IMPROVED ANTLION OPTIMIZATION ALGORITHM FOR QUADRATIC ASSIGNMENT PROBLEM

***Haydar Kılıç*[1], *Uğur Yüzgeç*[2*]**

[1,2]Department of Computer Engineering
Bilecik Seyh Edebali University
11210, Bilecik, Turkey

Email: haydar.kilic@bilecik.edu.tr[1], ugur.yuzgec@bilecik.edu.tr[2*] (corresponding author)

DOI: https://doi.org/10.22452/mjcs.vol34no1.3

ABSTRACT

The Antlion Optimization (ALO) algorithm is a meta-heuristic optimization algorithm based on the hunting of ants by antlions. The basic inadequacy of this algorithm is that it has long run time because of the random walk model used for the ant's movement. We improved some mechanisms in ALO algorithm, such as ants' random walking, reproduction, sliding ants towards antlion, elitism, and selection procedure. This proposed algorithm is called Improved Antlion Optimization (IALO) algorithm. To show the performance of the proposed IALO algorithm, we used different measurement metrics, such as mean best, standard deviation, optimality, accuracy, CPU time, and number of function evaluations (NFE). The proposed IALO algorithm was tested for different benchmark test functions taken from the literature. There are no studies regarding time analysis of ALO algorithm found in the literature. This study firstly aims to demonstrate the success of the proposed IALO algorithm especially in runtime analysis. Secondly, the IALO algorithm was also applied to the Quadratic Assignment Problem (QAP) known as a difficult combinatorial optimization problem. In QAP tests, the performance of the IALO algorithm was compared with the performances of the classic ALO algorithm and 14 well-known and recent meta-heuristic algorithms. The results of the benchmark test functions show that IALO algorithm is able to provide very competitive results in terms of mean best/standard deviation, optimality, accuracy, CPU time and NFE metrics. The CPU time results prove that IALO algorithm is faster than the classic ALO algorithm. As a result of the QAP tests, the proposed IALO algorithm has the best performance according to the mean cost, worst cost and standard deviation. The source codes of QAP with the proposed IALO algorithm are publicly available at https://github.com/uguryuzgec/QAP-with-IALO.

Keywords: ***Optimization, Benchmark, Quadratic Assignment Problem, Antlion***

1.0 INTRODUCTION

The hunting techniques of animals have always attracted the attention of scientists with their pitfalls and behaviors that they display. Antlion is one of these creatures, and the hunting technique it uses during the larval period was presented in 2015 by Seyedali Mirjalili [1]. Antlion Optimization Algorithm (ALO) was constructed on this hunting strategy. The ALO algorithm is principally based on the hunting strategy of antlions. It consists of five main steps: 1) ants' random walking; 2) building trap; 3) trapping in the antlion's pits; 4) sliding ants towards antlion; and 5) catching the prey and rebuilding the pit. There are some studies reported in the literature regarding applications or improvement of the ALO algorithm. Some of these are: PID controller parameters design [2], optimal non-convex and dynamic economic load [3], tournament selection based ALO algorithm for solving parallel machine scheduling [4] and quadratic assignment problem [5], optimal flexible process planning [6], optimal route planning for unmanned aerial vehicle [7], multi objective optimal generation scheduling [8], automatic generation control of interconnected power system [9], determining the optimal coefficients of IIR filters [10], and optimization of parameters on neuro-fuzzy inference system [11], [12].

Even though ALO algorithm gives effective results for different optimization problems on engineering area, it has some limitations. The main deficiency of ALO algorithm is the long runtime especially because of the random walking model. In this study, random walking distance was changed in model ant's movement in order to improve the ALO algorithm. The random walk distance is used as twenty percent of maximum iteration instead of the maximum iteration number in the original ALO algorithm. Furthermore, we added some new movements between lower and upper boundaries around the antlion into the phase of trapping antlion pits to ensure that ants walk more effectively around the selected antlion in the search space. In the improved ALO algorithm (IALO), the boundary checking process and the procedure about the catching prey and rebuilding the pit were improved.

The quadratic assignment problem (QAP) which is one of the most difficult combinatorial optimization problems is a facilities allocation problem. These facilities are located in many places that are already known and at the least costly ones. QAP was first presented in 1957 by Koopmans and Beckmann [13]. The cost function is the sum of the costs for each facility. The problem is solved by minimizing the total cost. The main reason for preferring QAP in this study is that it is a difficult optimization problem, and QAP has been solved with various optimization methods. In 1977, the location problem of the hospital departments was formulated with the QAP, and solved by heuristic method [14]. Experimental solution strategies of QAP were given by [15]. Then, simulated annealing algorithm was used for quadratic assignment problems [16]–[19]. The comparison of meta-heuristic algorithms and their application to QAP were presented in [20]–[23]. Afterwards, genetic algorithm were used to solve QAP [24]–[27]. In 1997, simulated annealing and genetic algorithm performance on QAP was proposed, followed by intelligent local search strategies in order to solve QAP in 1998 [28]. Ant colony optimization method were used to solve QAP [29][30]. Tabu search algorithm was applied to solve QAP [31]. In [32], Hafiz et al., presented the implementation of PSO variants for QAP. Another study on QAP is a hybrid method including tabu search and biogeography based optimization algorithms [33]. Chmiel et al. [34] compared meta-heuristic algorithms inspired by nature for quadratic assignment problem.

As the first objective of this study, we introduce an improved antlion optimization algorithm (IALO) to defeat the drawback of the original ALO algorithm's long runtime. The second aim of this study is to apply the IALO algorithm to the Quadratic Assignment Problem (QAP), which is known as a difficult combinational optimization problem. In Mirjalili's study [1], the algorithm's analysis was not carried out in terms of the CPU time or number of function evaluation. For this reason, firstly, ten benchmark functions with each having different characteristic were taken from the literature to evaluate the performance of the proposed IALO algorithm. In this stage, the IALO algorithm was compared with the well-known meta-heuristic algorithms in terms of mean best value, CPU time, number of function evaluations (NFE), optimality, and accuracy metrics. Then, the proposed IALO algorithm was adapted for QAP and its performance was compared with the original ALO algorithm, Genetic Algorithm (GA)[35], Firefly Algorithm (FA) [36], [37], Particle Swarm Optimization (PSO) [38], [39], Invasive Weed Optimization (IWO) [40]–[42], Imperialist Competitive Algorithm (ICA) [43], [44], Shuffled Frog Leaping Algorithm (SFLA) [45], [46], Biogeography-Based Optimization (BBO) [47], [48], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [49], [50], Harmony Search Algorithm (HSA) [51], [52], Cultural Optimization Algorithm (COA) [53], [54], Gray Wolf Optimization (GWO) [55], Dragonfly Optimization Algorithm (DA) [56], Grasshopper Optimization Algorithm (GOA) [57] and Moth-Flame Optimization (MFO) [58].

The rest of the paper is organized as follows: Section 2 presents the introduction of the original ALO algorithm. The proposed IALO algorithm and its novelty are explained in Section 3. In Section 4, the basic information about the quadratic assignment problem (QAP) is given briefly. For the benchmark and QAP tests, the performance of the proposed IALO algorithm is discussed in Section 5. Finally, in the last section, the conclusion and some suggestions are made for future studies.

2.0 ANTLION OPTIMIZER (ALO)

This section consists of the basic mechanisms used in the classic Antlion Optimization (ALO) algorithm. There are two important stages in the life cycles of antlions, the periods of larval periods and the periods of adulthood. ALO algorithm is based on the hunting tactic they use to feed during the larval periods of antlions. These hunting behaviors are quite unique and take place in a great mathematical structure. First of all, the antlions spiral into a cone-shaped trap that they pile themselves up at any place in a land of ants. To prevent the ants from coming out of this trap, they throw sand to the bottom of the trap, and eventually swallow the ants. After each hunt, they prepare the trap again for a new hunt. Fig. 1 illustrates the antlion’s hunting strategy.

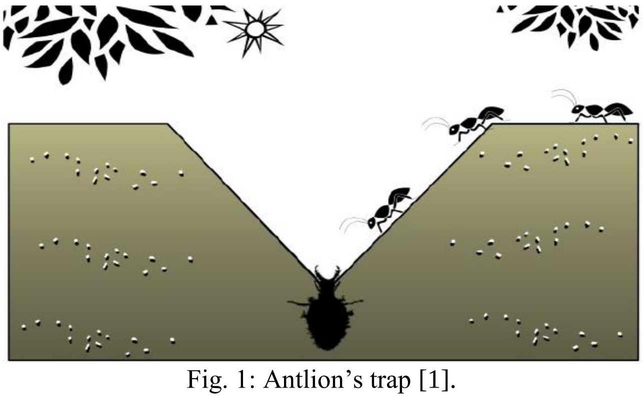


Fig. 1: Antlion’s trap [1].

The mathematical modeling of this interesting and unique hunting technique is briefly given below. After randomly selecting the first positions of ants and antlions in search space, random walks begin. The mathematical model of these walks is as follows:

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), \cdots, cumsum(2r(t_n) - 1)] \tag{1}$$

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

$$r(t) = \begin{cases} 1 & if\ rand > 0.5 \\ 0 & if\ rand \leq 0.5 \end{cases} \tag{2}$$

In order to keep random walks of ant in the search space, it has to be min-max normalized by the following equation:

$$X_i^t = \frac{(X_i^t - a_i)(d_i^t - c_i^t)}{b_i - a_i} + c_i^t \tag{3}$$

where *i* is value of the variable number, *t* is the iteration number, *a* is the minimum value of the random walk ($a = \min(X)$), b is the maximum value of the random walk ($b = \max(X)$), *c* stands for the lower value of the dynamic boundary around the antlion, *d* stands for the upper value of the dynamic boundary around the antlion.

When the ants fall down, the antlion starts throwing sand out of their way so they start sliding towards the bottom. In this way, the walks of the ants are affected by the antlion. The following math mode explains this situation.

$$c_i^t = Antlion_i^t + c^t \tag{4}$$
$$d_i^t = Antlion_i^t + d^t \tag{5}$$
$$c^t = c^t . I^{-1} \tag{6}$$
$$d^t = d^t . I^{-1} \tag{7}$$

where $Antlion_i^t$ is the position of the selected *i-th* antlion at *t-th* iteration, and *I* is the sliding ratio that can be changed in following conditions:

$$I = \begin{cases} 1 + 10^6 t/T_{max} & if\ 0.95T_{max} < t < T_{max} \\ 1 + 10^5 t/T_{max} & if\ 0.9T_{max} < t < 0.95T_{max} \\ 1 + 10^4 t/T_{max} & if\ 0.75T_{max} < t < 0.9T_{max} \\ 1 + 10^3 t/T_{max} & if\ 0.5T_{max} < t < 0.75T_{max} \\ 1 + 10^2 t/T_{max} & if\ 0.1T_{max} < t < 0.5T_{max} \\ 1 & otherwise \end{cases} \tag{8}$$

where T_{max} is the maximum iteration. After hunting, antlions update their positions with the positions of ants according to fitness values. R_A^t is antlion selected by roulette wheel method and R_E^t is elite antlion are obtained by Eq.(3) for each iteration. The ants are positioned around the elite antlion and the antlion selected by roulette wheel method with the following mathematical model.

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{9}$$

This interesting hunting mechanism inherent in antlions was discovered by Mirjalili and introduced to the literature in 2015 [1]. The pseudocode of the original ALO algorithm is given in Fig. 2.

Antlion Optimization Algorithm:

Input: Fitness function, ants and antlions, maximum iteration number, population size.
Output: The elite antlion position and its fitness value.
1) Initialize antlions' positions.
2) Calculate fitness values of antlions by using objective function.
3) Sort fitness values and save best antlion.
4) **while** $(iteration < Max\ iteration)$ and $(|f_{best} - f_{worst}| < VTR)$
  for every ant
    a) Select antlion by roulette wheel method for building trap.
    b) Slide randomly walking ants in trap.
    c) Create random walk for every ants around elite antlion and selected antlion.
    d) Normalize random walks (Eq.(3)).
    e) Update the ant position (Eq.(9)).
    f) Reposition the ants in case of outside search space.
  end for
  g) Calculate the fitness values of ants.
  h) Concatenate fitness of ants and antlions.
  i) Update antlions' positions.
  j) Save elite antlions' position and fitness value.
5) **end while**

Fig. 2: Pseudocode of the original ALO algorithm.

3.0 IMPROVED ANTLION OPTIMIZER (IALO)

The antlion algorithm reaches the optimal point later than other algorithms and does not give many good results in terms of accuracy. In the original ALO algorithm, the random walk model used for the movement of ants in the search space works as many as the number of ants in the population for the elite antlion and the antlion selected by the roulette wheel method in each iteration step. Since the size of each random walk model is the maximum iteration, these operations both slow down the algorithm and occupy too much memory unnecessarily. For this reason, the first proposed development in the antlion algorithm is achieved by reducing the size of the random walk. We conducted experiments with different random walk model sizes and observed that below 20 percent of the maximum iteration, the exploration and exploitation performance of the algorithm decreased. Therefore, we took the size of the random walk model as 20% of the maximum iteration number in this study.

The antlion, chosen by the roulette wheel, does not make any progress for the negative fitness values, and after a certain period of time, the same antlion in each iteration is selected. To solve it, the magnitudes of the fitness values have been entered on the roulette wheel, preventing the same selection every time for negative fitness values.

At the end of the algorithm, the elite antlion is updated; ant and antlion populations are combined and ranked according to their fitness values. Thus, half of the combined population is taken as antlion positions for the next iteration. Neglected ants are supposed to be eaten by antlions. Here, the novelty is that instead of combining and sorting the populations, ants and antlion's fitness values are compared for each pair of ant and antlion, and if the ant's fitness value is better than antlion's fitness, antlion's position is updated as ant's position.

Another novelty is related to the falling ants and ants out of search space. The falling ants are shifted at a certain shift rate, and these ratios have been modified to hunt the ants easier so that the accuracy of the algorithm is increased. Secondly, the ants outside the search space are left at the border in the ALO algorithm, and by changing this, the ants outside the border are moved to random positions in the search space. All these developments are explained with the following pseudocode.

Pseudocode of the Improved Antlion Optimization Algorithm (IALO):

***Input**: Fitness function, ants and antlions, maximum iteration number, population size.*
***Output**: The elite antlion position and its fitness value.*
Initialize antlions' positions.
Calculate fitness values of antlions by using objective function.
Sort fitness values and save best antlion.
while $(iter < Max\_iter)$ *and* $(|f_{best} - f_{worst}| < VTR)$

$$X(t) = [0, \cdots, cumsum(2r(t_n) - 1)], n = 1,2, \cdots, Max\_iter/5 \quad (10)$$

for *every ant*

Select antlion by roulette wheel method for building trap.

$$\frac{|f(Antlion_i^{t-1})|}{\sum_{j=1}^{n}|f(Antlion_j^{t-1})|}, i = 1,2,\cdots,n \tag{11}$$

Slide randomly walking ants in trap.

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

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

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

$$\left.\begin{aligned} c_i^t &= -Antlion_i^t - c^t \\ d_i^t &= -Antlion_i^t - d^t \end{aligned}\right\} if\ 0 < option < 0.25 \tag{15}$$

Create random walk for all ants around elite antlion and antlion selected by roulette wheel.
Normalize random walks (Eq.(3)) for elite and selected antlions.
Update the ant position.

$$Ant_i^t = \frac{R_A^{r(t_n)} + R_E^{r(t_n)}}{2}, r(t_n){:}\ rand\ number\ in\ [0\ t_n], \quad n = 1,2,\cdots, Max\_iter/5 \tag{16}$$

Reposition the ant in case of outside search space. They bring back them inside the search space unlike the original ALO.

$$Ant_i^t = b_{low} + rand \times (b_{up} - b_{low}) \quad if\ (Ant_i^t > b_{up})\ or\ (Ant_i^t < b_{low}) \tag{17}$$

end for

Calculate the fitness values of ants.
Compare fitness of ants and antlions. If ant has better fitness than antlion, the antlion position is updated as ant's position, otherwise antlion keeps its position.

$$Antlion_i^t = Ant_i^t \quad if\ f(Ant_i^t) < f(Antlion_i^t) \tag{18}$$

Update antlions' positions.
Save elite antlion's position and fitness value.

end while

Return elite antlion

where *option* in Eqs.(12-15) is chosen variable randomly, f_{best} stands for the best fitness, f_{worst} denotes the worst fitness, $r(t_n)$ is random number in interval $[0\ t_n]$, *n* is 20% of the maximum number of iteration, b_{low} is lower and b_{up} is upper boundary of the search space.

4.0 QUADRATIC ASSIGNMENT PROBLEM (QAP)

The Quadratic Assignment Problem (QAP) proposed for the first time by Koopmans and Beckman [13].The objective of the problem is to make total assignment cost minimum while assigning facilities to locations. We consider that w_{ij} the weight or the flow coefficients between *i-th* and *j-th* facilities and d_{pq} distance between *p-th* and *q-th* locations. The objective function of QAP is given below:

$$\begin{aligned} &min \sum_{i,j=1}^{n} \sum_{p,q=1}^{n} w_{ij}\, d_{pq} x_{ip} x_{jq} \\ &subject\ to \sum_{i=1}^{n} x_{ij} = 1, \\ &\qquad\qquad \sum_{j=1}^{n} x_{ij} = 1, \\ &x_{ij} \in \{0,1\}, 1 \le i,j \le n \end{aligned} \tag{19}$$

In the general form of QAP equation with an order *n*, there are two matrices: $W = [w_{ij}]$ and $D = [d_{pq}]$. W matrix includes the flow coefficients between the facilities and D matrix consists of the distances between all locations.

5.0 EXPERIMENTAL RESULTS AND DISCUSSIONS

To evaluate the performance of the IALO algorithm, multi-dimension benchmark tests have been realized with the other popular heuristic algorithms, and then the IALO algorithm has been implemented to quadratic assignment problem. This problem has been solved by IALO algorithm and its result has been compared with several recent meta-heuristic algorithms.

5.1 Evaluation Criteria

Algorithms are being analyzed in terms of various metrics with benchmark test functions and compared in terms of performance. The mathematical model of these metrics is examined below.

$$\gamma : X \subseteq \mathbb{R}^n \rightarrow \Gamma \tag{20}$$

where *n* is the dimension of solution space in search space. Let $x_0 \in \Gamma$ be the solution, $\gamma(x_0) = \gamma_0$ is considered to be the solution of the optimization problem, and $\gamma(\hat{x}_0) = \hat{\gamma}_0$ denotes closeness the solution found. Then, used metrics are defined as follows:

$$Optimality = 1 - \frac{\|\gamma_0 - \hat{\gamma}_0\|}{\left\|\overline{\gamma} - \underline{\gamma}\right\|} \in [0,1] \tag{21}$$

$$Accuracy = 1 - \frac{\|x_0 - \hat{x}_0\|}{\|\overline{x} - \underline{x}\|} \in [0,1] \tag{22}$$

$$Mean = \frac{1}{N}\sum_{i=1}^{N} \hat{\gamma}_0 \tag{23}$$

$$Standard\ Deviation\ (STD) = \sqrt{\frac{1}{N-1}\sum (\hat{\gamma}_0 - Mean)^2} \tag{24}$$

where, $\overline{\gamma}$ and $\underline{\gamma}$ are lower and upper bounds of γ, $\overline{x}$ denotes the lower bound and $\underline{x}$ denotes the upper bound of search space [59]. *Optimality* metric defines the relative closeness of an objective found. *Accuracy* metric shows the relative closeness of the solution found. *Mean* metric denotes the average of closeness of the solution found. Besides than these metrics, this study also used other metrics which are *CPU time* and *number of the function evaluations* (*NFE*), to give information about the run time of the algorithm.

5.2 Results and Discussion

5.2.1 Benchmark Test Results

In this study, ALO and IALO algorithms were tested with 10D benchmark test functions and compared with other popular and well-known heuristic algorithms. All benchmark test functions have different characteristics. The benchmark functions used are given as follows:

F_1: Ackley Function

$$f(x) = -20.exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\right) + 20 + \exp(1) \tag{25}$$

subject to $-35 \leq x_i \leq 35$, the global minima is $f(x) = 0$ at $x = (0, \cdots, 0)$

(26)

F_2: Griewank Function

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

subject to $-100 \le x_i \le 100$, the global minima is $f(x) = 0$ at $x = (0, \cdots, 0)$

F₃: Levy Function

$$f(x) = sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 \left[1 + 10sin^2(\pi w_i + 1)\right] + (w_d - 1)^2[1 + sin^2(2\pi w_d)] \tag{27}$$

$$w_i = 1 + \frac{x_i - 1}{4}, i = 1,2, \cdots, d$$

subject to $-10 \le x_i \le 10$, the global minima is $f(x) = 0$ at $x = (1, \cdots, 1)$

F_4: Rastrigin Function

$$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)] \tag{28}$$

subject to $-5.12 \le x_i \le 5.12$, the global minima is $f(x) = 0$ at $x = (0, \cdots, 0)$

F_5: Rosenbrock Function

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \tag{29}$$

subject to $-2.3 \le x_i \le 2.3$, the global minima is $f(x) = 0$ at $x = (1, \cdots, 1)$

F_6: Schwefel Function

$$f(x) = 418.9829d - \sum_{i=1}^{d} x_i sin\left(\sqrt{|x_i|}\right) \tag{30}$$

subject to $-500 \le x_i \le 500$, the global minima is $f(x) = 0$ at $x = (420.96, \cdots, 420.96)$

F_7: Sphere Function

$$f(x) = \sum_{i=1}^{d} x_i^2 \tag{31}$$

subject to $-5.12 \le x_i \le 5.12$, the global minima is $f(x) = 0$ at $x = (0, \cdots, 0)$

F_8: Styblinski-Tang Function

$$f(x) = \frac{1}{2}\sum_{i=1}^{d} (x_i^4 - 16x_i^2 + 5x_i) \tag{32}$$

subject to $-5 \le x_i \le 5$, the global minima is $f(x) = -39.16$ at $x = (-2.9, \cdots, -2.9)$

F_9: Sum Squares Function

$$f(x) = \sum_{i=1}^{d} ix_i^2 \tag{33}$$

subject to $-10 \le x_i \le 10$, the global minima is $f(x) = 0$ at $x = (0, \cdots, 0)$

F_{10}: Zakharov Function

$$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^4 \tag{34}$$

subject to $-5 \le x_i \le 10$, the global minima is $f(x) = 0$ at $x = (0, \cdots, 0)$

Ackley, Griewank, Rastrigin, Levy, Schwefel functions have many local minimum points. Ackley function appears to be approximately flat at the edge regions, but there are many local minimum points and a large hole at the center. In the Griewank function, there are many local minimum points uniformly distributed on the surface. Rastrigin function is a multimodal function, but the local minimum is regularly distributed as it is in Griewank. The Schwefel function is a complex function. Rosenbrock function is a valley-shaped unimodal function; convergence is difficult if the global minimum is in a narrow place. Sphere and Sum Squares functions are bowl-shaped functions. The sphere function is a unimodal function and has a local minimum point as dimension size. The Sum Squares function only has a global minimum. Similarly, Zakharov function has only a global minimum and is plate-shaped.

During benchmark works with multi-dimension, the population size is 100, and maximum iteration number is 1000. All algorithms have been run 50 times. All codes of heuristic algorithms have been run on PC with Intel(R) Core(TM) i7-6500U CPU@2.50GHz/8.00GB RAM. The parameters of algorithms that are used in this study are given in Table 1.

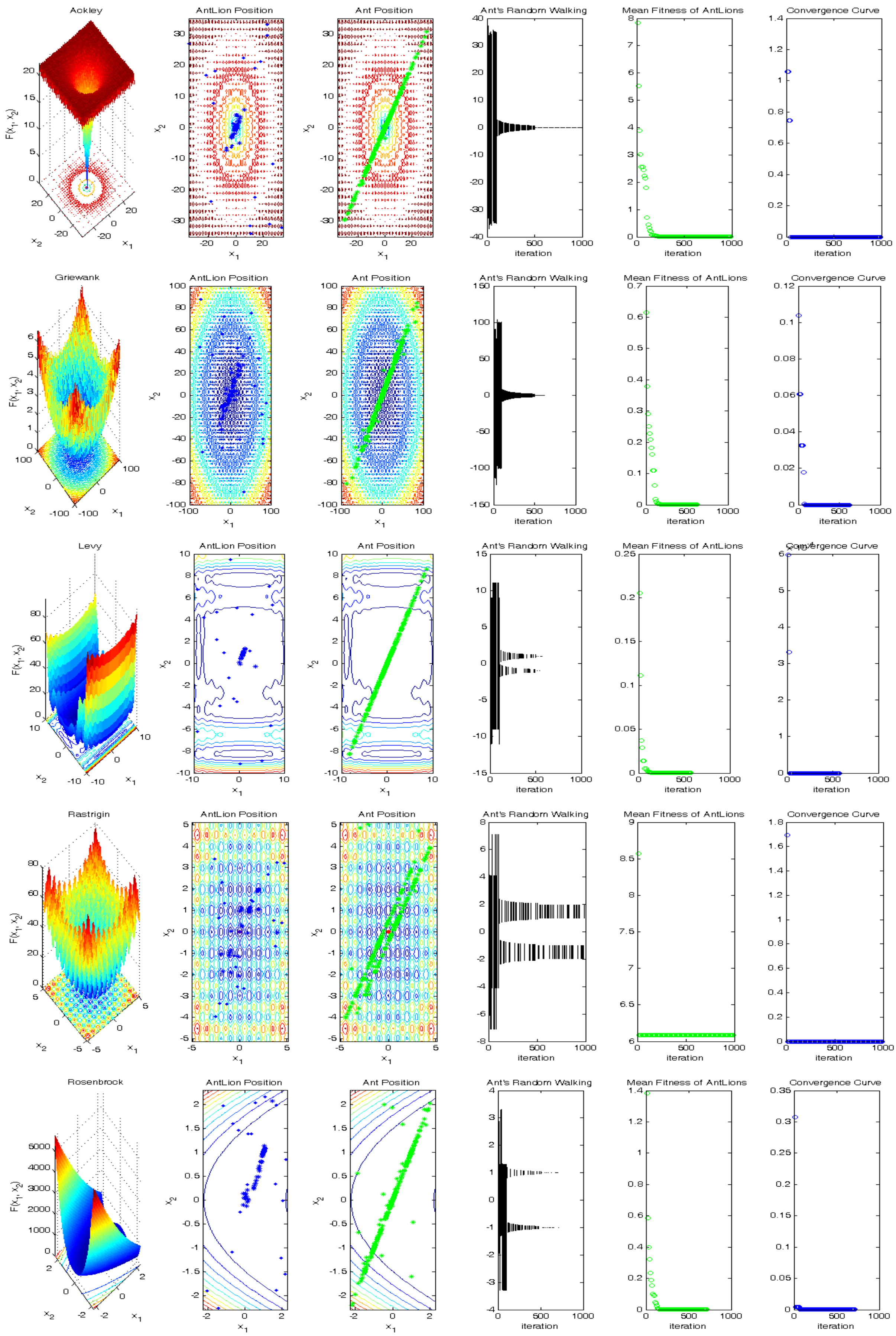
Table 1: Parameters of meta-heuristic algorithms for benchmark tests

Algorithm	**Parameters**
PSO [38]	Learning coefficients = 2.05, Constriction factor=0.7298
ABC [60]	Number of food sources=50, Limit of attempts=100
SA [61]	Temperature=current iteration/maximum iteration number
DE [62], [63]	Crossover probability=0.5, Differential weight=0.8, Differential strategy=DE/rand/1/bin
TACO [64]	Vaporing=0.1, Bit number=18
ALO [1]	Search agent=100
IALO	Search agent=100, random walk size =Max Iter/5

There are two criteria have been used to stop termination: one for reaching the maximum number of iterations, and the other for Value To Reach ($VTR = 10^{-6}$). *VTR* condition is given below:

$$if\ |f_{best} - f_{worst}| < VTR\ then\ stop\ the\ algorithm \tag{35}$$

where f_{best} denotes the best fitness value and f_{worst} denotes the worst fitness value in the population. The 3D images of the functions, the illustration of the positions of antlions and ants, the random walking of ants, mean fitness of antlions, and convergence curve during the optimization are shown in Fig. 3. From these figures, the antlion positions are located around the global solution, and the ant positions have been moved along a line or lines in the search space. For the problems with smooth surface and many local peaks, holes, random walking was produced differently unlike ALO algorithm. For all test functions, the solutions are shown to be reached in the short iteration numbers from the last two sub-figures. Tables 2-5 present the 10D benchmark test results for 50 independent runs. To compare their performances, results of seven meta-heuristic algorithms are presented in these tables. Four metrics, such as *mean best/standard deviation*, *number of function evaluation (NFE)/CPU time*, *optimality*, *accuracy* are used to show the performance of these algorithms. In terms of the *mean best/std.dev.*, IALO has the best value except for F6 function. According to the benchmark results in Table 3, *CPU time/NFE* results of the IALO algorithm are not the best, but the long-running time of the ALO algorithm has been shortened considerably with the proposed innovation on the random walkways. In some benchmark test functions, the IALO algorithm has reached an optimal result in 5-20 times less time than the original ALO algorithm.


Ackley
AntLion Position
Ant Position
Ant's Random Walking
Mean Fitness of AntLions
Convergence Curve
Griewank
Levy
Rastrigin
Rosenbrock
F(x1, x2)
x1
x2
iteration

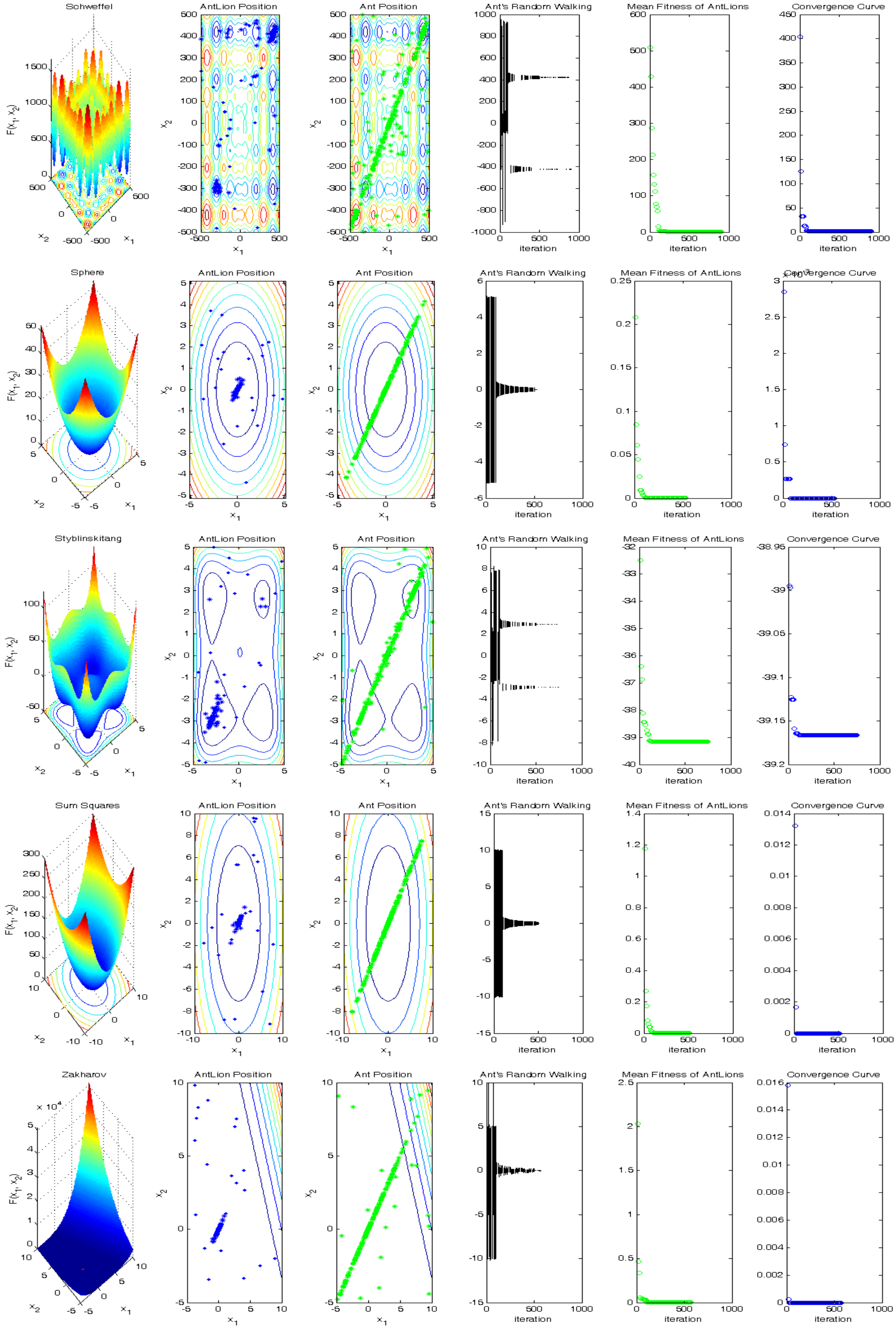


Fig. 3: IALO algorithm analysis for all benchmark functions

Table 2: Comparison results (*Mean Best & Std.Dev.*) with 50 independent runs of IALO algorithm, PSO, ABC, SA, DE, TACO and ALO algorithms. The best result of each function is emphasized in **boldface**.

	Mean Best (Std.Dev.)						
Function	PSO	ABC	SA	DE	TACO	ALO	IALO
FN1	4.37e+0	5.17e-10	1.51e+1	6.18e-7	1.52e+1	2.29e-1	**0.00e+0**
	(1.03e+0)	(5.25e-10)	(2.19e+0)	(1.29e-7)	(8.74e-1)	(5.51e-1)	**(0.00e+0)**
FN2	4.63e-1	1.46e-03	1.15e+0	1.32e-1	1.20e+0	1.72e-1	**0.00e+0**
	(1.47e-1)	(3.17e-03)	(1.28e-1)	(2.23e-2)	(1.83e-1)	(1.03e-1)	**(0.00e+0)**
FN3	3.01e-1	6.78e-13	2.46e+0	1.55e-7	1.25e+0	3.91e-1	**1.96e-14**
	(3.99e-1)	(2.14e-12)	(9.15e-1)	(4.09e-8)	(1.39e+0)	(5.67e-1)	**(1.39e-13)**
FN4	1.50e+1	3.89e-14	2.42e+1	6.18e-1	4.56e+1	1.61e+1	**0.00e+0**
	(6.95e+0)	(8.03e-14)	(4.36e+0)	(8.82e-1)	(1.06e+1)	(9.94e+0)	**(0.00e+0)**
FN5	1.17e+1	1.84e-1	6.42e+1	2.76e+0	2.33e+1	5.23e+0	**1.47e-11**
	(7.80e+0)	(1.95e-1)	(2.74e+1)	(1.26e-1)	(2.48e+1)	(2.38e+0)	**(3.59e-11)**
FN6	1.79e+3	**1.27e-4**	8.43e+2	**1.27e-4**	1.15e+3	1.48e+3	8.67e-2
	(2.48e+2)	**(2.78e-8)**	(1.47e+2)	(5.15e-8)	(2.56e+2)	(6.46e+2)	(2.33e-1)
FN7	1.15e-1	1.46e-12	1.86e+0	1.63e-7	8.49e-2	7.71e-9	**0.00e+0**
	(9.84e-2)	(2.39e-12)	(7.27e-1)	(4.79e-8)	(2.87e-1)	(2.38e-9)	**(0.00e+0)**
FN8	-3.36e+1	**-3.92e+1**	-3.57e+1	**-3.92e+1**	-2.99e+1	-3.61e+1	**-3.92e+1**
	(2.01e+0)	**(5.37e-15)**	(9.93e-1)	(4.60e-8)	(2.03e+0)	(1.96e+0)	(1.10e-4)
FN9	2.31e+0	1.57e-12	3.17e+1	1.62e-7	1.66e+0	4.85e-8	**0.00e+0**
	(1.68e+0)	(3.59e-12)	(1.44e+1)	(4.37e-8)	(5.23e+0)	(3.69e-8)	**(0.00e+0)**
FN10	5.83e+0	1.12e+1	6.21e+1	1.71e-2	2.07e+1	5.61e-10	**0.00e+0**
	(4.57e+0)	(5.28e+0)	(1.79e+1)	(9.64e-3)	(7.78e+0)	(2.13e-10)	**(0.00e+0)**

Table 3: Comparison results (*NFE & CPU Time*) with 50 independent runs of IALO algorithm, PSO, ABC, SA, DE, TACO and ALO algorithms. The best result of each function is emphasized in **boldface**.

	NFE (CPU Time)						
Function	PSO	ABC	SA	DE	TACO	ALO	IALO
FN1	**22320**	49835	100000	69516	100000	99258	99078
	(0.773s)	(1.823s)	(4.322s)	(2.423s)	(28.620s)	(49.307s)	(5.994s)
FN2	**21416**	50995	100000	100000	100000	95216	76690
	(0.780s)	(1.966s)	(4.467s)	(3.642s)	(28.810s)	(47.765s)	(4.864s)
FN3	**15666**	30603	100000	41078	100000	90332	74424
	(0.415s)	(0.901s)	(3.450s)	(1.104s)	(27.784s)	(45.179s)	(3.901s)
FN4	**26154**	51001	100000	100000	100000	95322	81860
	(0.931s)	(1.929s)	(4.416s)	(3.605s)	(28.920s)	(49.060s)	(5.213s)
FN5	**19122**	51002	100000	100000	100000	99736	81134
	(0.574s)	(1.656s)	(3.852s)	(3.027s)	(28.282s)	(50.874s)	(4.664s)
FN6	**25974**	51002	100000	86806	100000	98128	94342
	(0.806s)	(1.719s)	(3.931s)	(2.759s)	(28.609s)	(50.025s)	(5.521s)
FN7	**17272**	27681	100000	35722	100000	90298	54526
	(0.354s)	(0.638s)	(2.824s)	(0.742s)	(27.966s)	(45.319s)	(2.541s)
FN8	**17888**	44610	100000	44320	73482	90234	76240
	(0.476s)	(1.280s)	(3.411s)	(1.199s)	(20.485s)	(43.976s)	(4.010s)
FN9	**21716**	30290	100000	41650	100000	91420	72474
	(0.438s)	(0.693s)	(2.789s)	(0.864s)	(27.097s)	(43.960s)	(3.345s)
FN10	**29684**	51002	100000	100000	94824	95200	80588
	(0.652s)	(1.273s)	(2.999s)	(2.229s)	(26.025s)	(47.859s)	(4.001s)

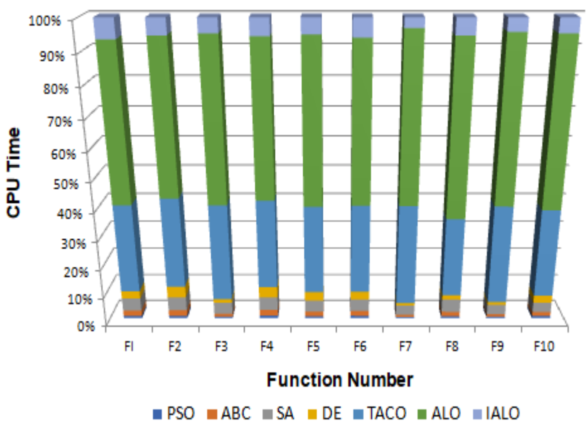
Table 4: Comparison results (*Optimality*) with 50 independent runs of IALO algorithm, PSO, ABC, SA, DE, TACO and ALO algorithms. The best result of each function is emphasized in **boldface**.

	Optimality						
Function	PSO	ABC	SA	DE	TACO	ALO	IALO
FN1	0.804	**1.000**	0.325	**1.000**	0.319	0.990	**1.000**
FN2	0.930	**1.000**	0.827	0.980	0.818	0.974	**1.000**
FN3	0.997	**1.000**	0.974	**1.000**	0.987	0.996	**1.000**
FN4	0.814	**1.000**	0.699	0.992	0.434	0.801	**1.000**
FN5	0.998	**1.000**	0.989	**1.000**	0.996	0.999	**1.000**
FN6	0.067	**1.000**	0.497	**1.000**	0.315	0.117	**1.000**
FN7	0.998	**1.000**	0.964	**1.000**	0.998	**1.000**	**1.000**
FN8	0.966	**1.000**	0.979	**1.000**	0.944	0.982	**1.000**
FN9	0.992	**1.000**	0.894	**1.000**	0.994	**1.000**	**1.000**
FN10	**1.000**	**1.000**	0.999	**1.000**	**1.000**	**1.000**	**1.000**

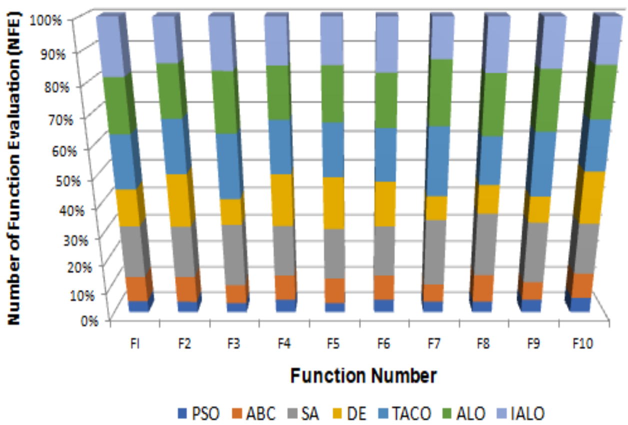
Table 5: Comparison results (*Accuracy*) with 50 independent runs of IALO algorithm, PSO, ABC, SA, DE, TACO and ALO algorithms. The best result of each function is emphasized in **boldface**.

	Accuracy						
Function	PSO	ABC	SA	DE	TACO	ALO	IALO
FN1	0.989	**1.000**	0.924	**1.000**	0.933	**1.000**	**1.000**
FN2	0.974	0.999	0.961	0.987	0.961	0.969	**1.000**
FN3	0.986	**1.000**	0.951	**1.000**	0.968	0.990	**1.000**
FN4	0.931	**1.000**	0.905	0.998	0.890	0.909	**1.000**
FN5	0.797	0.984	0.836	0.900	0.806	0.853	**1.000**
FN6	0.625	**1.000**	0.756	**1.000**	0.777	0.424	**1.000**
FN7	0.992	**1.000**	0.969	**1.000**	0.997	**1.000**	**1.000**
FN8	0.857	**1.000**	0.914	**1.000**	0.811	0.879	**1.000**
FN9	0.990	**1.000**	0.968	**1.000**	0.995	**1.000**	**1.000**
FN10	0.964	0.950	0.867	0.998	0.924	**1.000**	**1.000**

NFE/CPU time metric results of all meta-heuristic algorithms for each benchmark function are shown in Fig. 4. As can be seen from Fig.4b, the worst algorithm is the classic ALO algorithm. Fig. 5 presents the *mean best* values obtained by meta-heuristic algorithms for all benchmark functions. *Optimality* metric indicates how close to the global solution (fitness) and it varies from 0 to 1. *Accuracy* is a metric that varies between 0-1, indicating how close to the global solution points are. In terms of these metrics, the best algorithm is the proposed IALO algorithm. For all benchmark test functions, the global fitness values have been found at the global solution points with % 100 success by the IALO algorithm. In Fig. 6 and Fig.7, *Optimality* and *Accuracy* metric results of IALO algorithm and other meta-heuristic algorithms are presented for all benchmarks. The mean of cost value for all benchmark functions are shown in Fig.8. These graphics have been given as logarithmic plots in order to understand the comparison results of the algorithms better.



(a)



(b)

Fig.4: *NFE* (a) and *CPU Time* (b) results of meta-heuristic algorithms for benchmark problems

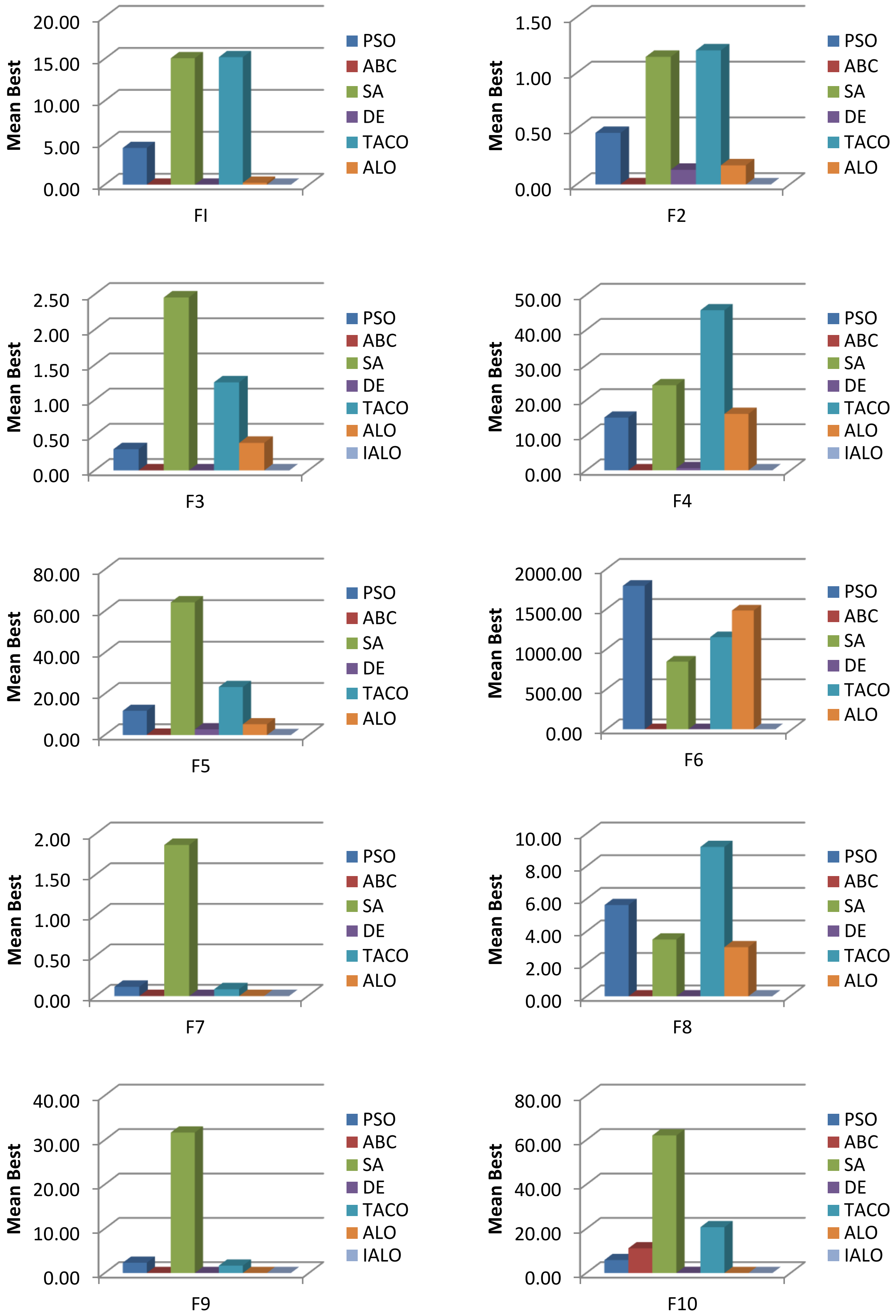


Fig. 5: Mean best results of meta-heuristic algorithms for benchmark problems

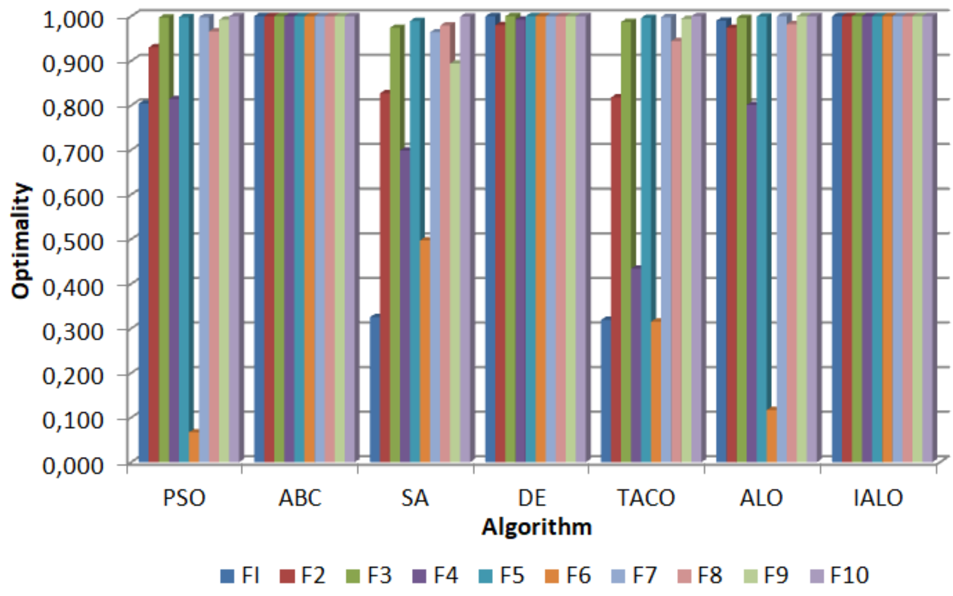


Fig.6: *Optimality* results of meta-heuristic algorithms for benchmark problems

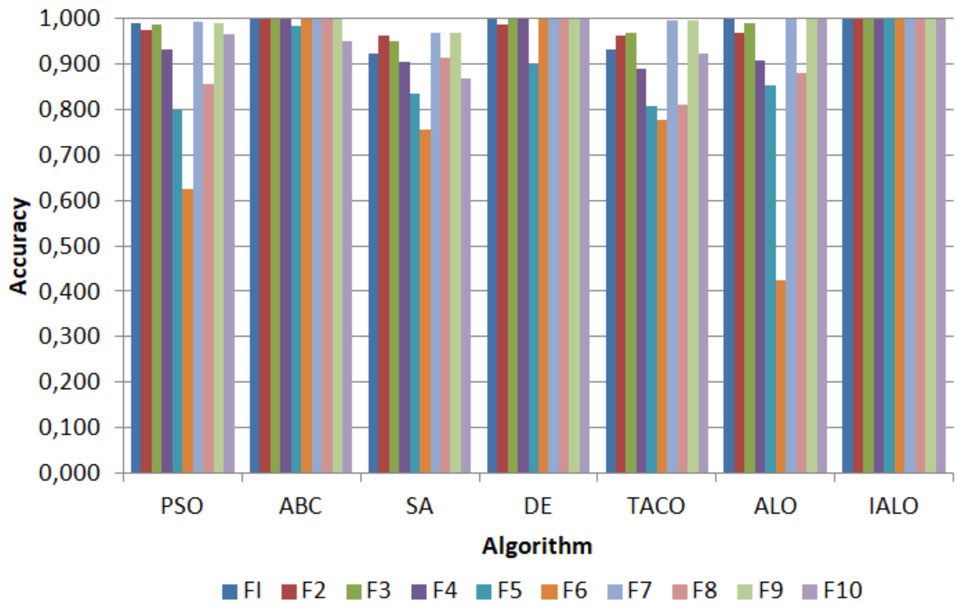
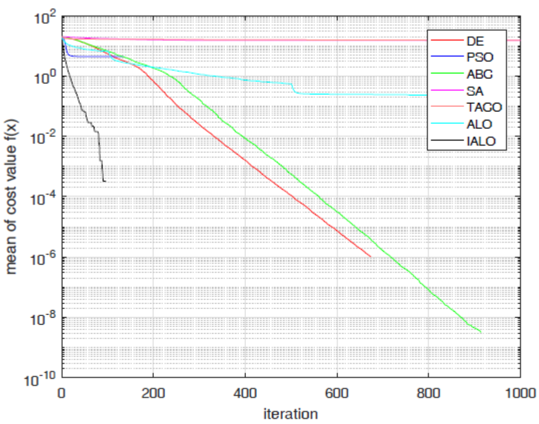
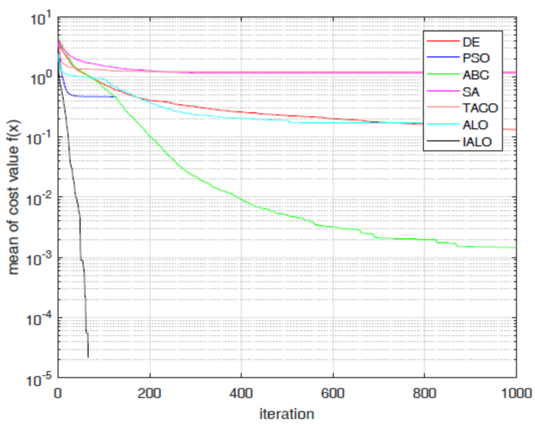


Fig.7: *Accuracy* results of meta-heuristic algorithms for benchmark problems



(*F*₁)



(*F*₂)

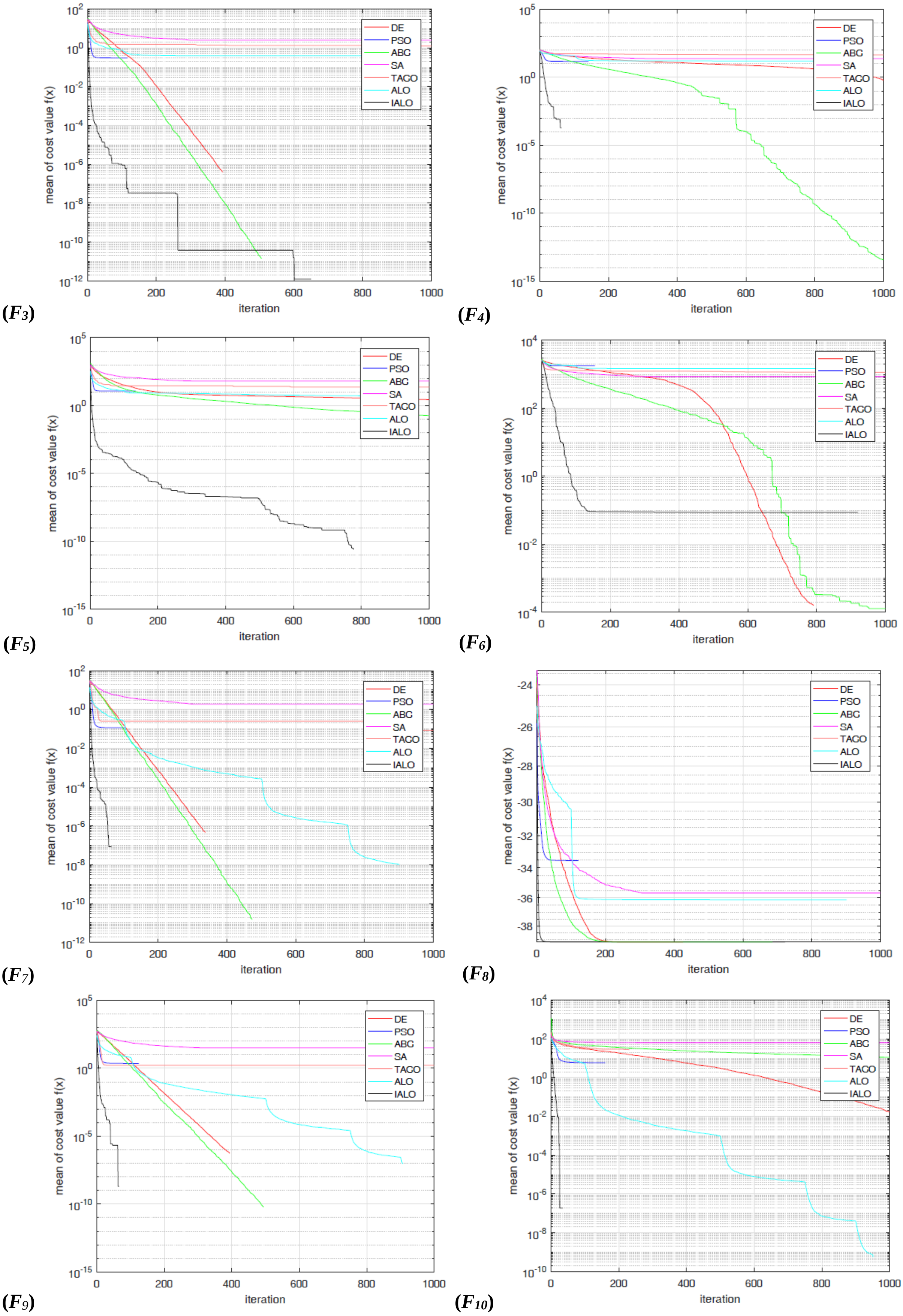


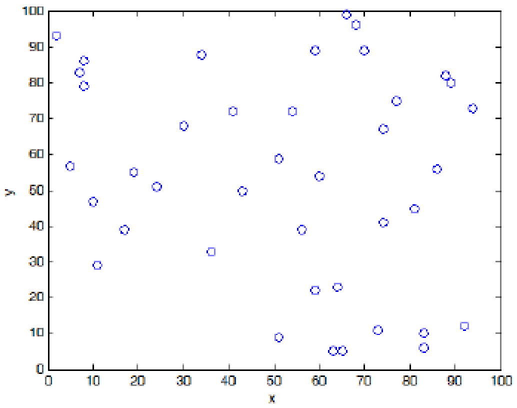
Fig.8: Comparison results of meta-heuristic algorithms on benchmark problems

5.2.2 QAP Test Results

In this study, QAP instance was taken from www.yarpiz.com web site [81]. This problem consists the $W[20 \times 20]$ weight matrix and $D[20 \times 20]$ distance matrix. This problem includes three different special situations. First, the 19th and 20th facilities must be as close as possible, then, the 11th and 16th facilities must be as close as possible. Finally, 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 values of this matrix are given in the appendix section. The locations of this QAP instance are shown in Fig. 9. There are 40 locations to be used in QAP.



QAP instance's locations
(70,89) (63,5) (11,29) (5,57)
(43,50) (94,73) (10,47) (2,93) (68,96) (74,67)
(24,51) (89,80) (59,22) (59,89) (41,72) (7,83)
(73,11) (86,56) (34,88) (88,82) (83,6) (19,55)
(66,99) (8,79) (64,23) (56,39) (92,12) (83,10)
(36,33) (77,75) (74,41) (8,86) (51,59) (30,68)
(60,54) (51,9) (17,39) (81,45) (65,5) (54,72)

Fig.9: Locations used for quadratic assignment problem

To solve QAP problem, IALO algorithm has been adapted to combinatorial optimization problem. For the example used in this study, we identified the problem dimension (N_p) as the number of locations. Fig. 10 shows how the solution of QAP derive from the antlion's position does. Initially, IALO algorithm randomly produces the positions of antlions in the range [0 1]. Then these position values are sorted 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 cost 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 IALO algorithm is given below:

Pseudo code about solving QAP problem by IALO Algorithm:

Input: weight matrix (W), location vectors (x, y), number of locations, number of facilities, candidate solutions produced by IALO.z
Output: total cost value.

1) Create facility list from candidate solution produced by IALO
2) Calculate distance between locations
   for i:number of locations
      for j=i+1:number of locations
         calculate distance (i, j) : $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
         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

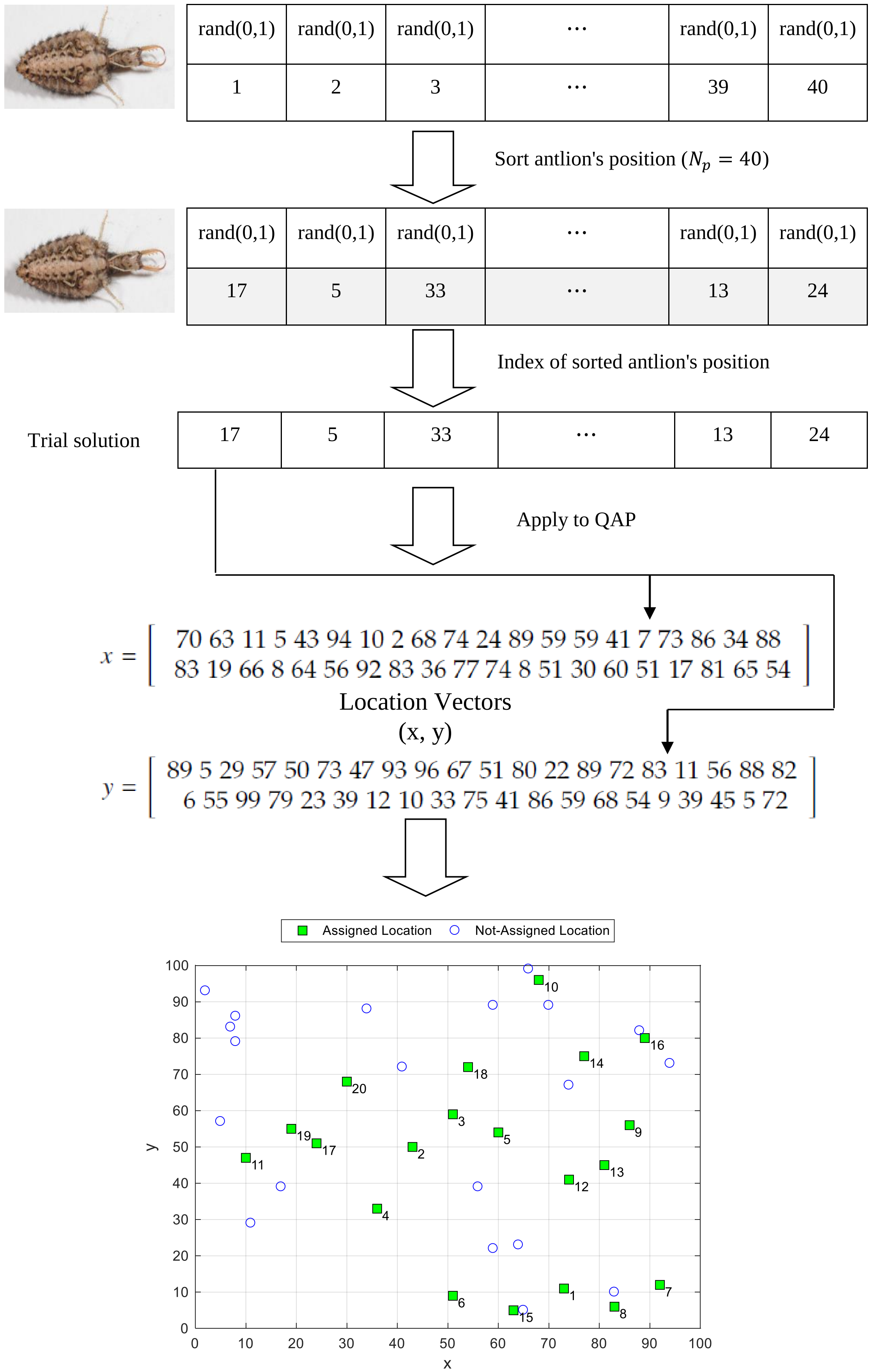


Fig. 10: Solving QAP with IALO algorithm.

For QAP tests, the performance of the proposed IALO algorithm was compared with the performances of the original ALO algorithm, Genetic Algorithm (GA), Firefly Algorithm (FA), Particle Swarm Optimization (PSO), Invasive Weed Optimization (IWO), Imperialist Competitive Algorithm (ICA), Shuffled Frog Leaping Algorithm (SFLA), Biogeography-Based Optimization (BBO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Harmony Search Algorithm (HSA), Cultural Optimization Algorithm (COA), Gray Wolf Optimization (GWO), Dragonfly Optimization Algorithm (DA), Grasshopper Optimization Algorithm (GOA) and Moth-Flame Optimization (MFO). All codes were run on PC with Intel(R) Core(TM) i7-6500U CPU@2.50GHz/8.00GB. For initial candidate solutions of these algorithms, same individuals have been used. Each algorithm was run for 10 times with 20 population size and 1000 maximum number of iterations. The parameters of meta-heuristic algorithms used for QAP performance tests are given in Table 6. The source codes of QAP with the proposed IALO algorithm are publicly available at https://github.com/uguryuzgec/QAP-with-IALO.

Table 6: Parameters of meta-heuristic algorithms for QAP tests

Algorithm	**Parameters**	**Algorithm**	**Parameters**
GA	Crossover Coefficient: 0.4	HSA	Number of New Harmonies: 20
	Mutation Coefficient: 0.8		Harmony Memory Consideration Rate: 0.9
	Selection Pressure Coefficient: 5		Pitch Adjustment Rate: 0.1
			Fret Width Damp Ratio: 0.995
PSO	Inertia Weight: 1.0	COA	Acceptance Ratio: 0.35
	Inertia Weight Damping Ratio: 0.99		Alpha: 0.3
	Personal Learning Coefficient: 1.5		
	Global Learning Coefficient: 2.0		
FA	Light Absorption Coefficient: 1.0	GWO	Number of Wolfs: 20
	Initial Attraction Coefficient: 2.0		
	Mutation Coefficient: 0.2		
	Mutation Coefficient Damping R. : 0.98		
IWO	Variance Reduction Exponent: 2	DA	Number of Dragonflies: 20
	Initial Value of Standard Deviation: 1		
	Final Value of Standard Deviation: 0.001		
	Minimum Number of Seeds: 0		
	Maximum Number of Seeds: 5		
ICA	Selection Pressure: 1	GOA	Number of Grasshoppers: 20
	Assimilation Coefficient: 2		cMax: 1
	Revolution Probability: 0.5		cMin: 0.00004
	Revolution Rate: 0.1		
	Colonies Mean Cost Coefficient: 0.1		
SFLA	Number of Memeplexes: 5	MFO	Number of Moth-Flames: 20
	Number of Offsprings: 3		
	Maximum Number of Iterations: 5		
	Step Size: 2		
BBO	Keep Rate: 0.2	ALO	Number of Antlions: 20
	Alpha: 0.9		
	Mutation Coefficient: 0.1		
CMA-ES	Number of Off-springs:	IALO	Number of Antlions: 20
	(4+round(3*log(nVar)))*10		
	nVar: number of variables		

The results obtained by the IALO and other meta-heuristic algorithms are shown in Fig. 11. These results are presented at the end of one-time run. IALO result has the second best cost value as -1078209.911. In the results of all 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. The convergence curves of the proposed IALO algorithm and other meta-heuristic algorithms are shown in Fig.12.

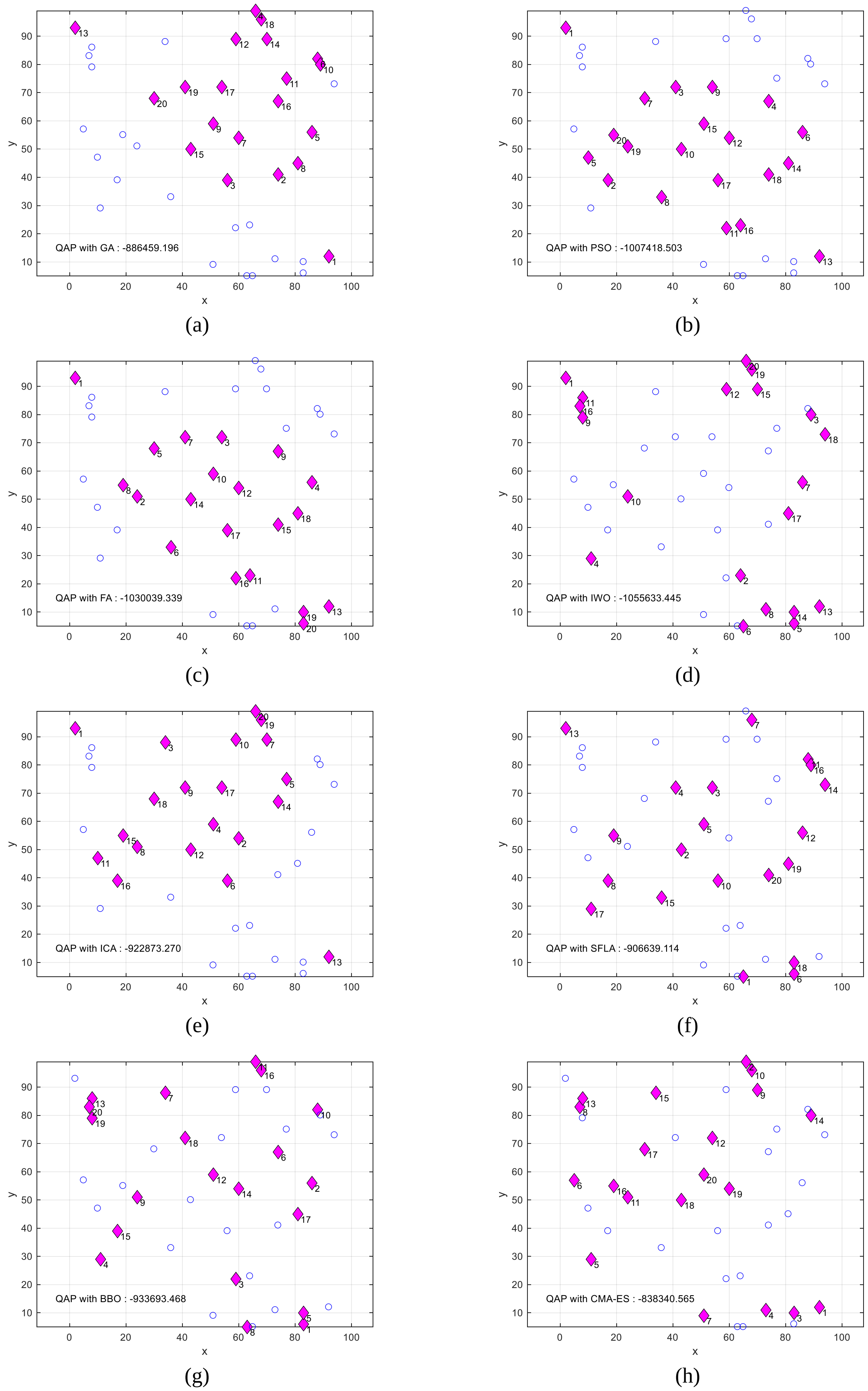


Fig. 11: QAP Results obtained by meta-heuristic algorithms,
(a) GA, (b) PSO, (c) FA, (d) IWO, (e) ICA, (f) SFLA, (g) BBO, (h) CMA-ES.

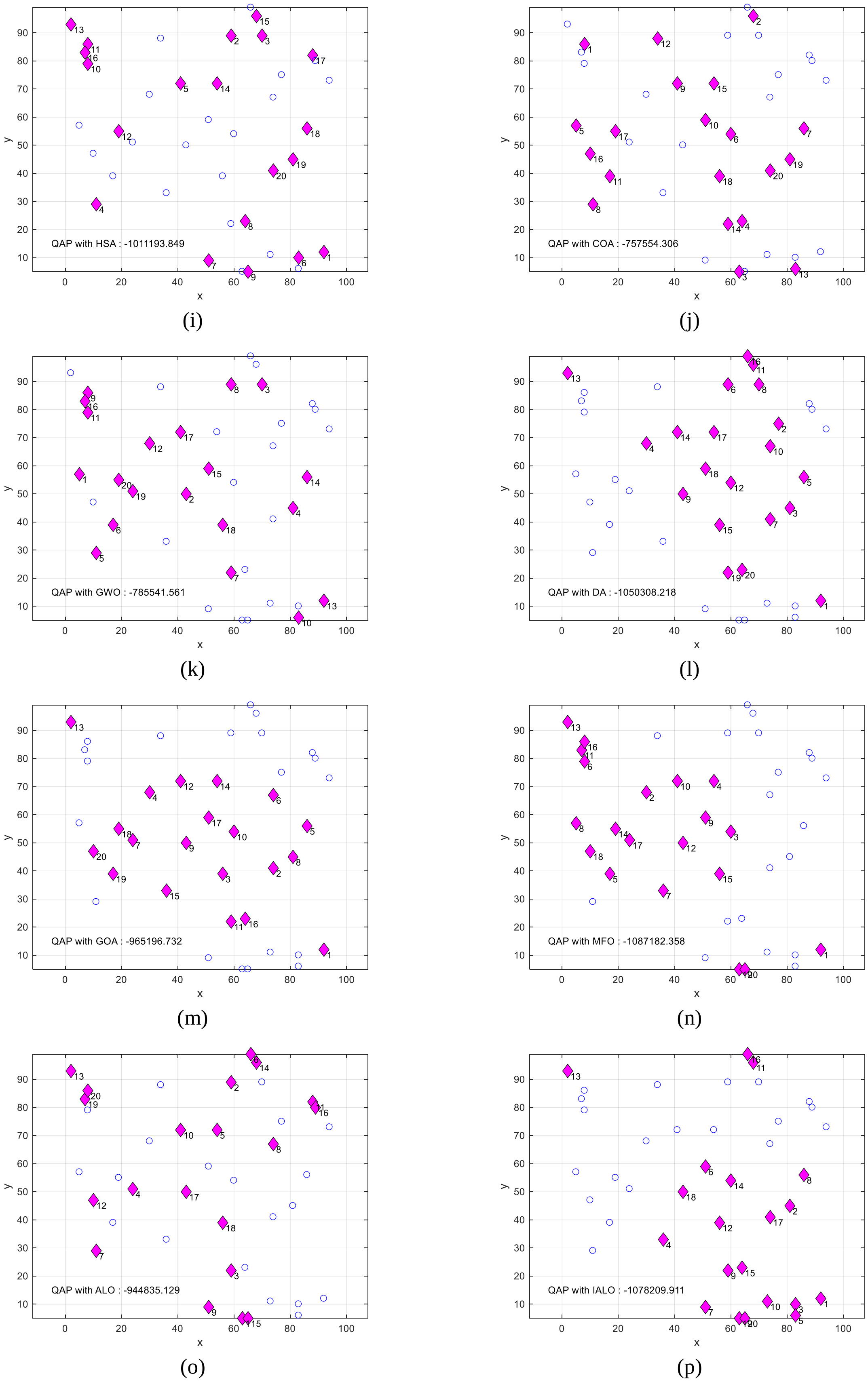


Fig. 11: (continued) QAP results obtained by meta-heuristic algorithms,
(i) HSA, (j) COA, (k) GWO, (l) DA, (m) GOA, (n) MFO, (o) ALO, (p) IALO.

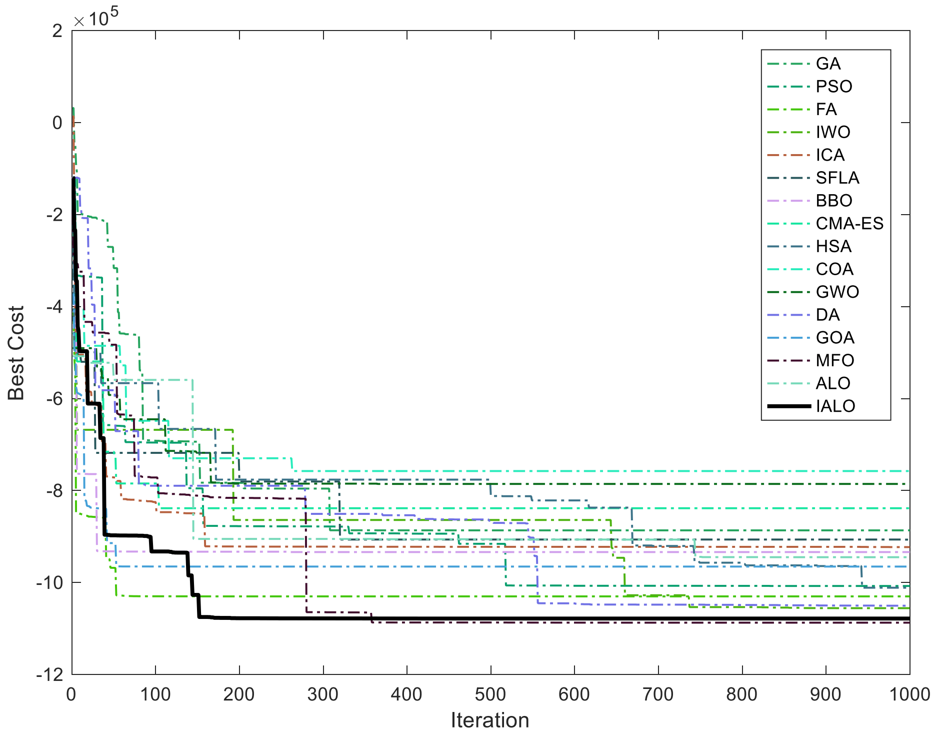


Fig. 12: Comparison result of IALO and other meta-heuristic algorithms for QAP

For QAP, the comparison results with 10 independent runs of IALO and the others are given in Table 7. This consists of mean cost, standard deviation, best cost and worst cost values from 10 runs. Based on the results shown in this table, the IALO algorithm has the best performance in terms of the mean cost, the standard deviation, and the worst cost metrics. The best values are shown bold in this table.

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

	Mean Cost	**Standard Dev.**	**Best Cost**	**Worst Cost**
GA	-1040715.59	32096.99	-1078899.84	-998807.47
PSO	-1050518.95	48744.72	-1094306.47	-972582.99
FA	-1039310.94	47073.92	**-1107239.04**	-947424.52
IWO	-1051046.31	32043.56	-1083975.02	-979314.77
ICA	-1049454.60	44573.31	-1094112.34	-966221.62
SFLA	-967258.89	60253.82	-1079771.33	-864700.71
BBO	-778150.82	198348.69	-1042232.90	-458813.87
CMA-ES	-1004125.65	57871.30	-1084741.29	-934860.75
HSA	-973304.68	97905.69	-1084095.47	-787085.25
COA	-815792.62	126273.52	-1079617.77	-691736.94
GWO	-945586.95	106512.14	-1089105.24	-780955.46
DA	-1055131.28	47818.36	-1105279.14	-976083.71
GOA	-997787.67	46922.63	-1093985.84	-928307.57
MFO	-968759.97	110631.05	-1098200.44	-822118.17
ALO	-776989.54	134951.44	-1035716.47	-605587.60
IALO	**-1061949.38**	**19146.89**	-1081731.48	**-1025641.54**

The convergence curves obtained by IALO algorithm for each runs are presented in Fig.13. As can be seen from this figure, the IALO algorithm has the most stable results for QAP. Fig. 14 presents the box plot regarding the performances of IALO algorithm and other meta-heuristic algorithms for 10 independent runs. This figure shows that the worst algorithm is BBO, while FA has the best fitness value (best cost) and the proposed IALO algorithm has the best mean cost value.

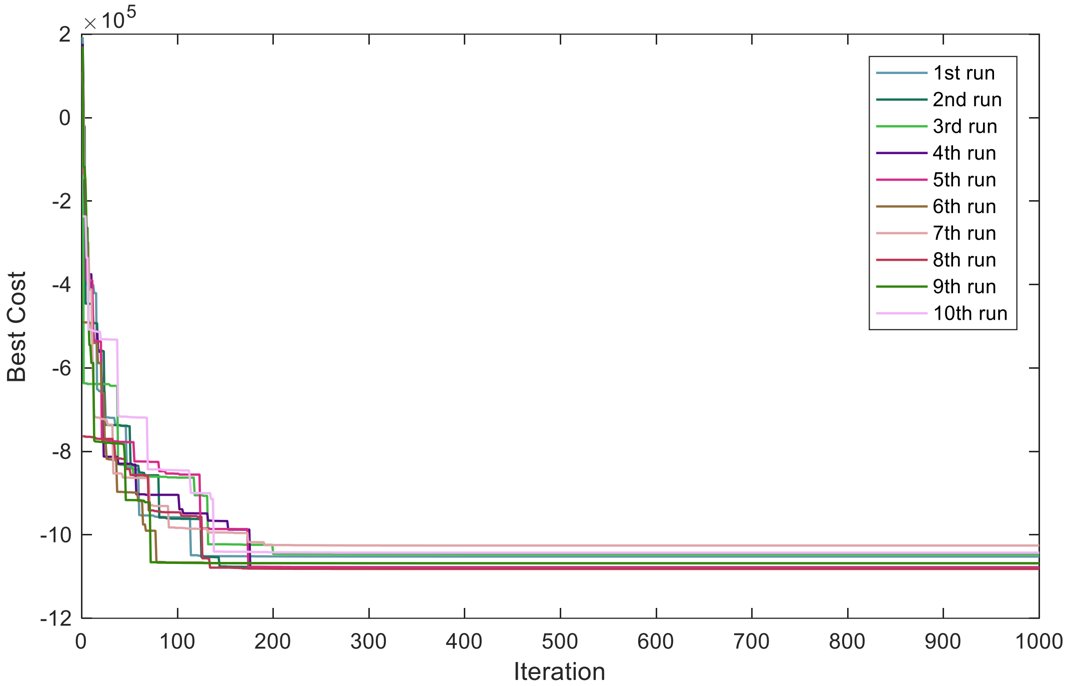


Fig. 13: Convergence curves of IALO algorithm with 10 independent runs

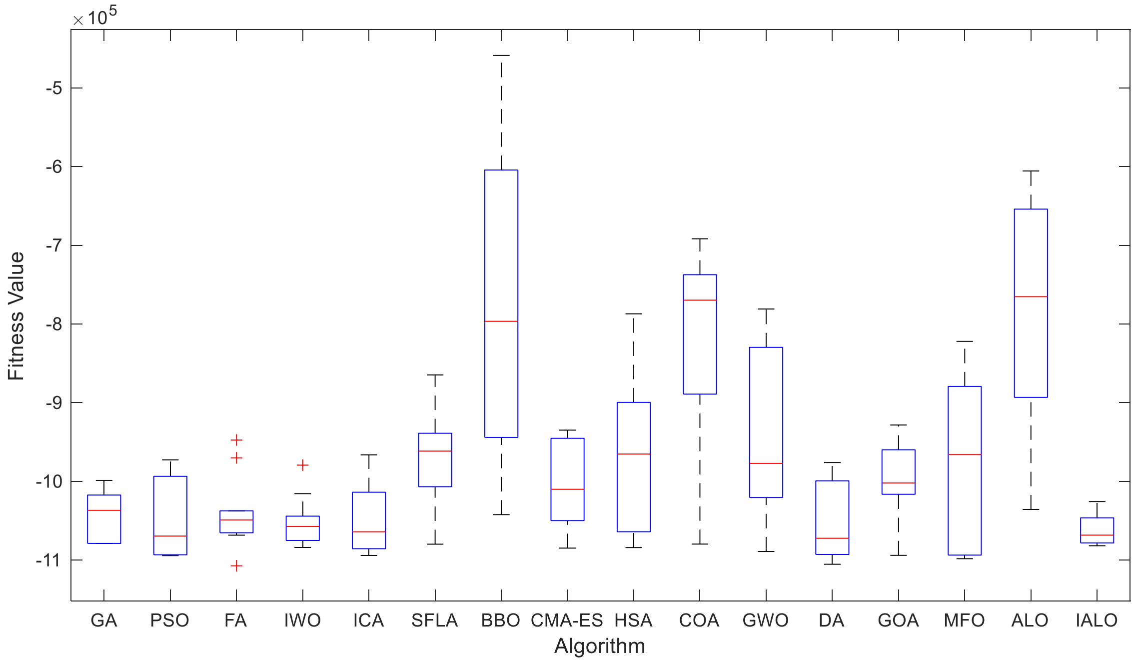


Fig. 14: Performances of IALO algorithm and other meta-heuristic algorithms for 10 independent runs

6.0 CONCLUSION AND FUTURE WORK

Antlion Optimization (ALO) that imitates the hunting mechanism of antlions has some drawbacks. In this study, the improved ALO algorithm which is called IALO was presented. The random walking mechanism and selection methods are some of the innovations made in ALO algorithm. The innovations made in the slip rates of the falling ants, and other adjustments, reveal the IALO algorithm. As there are no studies on time analysis of ALO algorithm in the literature, 10 well known benchmark functions were taken from the literature to show the performance of IALO algorithm according to CPU time and the number of function evaluations (NFE) metrics. The proposed IALO algorithm was compared with the other well-known meta-heuristic algorithms using these benchmark functions with multi-dimensions. The test results show that the proposed IALO has obtained the best performance in terms of different metrics, such as optimality, accuracy, mean best/std. IALO's run time was reduced by virtue of improvements made in the ALO algorithm, but, the best CPU-time results were not obtained in the benchmark tests. However, the CPU-time/NFE results show that the run-time of the IALO is much better than the original ALO.

For QAP tests, 15 recent meta-heuristic algorithms (GA, PSO, FA, IWO, ICA, SFLA, BBO, CMA-ES, HSA, COA, GWO, DA, GOA, MFO and ALO) were used. QAP results show that the proposed IALO algorithm obtained the best performance according to the mean cost, standard deviation and the worst cost except of the best cost. At the end of the QAP tests with 10 independent runs, IALO results present the stable convergence curves. This shows that this proposed algorithm resolves the QAP in different runs. For the future works, ALO algorithm's random walking mechanism can be improved to a further level, and IALO can be implemented to different real optimization problems, such as parallel machine scheduling, optimal robot path planning, capacitated vehicle routing problem, etc.

7.0 APPENDIX A

QAP Matrices

Weight matrix (W) is given below for QAP model used in this study [81]:

$$W = \begin{bmatrix}
4 & 9 & 9 & 6 & 7 & 3 & 9 & 7 & 7 & 7 & 5 & 4 & \boldsymbol{y} & 5 & 9 & 2 & 4 & 1 & 4 & 7 \\
9 & 3 & 7 & 1 & 5 & 7 & 7 & 9 & 5 & 5 & 3 & 5 & 6 & 6 & 5 & 8 & 5 & 1 & 2 & 4 \\
9 & 7 & 6 & 4 & 4 & 1 & 6 & 6 & 8 & 9 & 4 & 3 & 3 & 5 & 7 & 1 & 7 & 6 & 7 & 2 \\
6 & 1 & 4 & 5 & 3 & 7 & 4 & 4 & 7 & 6 & 6 & 7 & 7 & 4 & 3 & 1 & 4 & 5 & 8 & 1 \\
7 & 5 & 4 & 3 & 1 & 5 & 6 & 7 & 4 & 7 & 3 & 4 & 4 & 4 & 2 & 1 & 6 & 5 & 2 & 7 \\
3 & 7 & 1 & 7 & 5 & 9 & 3 & 6 & 6 & 7 & 5 & 3 & 6 & 8 & 6 & 7 & 6 & 4 & 2 & 1 \\
9 & 7 & 6 & 4 & 6 & 3 & 3 & 4 & 6 & 4 & 3 & 5 & 6 & 4 & 2 & 5 & 5 & 9 & 6 & 6 \\
7 & 9 & 6 & 4 & 7 & 6 & 4 & 4 & 2 & 3 & 6 & 5 & 7 & 3 & 1 & 6 & 9 & 4 & 1 & 3 \\
7 & 5 & 8 & 7 & 4 & 6 & 6 & 2 & 7 & 3 & 7 & 8 & 5 & 5 & 8 & 4 & 4 & 3 & 7 & 5 \\
7 & 5 & 9 & 6 & 7 & 7 & 4 & 3 & 3 & 9 & 9 & 7 & 4 & 6 & 2 & 4 & 5 & 3 & 9 & 5 \\
5 & 3 & 4 & 6 & 3 & 5 & 3 & 6 & 7 & 9 & 4 & 3 & 3 & 5 & 7 & \boldsymbol{x} & 6 & 6 & 5 & 4 \\
4 & 5 & 3 & 7 & 4 & 3 & 5 & 5 & 8 & 7 & 3 & 6 & 5 & 7 & 9 & 5 & 8 & 6 & 4 & 3 \\
\boldsymbol{y} & 6 & 3 & 7 & 4 & 6 & 6 & 7 & 5 & 4 & 3 & 5 & 1 & 8 & 4 & 7 & 5 & 5 & 7 & 5 \\
5 & 6 & 5 & 4 & 4 & 8 & 4 & 3 & 5 & 6 & 5 & 7 & 8 & 6 & 2 & 4 & 9 & 5 & 3 & 2 \\
9 & 5 & 7 & 3 & 2 & 6 & 2 & 1 & 8 & 2 & 7 & 9 & 4 & 2 & 7 & 4 & 7 & 7 & 9 & 5 \\
2 & 8 & 1 & 1 & 1 & 7 & 5 & 6 & 4 & 4 & \boldsymbol{x} & 5 & 7 & 4 & 4 & 4 & 4 & 7 & 5 & 4 \\
4 & 5 & 7 & 4 & 6 & 6 & 5 & 9 & 4 & 5 & 6 & 8 & 5 & 9 & 7 & 4 & 7 & 4 & 8 & 6 \\
1 & 1 & 6 & 5 & 5 & 4 & 9 & 4 & 3 & 3 & 6 & 6 & 5 & 5 & 7 & 7 & 4 & 4 & 4 & 5 \\
4 & 2 & 7 & 8 & 2 & 2 & 6 & 1 & 7 & 9 & 5 & 4 & 7 & 3 & 9 & 5 & 8 & 4 & 6 & \boldsymbol{x} \\
7 & 4 & 2 & 1 & 7 & 1 & 6 & 3 & 5 & 5 & 4 & 3 & 5 & 2 & 5 & 4 & 6 & 5 & \boldsymbol{x} & 9
\end{bmatrix}, \; x = 10000, \; y = -10000$$

In QAP model, the location vectors (x, y) are given below:

$$x = \begin{bmatrix} 70\ 63\ 11\ 5\ 43\ 94\ 10\ 2\ 68\ 74\ 24\ 89\ 59\ 59\ 41\ 7\ 73\ 86\ 34\ 88 \\ 83\ 19\ 66\ 8\ 64\ 56\ 92\ 83\ 36\ 77\ 74\ 8\ 51\ 30\ 60\ 51\ 17\ 81\ 65\ 54 \end{bmatrix}$$

$$y = \begin{bmatrix} 89\ 5\ 29\ 57\ 50\ 73\ 47\ 93\ 96\ 67\ 51\ 80\ 22\ 89\ 72\ 83\ 11\ 56\ 88\ 82 \\ 6\ 55\ 99\ 79\ 23\ 39\ 12\ 10\ 33\ 75\ 41\ 86\ 59\ 68\ 54\ 9\ 39\ 45\ 5\ 72 \end{bmatrix}$$

REFERENCES

[1] S. Mirjalili, "The ant lion optimizer," *Adv. Eng. Softw.*, vol. 83, pp. 80–98, 2015.

[2] M. Raju, L. C. Saikia, and 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.*, vol. 80, pp. 52–63, 2016.

[3] V. K. Kamboj, A. Bhadoria, and S. K. Bath, "Solution of non-convex economic load dispatch problem for small-scale power systems using ant lion optimizer," *Neural Comput. Appl.*, vol. 28, no. 8, pp. 2181–2192, 2017.

[4] H. Kılıç and U. Yüzgeç, "Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling," *Comput. Ind. Eng.*, vol. 132, no. June 2019, pp. 166–186, 2019.

[5] H. Kılıç and U. Yüzgeç, "Tournament selection based antlion optimization algorithm for solving quadratic assignment problem," *Engineering Science and Technology, an International Journal*, vol. 22, no. 2, Elsevier B.V., pp. 673–691, 01-Apr-2019.

[6] M. Petrović and Z. Miljković, "Biologically inspired optimization algorithms for flexible process planning," *Lect. Notes Mech. Eng.*, pp. 417–428, 2017.

[7] P. Yao and H. Wang, "Dynamic Adaptive Ant Lion Optimizer applied to route planning for unmanned aerial vehicle," *Soft Comput.*, vol. 21, no. 18, pp. 5475–5488, 2017.

[8] N. Chopra and S. Mehta, "Multi-objective optimum generation scheduling using Ant Lion Optimization," in *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3), INDICON 2015*, 2016.

[9] E. Gupta and A. Saxena, "Performance Evaluation of Antlion Optimizer Based Regulator in Automatic Generation Control of Interconnected Power System," *J. Eng. (United States)*, vol. 2016, 2016.

[10] S. S. Nair, K. P. S. Rana, V. Kumar, and A. Chawla, "Efficient Modeling of Linear Discrete Filters Using Ant Lion Optimizer," *Circuits, Syst. Signal Process.*, vol. 36, no. 4, 2017.

[11] H. Kilic, U. Yuzgec, and C. Karakuzu, "Improved Antlion Optimizer Algorithm and Its Performance on Neuro Fuzzy Inference System," *Neural Netw. World*, vol. 29, no. 4, pp. 235–254, 2019.

[12] H. Kilic, U. Yuzgec, and C. Karakuzu, "A novel improved antlion optimizer algorithm and its comparative performance," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3803–3824, 2020.

[13] T. C. Koopmans and M. Beckmann, "Assignment Problems and the Location of Economic Activities," *Econometrica*, vol. 25, no. 1, p. 53, 1957.

[14] A. N. Elshafei, "Hospital Layout as a Quadratic Assignment Problem," *Oper. Res. Q.*, vol. 28, no. 1, p. 167, 1977.

[15] R. S. Liggett, "The Quadratic Assignment Problem: An Experimental Evaluation of Solution Strategies," *Manage. Sci.*, vol. 27, no. 4, pp. 442–458, 1981.

[16] M. R. Wilhelm and T. L. Ward, "Solving Quadratic Assignment Problems by 'Simulated Annealing,'" *IIE Trans.*, vol. 19, no. 1, pp. 107–119, 1987.

[17] D. T. Connolly, "An improved annealing scheme for the QAP," *Eur. J. Oper. Res.*, vol. 46, no. 1, pp. 93–100, 1990.

[18] J. Bos, "Zoning in forest management: A quadratic assignment problem solved by simulated annealing," *J. Environ. Manage.*, vol. 37, no. 2, pp. 127–145, 1993.

[19] T. Peng, W. Huanchen, and Z. Dongme, "Simulated annealing for the quadratic assignment problem: A further study," *Comput. Ind. Eng.*, vol. 31, no. 3–4, pp. 925–928, 1996.

[20] V. Maniezzo, M. Dorigo, and A. Colorni, "Algodesk: An experimental comparison of eight evolutionary heuristics applied to the Quadratic Assignment Problem," *Eur. J. Oper. Res.*, vol. 81, no. 1, pp. 188–204, 1995.

[21] R. E. Burkard and E. Çela, "Heuristics for biquadratic assignment problems and their computational comparison," *Eur. J. Oper. Res.*, vol. 83, no. 2, pp. 283–300, 1995.

[22] G. Paul, "Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem," *Oper. Res. Lett.*, vol. 38, no. 6, pp. 577–581, 2010.

[23] M. Bashiri and H. Karimi, "Effective heuristics and meta-heuristics for the quadratic assignment problem with tuned parameters and analytical comparisons," *J. Ind. Eng. Int.*, vol. 8, no. 1, 2012.

[24] D. M. Tate and A. E. Smith, "A genetic approach to the quadratic assignment problem," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 73–83, 1995.

[25] R. Tavakkoli-Moghaddain and E. Shayan, "Facilities layout design by genetic algorithms," *Comput. Ind. Eng.*, vol. 35, no. 3–4, pp. 527–530, 1998.

[26] J. S. Kochhar, B. T. Foster, and S. S. Heragu, "HOPE: A genetic algorithm for the unequal area facility layout problem," *Comput. Oper. Res.*, vol. 25, no. 7–8, pp. 583–594, 1998.

[27] T. D. Mavridou and P. M. Pardalos, "Simulated Annealing and Genetic Algorithms for the Facility Layout Problem: A Survey," *Comput. Optim. Appl.*, vol. 7, pp. 111–126, 1997.

[28] C. Wen-Chyuan and C. Chi, "Intelligent local search strategies for solving facility layout problems with the quadratic assignment problem formulation," *Eur. J. Oper. Res.*, vol. 2217, no. 97, pp. 457–488, 1998.

[29] V. Maniezzo and A. Colorni, "The ant system applied to the quadratic assignment problem," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 5, pp. 769–778, 1999.

[30] E.-G. Talbi, O. Roux, C. Fonlupt, and D. Robillard, "Parallel Ant Colonies for the quadratic assignment problem," *Futur. Gener. Comput. Syst.*, vol. 17, no. 4, pp. 441–449, 2001.

[31] W. Zhu, J. Curry, and A. Marquez, "SIMD tabu search for the quadratic assignment problem with graphics hardware acceleration," *Int. J. Prod. Res.*, vol. 48, no. 4, pp. 1035–1047, 2010.

[32] F. Hafiz and A. Abdennour, "Particle Swarm Algorithm variants for the Quadratic Assignment Problems - A probabilistic learning approach," *Expert Syst. Appl.*, vol. 44, pp. 413–431, 2016.

[33] W. L. Lim, A. Wibowo, M. I. Desa, and H. Haron, "A biogeography-based optimization algorithm hybridized with tabu search for the quadratic assignment problem," *Comput. Intell. Neurosci.*, vol. 2016, 2016.

[34] W. Chmiel, P. Kadłuczka, J. Kwiecień, and B. Filipowicz, "A comparison of nature inspired algorithms for the quadratic assignment problem," *Bull. Polish Acad. Sci. Tech. Sci.*, vol. 65, no. 4, pp. 513–522, 2017.

[35] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. Addison-We. 1989.

[36] X. S. Yang, "Firefly Algorithm," *Nature-Inspired Metaheuristic Algorithms*, pp. 79–90, 2007.

[37] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5792 LNCS, pp. 169–178.

[38] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, vol. 4, pp. 1942–1948.

[39] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.

[40] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecol. Inform.*, vol. 1, no. 4, pp. 355–366, 2006.

[41] H. Y. Sang, P. Y. Duan, and J. Q. Li, "An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem," *Swarm Evol. Comput.*, vol. 38, pp. 42–53, 2018.

[42] X. Zhang, Y. Niu, G. Cui, and Y. Wang, "A modified invasive weed optimization with crossover operation," in *Proceedings of the 8th World Congress on Intelligent Control and Automation*, 2010, pp. 11–14.

[43] S. Hosseini and A. Al Khaled, "A survey on the Imperialist Competitive Algorithm metaheuristic: Implementation in engineering domain and directions for future research," *Applied Soft Computing Journal*, vol. 24. pp. 1078–1094, 2014.

[44] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, 2007, pp. 4661–4667.

[45] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization," *Eng. Optim.*, vol. 38, no. 2, pp. 129–154, 2006.

[46] K. K. Bhattacharjee and S. P. Sarmah, "Shuffled frog leaping algorithm and its application to 0/1 knapsack problem," *Appl. Soft Comput. J.*, vol. 19, pp. 252–263, 2014.

[47] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, 2008.

[48] H. Ma and D. Simon, "Blended biogeography-based optimization for constrained optimization," *Eng. Appl. Artif. Intell.*, vol. 24, no. 3, pp. 517–525, 2011.

[49] H. Beyer and B. Sendhoff, "Covariance matrix adaptation revisited–the CMSA evolution strategy–," *Parallel Probl. Solving from Nature–PPSN X*, pp. 123–132, 2008.

[50] M. Willjuice Iruthayarajan and S. Baskar, "Covariance matrix adaptation evolution strategy based design of centralized PID controller," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5775–5781, 2010.

[51] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[52] X. S. Yang, "Harmony search as a metaheuristic algorithm," *Studies in Computational Intelligence*, vol. 191. pp. 1–14, 2009.

[53] C. A. Coello Coello and R. L. Becerra, "Efficient evolutionary optimization through the use of a cultural algorithm," *Eng. Optim.*, vol. 36, no. 2, pp. 219–236, 2004.

[54] C. Soza, R. L. Becerra, M. C. Riff, and C. A. Coello Coello, "Solving timetabling problems using a cultural algorithm," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 337–344, 2011.

[55] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[56] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, 2016.

[57] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper Optimisation Algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, 2017.

[58] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Syst.*, vol. 89, pp. 228–249, 2015.

[59] W. Luo and Y. Li, "Benchmarking heuristic search and optimisation algorithms in Matlab," in *2016 22nd*

International Conference on Automation and Computing, ICAC 2016: Tackling the New Challenges in Automation and Computing, 2016, pp. 250–255.

[60] D. Karaboga, “An idea based on Honey Bee Swarm for Numerical Optimization,” *Tech. Rep. TR06, Erciyes Univ.*, no. TR06, p. 10, 2005.

[61] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[62] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[63] K. V Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, vol. 28. 2005.

[64] S. Sreejith, K. Chandrasekaran, and S. P. Simon, “Touring ant colony optimization technique for optimal power flow incorporating thyristor controlled series compensator,” in *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 2009, pp. 1127–1132.