is compared with the other ALO versions (binary ALO and chaotic ALO variants) for same ten benchmark functions. As last, TALO algorithm has been also implemented for the quadratic assignment problem (QAP). The QAP results has been compared with several well-known meta-heuristic algorithms. TALO's performance has been evaluated with those of binary ALO and chaotic ALO variants for same QAP instance. Finally, the solution quality of proposed TALO algorithm has been analyzed for solving QAP using some instances presented in QAPLIB site. The results provide the proposed TALO algorithm has the best performance in comparison with those of the other meta-heuristic algorithms. Due to the performance of TALO algorithm, we expect this version to be applied for different optimization problems.

1. Introduction

In mathematics and computer science, meta-heuristic algorithms are becoming an important role in solving to the optimization problems. Wide usage areas of meta-heuristic algorithms have come up over the last three decades, and many meta-heuristic algorithms such as genetic algorithm, differential evolution algorithm, particle swarm optimization algorithm, artificial bee colony algorithm, ant colony algorithm, etc., have become popular for real-world optimization problems. In general, the meta-heuristic algorithms imitate some mechanisms from nature as animal feeding habits, mating motivation or hunting techniques [1,2]. Some of the most popular algorithms are Genetic Algorithm (GA) [3], Particle Swarm Optimization (PSO) algorithm [4,5], Artificial Bee Colony (ABC) algorithm [6–8], Differential Evolution (DE) algorithm [9,10], Ant Colony (ACO) algorithm [11].

Combinatorial optimization problem is one of the research areas at the intersection of computer science and applied mathematics. Data allocation, traveling salesman, bin packing, facility layout, machine scheduling and quadratic assignment problems are given as examples of this problem. The quadratic assignment problem (QAP) which is one of the most difficult combinatorial optimization problems was presented in 1957 by Koopmans and Beckmann [12]. It is defined as a facilities allocation problem. These facilities are located in many places that are already known and at the least costly ones. In this problem, the sum of the costs for each facility is the general cost function. In literature, there are several works on solving QAP using the meta-heuristic algorithms. Some of these are: solving QAP by simulated annealing algorithm [13,14], by genetic algorithm [15], by ant colony algorithm [16], by tabu search algorithm [17] and by particle swarm optimization algorithm [18].

* Corresponding author.
E-mail addresses: haydar.kilic@bilecik.edu.tr (H. Kılıç), ugur.yuzgec@bilecik.edu.tr (U. Yüzgeç).
☆ This paper in short form was published in ICATCES 2018.
Peer review under responsibility of Karabuk University.

https://doi.org/10.1016/j.jestch.2018.11.013
2215-0986/

Antlion optimization algorithm (ALO) that was presented by Seyedali Mirjalili [19] in 2015 is a meta-heuristic algorithm. The antlions come from the Myrmeleontidae family of predatory insect species, which take their name from interesting nutritional behavior in the larval period. ALO algorithm mimics the hunting mechanism of the antlion in larvae phase. There are five main steps: the random walking mechanism of ants, building a trap, trapping in the antlion's pits, sliding ants towards antlion, catching the prey and rebuilding the pit [19,20]. There are some studies reported in the literature regarding real-world optimization implementation or increasing performance of the ALO algorithm. These are optimal non-convex and dynamic economic load [21], PID controller parameters design [22], optimal flexible process planning [23], automatic generation control of interconnected power system [24], optimal route planning for unmanned aerial vehicle [25], multi objective optimal generation scheduling [26], determining the optimal coefficients of IIR filters [27], feature selection problem [32–37].

Although original ALO algorithm presents the effective results for different benchmark optimization problems, it has got some disadvantages about the algorithm's mechanism. One of the handicaps of ALO algorithm is that it has got the long runtime at the end of the optimization process. The reason of this is the random walking model used in its code structure. In this study, TALO algorithm was developed by eliminating some deficiencies of the original algorithm, and a modified ALO algorithm is presented. We have changed the distance of random walking model as maximum iteration number/5. In Mirjalili's work [19], the antlion is selected from the population by roulette wheel method for random walking procedure. The tournament selection method is preferred rather than roulette wheel method in minimization problems. In the minimization problems, the tournament selection method is more efficient method [28,29]. For that reason, we preferred the tournament selection method to the roulette wheel method for the random walking mechanism. Furthermore, some new movements were defined between lower and upper boundaries around the antlion in the phase of trapping on antlion pits. These movements provide that ants walk more effectively around the selected antlion in the search space.

In this study, tournament selection based antlion optimization algorithm (TALO) is proposed to overcome the drawbacks of the original ALO algorithm. In Mirjalili's work, there was no time analysis (CPU time, number of function evaluation) about the original ALO algorithm. For this reason, the performance analysis of the proposed TALO algorithm is presented. From the literature, ten benchmark functions are taken to evaluate the performance of the proposed TALO algorithm. In comparison works, the TALO algorithm is compared with the original ALO algorithm in terms of the mean of the best value, CPU time, number of function evaluations (NFE) optimality, accuracy metrics. The detailed analyzes of the proposed TALO algorithm such as the convergence analysis, statistical analysis, search history analysis, trajectory analysis, average distance analysis, computational complexity analysis are performed. The proposed TALO algorithm is also compared with the other ALO versions (binary ALO and chaotic ALO variants).

The major purpose of this study is to apply the proposed TALO algorithm to the QAP problem. The proposed TALO algorithm's performance on QAP instance was compared with the well-known meta-heuristic algorithms such as ALO algorithm, Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Firefly Optimization Algorithm (FOA). At the same time, TALO algorithm's performance has been evaluated with those of binary ALO and chaotic ALO variants for same QAP instance. Finally, we present the solution quality of proposed TALO algorithm for solving QAP using some instances in QAPLIB.

The rest of the paper is organized as follows:

In Section 2, the basic information about the Quadratic Assignment Problem (QAP) problem is given briefly. Section 3 introduces the brief of the original ALO algorithm. The proposed TALO algorithm and its innovation mechanisms are presented in Section 4. In the benchmark and QAP tests, the performance of the TALO algorithm is discussed in Section 5. In the last section, conclusion and the future works are presented.

2. Quadratic assignment problem (QAP)

The Quadratic Assignment Problem (QAP) was presented for the first time by Koopmans and Beckman [1]. In solving this problem, the main aim is to find minimum cost for total assignment while

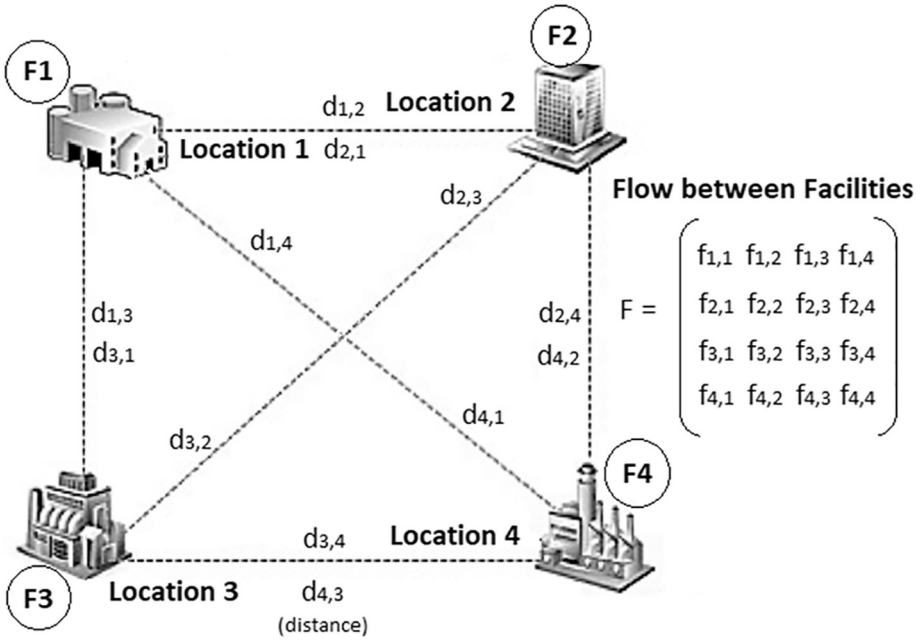

Fig. 1. Quadratic Assignment Problem (QAP).

assigning facilities to locations. The mathematical model objective function of QAP is given below:

$$min \sum_{i,j=1}^{n} \sum_{p,q=1}^{n} w_{ij} d_{pq} x_{ip} x_{jq} \tag{1}$$

$$subjectto \sum_{i=1}^{n} x_{ij} = 1, \sum_{j=1}^{n} x_{ij} = 1, x_{ij} \in \{0,1\}, 1 \le i,j \le n \tag{2}$$

where w_{ij}stands for the weight/flow coefficients in range i_{th} and j_{th} facilities and d_{pq} denotes the distance in range p_{th} and q_{th} locations. Fig. 1 shows the example of QAP with four facilities.

3. AntLion optimizer (ALO)

The mathematical model of the original ALO algorithm is given briefly in this section. The hunting strategy of antlions shows their unique hunting behaviour. The hunting technique consists of setting up a trap, hiding in this trap, waiting for its prey, preventing the prey escape from the trap, catching its prey and rebuilding the trap. The ALO algorithm imitates hunter-prey relationship between antlions and ants in the trap. After initializing the antlion population, the random walk model of ants is started. The mathematical model of these walks is given below:

$$X(t) = \begin{bmatrix} 0 \\ cumsum(2r(t_1) - 1) \\ cumsum(2r(t_2) - 1) \\ \vdots \\ cumsum(2r(t_n) - 1) \end{bmatrix} \tag{3}$$

where n stands for the maximum number of iteration, *cumsum* is the cumulative sum, t denotes the step of random walk, and $r(t)$ is the stochastic function as defined:

$$r(t) = \begin{cases} 1, if rand > 0.5 \\ 0, if rand \le 0.5 \end{cases} \tag{4}$$

In order to keep ant's random walks in the search space, it is normalized by the equation given below:

$$X_i^t = (X_i^t - a_i)\left(d_i^t - c_i^t\right)(b_i - a_i)^{-1} + c_i^t \tag{5}$$

where i denotes the variable number's value, t is the iteration number, a stands for the minimum value of the random walk ($a = min(X)$), b is the maximum value of the random walk ($b = max(X)$), c denotes the lower value of the dynamic limit around the antlion, d is the upper value of the dynamic limit around the antlion.

Once the ants begin to fall into the trap, the antlions throw sand to hold in the trap and slide them down towards the center of the trap. The mathematical model regarding this behaviour is given below:

$$c_i^t = Antlion_i^t + c^t \tag{6}$$

$$d_i^t = Antlion_i^t + d^t \tag{7}$$

$$c^t = c^t . I^{-1} \tag{8}$$

$$d^t = d^t . I^{-1} \tag{9}$$

where $Antlion_i^t$ denotes the position of the selected *i-th* antlion at *t-th* iteration, and I stands for the sliding ratio. The ants are moved around the elite antlion and the antlion selected by roulette wheel method using the model given in Eq. (10).

$$Ant_i^t = 0.5\left(R_A^t + R_E^t\right) \tag{10}$$

where R_A^t denotes the antlion selected by roulette wheel method and R_E^t denotes the elite antlion obtained by Eq. (5) at each iteration. After catching the prey, antlions update their positions according to the eaten ants' fitness values. Eq. (11) gives the update mechanism of antlion for minimization problems:

$$iff\left(Ant_i^t\right) < f\left(Antlion_i^t\right), Antlion_i^t = Ant_i^t \tag{11}$$

ALO algorithm's pseudo code is given follows:

Algorithm 1 (*Pseudo Code of ALO Algorithm.*).

1. Initialize the positions of antlions
2. Calculate the cost values of antlions
3. Save the best antlion and its position (elite antlion)
4. **while** (*iteration < maximum iteration*)
5. **for** (each antlion)
6. Select antlion using roulette wheel method
7. Slide randomly walking ants in a trap
8. Generate ant's random walk route around elite antlion
9. Generate ant's random walk route around selected antlion
10. Normalize random walks
11. Calculate the position of ant using Eq. (8)
12. **end for**
13. Calculate the cost values of ants
14. Combine ants and antlions
15. Sort according to their costs and take first population size
16. Update the elite antlion
17. **end while**

4. Tournament selection based antlion optimization algorithm (TALO)

In this study, we developed ALO algorithm by improvements on the random walk model, hunting mechanism, selection procedure and etc. In the original ALO code, the random walking mechanism uses the maximum iteration number to generate the ant's random walking route. This is not effective method for run time of the algorithm. For that reason, first innovation on the ALO algorithm was realized by reducing the random walk size. We used n value as maximum iteration number/5 in Eq. (3). Therefore, the long running time of the ALO algorithm has been shortened significantly. In the stage of sliding ants towards the antlion's trap, the ants are shifted toward the antlion by a certain rate of slippage. We made some improvements on the antlion's pit and on the shifting of the ants by throwing sand. It is given by the following equations.

$$\left.\begin{aligned} c_i^t &= Antlion_i^t + c^t \\ d_i^t &= Antlion_i^t + d^t \end{aligned}\right\} if \;\; 0.75 < opt < 1 \tag{12}$$

$$\left.\begin{aligned} c_i^t &= Antlion_i^t - c^t \\ d_i^t &= Antlion_i^t - d^t \end{aligned}\right\} if \;\; 0.5 < opt < 0.75 \tag{13}$$

$$\left.\begin{aligned} c_i^t &= -Antlion_i^t + c^t \\ d_i^t &= -Antlion_i^t + d^t \end{aligned}\right\} if \;\; 0.25 < opt < 0.5 \tag{14}$$

$$\left.\begin{aligned} c_i^t &= -Antlion_i^t - c^t \\ d_i^t &= -Antlion_i^t - d^t \end{aligned}\right\} if \;\; opt < 0.25 \tag{15}$$

where *opt* denotes a variable chosen randomly. Thanks to the updating the shift rates, we have provided a more accurate and faster hunting mechanism. Unlike antlion updating mechanism used at the end of iterations in the ALO algorithm, in the new updating mechanism, cost values of the ants and antlions are compared for each pair of ant and antlion. If the ant's cost value is better than antlion's cost, antlion's position is updated as ant position. Another innovation is about ants which go out of the search area. When the ant's position is out of the search space, they come back the search space again unlike the original ALO algorithm. This idea ensures that the ants take the random positions in the search space.

$$Ant_i^t = b_{low} + rand \times (b_{up} - b_{low}), \quad if \;\; Ant_i^t > b_{up} or Ant_i^t < b_{low} \tag{16}$$

where *rand* stands for a random number in interval [0,1], b_{low} denotes the lower limit and b_{up} is the upper limit of the search space. In meta-heuristic algorithms, the selection procedure is used for selecting better individuals from the population. Some of examples are roulette wheel method, tournament selection method, truncation selection, linear ranking selection and exponential ranking selection. For especially minimization problems, tournament selection is the most efficient method [29,30].

In this method, a tournament is realized between individuals selected randomly and the individual with the best cost value becomes the winner of the tournament. The parameter of this method is the tournament size, known as tour. This parameter may be become a value ranging from 2 to number of population.

In this study, we focused a minimization problem known as QAP, so we preferred the tournament selection method instead of the roulette wheel selection method used in the original ALO algorithm. The tour size was selected as 2. In the tournament method, two groups from the population are randomly selected and the size of each group is found by division of the population size to tournament size. Tournament Selection based Antlion Optimization (TALO) algorithm's pseudo code is given below:

Algorithm 2 (*Pseudo Code of TALO Algorithm.*).

```
1. Initialize the positions of antlions
2. Calculate the cost values of antlions
3. Save the best antlion and its position (elite antlion)
4.   while (iteration < maximum iteration)
5.     for (each antlion)
6.       Select antlion using tournament selection method
7.       Slide randomly walking ants in a trap as Eqs. (12)–(15)
8.       Generate ant's random walk route around elite
         antlion
9.       Generate ant's random walk route around selected
         antlion
10.      Normalize random walks
11.      Calculate the position of ant
12.        if Ant is out of the search space
13.          Relocate the ant in search space
14.        end if
15.    end for
16.    Calculate the cost values of ants
17.    for (each antlion)
18.      if Cost value of ant is better than that of antlion
19.        antlion eats ant (update the position of antlion)
20.      end if
21.    end for
22.    Update the elite antlion
23. end while
```

5. Experimental results and discussions

In this section, firstly, a number of benchmark tests are taken from the literature to show the performance of the proposed TALO algorithm. The performance comparison was realized between TALO algorithm and the original ALO algorithm according to the different metrics. Secondly, the proposed TALO algorithm has been implemented to QAP.

5.1. Benchmark results

We took ten different benchmark functions from the literature to evaluate the proposed TALO algorithm with original ALO algorithm. All benchmark functions includes different characteristics. Four metrics are used as defined below:

$$Optimality = 1 - \frac{\| \gamma_0 - \hat{\gamma}_0 \|}{\| \bar{\gamma} - \underline{\gamma} \|} \in [0,1] \tag{17}$$

$$Accuracy = 1 - \frac{\| x_0 - \hat{x}_0 \|}{\| \bar{x} - \underline{x} \|} \in [0,1] \tag{18}$$

$$Mean = \frac{1}{N} \sum_{i=1}^{N} \hat{\gamma}_0 \tag{19}$$

$$StandardDeviation(Std) = \sqrt{\frac{1}{N-1} \sum \left(\hat{\gamma}_0 - Mean\right)^2} \tag{20}$$

where x_0 is a position in the search space, $\gamma(x_0) = \gamma_0$ is the solution of the optimization problem, $\gamma(\hat{x}_0) = \hat{\gamma}_0$ denotes the candidate solution found by the algorithm, $\bar{\gamma}$ and γ denote lower and upper bounds of γ, $\bar{x}$ and x denote the lower and upper bounds of the search space. *Optimality* metric gives the relative closeness of an objective function value of the candidate solution to the global solution. *Accuracy* metric defines the relative closeness of a candidate solution's position to the global solution position. *Mean* metric represents the average solution. *CPU time* and *Number of the Function Evaluations* (*NFE*) give some information regarding the runtime of the algorithm. The benchmark functions used for the comparison work are given in Table 1. *d* represents the dimension of the problem in the benchmark mathematical formulas. Fig. 2 shows the 3D graphs of ten benchmark functions used in this study.

Two criteria for stop termination have been used for benchmark tests, one is to reach the maximum number of iterations, and the other is given below:

$$if \;\; |f_{best} - f_{worst}| < VTR \;\; then \;\; stop \;\; the \;\; algorithm \tag{21}$$

where f_{best} is the best fitness value, f_{worst} is the worst fitness value in the population, *VTR* stands for the value to reach and this value is used as $10e-6$ in this study. The population size is determined as 10 times the number of the optimization parameters (problem dimension) and we defined the dimension of benchmarks as 10. In the benchmark tests, the population size is 100, and maximum iteration number is 1000. Both ALO and TALO algorithms have been run 50 times. The codes of these algorithms have been run on PC with Intel(R) Core(TM) i7-6500U CPU@2.50 GHz/8.00 GB RAM. In the comparison results, four metrics were evaluated such as *mean best*/*standard deviation*, *number of function evaluation (NFE)*/*CPU time*, *optimality*, *accuracy*. The results of 10D benchmark tests for TALO and ALO algorithms are summarized in Table 2.

As can be seen from this table results in terms of the *mean best*/*StDev*, the proposed TALO algorithm has the best performance for all benchmark functions. The results of the NFE/CPU Time metrics

Table 1
Benchmark functions.

Function	Dim	Range	Solution
Ackley Function $f_1(x) = -20.exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}cos(2\pi x_i)\right) + 20 + \exp(1)$	10	[35,35]	$f(x) = 0$ at $x = (0,\cdots,0)$
Griewank Function $f_2(x) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	10	[−100,100]	$f(x) = 0$ at $x = (0,\cdots,0)$
Levy Function $f_3(x) = sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i-1)^2\left[1+10sin^2(\pi w_i+1)\right] + (w_d-1)^2\left[1+sin^2(2\pi w_d)\right]$ $w_i = 1+\frac{x_i-1}{4}, i = 1,2,\cdots,d$	10	[−10,10]	$f(x) = 0$ at $x = (1,\cdots,1)$
Rastrigin Function $f_4(x) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	10	[−5.12,5.12]	$f(x) = 0$ at $x = (0,\cdots,0)$
Rosenbrock Function $f_5(x) = \sum_{i=1}^{d-1}\left[100(x_{i+1}-x_i^2)^2 + (1-x_i)^2\right]$	10	[−2.3,2.3]	$f(x) = 0$ at $x = (1,\cdots,1)$
Schwefel Function $f_6(x) = 418.9829d - \sum_{i=1}^{d}x_i sin(\sqrt{\lvert x_i\rvert})$	10	[−500,500]	$f(x) = 0$ at $x = (420.96,\cdots,420.96)$
Sphere Function $f_7(x) = \sum_{i=1}^{d}x_i^2$	10	[−5.12,5.12]	$f(x) = 0$ at $x = (0,\cdots,0)$
Styblinski-Tang Function $f_8(x) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$	10	[−5,5]	$f(x) = -39.16$ at $x = (-2.9,\cdots,-2.9)$
Sum Squares Function $f_9(x) = \sum_{i=1}^{d}ix_i^2$	10	[−10,10]	$f(x) = 0$ at $x = (0,\cdots,0)$
Zakharov Function $f_{10}(x) = \sum_{i=1}^{d}x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d}ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d}ix_i\right)^4$	10	[−5,10]	$f(x) = 0$ at$x = (0,\cdots,0)$

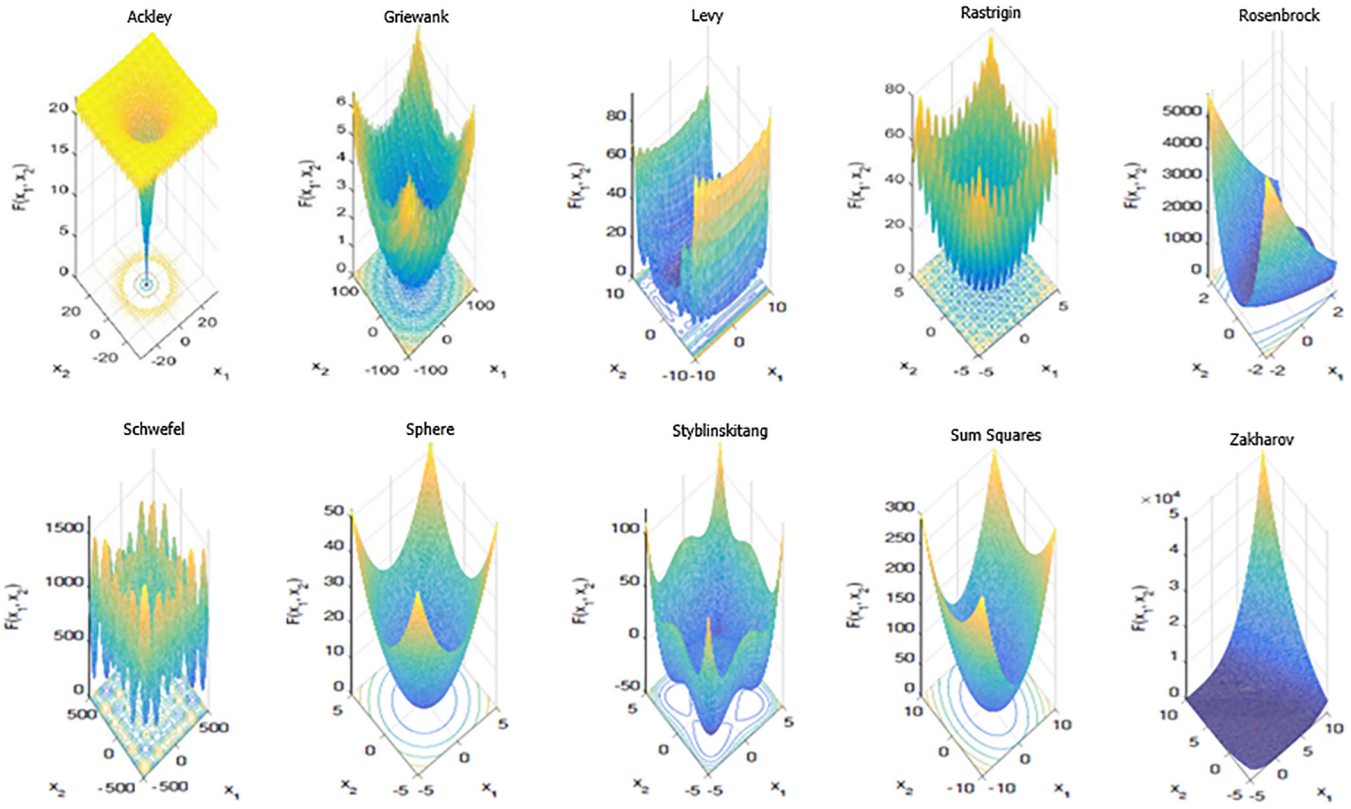

Fig. 2. Benchmark Function 3D graphs.

of both algorithms show that TALO algorithm is faster 10–20 times than the original ALO algorithm. *Optimality* metric shows how close to the global solution. TALO algorithm proves the best value (1) for all benchmarks. Original ALO algorithm has the best value for only FN7, FN9 and FN10 functions. *Accuracy* metric indicates how close to the global solution points. According to this metric, the best result is TALO for all benchmark functions. ALO algorithm has the best value for only four benchmarks.

Table 2
Comparison results with 50 independent runs of TALO and ALO algorithms. The best result of each function is emphasized in **boldface**.

Function	ALO				TALO			
	Mean Best (StDev)	NFE (CPUTime)	Optimality	Accuracy	Mean Best (StDev)	NFE (CPUTime)	Optimality	Accuracy
FN1	2.14e−01 (5.05e−01)	99,258 (49.881 s)	0.990	**1.000**	**0.00e+00 (0.00e+00)**	**98,158 (5.745 s)**	**1.000**	**1.000**
FN2	1.42e−01 (7.52e−02)	95,216 (48.518 s)	0.979	0.973	**0.00e+00 (0.00e+00)**	**74,826 (4.435 s)**	**1.000**	**1.000**
FN3	2.49e−01 (3.49e−01)	90,324 (46.103 s)	0.997	0.992	**5.04e−06 (1.16e−05)**	**76,242 (3.984 s)**	**1.000**	**1.000**
FN4	1.76e+01 (7.44e+00)	95,324 (48.447 s)	0.781	0.900	**0.00e+00 (0.00e+00)**	**77,482 (4.564 s)**	**1.000**	**1.000**
FN5	4.94e+00 (2.22e+00)	99,812 (49.180 s)	0.999	0.858	**6.88e−05 (1.40e−04)**	**82,128 (4.421 s)**	**1.000**	**1.000**
FN6	1.62e+03 (5.73e+02)	98,496 (48.897 s)	0.036	0.390	**4.13e−01 (9.66e−01)**	**94,900 (5.228 s)**	**1.000**	**1.000**
FN7	8.68e−09 (3.49e−09)	90,288 (45.223 s)	**1.000**	**1.000**	**1.93e−35 (9.53e−35)**	**53,814 (2.465 s)**	**1.000**	**1.000**
FN8	−3.66e+01 (2.45e+00)	90,250 (45.690 s)	0.984	0.897	**−3.92e+01 (5.68e−05)**	**75,756 (4.016 s)**	**1.000**	**1.000**
FN9	3.82e−08 (3.19e−08)	91,670 (45.966 s)	**1.000**	**1.000**	**0.00e+00 (0.00e+00)**	**71,386 (3.256 s)**	**1.000**	**1.000**
FN10	6.13e−10 (2.28e−10)	95,200 (47.429 s)	**1.000**	**1.000**	**6.92e−14 (4.89e−13)**	**79,666 (3.658 s)**	**1.000**	**1.000**

5.2. Convergence analysis

One of the important issues on the performance of the algorithm is its convergence behaviour and we compared the classic ALO algorithm and the proposed TALO algorithm for benchmark functions. Three benchmark functions (FN1, FN4 and FN6) were taken to show their convergence behaviors. Fig. 3 shows the convergence graphs of three benchmark functions FN1, FN4 and FN6 for ALO and TALO algorithms.

In left column of this figure, the average best of cost/objective function values on Y-axis are plotted against the number of iterations on X-axis. In right hand of this figure, the convergence curves of both algorithms are depicted on log scale. It is clearly evident from these graphs that the proposed TALO algorithm starts to converge earlier than the ALO algorithm. We used two stopping criteria in ALO and TALO algorithms, one of which is to stop with convergence as given in Eq. (21). As can be seen from logarithmic scale graphs, TALO algorithm stops the optimization process earlier than standard ALO algorithm due to the convergence stopping criterion.

This convergence behaviour of the proposed TALO algorithm provides that the tournament selection and the other improvements on the algorithm ensure to search the candidate solutions which are nearer to global optimum point. As a result, the proposed TALO algorithm has the better convergence rate than the standard ALO algorithm for benchmark functions.

5.3. Statistical analysis

We used the Wilcoxon ranksum test that is one of the nonparametric tests to compare the proposed TALO algorithm and standard ALO algorithm in terms of statistical significance. Wilcoxon ranksum test is very popular statistical analysis method to evaluate the metaheuristic algorithms. This statistical test is used to compare two samples or repeated measurements on a single sample. In this study, we used a confidence level of 0.95 for statistical analysis. Table 3 summarizes the Wilcoxon ranksum test results from a pair of samples for two algorithms of 50 independent studies to test the null hypothesis for benchmark functions in 10 dimensions. In this table, the outcome comprises three different signs. '+' represents the significant statistical difference at 0.05 level of significance, '−' stands for no significant difference and '=' denotes that the sample pair is same. According to obtained p values, the proposed TALO algorithm is statistical significant in comparison with the classic ALO algorithm for 9 benchmark functions except of FN3.

5.4. Search history analysis

To show the searching ability of the antlions in the proposed TALO algorithm, the positions of the search agents were examined during the optimization process for some benchmark functions (FN1, FN5, FN6, FN8). Fig. 4 shows the search history of the TALO algorithm with ALO algorithm as contour plot.

This analysis indicates the search agents' movements on the surface of the objective function. In these subfigures, asterisk symbol denotes the antlions' positions of TALO algorithm, and blue square symbol denotes the positions of the antlions updated by ALO algorithm. These graphs show the distribution of the populations obtained in each of 100 iterations in the search space of the antlions whose positions are updated throughout the optimization. As can be seen from these subfigures, search agents of the proposed TALO algorithm are more efficient to search promising region of the search space in comparison with ALO algorithm.

5.5. Trajectory analysis

In this analysis, we examined the position of elite antlion during the optimization. Trajectory analysis shows how the TALO determines the nearby position of global solution point. Figs. 5 and 6 show the trajectory analysis results for FN3 and FN7. In these figures, there are three plots for each benchmark function. On the left side of the figure, the positions of the elite antlion for both algorithms were given on the contour surface of the benchmark function. On the right side of these figures, two subfigures shows the position changes of the elite antlion on x-axis and y-axis from start to end of the optimization. It is clearly observed that thanks to using the tournament selection and the other improvements, elite antlion in the TALO realizes superior exploration than that in ALO. At early iterations, the candidate antlions quickly come close to the elite antlion in TALO algorithm. At the later iterations, elite antlion's position exhibits the steady trend at the global solution positions.

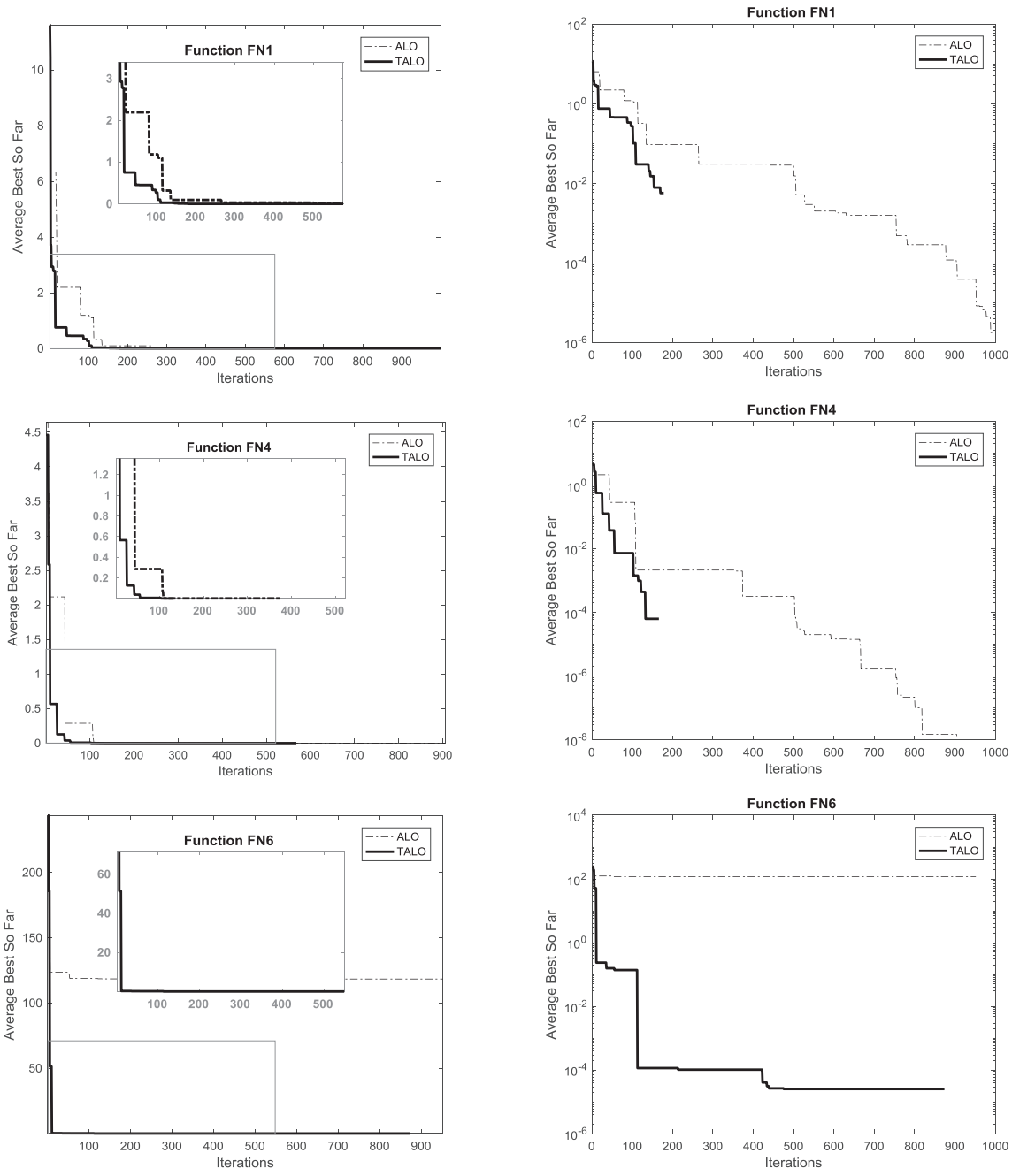

Fig. 3. Convergence analysis of TALO algorithm.

Table 3
Wilcoxon ranksum test results for benchmark functions.

	TALO vs ALO	
Function	*p* values	Outcome
FN1	4.7330e−20	+
FN2	6.6327e−20	+
FN3	9.7300e−02	−
FN4	9.1122e−20	+
FN5	7.9688e−18	+
FN6	8.8626e−16	+
FN7	6.6308e−20	+
FN8	2.2041e−05	+
FN9	3.3111e−20	+
FN10	4.7330e−20	+

5.6. Average distance analysis

To show the exploratory or exploitative search behaviors of the proposed TALO algorithm, we used the average distance analysis between initial and the updated positions of the antlions. In this analysis, the distances between search agents (antlions) for D dimensional search space are calculated in Eq. (22) known as Euclidean distance:

$$dist_{X,Y} = \| X, Y \| = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2} \tag{22}$$

This calculation method can be simplified for a one-dimensional search space:

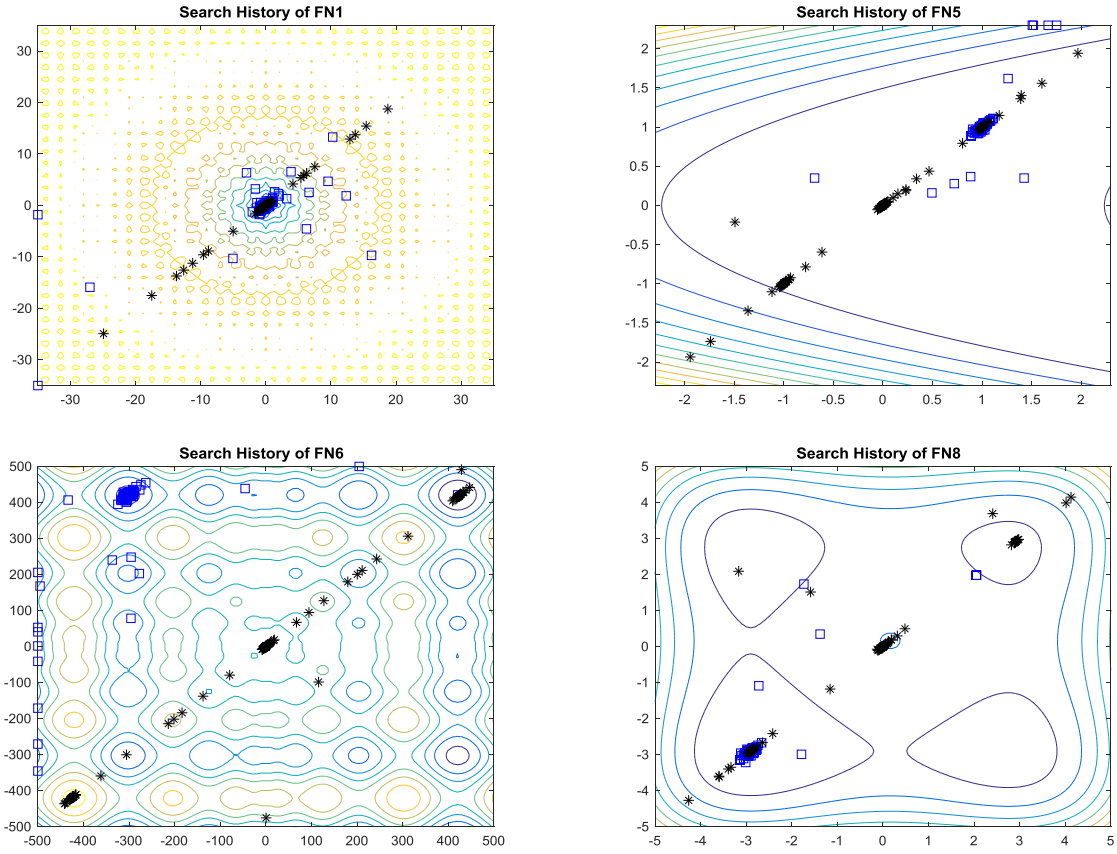

Fig. 4. Search history analysis of TALO algorithm (*: TALO, □: ALO).

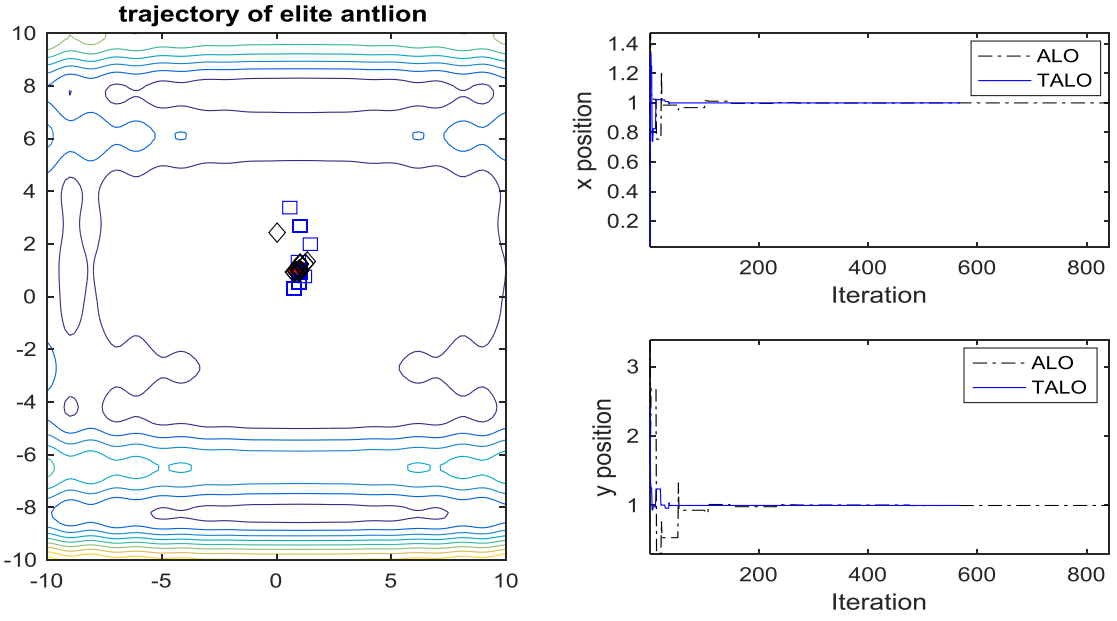

Fig. 5. Trajectory of elite antlion for FN3 benchmark function (◇: TALO, □: ALO).

$$dist_{X,Y} = \| X, Y \| = |x - y| \tag{23}$$

For this analysis, we used the absolute value of the average distance in each iteration for the proposed TALO algorithm and compared with classic ALO algorithm. For 8 benchmark functions, the average distance analysis results were performed between first dimension of first antlion and those of the rest antlions in the population. In Fig. 7, the average distance analysis results obtained by both algorithms are shown for benchmark functions. It can be seen that the proposed TALO algorithm converges faster than ALO algorithm. In the first 100 iterations, ALO algorithm fluctuates in the certain range. But, it is observed from the average distance curves that the TALO algorithm has no fluctuation and provides significant closer solutions to each other in the first 100 iterations. As a result of this analysis, it can be said that the proposed TALO

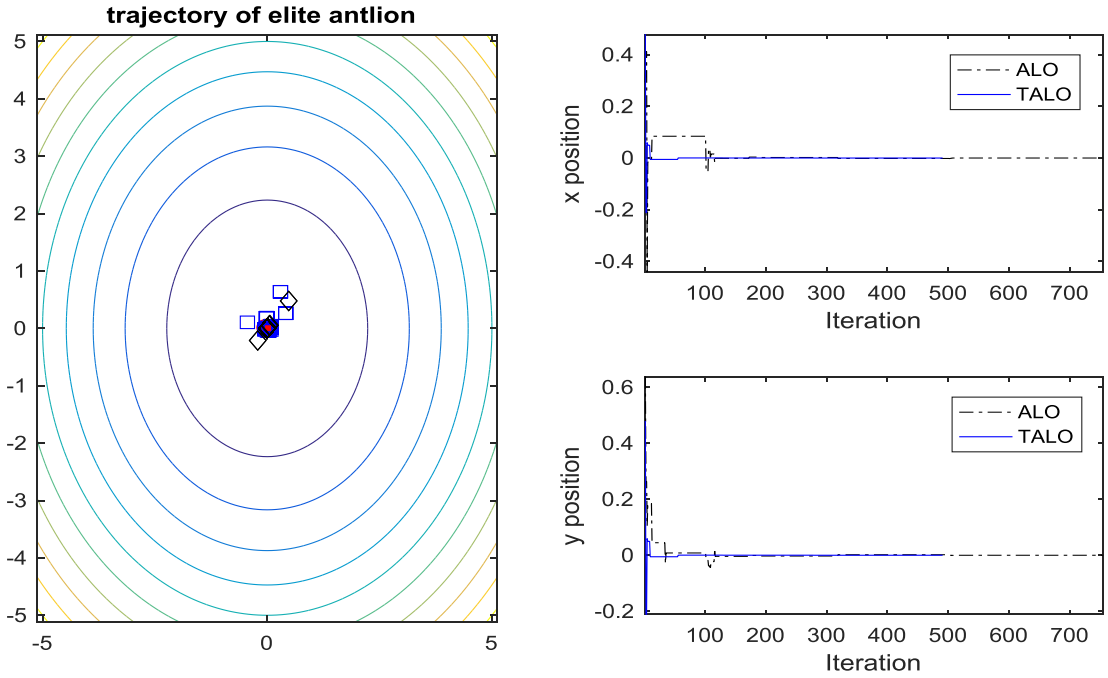

Fig. 6. Trajectory of elite antlion for FN7 benchmark function (◇: TALO, □: ALO).

algorithm provides the fast convergence thanks to the tournament method and the proposed improvements on the classic ALO algorithm.

5.7. Computational complexity analysis

Algorithm complexity is an important metric used to evaluate the algorithm's power. We realized time complexity analysis of the proposed TALO algorithm. In time complexity analysis, the scenario of worst-case complexity is considered. The time complexity is found according to the population size, iteration number, number of loops, function evaluations. The time complexity analysis for classic ALO and proposed TALO algorithm is defined below.

The initialization of the antlions' positions in the population of size NP for both algorithms in step 1 has time complexity of *O(NP)*. In step 2, the cost values of the antlions are calculated with time complexity *O(NP)*O(F(x))* where *F(x)* represents the objective/cost function.

The main loop (while) starts from step 4 and it stops with that the iteration reached the maximum iteration. In this loop, the complexity of each line is multiplied by maximum iteration ($Iter_{max}$) according to the scenario of worst-case. In the other loop (*for (each antlion)*) at step 5, the steps in this loop are executed *NP* times with time complexities ($Iter_{max}$**NP*) for both algorithms. Generating the ant's random walk is realized for each dimension (*D*) of antlion's position, so time complexities of the lines in steps 8–10 are determined as $O(Iter_{max}*NP*D)$. Calculating the position of all ants in step 11 was executed with time complexity $O(Iter_{max}*NP)$. The cost values of the ants with updated positions are calculated with time complexity $O(Iter_{max}*NP)*$ *O(F(x))*. As a result of the complexity analysis, the time complexity of worst-case scenario for both algorithms (ALO and TALO) occurs to $O(Iter_{max}*NP*D)*$ *O(F(x))*.

5.8. Comparison results with other ALO algorithms

In this subsection, the proposed TALO algorithm is compared with the other ALO versions for benchmark functions. Binary ALO [32] and chaotic ALO [33] versions are used in the comparison. In Figs. 8–10, the comparison results are shown for all benchmarks. We used three binary variants of ALO (bALO-1/S/V) and five chaotic variants of ALO (CALO).

It is clearly evident from these graphs that the proposed TALO algorithm has the best performance except of FN3 and FN5 functions. For FN3, the bALO-V exhibits superiority over the proposed TALO, CALO variants and other bALO variants. CALO-Singer and CALO-Tent show better convergence than the proposed TALO algorithm for FN5.

Table 4 summarizes the comparison results of the proposed TALO, binary ALO variants and chaotic ALO variants. In this table, there are four statistical metrics: mean, best, standard deviation and worst. It is clear from the comparison table that the proposed TALO algorithm outperforms the bALO and CALO variants for most benchmarks. For only FN3 and FN5, the proposed TALO has not the best performance for worst, stdev and mean metric values.

The number of function evaluation (NFE) and CPU time results are given in Table 5. It is can be clearly observed from table results that the proposed TALO algorithm has the best CPU time and NFE values in comparison with the bALO and CALO variants. As can be seen from this table results, bALO-S and bALO-V have worst CPU time values for all benchmark functions. Thanks to the proposed improvements on TALO algorithm, the best candidate solutions reach the global optima in short time.

5.9. QAP comparison results

5.9.1. Comparison with ALO, GA, PSO and FOA

In this study, QAP instance has been taken from www.yarpiz.com web site [31]. This instance has been solved by TALO algorithm and it's result has been compared with the several well-known meta-heuristic algorithms, such as ALO algorithm, Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Firefly Optimization Algorithm (FOA). This problem includes the $W[20 \times 20]$ weight matrix and $D[20 \times 20]$ distance matrix. This problem comprises three different special situations. First of all, the 19th and 20th facilities must be as close as possible, then, the 11th and 16th facilities must be as close as possible. Finally,

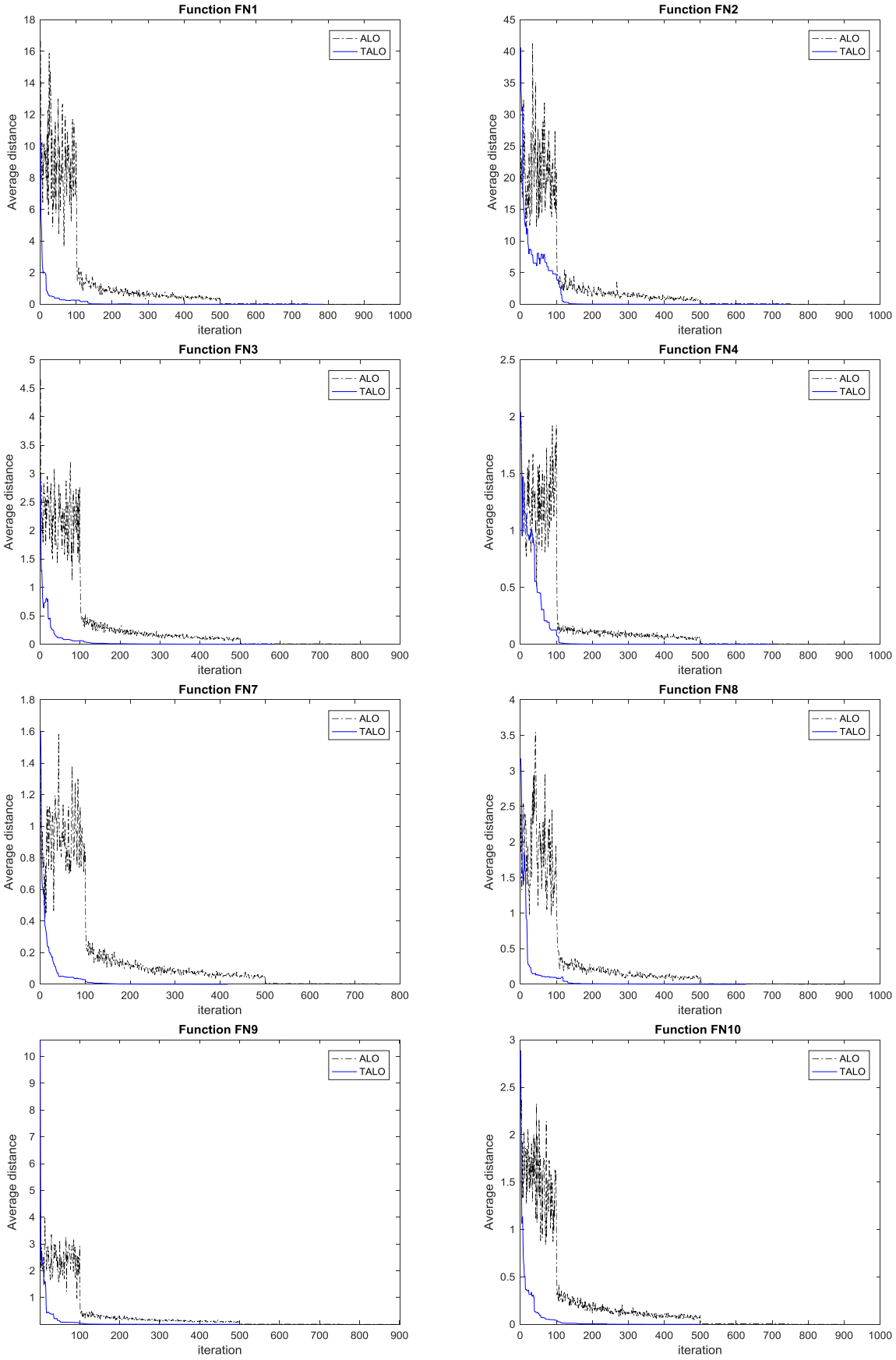

Fig. 7. Average distance analysis between antlions for benchmark functions.

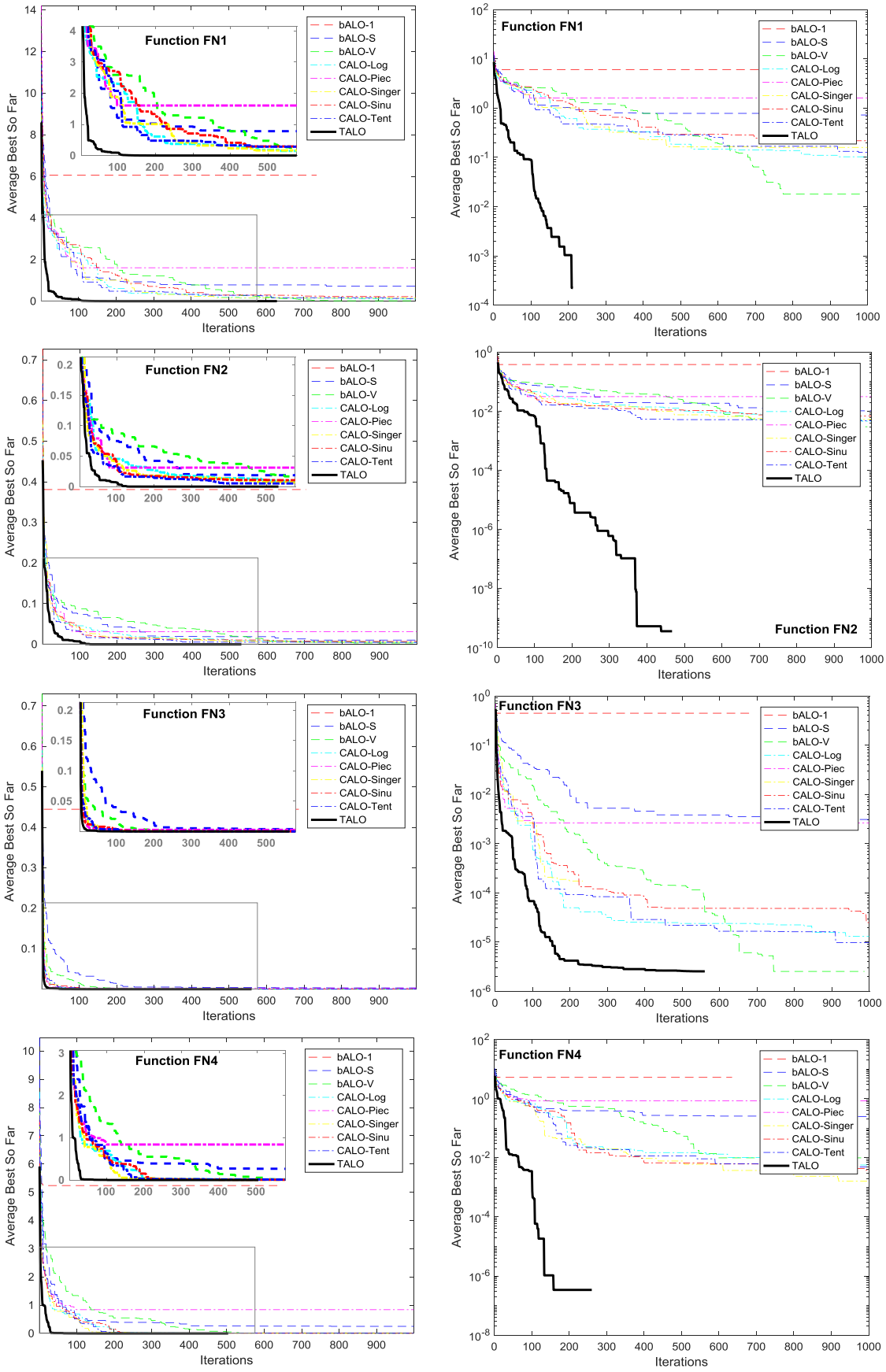

Fig. 8. Convergence curves of TALO and other ALO versions (FN1-FN4).

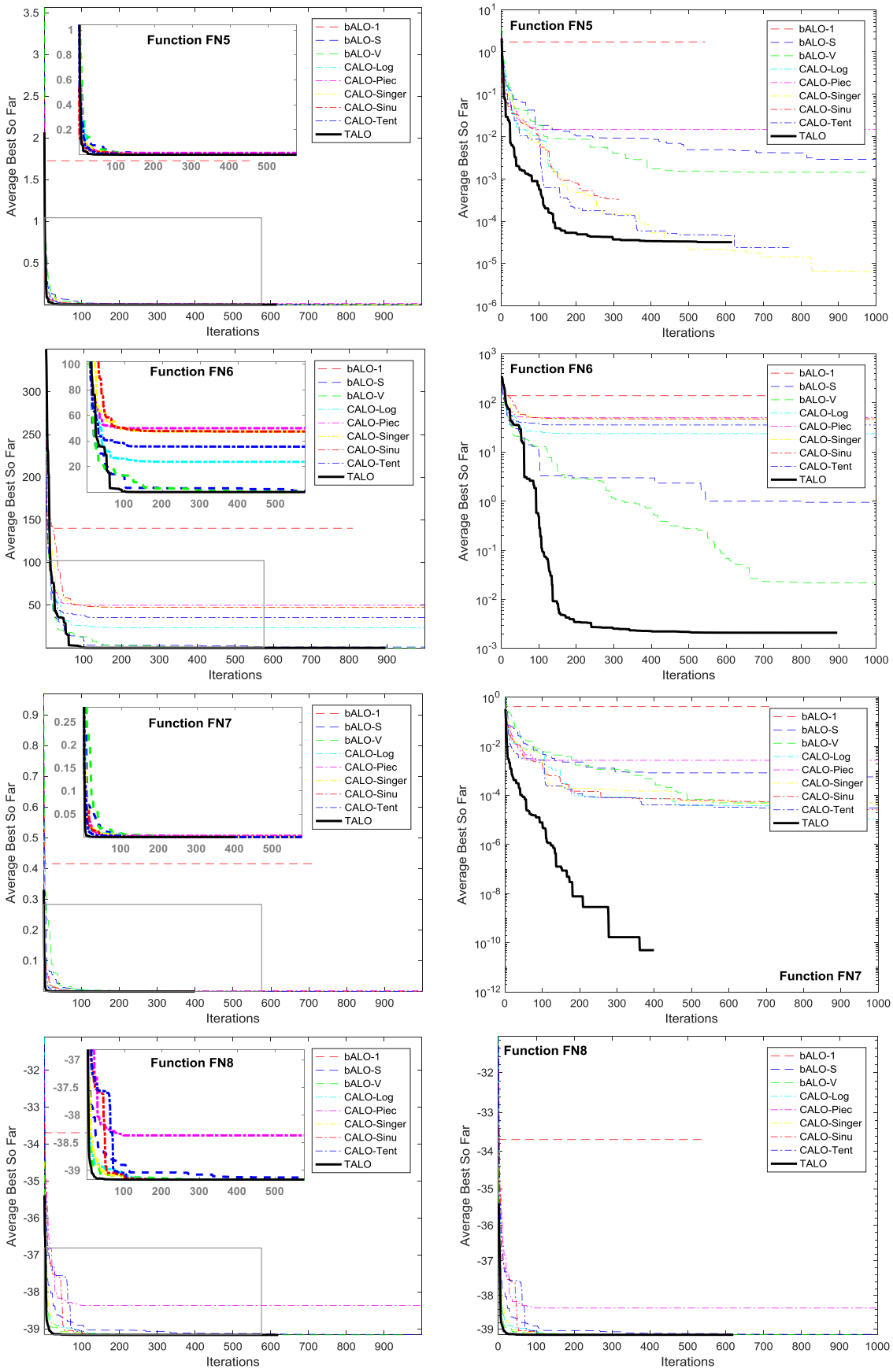

Fig. 9. Convergence curves of TALO and other ALO versions (FN5-FN8).

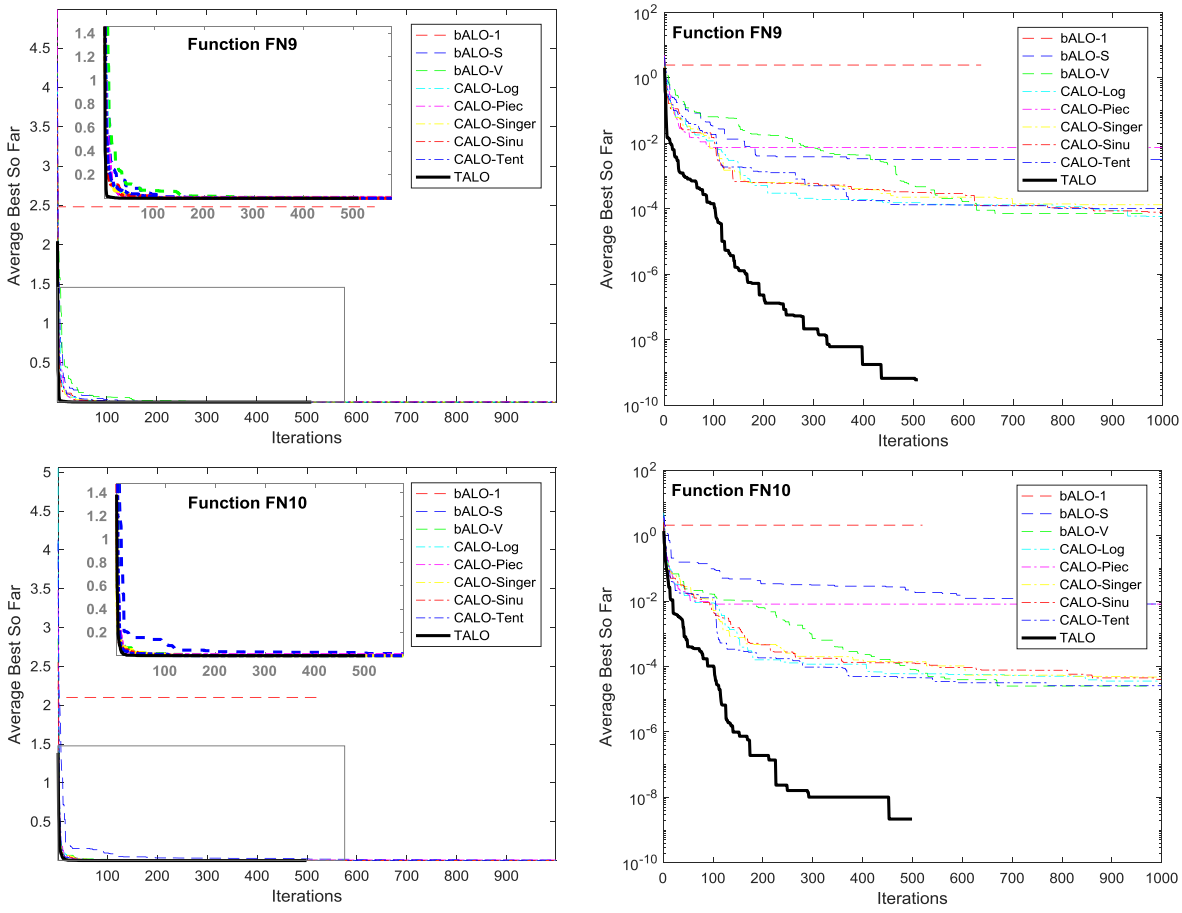

Fig. 10. Convergence curves of TALO and other ALO versions (FN9 & FN10).

the 1st and 13th facilities must be as far as possible. These three critical states are indicated in the weight matrix as follows:

$$w(19,20) = w(20,19) = 10000$$

$$w(11,16) = w(16,11) = 10000$$

$$w(1,13) = w(13,1) = -10000$$

The codes of TALO and other algorithms have been run on PC with Intel(R) Core(TM) i5-3230 M CPU@2.60 GHz RAM/8. Each algorithm has been run 10 times. Population size is 20, maximum number of iterations is 1000. The parameters of meta-heuristic algorithms used for QAP performance tests are given in Table 6. The locations of this QAP instance are shown in Fig. 11. There are 40 locations *(not assigned facility)* to be used in QAP. The population size set as the number of assigned facilities in QAP. In this work, the population size was used as the half of the number of facilities.

To solve QAP problem, TALO algorithm has been adapted to combinatorial optimization problem. For the QAP scenario used in this study, we assumed that the problem dimension (N_p) equals the number of locations. Initially, TALO algorithm produces the positions of antlions in the range [01] randomly. Then sorting these position values and index values of the sorted positions are used as the locations of facilities in QAP. According to assigned locations of facilities, QAP's total fitness value is calculated using $D[20 \times 20]$ distance matrix and $W[20 \times 20]$ weight matrix. Pseudo code of how to solve QAP by TALO algorithm is given below:

Algorithm 3 (*Pseudo code about solving QAP problem by TALO Algorithm.*).

Input: weight matrix (W), location vectors (x, y), number of locations, number of facilities, candidate solutions produced by TALO.
Output: total cost value.
1) Create facility list from candidate solution produced by TALO
2) Calculate distance between locations
for i:number of locations
for j = i + 1:number of locations
calculate distance (i, j) :$d_{i,j} = \sqrt{(x_i - x_j)^2 + \left(y_i - y_j\right)^2}$
distance (i, j) = distance (j, i)
end for
end for
3) Calculate total cost
cost = 0
for i:number of facilities
for j = i + 1:number of facilities
cost = cost + weight (i, j)*distance(facility(i), facility(j))
end for
end for

Fig. 12 shows the results obtained at the end of one-time run by the TALO and other meta-heuristic algorithms. In these figures, blue squares denote assigned facilities and pink circles denote

Table 4
Comparison results with 10 independent runs for benchmark functions.

Fn	Method	bALO-1	bALO-S	bALO-V	CALO Log	CALO Piec	CALO Singer	CALO Sinu	CALO Tent	TALO
FN1	Mean	6.062e+00	7.251e−01	1.806e−02	1.019e−01	1.606e+00	1.593e−01	2.173e−01	1.259e−01	**0.000e+00**
	Best	2.597e+00	1.806e−02	1.806e−02	1.349e−02	1.689e−01	3.020e−02	4.679e−02	1.786e−02	**0.000e+00**
	StDev	2.632e+00	7.201e−01	**0.000e+00**	9.199e−02	1.149e+00	8.692e−02	1.918e−01	9.323e−02	**0.000e+00**
	Worst	11.177e+00	1.841e+00	1.806e−02	3.084e−01	3.134e+00	2.663e−01	6.949e−01	2.744e−01	**0.000e+00**
FN2	Mean	3.809e−01	1.033e−02	2.997e−03	4.608e−03	3.118e−02	5.279e−03	6.917e−03	4.736e−03	**0.000e+00**
	Best	1.164e−01	7.524e−04	6.989e−06	3.925e−04	2.964e−03	5.051e−04	5.519e−05	1.795e−05	**0.000e+00**
	StDev	2.519e−01	7.979e−03	3.860e−03	3.650e−03	2.081e−02	3.141e−03	3.095e−03	3.339e−03	**0.000e+00**
	Worst	9.594e−01	2.376e−02	7.559e−03	1.084e−02	7.669e−02	8.865e−03	1.034e−02	9.006e−03	**0.000e+00**
FN3	Mean	4.446e−01	3.083e−03	**2.545e−06**	1.319e−05	2.635e−03	8.681e−05	2.495e−05	9.819e−06	2.552e−06
	Best	4.503e−03	1.365e−04	2.545e−06	2.012e−07	3.281e−06	2.998e−07	4.693e−08	4.185e−07	**4.329e−31**
	StDev	4.399e−01	2.782e−03	**4.464e−22**	1.691e−05	2.351e−03	1.724e−04	3.582e−05	8.923e−06	4.148e−06
	Worst	1.265e+00	8.312e−03	**2.545e−06**	4.622e−05	6.652e−03	5.483e−04	1.113e−04	2.543e−05	1.351e−05
FN4	Mean	5.237e+00	2.503e−01	9.938e−03	5.674e−03	8.403e−01	1.605e−03	3.759e−03	4.852e−03	**0.000e+00**
	Best	2.046e+00	9.938e−03	9.938e−03	2.387e−04	3.315e−02	2.294e−04	2.645e−04	7.119e−04	**0.000e+00**
	StDev	3.177e+00	3.882e−01	**0.000e+00**	7.689e−03	4.954e−01	1.553e−03	3.944e−03	5.735e−03	**0.000e+00**
	Worst	1.143e+01	9.952e−01	9.938e−03	2.363e−02	1.419e+00	5.419e−03	1.294e−02	1.939e−02	**0.000e+00**
FN5	Mean	1.726e+00	2.878e−03	1.433e−03	5.666e−04	1.459e−02	**6.461e−06**	5.214e−05	1.743e−05	3.205e−05
	Best	4.829e−02	1.743e−03	1.407e−03	1.150e−06	4.399e−04	1.702e−08	5.327e−06	4.424e−07	**0.000e+00**
	StDev	2.569e+00	8.189e−04	8.241e−05	1.755e−03	1.983e−02	**9.914e−06**	6.120e−05	2.362e−05	3.650e−05
	Worst	8.767e+00	4.383e−03	1.667e−03	5.561e−03	5.227e−02	**3.258e−05**	1.651e−04	8.178e−05	9.998e−05
FN6	Mean	1.401e+02	9.012e−01	2.159e−02	2.369e+01	5.001e+01	4.741e+01	4.740e+01	3.555e+01	**2.091e−03**
	Best	1.551e+00	4.612e−02	3.840e−05	1.892e−03	1.021e−01	2.037e−03	1.797e−03	1.937e−03	**2.546e−05**
	StDev	9.568e+01	5.985e−01	4.445e−02	4.994e+01	6.181e+01	6.116e+01	6.116e+01	5.719e+01	**3.948e−03**
	Worst	2.748e+02	1.758e+00	1.059e−01	1.185e+02	1.258e+02	1.185e+02	1.185e+02	1.184e+02	**9.802e−03**
FN7	Mean	4.156e−01	5.711e−04	5.009e−05	1.089e−05	2.707e−03	4.852e−05	2.596e−05	3.097e−05	**4.992e−11**
	Best	3.452e−02	5.009e−05	5.009e−05	1.893e−07	2.884e−04	3.787e−07	3.953e−08	4.274e−07	**0.000e+00**
	StDev	3.594e−01	9.168e−04	**1.429e−20**	1.313e−05	3.182e−03	9.013e−05	4.247e−05	3.670e−05	1.579e−10
	Worst	1.306e+00	3.056e−03	5.009e−05	3.891e−05	1.081e−02	2.786e−04	1.371e−04	1.127e−04	**4.992e−10**
FN8	Mean	−3.369e+01	−3.915e+01	−3.917e+01	**−3.916e+01**	−3.837e+01	**−3.916e+01**	**−3.916e+01**	−3.915e+01	**−3.916e+01**
	Best	−3.902e+01	**−3.917e+01**	**−3.917e+01**	**−3.917e+01**	−3.915e+01	**−3.917e+01**	**−3.917e+01**	**−3.917e+01**	**−3.917e+01**
	StDev	4.441e+00	1.522e−02	**0.000e+00**	3.891e−03	2.219e+00	2.697e−03	3.135e−03	3.771e−02	2.616e−04
	Worst	−2.707e+01	−3.911e+01	−3.917e+01	−3.891e−03	−3.206e+01	**−3.916e+01**	**−3.916e+01**	−3.905e+01	−3.917e+01
FN9	Mean	2.482e+00	3.241e−03	7.159e−05	5.804e−05	7.468e−03	1.332e−04	7.907e−05	1.024e−04	**5.962e−10**
	Best	4.904e−01	7.159e−05	7.159e−05	1.339e−07	2.155e−03	6.511e−06	5.330e−06	8.444e−06	**0.000e+00**
	StDev	2.338e+00	3.939e−03	1.429e−20	6.138e−05	4.231e−03	1.142e−04	1.106e−04	8.877e−05	**1.885e−09**
	Worst	7.329e+00	1.076e−02	7.159e−05	1.884e−04	1.685e−02	3.638e−04	3.671e−04	2.442e−04	**5.962e−09**
FN10	Mean	2.101e+00	8.166e−03	2.536e−05	3.598e−05	8.110e−03	4.779e−05	4.071e−05	2.595e−05	**1.021e−09**
	Best	8.318e−02	6.340e−04	2.536e−05	4.993e−07	2.160e−04	9.029e−07	6.757e−06	1.779e−06	**0.000e+00**
	StDev	2.220e+00	1.048e−02	3.571e−21	4.222e−05	6.883e−03	6.599e−05	2.941e−05	2.299e−05	**3.229e−09**
	Worst	7.227e+00	3.577e−02	2.536e−05	1.156e−04	2.038e−02	2.038e−04	9.110e−05	6.898e−05	**1.021e−08**

Table 5
NFE/CPU Time results with 10 independent runs for benchmark functions.

	Number of Function Evaluation (NFE), (CPU time)								
Fn	bALO-1	bALO-S	bALO-V	CALO Log	CALO Piec	CALO Singer	CALO Sinu	CALO Tent	TALO
FN1	**16,156**	20,000	19,952	20,000	20,000	20,000	20,000	20,000	16,658
	5.49 s	110.73 s	100.72 s	10.50 s	10.48 s	11.26 s	10.55 s	10.54 s	**2.84 s**
FN2	16,256	20,000	19,922	20,000	20,000	20,000	20,000	20,000	**12,862**
	4.97 s	117.32 s	78.27 s	10.59 s	10.61 s	11.21 s	10.41 s	10.32 s	**0.88 s**
FN3	14,766	20,000	19,850	20,000	20,000	18,470	20,000	20,000	**12,178**
	1.56 s	32.84 s	32.12 s	3.87 s	3.86 s	3.56 s	3.99 s	3.84 s	**0.71 s**
FN4	15,650	20,000	19,896	20,000	20,000	20,000	20,000	20,000	**12,572**
	1.74 s	30.88 s	30.30 s	3.94 s	3.94 s	3.95 s	3.99 s	3.96 s	**0.90 s**
FN5	15,560	20,000	19,866	17,536	20,000	20,000	15,226	19,542	**14,362**
	1.57 s	27.60 s	27.95 s	3.32 s	3.77 s	3.80 s	2.87 s	3.69 s	**0.83 s**
FN6	**16,908**	20,000	19,980	20,000	20,000	20,000	20,000	20,000	18,192
	1.88 s	50.13 s	49.60 s	3.86 s	3.86 s	3.88 s	4.01 s	3.89 s	**1.10 s**
FN7	15,718	20,000	19,864	20,000	20,000	20,000	20,000	20,000	**9408**
	1.58 s	30.30 s	29.98 s	3.68 s	3.80 s	3.67 s	3.67 s	3.67 s	**0.50 s**
FN8	14,630	20,000	19,694	12,170	20,000	15,660	12,396	**7920**	14,248
	1.68 s	31.27 s	30.04 s	2.56 s	4.37 s	3.01 s	2.38 s	1.52 s	**0.83 s**
FN9	15,198	20,000	19,822	20,000	20,000	20,000	20,000	20,000	**10,564**
	1.51 s	33.05 s	32.44 s	3.70 s	3.68 s	3.70 s	3.73 s	3.74 s	**0.55 s**
FN10	14,320	20,000	19,758	20,000	20,000	20,000	20,000	20,000	**10,562**
	1.46 s	33.79 s	33.03 s	3.88 s	3.80 s	4.08 s	3.87 s	3.86 s	**0.56 s**

Table 6
Parameters of meta-heuristic algorithms used for solving QAP.

Algorithm	Parameters
Genetic Algorithm (GA)	Crossover Coefficient: 0.4 Mutation Coefficient: 0.8 Selection Pressure Coefficient: 5
Particle Swarm Optimization (PSO)	Inertia Weight: 1.0 Inertia Weight Damping Ratio: 0.99 Personal Learning Coefficient: 1.5 Global Learning Coefficient: 2.0
Firefly Optimization Algorithm (FOA)	Light Absorption Coefficient: 1.0 Initial Attraction Coefficient: 2.0 Mutation Coefficient: 0.2 Mutation Coefficient Damping R.: 0.98
Antlion Optimization Algorithm (ALO)	Number of Antlions: 20
Tournament Selection based Antlion Optimization Algorithm (TALO)	Number of Antlions: 20 Tournament Size: 2

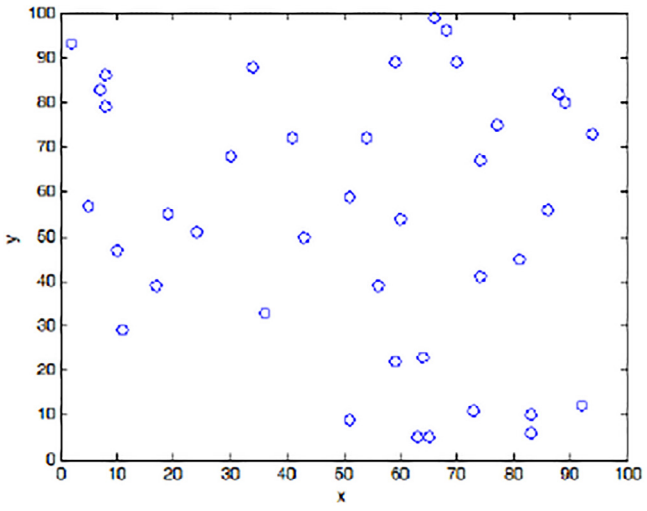

Fig. 11. Locations used for quadratic assignment problem.

empty location (not assigned facility). TALO result has the best cost value (-1049377.204) among all results. According to the results of all meta-heuristic algorithms, facility pairs (19–20), (11–16) are shown to be at close locations and facility pairs (1,13) be at far locations from each other. In Fig. 13, the convergence curves are shown for TALO and meta-heuristic algorithms used in QAP test.

For the comparison results with 10 independent runs of TALO and the other meta-heuristic algorithms, we used four metrics: mean cost, standard deviation, best cost and worst cost. Fig. 14 shows the best cost curves obtained by TALO algorithm for each runs. Table 7 summarizes QAP results with 10 independent runs. Fig. 15 shows the statistical results regarding the performances of TALO algorithm and other meta-heuristic algorithms for 10 independent runs.

As can be seen from Table 7 and Fig. 15, the worst algorithm is ALO algorithm in terms of all metrics. The proposed TALO algorithm has the best cost value in comparison with other algorithms.

5.9.2. Comparison with ALO variants

In this section, we present the comparison result with the proposed TALO algorithm and the other ALO variants taken from [32,33] for QAP example given in the previous section. Firstly, binary ALO and chaotic ALO variants were adapted to QAP example. In this comparison work, three binary variants of ALO (bALO-1, bALO-S, bALO-V) and five chaotic variants of ALO (CALO-Log, CALO_Piec, CALO_Singer, CALO_Sinu, CALO_Tent) were used. The comparison results are taken for 1000 as maximum iteration and 20 as population size. The convergence curves of the proposed TALO algorithm, binary ALO and chaotic ALO variants are shown in Fig. 16. It is obvious from these convergence graphs that the proposed TALO has the best convergence for QAP. Table 8 presents the obtained results with 10 runs of TALO and ALO variants. The table results are observed in terms of mean cost, standard deviation, best and worst cost.

The obtained results clearly show that the proposed improvements of TALO algorithm provide the best values on mean cost, worst cost and mean cost metrics. For the standard deviation metric, the bALO-S algorithm is the best, but the proposed TALO algorithm has the second best standard deviation value. The low standard deviation of TALO algorithm show that the repetitive results tend to be close the mean cost. Fig. 17 shows the statistical results of TALO, bALO variants and CALO variants with 10 runs. It is shown from this figure that the best binary ALO variant is bALO-V and the best chaotic ALO variant is CALO-Log algorithm.

It is clearly evident from this figure that the performance of the proposed TALO algorithm is better than those of binary ALO and chaotic ALO variants. The reason of this is due to the integration of the tournament selection and improvements on updating mechanism of ALO.

Finally, we analyzed the solution quality of proposed TALO algorithm for solving QAP using some instances presented in QAPLIB site [38]. The statistic results were obtained by the proposed TALO algorithm from 10 runs for some QAP instances presented by Taillard [39]. In Table 9, the information about QAP instances (size, solution) and statistical results (mean, best, worst, best error) are summarized. The results were obtained for 4×dimension as maximum iteration and 150 as population size. The best error is calculated by the following equation:

$$besterror = 100 \times \left(\frac{solution - optimalsolution}{optimalsolution} \right) \quad (24)$$

As can be shown from this table result, the best error percentage values of the proposed TALO algorithm are between 0.000 and 5.263.

6. Conclusion

Antlion optimization algorithm is one of the meta-heuristic algorithms, proposed in recent years, but it has some drawbacks, such as long runtime during the optimization process. In this study, antlion optimization algorithm is developed by some improvements and innovations on ALO algorithm. The improvements on original ALO algorithm are essentially about the random walking mechanism and selection method. In the selection mechanism of antlion used for the random walk model, the tournament selection method is implemented instead of the roulette wheel method into ALO algorithm. In these improvements, new movements were also defined between lower and upper boundaries around the antlion in the phase of trapping in antlion pits. These

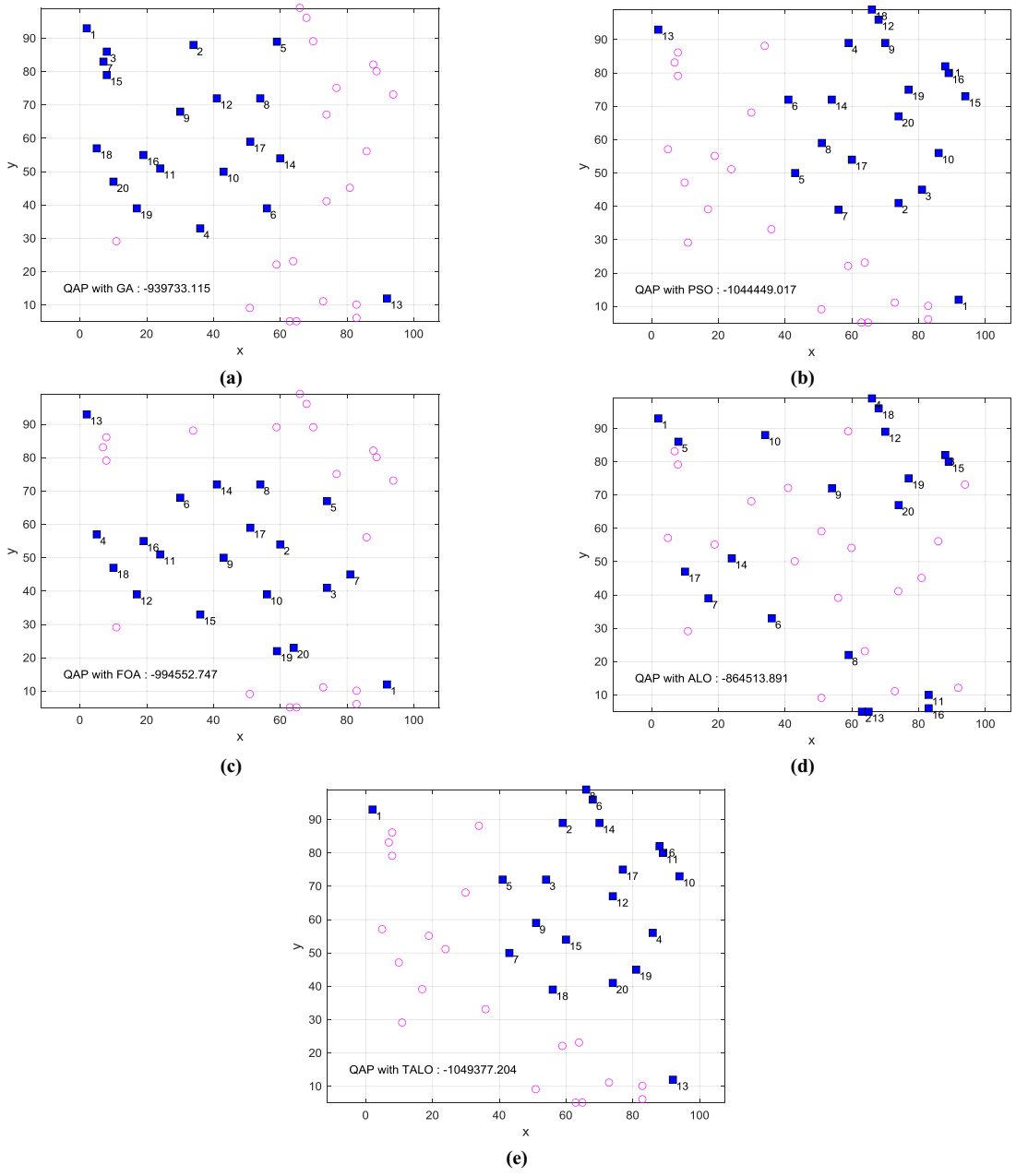

Fig. 12. QAP Results obtained by meta-heuristic algorithms, (a) GA, (b) PSO, (c) FA, (d) ALO, (e) TALO.

movements provide that ants walk more effectively around the selected antlion in the search space.

In this study, the tournament selection based ALO algorithm (TALO) was presented and the proposed TALO algorithm was adapted to the difficult combinatorial optimization problem known as quadratic assignment problem (QAP). To compare the performance of TALO and ALO algorithms, a benchmark study was performed on some functions for different metrics. Four metrics were used such as optimality, accuracy, mean best/standard deviation and CPU time/number of function evaluations (NFE). The results show that the proposed TALO has the best performance in terms of all metrics. By the improvements made in the original ALO algorithm, the run time of the TALO algorithm was decreased highly. In terms of the CPU-time/NFE metric results, the run time of the TALO algorithm is better than that of the original ALO algorithm. In detail algorithm analyses, we have performed the convergence analysis, statistical analysis, search history analysis, trajectory analysis, average distance analysis, computational complexity analysis. As a result of these analyzes, it is clearly evident that the proposed improvements on TALO algorithm provide developments on the exploration and exploitation search behaviors. We have compared the proposed TALO with binary ALO and chaotic ALO variants for benchmarks. It is obvious from the comparison results that the proposed TALO outperforms the bALO and CALO variants for benchmarks. Thanks to the proposed improvements of TALO, the best candidate solutions reach the global optima in short time.

To evaluate the algorithm's performance on QAP instance, we have firstly used four well-known meta-heuristic algorithms (ALO, GA, PSO, FOA). Secondly, TALO algorithm has been compared with three binary ALO variants and five chaotic ALO variants for the same QAP instance. The comparison outcomes show that the

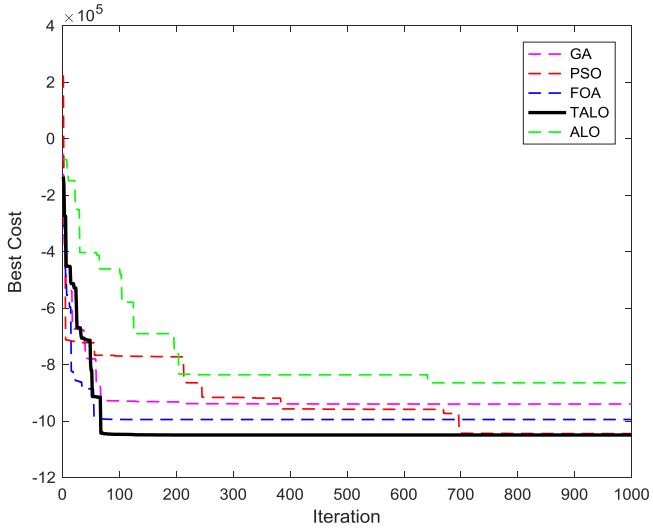

Fig. 13. Comparison results of convergence curves.

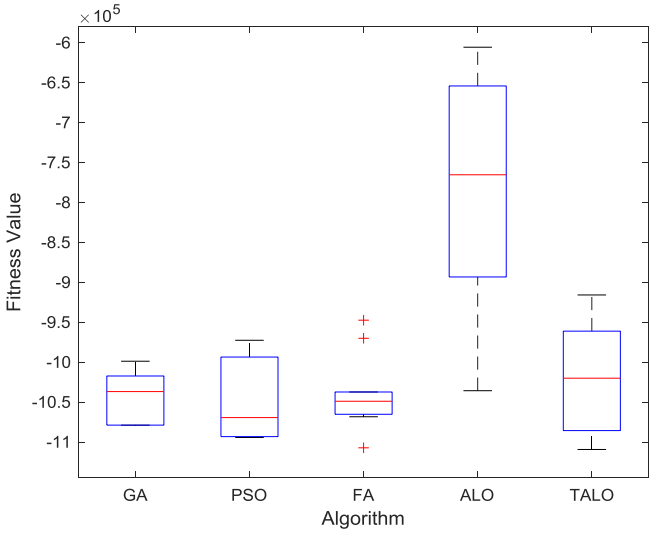

Fig. 15. Statistical results of TALO and other meta-heuristic algorithms with 10 runs.

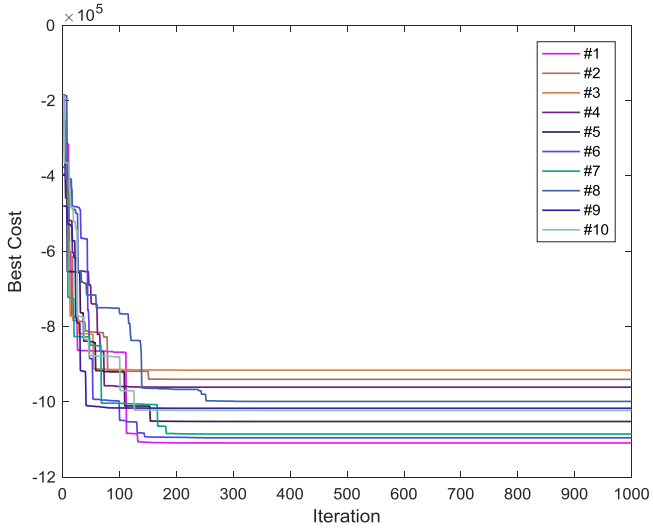

Fig. 14. Convergence curves of TALO algorithm with 10 runs.

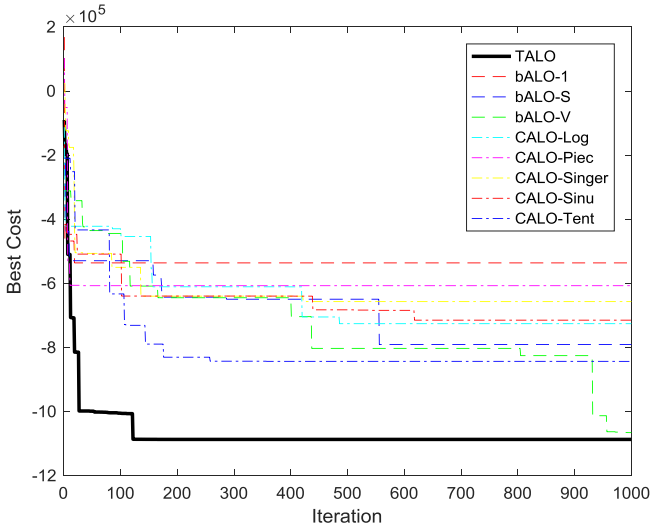

Fig. 16. Convergence curves of ALO variants.

proposed TALO algorithm is able to provide very competitive results well-known meta-heuristic algorithms. The performance of the proposed TALO algorithm is better than that of the original ALO algorithm according to all indicators. In comparison between TALO and ALO variants (bALO, CALO), the results clearly show that the proposed improvements of TALO algorithm provide the best values on mean cost, worst cost and mean cost metrics. As last, we have analyzed the solution quality of proposed TALO algorithm for some QAP instances (Tai*) from in QAPLIB site. The obtained statistical results show that the best error percentage values of the TALO algorithm are in range 0–5.263%.

In the future studies, the useful mechanisms can be investigated to increase the performance of the ALO algorithm. TALO algorithm can be implemented in real optimization problems in different fields, such as optimal robot path planning, bin packing problem, hub location allocation problem, capacitated vehicle routing problem, minimum spanning tree etc.

Table 7
The QAP results with 10 runs of TALO and other meta-heuristic algorithms.

Algorithm	Mean Cost	StDev	Best Cost	Worst Cost
GA	−1040715.59	**32096.99**	−1078899.84	**−998807.47**
PSO	**−1050518.95**	48744.72	−1094306.47	−972582.99
FOA	−1039310.94	47073.92	−1107239.04	−947424.52
ALO	−776989.54	134951.44	−1035716.47	−605587.60
TALO	−1019909.74	66819.23	**−1109319.29**	−915827.81

Table 8
The QAP results with 10 runs of TALO and ALO variants.

Algorithm	Mean Cost	StDev	Best Cost	Worst Cost
bALO-1	−673632.17	88724.09	−790631.35	−498817.24
bALO-S	−816076.00	**66412.94**	−920611.89	−698078.88
bALO-V	−921934.71	103698.37	−1071290.18	−772033.49
CALO-Log	−956141.54	91521.39	−1066654.07	−771761.13
CALO-Piec	−561195.68	78673.02	−649200.76	−412287.52
CALO-Singer	−860704.18	106987.04	−1046642.10	−678161.70
CALO-Sinu	−841341.43	105076.97	−964691.41	−678328.80
CALO-Tent	−857609.68	80066.99	−945542.43	−666862.63
TALO	**−1019909.74**	66819.23	**−1109319.29**	**−915827.81**

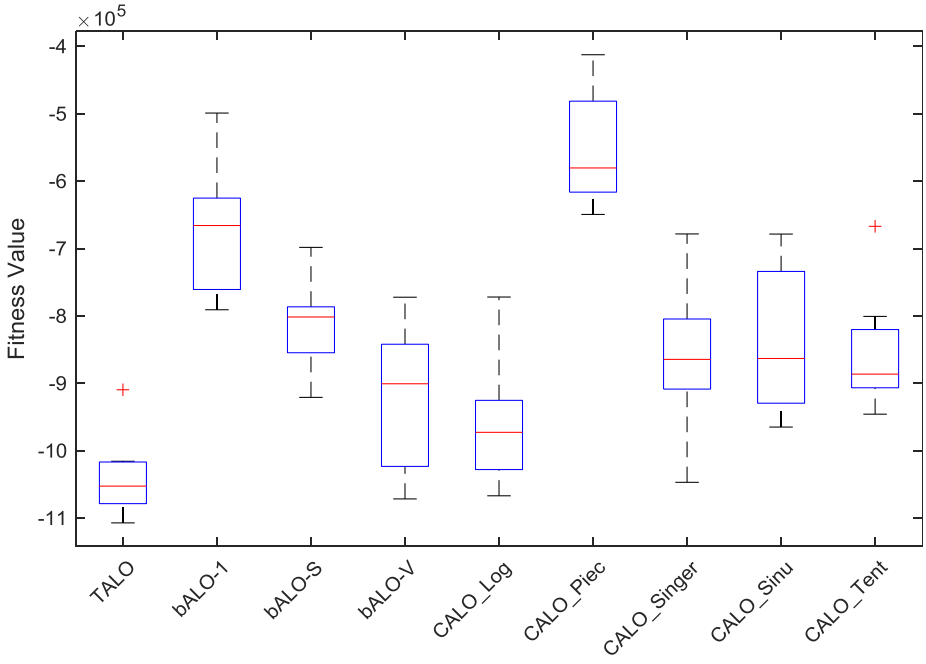

Fig. 17. Statistical results of TALO and other meta-heuristic algorithms with 10 runs.

Table 9
The QAPLIB results with 10 runs of TALO algorithm.

Problem	Size	Solution	Mean	Best	Worst	Best Error
Tai12a	12	224,416	231,761	224,416	241,846	0.000
Tai15a	15	388,214	398,912	391,942	404,600	0.960
Tai17a	17	491,812	510,300	496,598	527,336	0.973
Tai20a	20	703,482	734,013	722,346	748,902	2.682
Tai25a	25	1,167,256	1,233,443	1,223,348	1,244,440	4.806
Tai30a	30	1,818,146	1,899,847	1,872,376	1,923,304	2.983
Tai35a	35	2,422,002	2,541,491	2,528,052	2,564,760	4.379
Tai40a	40	3,139,370	3,307,021	3,277,580	3,337,568	4.403
Tai50a	50	4,938,796	5,214,354	5,198,732	5,248,334	5.263
Tai60a	60	7,205,962	7,601,621	7,553,190	7,647,488	4.819

Declaration of interest

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of this article.

References

[1] S.J. Nanda, G. Panda, A survey on nature inspired metaheuristic algorithms for partitional clustering, Swarm Evol. Comput. 16 (2014) 1–18.
[2] Z. Beheshti, S.M.H. Shamsuddin, A review of population-based meta-heuristic algorithm, Int. J. Adv. Soft Comput. Appl. 5 (1) (2013).
[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-We., 1989.
[4] J. Kennedy, R. Eberhart, "Particle swarm optimization", Neural Networks, Proceedings., IEEE Int. Conf. 4 (1995) 1942–1948.
[5] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.
[6] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. J. 8 (1) (2008) 687–697.
[7] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132.
[8] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57.
[9] R. Storn, K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.

[10] K.V. Price, R.M. Storn, J.A. Lampinen, Differential evolution: a practical approach to global, Optimization vol (2005) 28.
[11] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39.
[12] T.C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica 25 (1) (1957) 53.
[13] T. Peng, W. Huanchen, Z. Dongme, Simulated annealing for the quadratic assignment problem: a further study, Comput. Ind. Eng. 31 (3–4) (1996) 925–928.
[14] M.R. Wilhelm, T.L. Ward, Solving quadratic assignment problems by 'Simulated Annealing', IIE Trans. 19 (1) (1987) 107–119.
[15] D.M. Tate, A.E. Smith, A genetic approach to the quadratic assignment problem, Comput. Oper. Res. 22 (1) (1995) 73–83.
[16] E.-G. Talbi, O. Roux, C. Fonlupt, D. Robillard, Parallel Ant Colonies for the quadratic assignment problem, Futur. Gener. Comput. Syst. 17 (4) (2001) 441–449.
[17] W. Zhu, J. Curry, A. Marquez, SIMD tabu search for the quadratic assignment problem with graphics hardware acceleration, Int. J. Prod. Res. 48 (4) (2010) 1035–1047.
[18] F. Hafiz, A. Abdennour, Particle swarm algorithm variants for the quadratic assignment problems – A probabilistic learning approach, Expert Syst. Appl. 44 (2016) 413–431.
[19] S. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.
[20] H. Kiliç, U. Yüzgeç, Improved antlion optimization algorithm, 2nd International Conference on Computer Science and Engineering, UBMK, 2017, 2017.
[21] V.K. Kamboj, A. Bhadoria, S.K. Bath, Solution of non-convex economic load dispatch problem for small-scale power systems using ant lion optimizer, Neural Comput. Appl. 28 (8) (2017) 2181–2192.
[22] M. Raju, L.C. Saikia, N. Sinha, Automatic generation control of a multi-area system using ant lion optimizer algorithm based PID plus second order derivative controller, Int. J. Electr. Power Energy Syst. 80 (2016) 52–63.
[23] M. Petrović, Z. Miljković, Biologically inspired optimization algorithms for flexible process planning, Lect. Notes Mech. Eng. (2017) 417–428.
[24] E. Gupta, A. Saxena, Performance evaluation of antlion optimizer based regulator in automatic generation control of interconnected power system, J. Eng. (United States) 2016 (2016).
[25] P. Yao, H. Wang, Dynamic Adaptive Ant Lion Optimizer applied to route planning for unmanned aerial vehicle, Soft Comput. 21 (18) (2017) 5475–5488.
[26] N. Chopra, S. Mehta, Multi-objective optimum generation scheduling using Ant Lion Optimization, 12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3), INDICON 2015, 2016.
[27] S.S. Nair, K.P.S. Rana, V. Kumar, A. Chawla, Efficient modeling of linear discrete filters using ant lion optimizer, Circuits, Syst. Signal Process. 36 (4) (2017).
[28] M. Noraini, J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP", World Congr. Eng., vol. II, no. 978-988-19251-4-5, pp. 4–9, 2011.
[29] N.M. Razali, J. Geraghty, Genetic algorithm performance with different selection strategies in solving TSP, Int. Conf. Comput. Intell. Intell. Syst. (2011).
[30] T. Blickle, L. Thiele, A comparison of selection schemes used in evolutionary algorithms, Evol. Comput. 4 (4) (1996) 361–394.
[31] Yarpiz, Quadratic Assignment Problem (QAP) using GA, PSO and FA. http://yarpiz.com/359/ypap104-quadratic-assignment-problem/ Accessed 8 August 2017.
[32] E. Emary, Hossam M. Zawbaa, Aboul Ella Hassanien, Binary ant lion approaches for feature selection, Neurocomputing Elsevier 213 (2016) 54–65.
[33] Hossam M. Zawbaa, E. Emary, Crina Grosan, Feature selection via chaotic antlion optimization, PLoS ONE 11 (3) (March 2016).
[34] Hossam M. Zawbaa, E. Emary, B. Parv, Feature Selection Based on Antlion Optimization Algorithm, in: 3rd World conference on complex systems (WCCS), pp. 1-7, Morocco, November 2015.
[35] Hossam M. Zawbaa, E. Emary, Crina Grosan, Vaclav Snasel, Large-dimensionality small-instance set feature selection: a hybrid bioinspired heuristic approach, Swarm Evol. Comput. 42 (2018).
[36] E. Emary, Hossam M. Zawbaa, Feature selection via Levy Antlion optimization, Pattern Anal. Appl. (PAAA) (2018).
[37] E. Emary, Hossam M. Zawbaa, Impact of chaos functions on modern swarm optimizers, PLoS ONE 11 (7) (2016).
[38] R.E. Burkard, S.E. Karisch, F. Rendl, QAPLIB–a quadratic assignment problem library, J. Global Optim. 10 (4) (1997) 391–403.
[39] É. Taillard, Robust taboo search for the quadratic assignment problem, Parallel Comput. 17 (4–5) (1991) 443–455.