Computers & Industrial Engineering 132 (2019) 166–186

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling

Haydar Kılıç, Uğur Yüzgeç*

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

ARTICLE INFO

Keywords:
Antlion
Parallel Machine Scheduling
Optimization
Tournament selection

ABSTRACT

In this paper, we consider the parallel machine scheduling which is combinatorial optimization problem. An improved meta-heuristic method is proposed to solve this problem. Antlion optimization algorithm (ALO) is one of the new meta-heuristic optimization algorithms and it is inspired by the hunting mechanism of antlion in nature. The original algorithm has some deficiencies like the runtime. Some improvements on the classic ALO algorithm are presented in this study. One of these improvements is to use the tournament method instead of the roulette wheel method in the selection of the antlion from the population for the random walk which is the most important mechanism on the original ALO algorithm. Some mechanisms in the original ALO algorithm have been improved, such as ant random walking, reproduction, sliding of ants towards antlion, elitism and selection procedure. There is no time analysis result about ALO algorithm in the literature. Due to this reason, it is aimed to show the performance of the proposed IALOT algorithm especially in runtime analysis. The proposed improved ALO algorithm via tournament selection method (IALOT) has been compared with the other popular well-known meta-heuristic algorithms on the multi-dimensions benchmark test functions and the Parallel Machine Scheduling (PMS) problem to evaluate its performance. The obtained results show that the IALOT has the best performance in terms of the optimality, accuracy and mean best metrics for the benchmark tests. According to the PMS results, the proposed IALOT algorithm is the best algorithm in comparison with the other meta-heuristic algorithms.

1. Introduction

In today's engineering sciences, meta-heuristic algorithms have a great advantage in solving optimization problems. Optimization techniques are used to obtain the best final product to ensure that the prototype models created during research and development adapt to real life. In addition to well-known popular meta-heuristic algorithms, the use of new optimization algorithms in real-life optimization problems is increasing in recent years. Looking at the literature, it can be seen that meta-heuristic algorithms inspired by nature are more common and give good results. Scientists often use routines such as animal hunting techniques, nutrition methods, mating habits in the invention of such algorithms. As an example of these algorithms, particle swarm algorithm, artificial bee colony algorithm can be given.

Meta-heuristic algorithms are classified into evolutionary based algorithms, physical based algorithms, swarm intelligence algorithms, bio-inspired algorithms and other nature-inspired algorithms (Beheshti & Shamsuddin, 2013; Nanda & Panda, 2014; Tayarani, Yao, & Xu, 2015). The evolutionary based algorithms are inspired by natural selection in Darwin theory (Nanda & Panda, 2014). These algorithms begin with a random population and update the population according to various mechanisms of evolution such as mutation and crossover. Genetic algorithm (GA) (Goldberg, 1989; Holland, 1975); Differential Evolution (DE) algorithm (Price, Storn, & Lampinen, 2005; Storn & Price, 1997) are the well known examples of the evolutionary based meta-heuristic algorithms. In the physical based algorithms, the optimization process starts with a single solution and it is changed by physical equations during the iterations. Tabu Search algorithm (TS) (Glover, 1986), Simulated Annealing algorithm (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983); Harmony Search algorithm (HSA) (Geem, Kim, & Loganathan, 2001; Yang, 2009) and Shuffled Frog Leaping algorithm (SFLA) (Bhattacharjee & Sarmah, 2014; Eusuff, Lansey, & Pasha, 2006) are the most popular well-known examples of the physical based meta-heuristic algorithms. Swarm intelligence algorithms are the group of natural meta-heuristic algorithms which mimic the collective intelligence, such as flocks of birds, ant colonies, fish swarm etc. (Nanda & Panda, 2014). The main such algorithms include: Particle Swarm Optimization (PSO) algorithm by Kennedy & Eberhart (Eberhart &

* Corresponding author.
E-mail addresses: haydar.kilic@bilecik.edu.tr (H. Kılıç), ugur.yuzgec@bilecik.edu.tr (U. Yüzgeç).

https://doi.org/10.1016/j.cie.2019.04.029
Received 27 June 2018; Received in revised form 16 April 2019; Accepted 18 April 2019
Available online 20 April 2019
0360-8352/

Kennedy, 1995; Kennedy & Eberhart, 1995), Artificial Bee Colony (ABC) algorithm by Karaboga and his colleagues (Karaboga & Ozturk, 2011; Karaboga, Gorkemli, Ozturk, & Karaboga, 2014; Karaboga, 2005), Ant Colony Optimization (ACO) algorithm by Dorigo et al. (1996, 2006) and Fish Swarm Algorithm (FSA) by Li et al. (Li, Shao, & Qian, 2002; Li, 2003). The bio-inspired algorithms include natural meta-heuristics derived from living phenomena and activities of biological organisms (Nanda & Panda, 2014). The most important examples of such algorithms are Artificial Immune algorithm (AI) (Dasgupta, 1999; de Charsto & Timmis, 2002) and Bacterial Foraging Optimization algorithm (BFO) (Mishra, 2005; Passino, 2002; Tang, Wu, & Saunders, 2006). There are a lot of studies regarding the other nature inspired meta-heuristic algorithms and application to real optimization problems. Some of these recent algorithms are Biogeography-Based Optimization algorithm (BBO) (Ma & Simon, 2011; Simon, 2008), Cuckoo Search algorithm (CSA) (Yang and Deb, 2009, 2014), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Willjuice Iruthayarajan & Baskar, 2010), Cultural Optimization algorithm (COA) (Coello & Becerra, 2004; Soza, Becerra, Riff, & Coello Coello, 2011), Dragonfly Optimization algorithm (DA) (Mirjalili, 2016; Sree Ranjini & Murugan, 2017), Firefly algorithm (FA) (Yang, 2009, 2010), Fruit Fly Optimization algorithm (FOA) (Pan, 2012), Gray Wolf Optimization algorithm (GWO) (Medjahed, Ait Saadi, Benyettou, & Ouali, 2016; Mirjalili, Mirjalili, & Lewis, 2014; Mirjalili, 2015), Grasshopper Optimization algorithm (GOA) (Saremi, Mirjalili, & Lewis, 2017), Gravitational Search algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009; Sabri, Puteh, & Mahmood, 2013), Imperialist Competitive algorithm (ICA) (Atashpaz-Gargari & Lucas, 2007; Hosseini & Al Khaled, 2014), Invasive Weed Optimization algorithm (IWO) (Mehrabian & Lucas, 2006; Sang, Duan, & Li, 2018), Moth-Flame Optimization algorithm (MFO) (Mirjalili, 2015; Sayed & Hassanien, 2017), Touring Ant Colony Optimization algorithm (TACO) (Sreejith, Chandrasekaran, & Simon, 2009) and Whale Optimization algorithm (WOA) (Kaveh & Ghazaan, 2017; Mirjalili & Lewis, 2016).

Antlion optimization algorithm (ALO) is one of the meta-heuristic algorithms that are inspired by nature and animals. The antlions are from the Myrmeleontidae family of predatory insect species, which take their name from extremely interesting nutritional behavior in the larval period. The larvae hunting behavior of the antlion is examined by Seyedali Mirjalili and presented as an antlion optimization algorithm in 2015 (Mirjalili, 2015). The ALO algorithm is basically derived from the hunting strategy of antlions. It comprises 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. There are some studies reported in the literature about implementations or improvement of the ALO algorithm. Some of these are summarized in Table 1.

Although ALO algorithm provides effective results for different optimization problems in engineering, it comprises some disadvantages. The greatest deficiency of ALO algorithm is the long runtime because of the random walking mechanism. In this study, antlion optimization algorithm is improved by eliminating some handicaps and a new improved ALO algorithm is proposed. Firstly, we have changed the random walking distance as 20% of maximum iteration number in the original ALO algorithm. In Mirjalili's paper (Mirjalili, 2015), the antlion is selected from the population by roulette wheel method for each ant's random walking. As can be seen from the optimization works, the roulette wheel method is more preferred for maximization problems. In the minimization problems, the tournament selection method is more efficient method (Noraini & Geraghty, 2011; Razali & Geraghty, 2011). Therefore, the tournament selection method has been used in this study instead of the roulette wheel method on the random walking mechanism. Because the tournament method has been used to get rid of the handicap of the roulette wheel method used in the selection of antlions in ALO, this proposed algorithm has been called the improved antlion optimization algorithm via tournament selection (IALOT). Moreover, we added some new movements 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 the proposed IALOT algorithm, the procedures about the catching prey, rebuilding the pit and the boundary checking process have been developed.

The Parallel Machine Scheduling problem (PMS), which is one of the best known combinatorial optimization problems, is a set of independent jobs to be scheduled on a number of parallel machines. Many of PMS problems are known to be NP-hard problems. They are hard to solve and computationally expensive because of the fact that the spending time to find a feasible solution increases exponentially with problem size. Due to the complexity of the problem, it is more preferable to find a suitable solution by heuristic algorithms instead of an optimal solution for the parallel-machine programming problem. Thus, the literature regarding PMS works shows that heuristics and meta-heuristic algorithms (Caniyilmaz, Benli, & Ilkay, 2015; Chen, Cheng, Wang, & Chen, 2012; Kizilay, Tasgetiren, Bulut, & Bostan, 2014; Moslehi & Mahnam, 2011; Pakzad-Moghaddam, 2016; Wang, Wang, Zhao, Zhang, & Xu, 2013; Zhang, Song, & Wu, 2013) can be used to tackle these types of problems. In this study, we have dealt with the unrelated parallel machine scheduling problem with setup times. This unrelated PMS problem is strongly NP-hard to minimize makespan $R||C_{max}$ according to the machine environment, job characteristics and optimality criterion (Graham, Lawler, Lenstra, & Rinnooy Kan, 1979). Because of the complexity of this problem, exact algorithms for medium

Table 1
Literature brief about implementations and improvements of ALO.

Ref.	Implementations of ALO	Ref.	Improvements of ALO
Ali et al. (2016, 2017)	Optimal allocation and sizing of renewable distributed generations	Emary, Zawbaa, and Hassanien (2016)	Binary ALO approaches
Dubey, Pandit, and Panigrahi (2016)	Short-term wind integrated hydrothermal power generation scheduling	Zawbaa, Emary and Grosan (2016), Tharwat and Hassanien (2018)	Chaotic based ALO variants
Kamboj, Bhadoria, and Bath (2017)	Non-convex economic load dispatch problem	Emary and Zawbaa (2018)	Levy flight based ALO approach
Raju, Saikia, and Sinha (2016)	Optimal controller design	Wang, Zhou, and Zheng (2018)	ALO with adaptive boundary mechanism
Petrovic et al. (2015)	Optimal process planning	Dinkar and Deep (2018)	Opposition based Levy flight ALO
Yao and Wang (2016)	Optimal route planning for unmanned aerial vehicle	Yuan, Zheng, Lu, Li, and Masood (2018)	ALO with differential mutation operator
Chopra and Mehta (2015)	Multi-objective optimal generation scheduling	Dinkar and Deep (2017)	Opposition based Laplacian ALO algorithm
Gupta and Saxena (2016)	Automatic generation control of interconnected power system	Rajan, Jeevan, and Malakar (2017)	Weighted elitism based ALO algorithm
Nair, Rana, Kumar, and Chawla (2016)	Determining the optimal coefficients of IIR filters	Dinkar and Deep (2019)	Accelerated opposition-based ALO algorithm

and large-sized problems are insufficient (Ramezanian & Saidi-Mehrabad, 2012). Due to the fact that this type PMS problem is strongly NP-hard, IALOT algorithm is proposed to find good quality solutions for the presented problem.

In this study, we propose the improved antlion optimization algorithm (IALOT) via tournament selection method to overcome the drawback of the long runtime of the original ALO algorithm. In Mirjalili's work (Mirjalili, 2015), there was no time analysis about the original ALO algorithm. For this reason, we would like to show the performance of the proposed IALOT algorithm, especially on the runtime analysis. From the literature, ten benchmark functions were taken to evaluate the performance of the proposed IALOT algorithm. In comparison works, the IALOT algorithm was compared with the well known meta-heuristic algorithms according to the mean of the best value, CPU time, number of function evaluations (NFE) optimality, accuracy metrics. The main purpose of this study is to apply the proposed IALOT algorithm to the PMS problem. The proposed IALOT algorithm's performance on PMS problem was compared with the well known recent meta-heuristic algorithms such as the original ALO algorithm, Genetic algorithm (GA), Particle Swarm Optimization algorithm (PSO), Firefly algorithm (FA), Invasive Weed Optimization algorithm (IWO), Imperialist Competitive algorithm (ICA), Shuffled Frog Leaping algorithm (SFLA), Biogeography-Based Optimization algorithm (BBO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Harmony Search algorithm (HSA), Cultural Optimization algorithm (COA), Gray Wolf Optimization algorithm (GWO), Dragonfly Optimization algorithm (DA), Grasshopper Optimization algorithm (GOA) and Moth-Flame Optimization algorithm (MFO). The rest of the paper is organized as follows:

Section 2 introduces the original ALO algorithm. The proposed IALOT algorithm and its novelty and improvement are presented in Section 3. In Section 4, the basic information about the Parallel Machine Scheduling (PMS) problem is given briefly. For the benchmark and PMS tests, the performance of the proposed IALOT algorithm is provided in Section 5. In the last section, conclusion and the future works are mentioned.

2. AntLion optimizer (ALO)

The antlion is the common name of the predatory insect species, which take their names from the Myrmeleontidae family about extremely interesting nutritional behaviour in larvae. The antlion's 2.5–3 years of life has about 2-years larval period. They are fed as carnivores in this period and usually eat ants. Although they are nurtured as carnivores during the larval period, they are fed as herbivores during the adulthood that they metamorphosed. Fig. 1 illustrates the hunting mechanism of an antlion.

Feeding and hunting methods of antlions are a great show that is exhibited in a mathematical order. First of all, they dig a spiral road and bury themselves in the bottom of their cone-shaped trap, waiting for the ants to fall. When the falling ants want to go out, they throw sand from the place that they bury themselves and allow the ants to slip towards the bottom and eventually consume their prey. After this interesting hunting, they will be trapped for another hunt and start to wait again. The antlion's magnificent hunting technique was examined by Seyedali Mirjalili and announced in 2015 as Antlion Optimization Algorithm (ALO) (Mirjalili, 2015).

This interesting hunting technique starts with the random walk of ants and ant lions. The mathematical model of these walks is given below.

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

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

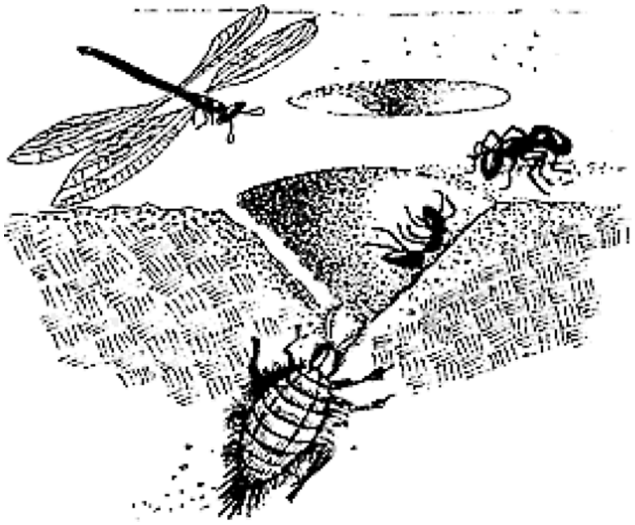

Fig. 1. Antlion's trap (web site, 2018).

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

where *rand* denotes the random number between 0 and 1. In order to keep the random walk of an 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 t is the iteration number, i is value of the variable number, X_i^t represents the normalized random walk vector of i-th variable at t-th iteration, a is the minimum random walk ($a = \min(X)$), b is the maximum random walk ($b = \max(X)$), c denotes the lower value of the dynamic boundary around the antlion, d denotes the upper value of the dynamic boundary around the antlion.

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

$$c_i^t = Antlion_i^t + c^t \tag{4}$$

$$d_i^t = Antlion_i^t + d^t \tag{5}$$

$$c^t = c^t . I^{-1} \tag{6}$$

$$d^t = d^t . I^{-1} \tag{7}$$

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

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

where *iter* is the current iteration and *MaxIter* is the maximum iteration. After hunting, antlions update their positions according to fitness values of ants. In Eq. (9), R_A^t is the antlion selected by roulette wheel method and R_E^t is the elite antlion. R_A^t and R_E^t are obtained by Eq. (3) for each iteration. The ants are positioned around the elite antlion and the antlion which is selected by roulette wheel method, with the following mathematical model.

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

These behaviors of antlions during hunting were discovered by Seyedali Mirjalili and announced in 2015 (Mirjalili, 2015). Its pseudo code is as follows:

The pseudo code of original Antlion Optimization algorithm:
Input: Fitness function, ants and antlions, maximum iteration number, population size.
Output: The elite antlion and its fitness value.
1) Initialize antlions positions.
2) Calculate fitness values of antlions using an objective function.
3) Sort fitness values and find the best antlion.
4) *while*(*iter* < *Max_iter*)
a) Select antlion by using roulette wheel method for building trap.
b) Slide randomly walking ants in a trap.
c) Create random walk for every ant around elite antlion and antlion selected by roulette wheel method.
d) Normalize random walk.
e) Calculate the fitness values of ants.
f) Merge fitness of ants and antlions.
g) Sort all population fitness and get rows that are up to population size.
h) Update the elite antlion.
5) *end while*

3. Improved AntLion Optimizer via Tournament Selection (IALOT)

In this study, original ALO algorithm is developed by improvements on the random walk, hunting of the ants, selection mechanism of antlions, combination of fitness values. The random walking mechanism on the original ALO algorithm gives the model of the ants' walking using the maximum iteration number, and this modeling mechanism affects the run time of the algorithm. Therefore, our first novelty on the ALO algorithm is achieved by reducing the distance of the random walk. As given in Eq.10, this distance value is used as 20% of maximum iteration number.

$$X(t) = [0, cumsum(2r(t_1) - 1), \cdots, cumsum(2r(t_n) - 1)],\ n = 1, 2, \cdots, MaxIter/5 \tag{10}$$

Thus, the long running time of the ALO algorithm has been shortened considerably with the innovation of random walkways. In some test functions, IALOT algorithm runs %500–2000 faster than the original ALO algorithm. In the phase of sliding ants towards the antlion on the original ALO code, the ants are shifted toward the antlion by a certain rate of slippage. Innovations made on the antlion's pit and on the shifting of the ants by throwing sand are explained by the following mathematical model.

$$\left.\begin{aligned} c_i^t &= Antlion_i^t + c^t \\ d_i^t &= Antlion_i^t + d^t \end{aligned}\right\} \quad if \quad 0.75 < option < 1 \tag{11}$$

$$\left.\begin{aligned} c_i^t &= Antlion_i^t - c^t \\ d_i^t &= Antlion_i^t - d^t \end{aligned}\right\} \quad if \quad 0.5 < option < 0.75 \tag{12}$$

$$\left.\begin{aligned} c_i^t &= -Antlion_i^t + c^t \\ d_i^t &= -Antlion_i^t + d^t \end{aligned}\right\} \quad if \quad 0.25 < option < 0.5 \tag{13}$$

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

where *option* is a randomly chosen variable. By updating the values of these shift rates according to the conditions, a more accurate and faster hunting mechanism has been established. Innovations in retrieving the ants from the search space into the redirection space have provided more precise results in the test functions.

At the end of each iteration in the original ALO algorithm, the elite antlion is updated, the populations of ant and antlion are combined and they are ranked according to their fitness values. For the next iteration, half of the combined population is taken as antlion positions. In the proposed IALOT algorithm, the improvement on the selection mechanism is that ants and antlion's fitness values are compared for each pair of ant and antlion instead of combining and sorting the population. In this comparison, if the ant's fitness value is better than antlion's fitness, antlion position is updated as ant position.

$$Antlion_i^t = Ant_i^t \quad if \quad f(Ant_i^t) < f(Antlion_i^t) \tag{15}$$

where $Antlion_i^t$ is the position of the selected *i-th* antlion at *t-th* iteration, Ant_i^t is the position of the selected *i-th* ant at *t-th* iteration, $f(Ant_i^t)$ stands for the fitness of the *i-th* ant at *t-th* iteration, $f(Antlion_i^t)$ denotes the fitness of the *i-th* antlion at *t-th* iteration. There is another innovation about ants that go out of the search area. In the original ALO code, there is the search space boundary checking mechanism. If the ants go beyond the boundaries of the search space, ants' positions are assumed at the boundaries of search space. This procedure causes the fact that the most of the search agents can be located at the same boundary points in the population. We proposed the new mechanism instead of this boundary mechanism in original ALO code. Unlike the original ALO algorithm, once the ants exit the search space, they return to the search space again. This mechanism increases the search performance of the improved ALO algorithm. This idea leaves the ants randomly in the search space by the following mathematical model:

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

where *rand* denotes random number in interval (Nanda & Panda, 2014); b_{low} is lower boundary and b_{up} is upper boundary of the search space. In the original ALO algorithm, it is assumed that every ant moves around an antlion selected by the roulette wheel method from the antlion population. In meta-heuristic algorithms, there are some selection methods as an alternative to the roulette wheel method. Some of them are the truncation selection, the linear ranking selection, the exponential ranking selection and the tournament selection method, etc. The tournament method which is one of the selection methods, arranges a tournament among randomly selected individuals from the population and the individual with the best fitness is the winner of the tournament. The tournament size or tour is the parameter of the tournament method. The tournament size takes values from 2 to population size (Blickle & Thiele, 1996).

The roulette wheel method that was used in Mirjalili's work (Mirjalili, 2015) is less efficient than the other methods for the minimization problems. For that reason, we used the tournament selection method, which is more useful in the minimization problems, rather than the roulette wheel method. In this study, the tournament size is determined as 2, so the tournament method takes randomly two groups from the population and the size of these groups is calculated by division of the population size to the tournament size. IALOT algorithm's pseudo code is given below:

The pseudo code of Improved Antlion Optimization Algorithm via Tournament Selection:
Input: *FitnessFunction*, *Positions*, *Max_iter*, *Pop_size*, *Tour_size*.
Output: The elite antlion position and its fitness value.
1) Initialize antlions' positions and calculate their fitness values.
2) Save the best antlion.
3) while (*iter* < *Max_iter*) and($|f_{best} - f_{worst}| < 10^{-6}$)
a) Calculate $X(t)$ as Eq.10
b) for every ant
i. Select antlion by tournament method for building a trap.
ii. Slide randomly walking ants in the trap as Eqs. (11)–(14).
iii. Create random walk for every ant around the elite antlion and selected antlion.
iv. Update the ant position.
v. Relocate the ants in case of outside search space using Eq. (16).

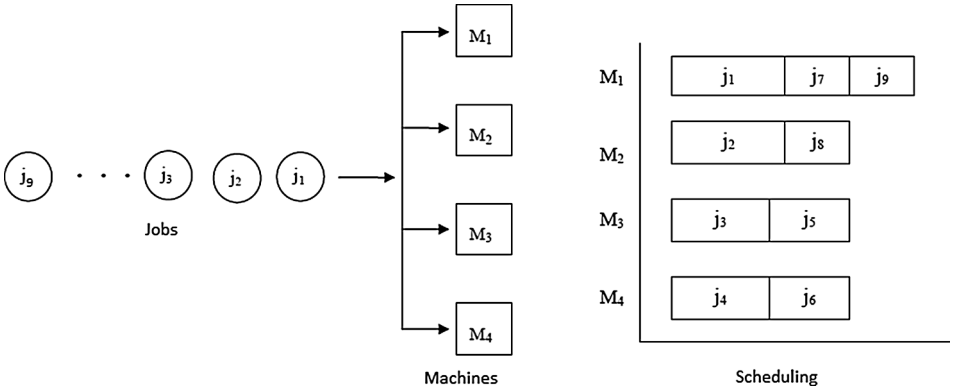

Fig. 2. PMS problem with four machines and nine jobs.

Table 2
Benchmark test 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 $f_3(x) = sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + sin^2(2\pi w_d)]$ $w_i = 1 + \frac{x_i - 1}{4},\ i = 1, 2, \cdots, d$	10	[−10,10]	$f(x) = 0$ at $x = (1, \cdots, 1)$
Rastrigin function $f_4(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$	10	[−5.12,5.12]	$f(x) = 0$ at $x = (0, \cdots, 0)$
Rosenbrock function $f_5(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	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 $x = (0, \cdots, 0)$

* d denotes the dimension of the problem in the benchmark mathematical formulas.

c) end for
d) Calculate the fitness values of ants.
e) Compare fitness of ants and antlions and update the positions of the antlions according to Eq. (15).
f) Save the elite antlion's position and its fitness value.
4) end while
5) Return the elite antlion

4. Parallel Machine Scheduling (PMS)

The Parallel Machine Scheduling (PMS) problem can be used when several machines of similar performance are available to process multiple jobs, simultaneously (McNaughton, 1959). An example of a PMS environment with four machines and nine jobs is shown in Fig. 2.

PMS problem deals with scheduling m parallel machines for n jobs waiting in a queue and the main objective of PMS is to minimize the completion time. The basic mathematical model of PMS is given below (Rabadi, Moraga, & Al-Salem, 2006):

$$
\begin{aligned}
&min\ \ C_{max}\\
&subject\ \ to\\
&\sum_{\substack{i=0\\ i\neq j}}^{n}\sum_{k=1}^{m} x_{ijk} = 1,\ \forall\, j = 1, 2, \cdots, n\\
&\sum_{j=0}^{n} x_{0jk} = 1, \quad \forall\, k = 1, 2, \cdots, m\\
&\sum_{\substack{i=0\\ i\neq h}}^{n} x_{ihk} - \sum_{\substack{j=0\\ j\neq h}}^{n} x_{hjk} = 0, \quad \forall\, h = 1, 2, \cdots, n\ \forall\, k = 1, 2, \cdots, m\\
&C_j \geq C_i + \sum_{k=1}^{m} x_{ijk}(S_{ijk} + p_{jk}) + M\left(\sum_{k=1}^{m} x_{ijk} - 1\right),\ \forall\, i = 0, 1, \cdots, n\ \forall\, j\\
&\quad = 1, 2, \cdots, n\\
&x_{ijk} \in \{0, 1\},\ \forall\, i = 0, 1, \cdots, n\ \forall\, j = 1, 2, \cdots, n\ \forall\, k = 1, 2, \cdots, m\\
&C_0 = 0,\ C_j \geq 0,\ \forall\, j = 1, 2, \cdots, n
\end{aligned}
\tag{17}
$$

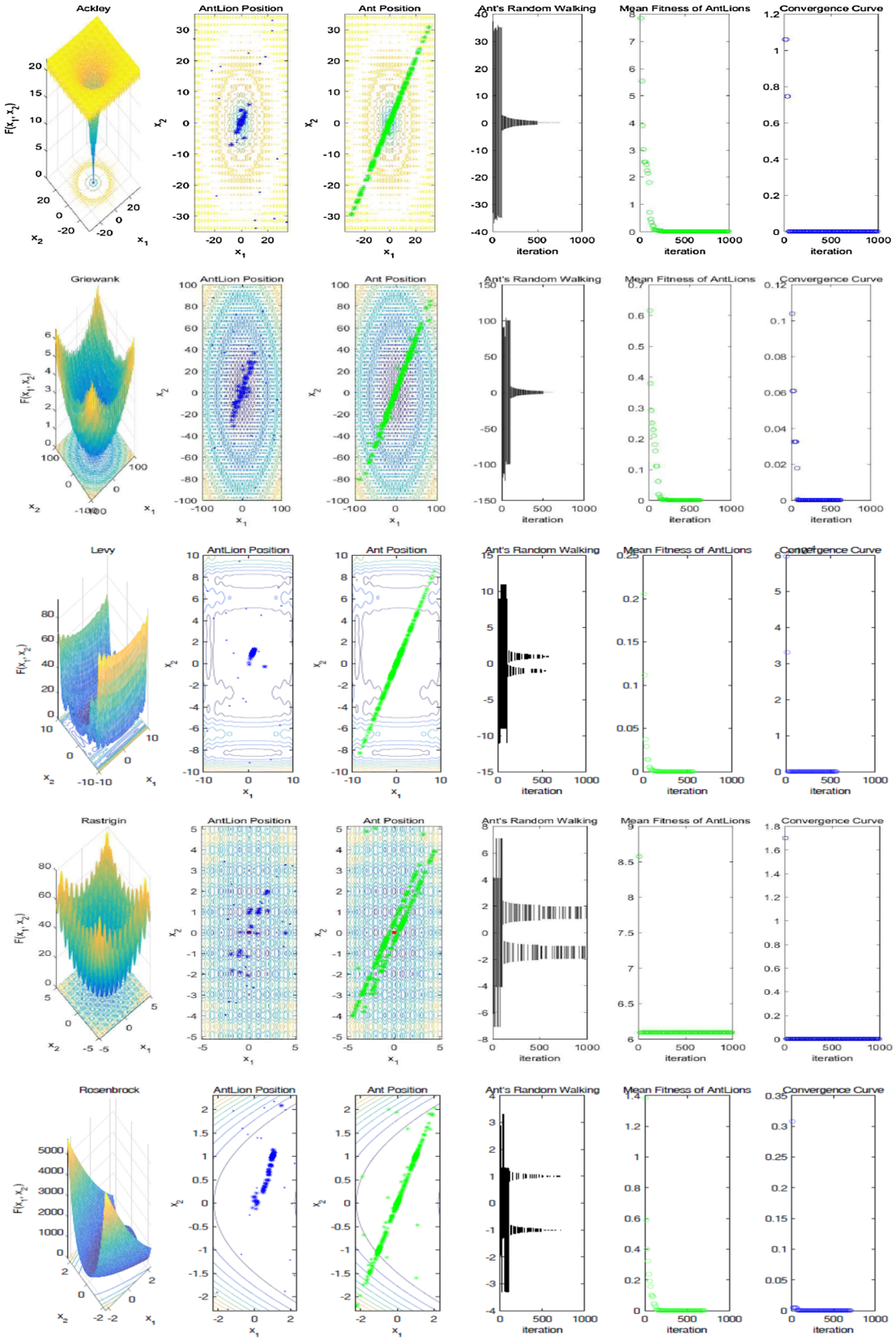

Fig. 3. IALOT algorithm analysis for all benchmark functions.

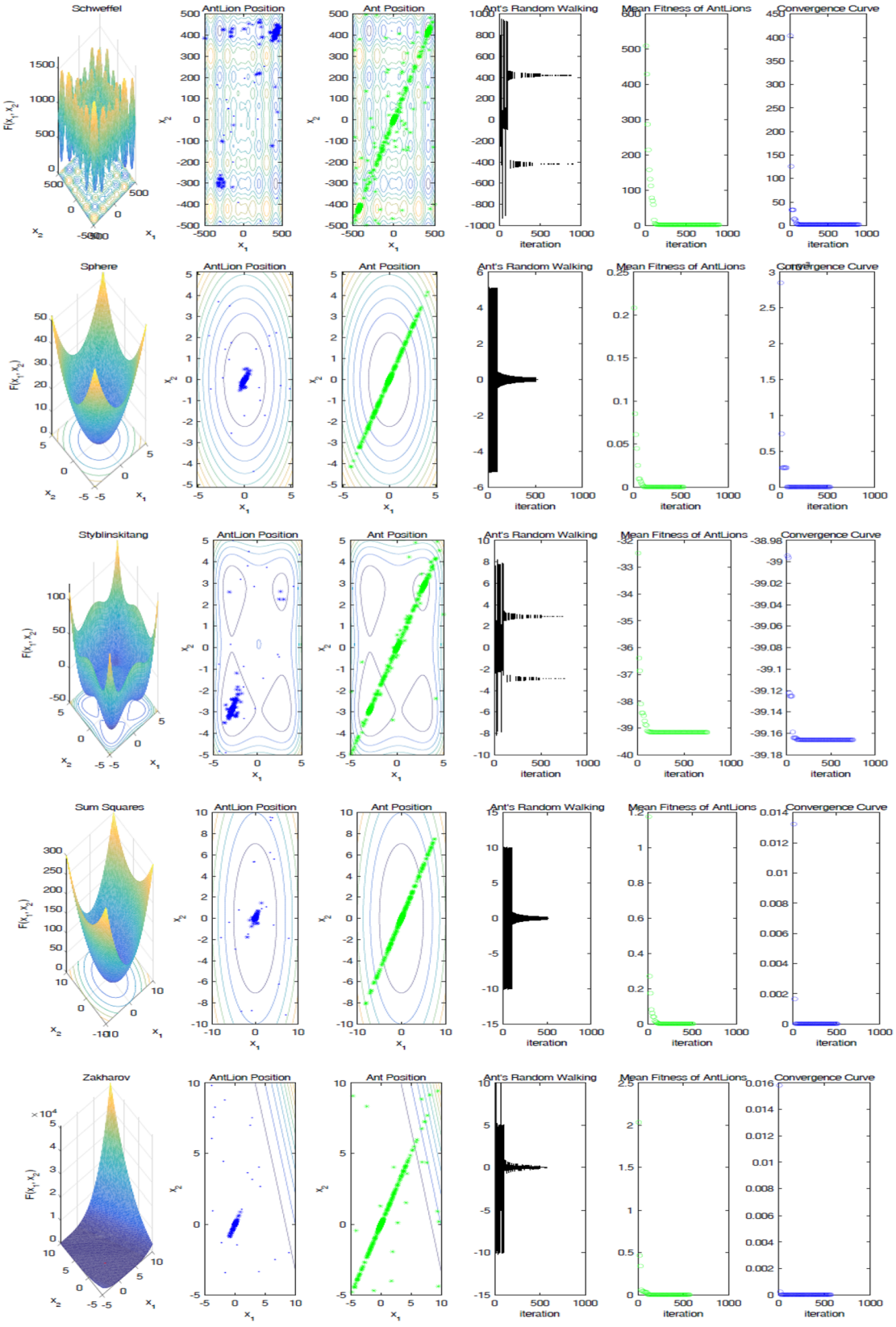

Fig. 3. (*continued*)

Table 3
Parameters of meta-heuristic algorithms for benchmark tests.

Algorithm	Parameters
PSO (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995)	Learning coefficients = 2.05, Constriction factor = 0.7298
ABC (Karaboga, 2005)	Number of food sources = 50, limit of attempts = 100
SA (Kirkpatrick et al., 1983)	Temperature = current iteration/maximum iteration number
DE (Price et al., 2005; Storn & Price, 1997)	Crossover probability = 0.5, differential weight = 0.8,differential strategy = DE/rand/1/bin
TACO (Sreejith et al., 2009)	Vaporing = 0.1, bit number = 18
ALO (Mirjalili, 2015)	Search agent = 100
IALOT	Search agent = 100, random walk size = Max Iter/5, tournament size = 2

Table 4
Comparison results of the proposed IALOT with the other algorithms.*

	Mean best (Std.Dev.)						
Function	PSO	ABC	SA	DE	TACO	ALO	IALOT
FN1	4.21e + 00	4.99e−10	1.57e+01	5.97e−07	1.51e+01	2.14e−01	**0.00e+00**
	(8.91e−01)	(6.42e−10)	(2.17e+00)	(1.31e−07)	(1.12e+00)	(5.05e−01)	**(0.00e+00)**
FN2	5.13e−01	2.47e−03	1.15e+00	1.20 e−01	1.25e+00	1.42e−01	**0.00e+00**
	(1.44e−01)	(4.37e−03)	(1.19e−01)	(2.30e−02)	(1.65e−01)	(7.52e−02)	**(0.00e+00)**
FN3	2.22e−01	**1.29e−12**	2.34e+00	1.47e−07	1.38e+00	2.49e−01	5.04e−06
	(1.89e−01)	**(5.44e−12)**	(8.741e−01)	(4.39e−08)	(1.19e+00)	(3.49e−01)	(1.16e−05)
FN4	1.48e+01	3.44e−14	2.29e+01	6.14e−01	4.75e+01	1.76e+01	**0.00e+00**
	(5.77e+00)	(9.00e−14)	(5.66e+00)	(8.68e−01)	(1.04e+01)	(7.44e+00)	**(0.00e+00)**
FN5	1.08e+01	1.86e−01	5.92e+01	2.74e+00	2.93e+01	4.94e+00	**6.88e−05**
	(4.95e+00)	(1.69e−01)	(2.38e+01)	(1.39e−01)	(3.34e+01)	(2.22e+00)	**(1.40e−04)**
FN6	1.68e+03	**1.27e−04**	8.75e+02	**1.27e−04**	1.13e+03	1.62e+03	4.13e−01
	(3.26e+02)	**(3.63e−08)**	(1.18e+02)	**(4.53e−08)**	(3.08e+02)	(5.73e+02)	(9.66e−01)
FN7	1.13e−01	1.04e−11	1.89e+00	1.51e−07	2.29e−01	8.68e−09	**1.93e−35**
	(6.55e−02)	(4.09e−11)	(7.55e−01)	(4.56e−08)	(9.58e−01)	(3.49e−09)	**(9.53e−35)**
FN8	−3.38e+01	**−3.92e+01**	−3.53e+01	**−3.92e+01**	−3.01e+01	−3.66e+01	**−3.92e+01**
	(2.31e+00)	**(7.71e−14)**	(8.02e−01)	**(4.40e−08)**	(1.70e+00)	(2.45e+00)	**(5.68e−05)**
FN9	2.26e+00	1.49e−12	3.39e+01	1.55e−07	2.28e+00	3.82e−08	**0.00e+00**
	(1.59e+00)	(4.13e−12)	(1.47e+01)	(5.46e−08)	(6.07e+00)	(3.19e−08)	**(0.00e+00)**
FN10	6.86e+00	9.46e+00	6.69e+01	1.78e−02	2.01e+01	6.13e−10	**6.92e−14**
	(7.28e+00)	(5.39e+00)	(1.89e+01)	(1.19e−02)	(6.19e+00)	(2.28e−10)	**(4.89e−13)**

* The results are achieved from 50 independent runs of the algorithms, and the bold values in each case indicate a better performance.

where,

C_{max}: Maximum completion time of jobs on machines ($C_{max} = \sum_j C_j$)

C_j : Completion time of job j

p_{jk} : Processing time of job j on machine k

S_{ijk} : Sequence dependent setup time to process job j after job i on machine k

S_{0jk} : Setup time to process job j first on machine k

x_{ijk} : 1 if job j is processed directly after job i on machine k and 0 otherwise

x_{0jk} : 1 if job j is the first job to be processed on machine k and 0 otherwise

x_{j0k} : 1 if job j is the last job to be processed on machine k and 0 otherwise

M : A large positive number

The first two constraints guarantee that each job is processed once and that each machine is used at most once, respectively. The third constraint ensures that the assignment of jobs to machines is precise. The fourth constraint defines the completion time.

5. Experimental results and discussions

In this section, a number of benchmark tests are selected to evaluate the performance of the proposed IALOT algorithm. For the verification of the IALOT benchmark results, well-known meta-heuristic algorithms are employed: PSO and ABC algorithms as the popular swarm intelligence algorithms, DE algorithm as the best evolutionary based algorithm, SA algorithm as one of the physical based algorithms and TACO algorithm as the other meta-heuristic algorithm. Next, the analysis of the proposed IALOT algorithm is presented in this section. Lastly, the proposed IALOT algorithm is adapted to the parallel machine scheduling problem. This problem has been solved by IALOT algorithm and compared with the several recent meta-heuristic algorithms to show the performance of the proposed IALOT algorithm.

5.1. Evaluation criteria

Algorithms have been investigated according to a range of metrics for the benchmark test functions and compared in terms of their performances. The mathematical model of these metrics is examined below.

$$\gamma: X \subseteq \mathbb{R}^n \rightarrow \Gamma \tag{18}$$

where n denotes the dimension of solution space in search space. Let $x_0 \in \Gamma$ is the position, $\gamma(x_0) = \gamma_0$ is the optimization problem solution, and $\gamma(\hat{x}_0) = \hat{\gamma}_0$ denotes the solution found by the algorithm. Then used metrics defined below:

$$Optimality = 1 - \frac{||\gamma_0 - \hat{\gamma}_0||}{||\bar{\gamma} - \gamma||} \in [0, 1] \tag{19}$$

$$Accuracy = 1 - \frac{||x_0 - \hat{x}_0||}{||\bar{x} - x||} \in [0, 1] \tag{20}$$

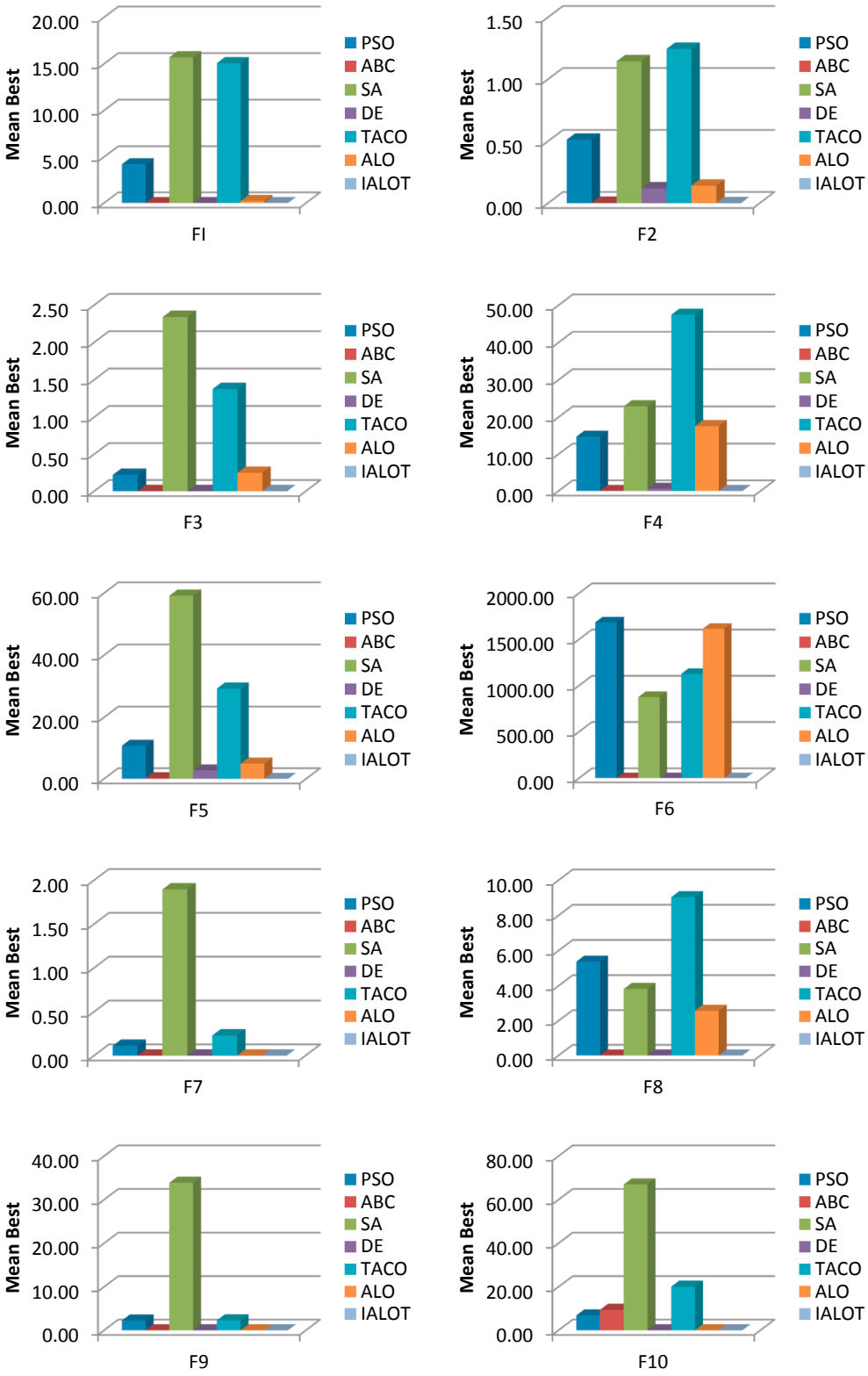

Fig. 4. Mean best results of meta-heuristic algorithms for benchmark problems.

$$Mean = \frac{1}{N}\sum_{i=1}^{N}\hat{\gamma}_0 \tag{21}$$

$$Standard\ \ Deviation\ \ (STD) = \sqrt{\frac{1}{N-1}\sum(\hat{\gamma}_0 - Mean)^2} \tag{22}$$

where $\overline{\gamma}$ and γ stand for lower and upper bounds of γ, $\bar{x}$ and x denote the lower and upper bounds of the search space (Luo & Li, 2016). *Optimality* metric shows the relative closeness of an objective function value found. *Accuracy* metric defines the relative closeness of a solution found. *Mean* metric stands for the average of the closeness of the solution found. *CPU time* and *Number of the Function Evaluations* (*NFE*) give some information regarding the runtime of the algorithm. *Standard Deviation (STD)* shows the diversity of the solutions found. Optimality

Table 5
Comparison results (*NFE & CPU time*) of the IALOT with the other algorithms.*

	NFE (CPU time)						
Function	PSO	ABC	SA	DE	TACO	ALO	IALOT
FN1	**23,824**	50,347	100,000	69,488	100,000	99,258	98,158
	(0.804 s)	(1.804 s)	(4.214 s)	(2.369 s)	(28.619 s)	(49.881 s)	(5.745 s)
FN2	**22,404**	51,004	100,000	100,000	100,000	95,216	74,826
	(0.763 s)	(1.853 s)	(4.275 s)	(3.446 s)	(28.904 s)	(48.518 s)	(4.435 s)
FN3	**18,928**	30,893	100,000	41,192	100,000	90,324	76,242
	(0.507 s)	(0.922 s)	(3.654 s)	(1.121 s)	(28.735 s)	(46.103 s)	(3.984 s)
FN4	**23,770**	51,001	100,000	100,000	100,000	95,324	77,482
	(0.807 s)	(1.832 s)	(4.237 s)	(3.446 s)	(29.086 s)	(48.447 s)	(4.564 s)
FN5	**20,678**	51,010	100,000	100,000	100,000	99,812	82,128
	(0.600 s)	(1.577 s)	(3.696 s)	(2.920 s)	(27.852 s)	(49.180 s)	(4.421 s)
FN6	**28,058**	51,006	100,000	85,238	100,000	98,496	94,900
	(0.863 s)	(1.658 s)	(3.883 s)	(2.922 s)	(28.826 s)	(48.897 s)	(5.228 s)
FN7	**16,162**	26,837	100,000	36,086	100,000	90,288	53,814
	(0.331 s)	(0.624 s)	(2.839 s)	(0.755 s)	(28.064 s)	(45.223 s)	(2.465 s)
FN8	**21,188**	44,658	100,000	44,556	72,678	90,250	75,756
	(0.577 s)	(1.305 s)	(3.490 s)	(1.232 s)	(20.589 s)	(45.690 s)	(4.016 s)
FN9	**20,954**	30,471	100,000	41,428	100,000	91,670	71,386
	(0.424 s)	(0.701 s)	(2.825 s)	(0.856 s)	(27.983 s)	(45.966 s)	(3.256 s)
FN10	**30,654**	51,001	100,000	100,000	96,774	95,200	79,666
	(0.634 s)	(1.185 s)	(2.842 s)	(2.104 s)	(26.741 s)	(47.429 s)	(3.658 s)

* The results are achieved from 50 independent runs of the algorithms, and the bold values in each case indicate a better performance.

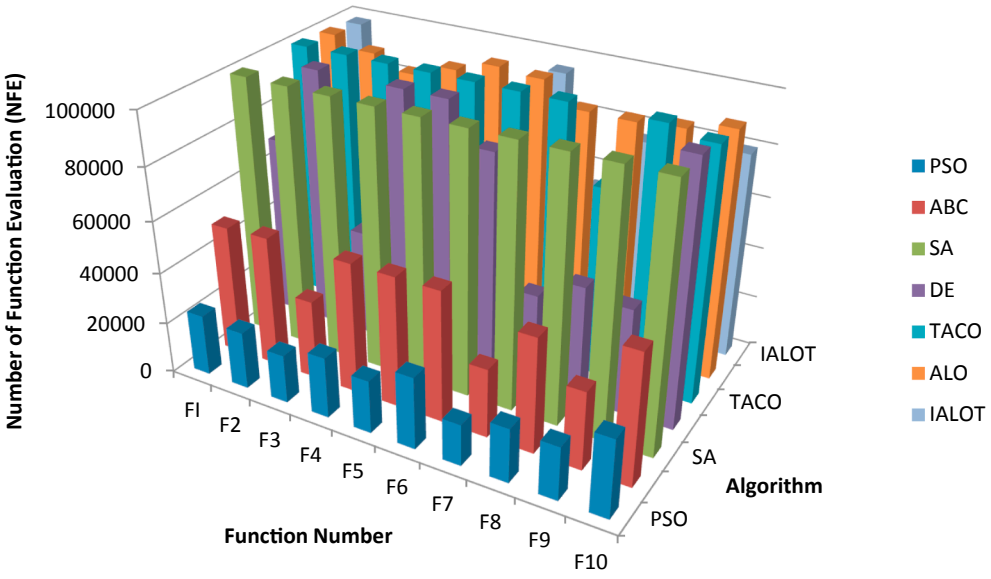

Fig. 5. *NFE* results of meta-heuristic algorithms for benchmark problems.

and Accuracy metrics are in range [01] and it is good to obtain values that are close to 1 for both of these metrics.

5.2. Results and discussion

5.2.1. Analysis of the proposed IALOT algorithm

In this subsection, we examine the convergence and the behavior of the proposed IALOT algorithm for the optimization benchmark problems. All benchmark functions comprise different characteristics. These benchmark functions used for the algorithm analysis are given in Table 2.

In Fig. 3, the IALOT algorithm analysis is shown for different metrics. All of the benchmark functions are solved by twenty antlions up to 1000 iterations. The first column in this figure presents the 3D images of the benchmark functions. During the optimization process of the benchmark functions, the position history of the ant and antlions are shown in the second and the third columns in Fig. 3.

The results show that the antlions are located in promising regions in the search spaces on all benchmark functions. Ants are located in a larger area than antlions in the search space because of the random walking model. The ants' random walks in the 4th column of Fig. 3 show that limits of the random walks are regulated by updating the sliding ratio in Eq. (8). Unlike ALO algorithm, the adaptive shrinking model of the random walking limits provides more exploitation and exploration thanks to the improvements given in Eqs. (11)–(14).

In the last two columns in Fig. 3, the average fitness curves of antlions and the fitness curves of best antlion (convergence curve) are shown for all test functions. The mean fitness curves of antlions present decreasing trends for all benchmark functions. As can be seen from these sub-figures, the fitness curves of the proposed IALOT algorithm stop at different iteration numbers due to the new stop criteria that is given in next sub-section. The convergence curves of the best antlion show the same behaviour as mean fitness curves. The both of the fitness curves show that IALOT algorithm reaches the solutions of the test

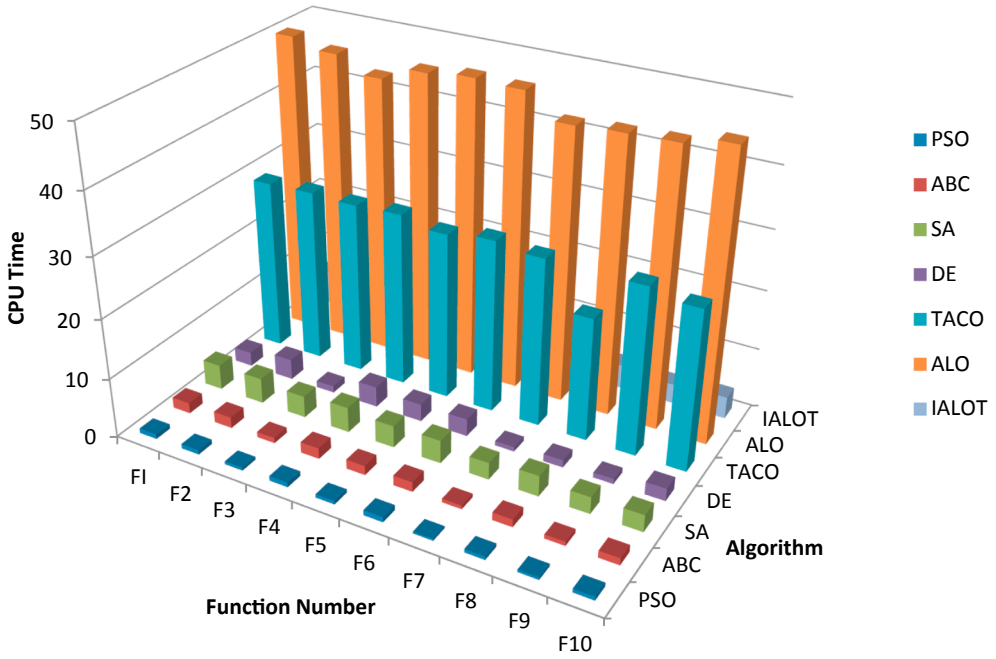

Fig. 6. *CPU Time* results of meta-heuristic algorithms for benchmark problems.

Table 6
Comparison results (*Optimality*) of the IALOT with the other algorithms.*

	Optimality						
Function	PSO	ABC	SA	DE	TACO	ALO	IALOT
FN1	0.811	**1.000**	0.296	**1.000**	0.324	0.990	**1.000**
FN2	0.922	**1.000**	0.827	0.982	0.812	0.979	**1.000**
FN3	0.998	**1.000**	0.975	**1.000**	0.986	0.997	**1.000**
FN4	0.817	**1.000**	0.715	0.992	0.411	0.781	**1.000**
FN5	0.998	**1.000**	0.990	**1.000**	0.995	0.999	**1.000**
FN6	0.065	**1.000**	0.478	**1.000**	0.327	0.036	**1.000**
FN7	0.998	**1.000**	0.964	**1.000**	0.996	**1.000**	**1.000**
FN8	0.967	**1.000**	0.977	**1.000**	0.945	0.984	**1.000**
FN9	0.992	**1.000**	0.887	**1.000**	0.992	**1.000**	**1.000**
FN10	**1.000**	**1.000**	0.999	**1.000**	**1.000**	**1.000**	**1.000**

* The results are achieved from 50 independent runs of the algorithms, and the bold values in each case indicate a better performance.

Table 7
Comparison results (*Accuracy*) of the IALOT with the other algorithms.*

	Accuracy						
Function	PSO	ABC	SA	DE	TACO	ALO	IALOT
FN1	0.989	**1.000**	0.916	**1.000**	0.933	**1.000**	**1.000**
FN2	0.973	0.999	0.963	0.986	0.959	0.973	**1.000**
FN3	0.989	**1.000**	0.955	**1.000**	0.966	0.992	**1.000**
FN4	0.932	**1.000**	0.907	0.998	0.888	0.900	**1.000**
FN5	0.797	0.985	0.840	0.901	0.809	0.858	**1.000**
FN6	0.671	**1.000**	0.705	**1.000**	0.777	0.390	**1.000**
FN7	0.992	**1.000**	0.969	**1.000**	0.995	**1.000**	**1.000**
FN8	0.867	**1.000**	0.907	**1.000**	0.824	0.897	**1.000**
FN9	0.990	**1.000**	0.967	**1.000**	0.994	**1.000**	**1.000**
FN10	0.961	0.955	0.861	0.998	0.925	**1.000**	**1.000**

* The results are achieved from 50 independent runs of the algorithms, and the bold values in each case indicate a better performance.

functions at the short iteration numbers.

5.2.2. Results of the algorithms for benchmark test functions

In this study, IALOT algorithms have been tested with 10D benchmark test functions and compared with other well-known meta-heuristic algorithms (PSO, DE, ABC, SA, TACO, ALO). For the benchmark tests, the population size is 100, and maximum iteration number is 1000. All meta-heuristic algorithms have been run 50 times. All codes of meta-heuristic algorithms have been run on PC with Intel(R) Core (TM) i7-6500U CPU@2.50 GHz/8.00 GB RAM. The parameters of the meta-heuristic algorithms that used in this work are given in Table 3. For the random walking size values less than 0.2*maximum iteration, the proposed IALOT algorithm shows less exploration and exploitation in search space (Kilic, Yuzgec, & Karakuzu, 2018). The tournament size was determined as 2 according to the trial and error method. Using higher tournament size did not change the optimization result; unlike it increased algorithm's running time.

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:

$$if\ |f_{best} - f_{worst}| < VTR\ \ then\ \ stop\ \ the\ \ algorithm \tag{23}$$

where f_{best} stands for the best fitness value, f_{worst} is the worst fitness value in the population, *VTR* denotes the value to reach and this value is 10^{-6} in this study. The comparison results are presented for four metrics, such as *mean best/standard deviation*, *number of function evaluation (NFE)/CPU time*, *optimality*, *accuracy,* to show the performance of the IALOT algorithm. The first metric results of 10D benchmark tests for IALOT and six well-known meta-heuristic algorithms are presented in Table 4.

According to the results of the algorithms in terms of the *mean best/std.dev.*, the proposed IALOT algorithm has the best performance for 80% of the benchmark functions except F3 and F6. Fig. 4 shows the mean values of the best fitness in 50 runs for all benchmark functions. As can be seen from this figure, IALOT algorithm's performance is better than those of the other algorithms. For only F8 function, the inverse of the mean best fitness values is shown by subtracting its global solution. *CPU time/NFE* metric results are given in Table 5. PSO algorithm is the best among all benchmark functions, but the PSO algorithm did not find the solution of any test problems according to the results in Table 4.

The reason of the success of the PSO algorithm in terms of *CPU time /NFE* metrics is the new stopping criterion (Eq. (23)) used in the algorithm tests. The results of the proposed IALOT algorithm are not the

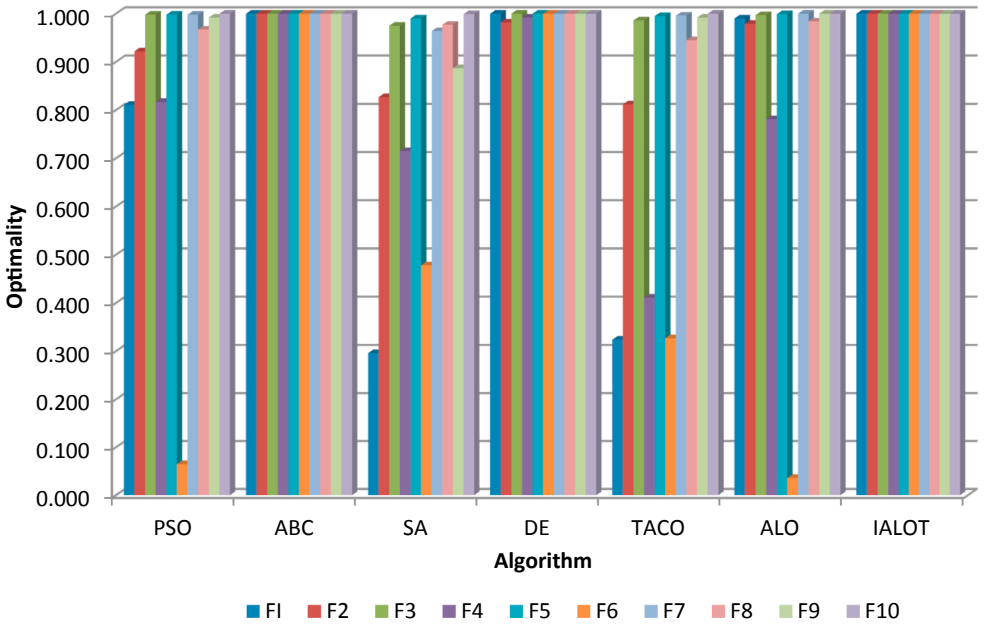

Fig. 7. *Optimality* results of meta-heuristic algorithms for benchmark problems.

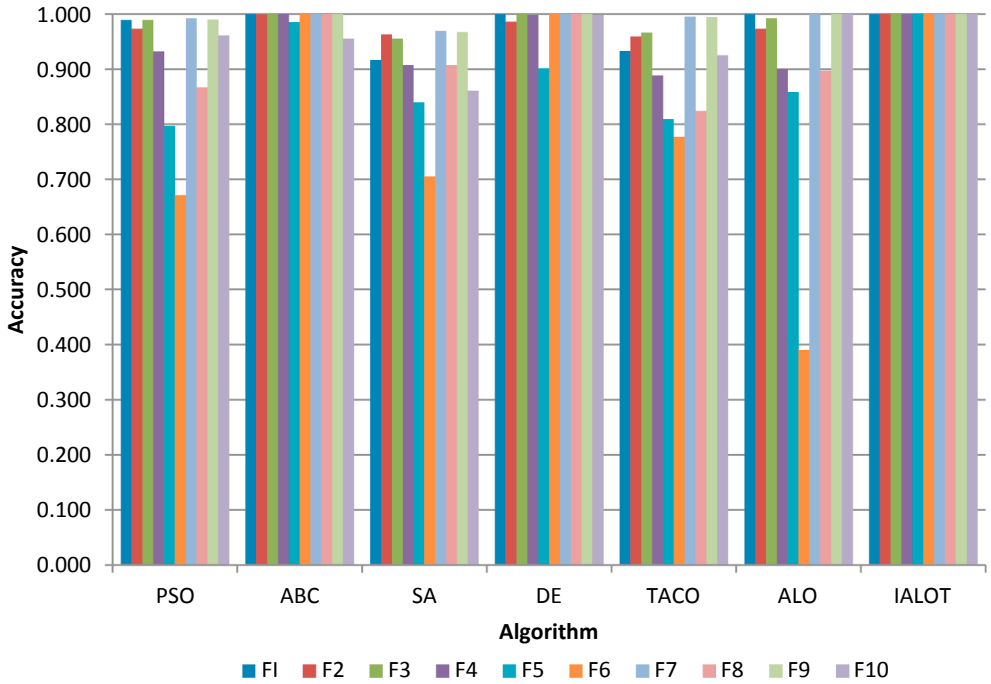

Fig. 8. *Accuracy* results of meta-heuristic algorithms for benchmark problems.

best, but they show that the proposed IALOT is faster 10–20 times than the original ALO algorithm. In Figs. 5 and 6, *NFE/CPU time* metric results of the meta-heuristic algorithms are presented for all benchmark functions. As can be seen from Fig. 6, the original ALO algorithm is the worst algorithm.

Table 6 summarizes *Optimality* metric results of all meta-heuristic algorithms. *Optimality* metric shows how close to the global solution. Both of IALOT and ABC algorithms prove the best value (1) for %100 of the benchmarks. In Table 7, *Accuracy* metric results for IALOT algorithm and other meta-heuristic algorithms are presented for all benchmarks. *Accuracy* metric indicates how close to the global solution points. According to these results given in Table 7, the best algorithm is IALOT for all benchmark functions.

Optimality and *Accuracy* metric results of the proposed IALOT algorithm and other meta-heuristic algorithms are shown in Figs. 7 and 8. When looking at these figures, the achievement of the IALOT algorithm is once again seen. As a result, the global fitness values are found at the global solution points with % 100 success by our improvements on the original ALO algorithm. The comparison results of all meta-heuristic algorithms are shown in Fig. 9 for each benchmark function. These graphics are given in logarithmic plots for a better understanding of the comparison results of the meta-heuristic algorithms. The logarithmic curves of mean cost value (fitness) show that the IALOT algorithm provides the fastest convergence on benchmark functions.

The reason for the fact that the meta-heuristic algorithms stop at different iterations is the new stop criteria given in Eq. (23). Thanks to

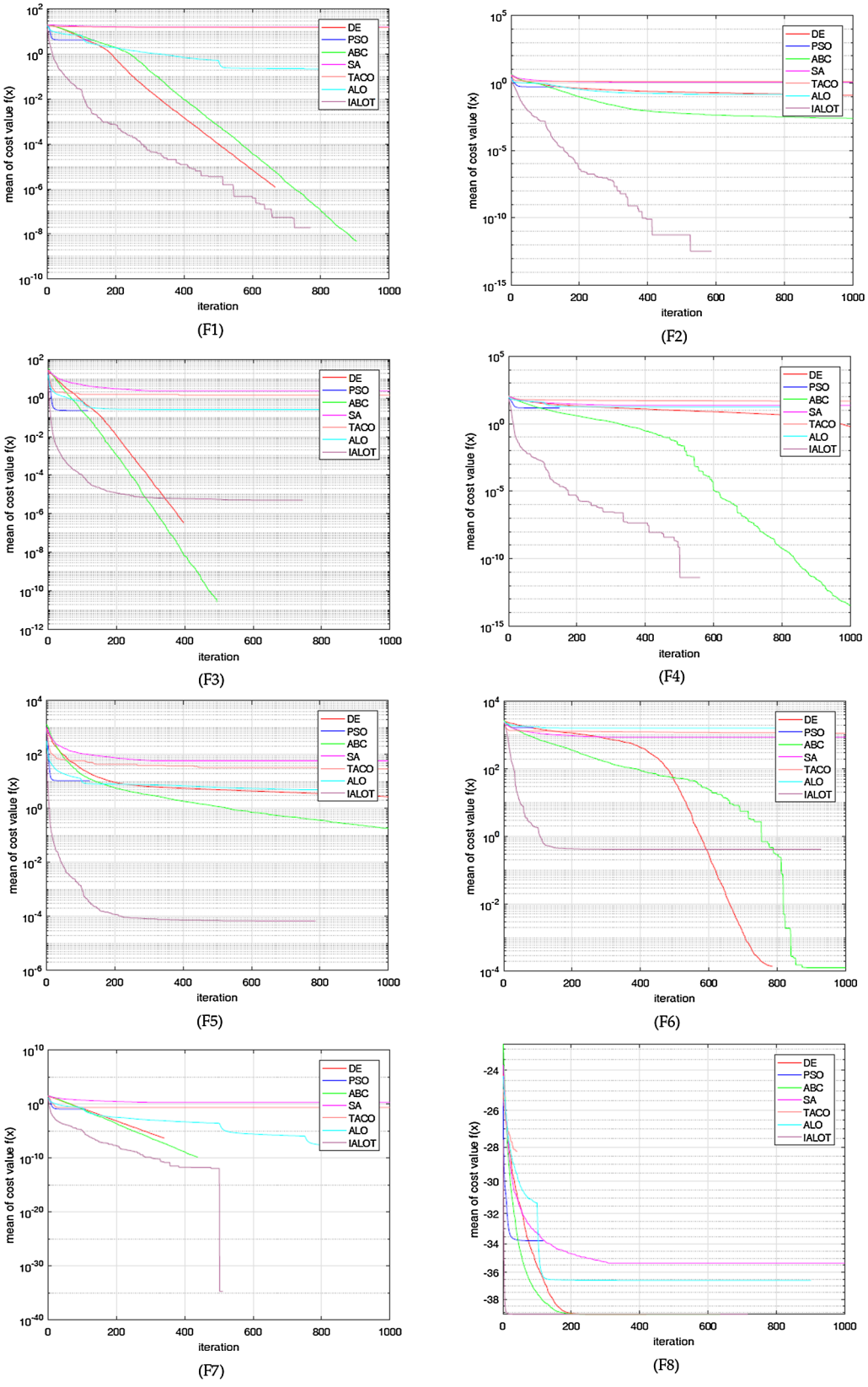

Fig. 9. Comparison results of meta-heuristic algorithms on benchmark problems.

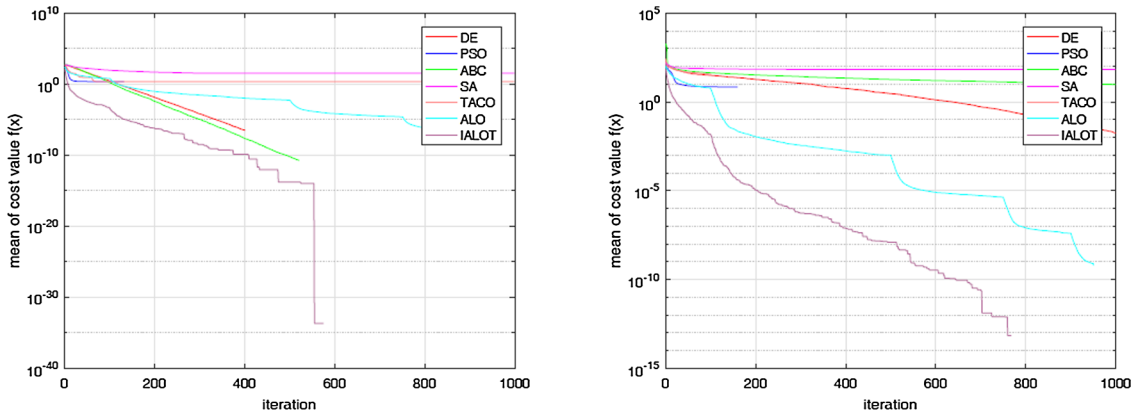

Fig. 9. (*continued*)

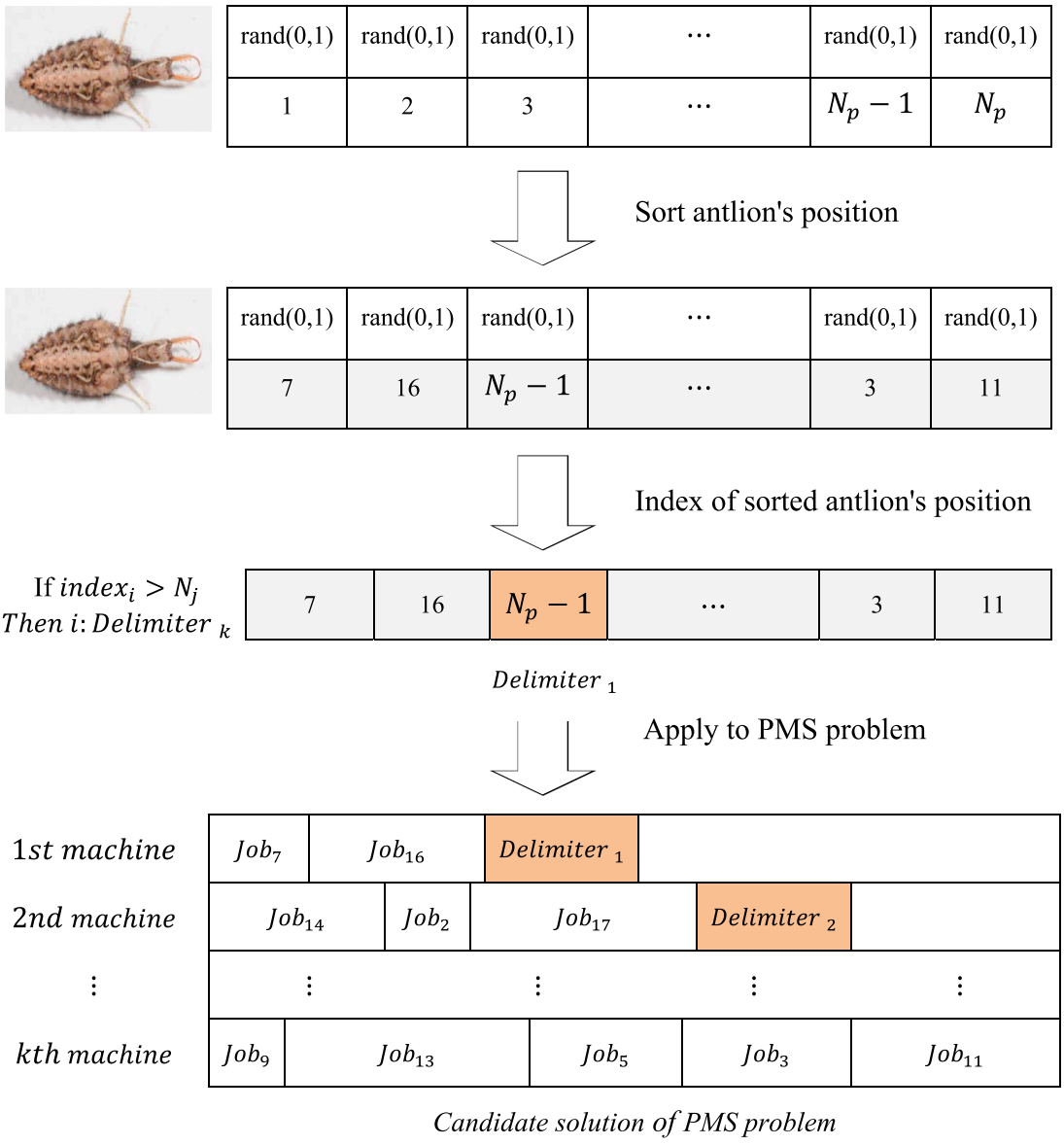

Fig. 10. Solving PMS problem with IALOT algorithm.

Table 8
Parameters of meta-heuristic algorithms for PMS tests.

Algorithm	Parameters	Algorithm	Parameters
GA	Crossover coefficient : 0.4 Mutation coefficient : 0.8 Selection pressure coefficient : 5	HSA	Number of new harmonies : 20 Harmony memory consideration rate: 0.9 Pitch adjustment rate : 0.1 Fret width damp ratio : 0.995
PSO	Inertia weight : 1.0 Inertia weight damping ratio : 0.99 Personal learning coefficient : 1.5 Global learning coefficient : 2.0	COA	Acceptance ratio : 0.35 Alpha : 0.3
FA	Light absorption coefficient : 1.0 Initial attraction coefficient: 2.0 Mutation coefficient : 0.2 Mutation coefficient damping R. : 0.98	GWO	Number of wolfs : 20
IWO	Variance reduction exponent : 2 Initial value of standard deviation : 1 Final value of standard deviation : 0.001 Minimum number of seeds : 0 Maximum number of seeds : 5	DA	Number of dragonflies : 20
ICA	Selection pressure : 1 Assimilation coefficient : 2 Revolution probability : 0.5 Revolution rate : 0.1 Colonies mean cost coefficient : 0.1	GOA	Number of grasshoppers : 20 cMax : 1 cMin : 0.00004
SFLA	Number of memeplexes : 5 Number of offsprings : 3 Maximum number of iterations :5 Step size : 2	MFO	Number of moth-flames : 20
BBO	Keep rate : 0.2 Alpha : 0.9 Mutation coefficient : 0.1	ALO	Number of antlions : 20
CMA-ES	Number of offsprings : (4 + round(3*log(nVar)))*10 nVar: number of variables	IALOT	Number of antlions : 20 random walk size = Max Iter/5

the improvements on the classic ALO algorithm in this study, IALOT algorithm provides the best convergence curves in six of benchmark functions.

5.2.3. Parallel Machine Scheduling (PMS) test results

In this study, the proposed IALOT algorithm is used to solve the Parallel Machine Scheduling (PMS) problem. PMS instance has been taken from the website www.yarpiz.com (Parallel Machine Scheduling using Simulated Annealing- Yarpiz, 2018). In this study, there are four parallel machines and the number of total jobs is 20. The processing time of jobs is a known matrix that can be shown as $p[20 \times 4]$. For each machine, the values of setup time to process job j after job i on machine have been kept on the $s[20 \times 20 \times 4]$ matrix. These matrices have been given in the Appendix section.

To solve PMS problem, IALOT algorithm has been adapted to combinatorial optimization. For the example used in this study, the number of problem dimension (N_p) has been defined as following:

$$N_p = N_j + N_m - 1 \tag{24}$$

where N_j denotes the number of jobs and N_m is the number of machines. Here, $N_m - 1$ variables are added into the N_j variables to know how many jobs can be scheduled in a suitable machine. Fig. 10 presents how the candidate solution of PMS problem derives from the position of antlion does. Firstly, IALOT algorithm generates randomly the positions of antlions in the range [0 1]. These positions are sorted and index vectors of the sorted positions are used as the candidate solutions of PMS problem. If any index value is bigger than number of jobs (N_j), then delimiter position is determined. The delimiter specifies switching to job scheduling for next parallel machine.

For each parallel machine, job lists are created by these index vectors with delimiter positions. To calculate the completion time of each machine, start, process, setup and final time values of jobs on machine are determined using processing time matrix, setup time matrix. The cost value of PMS problem is maximum value of completion time of machines. Pseudo code of how to solve PMS problem by IALOT algorithm is given below:

The pseudo code about solving PMS problem by IALOT Algorithm:
Input: number of jobs, number of machines, processing time matrix, setup time matrix, candidate solution produced by IALOT.
Output: jobs list, start times, processing times, final times, completion times, maximum completion time.
1) Delimiters position (candidate solution > number of jobs)
2) Determine start and end of machines job sequence
3) for every machine
Create job list from candidate solution produced by IALOT
end for
4) for i:number of machines
for j:job list of machine(i)
k = index of job list
if first job on machine
start time = 0
else
previous job = job list of machine (k − 1)
start time = final time(previous job) + setup time (previous job, j, i)
end if
process time = processing time(j, i)
final time = start time + process time
end for
if job list of machine(i) is not empty
completion time(i) = final time(job list of machine(i)(end))
end if
end for
5) maximum completion time = max(completion time of machines)

PMS problem has been solved by the proposed IALOT algorithm and its performance has been compared with the performances of the original ALO algorithm, Genetic algorithm (GA), Particle Swarm Optimization algorithm (PSO), Firefly algorithm (FA), Invasive Weed Optimization algorithm (IWO), Imperialist Competitive algorithm (ICA), Shuffled Frog Leaping algorithm (SFLA), Biogeography-Based Optimization algorithm (BBO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Harmony Search algorithm (HSA), Cultural Optimization

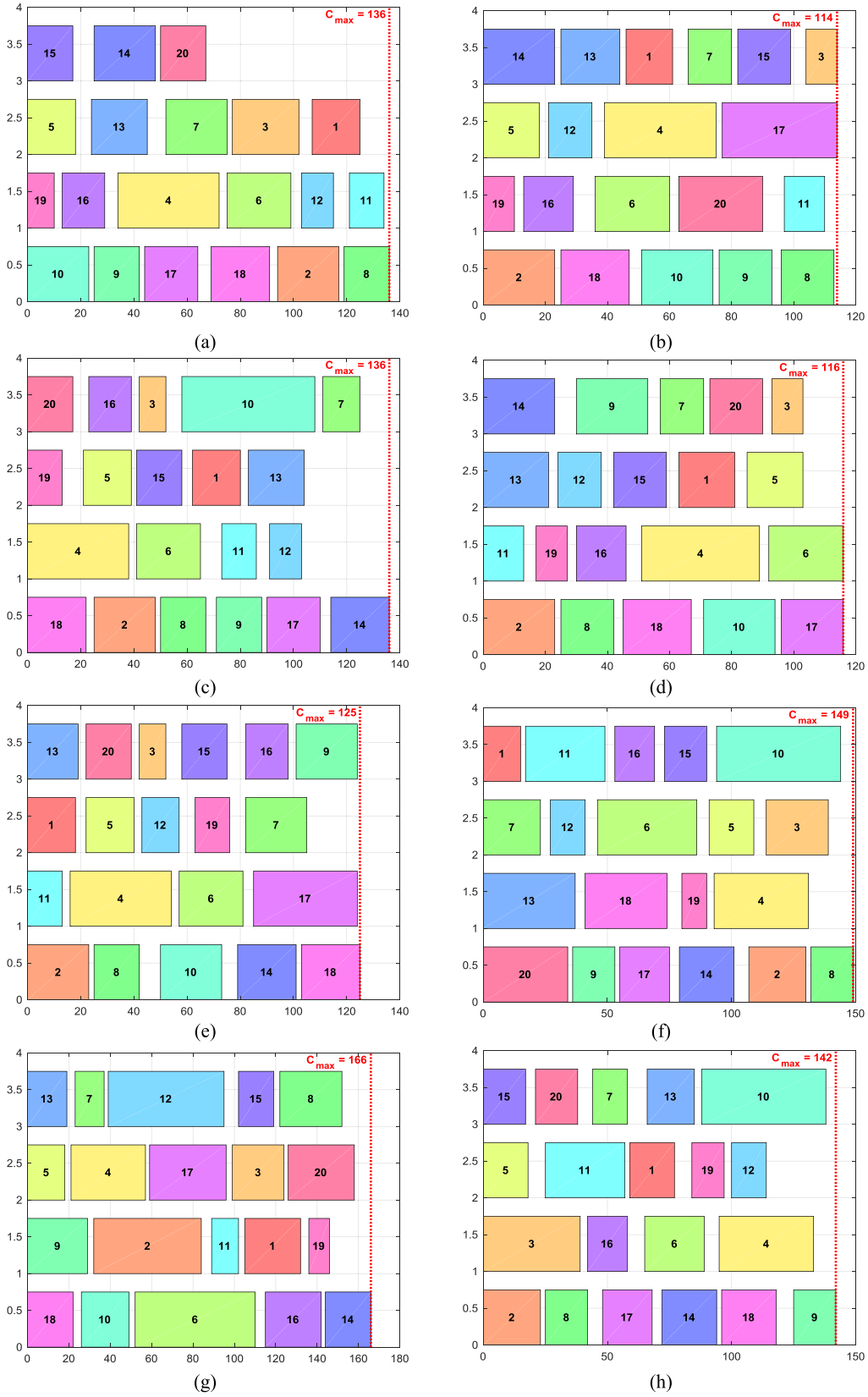

Fig. 11. Results obtained by meta-heuristic algorithms for Parallel Machine Problem. (a) GA, (b) PSO, (c) FA, (d) IWO, (e) ICA, (f) SFLA, (g) BBO, (h) CMA-ES. Results obtained by meta-heuristic algorithms for Parallel Machine Scheduling. (i) HSA, (j) COA, (k) GWO, (l) DA, (m) GOA, (n) MFO, (o) ALO, (p) IALOT.

algorithm (COA), Gray Wolf Optimization algorithm (GWO), Dragonfly Optimization algorithm (DA), Grasshopper Optimization algorithm (GOA) and Moth-Flame Optimization algorithm (MFO). All meta-heuristic algorithm codes for PMS problem have been run on PC with Intel(R) Core(TM) i5-3230 M CPU@2.60 GHz, RAM/8. In the initial populations of these algorithms, same individuals have been used. Each

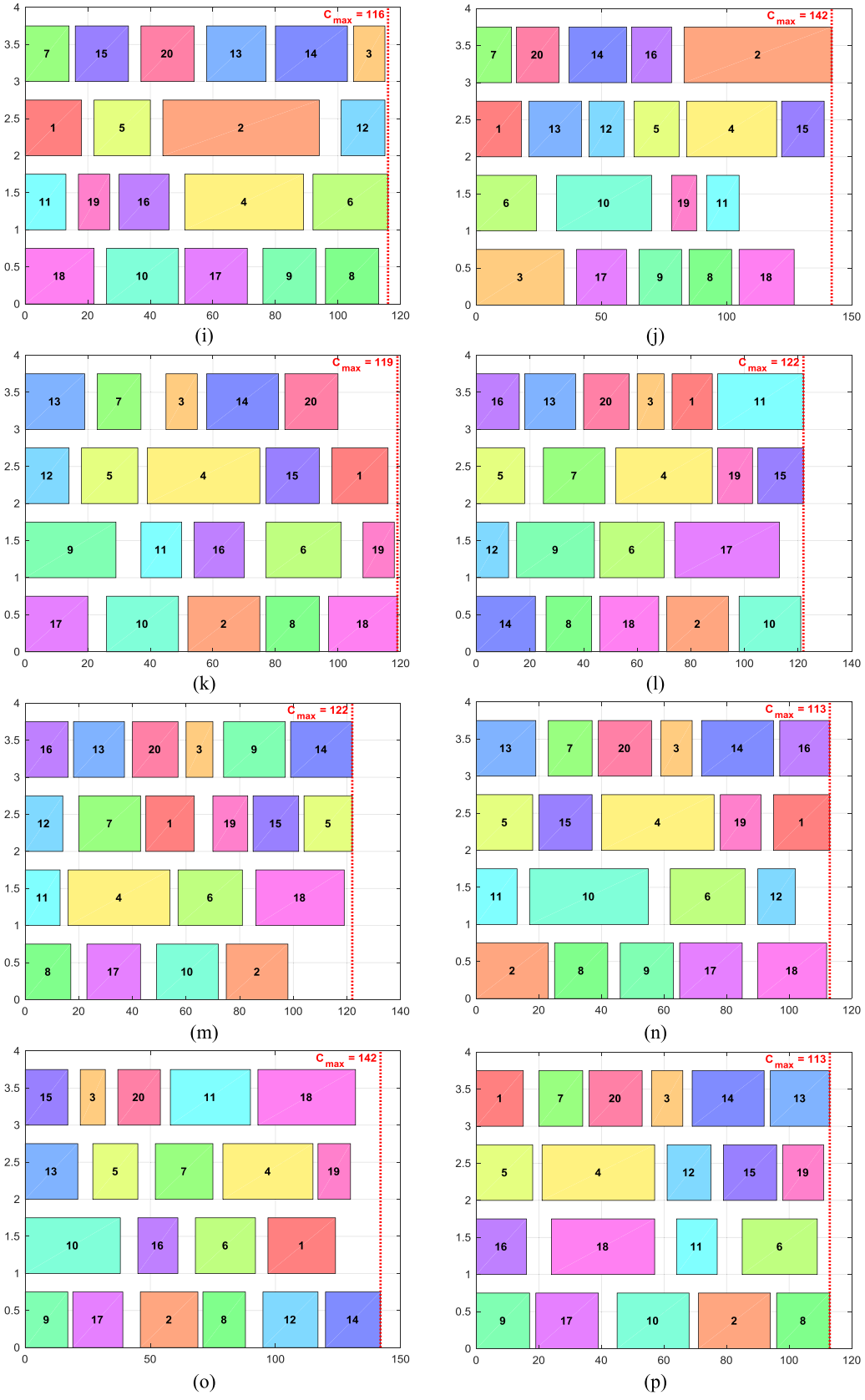

Fig. 11. (*continued*)

of the meta-heuristic algorithm has been run 10 times. Population size is selected as 20 for all algorithms, the maximum number of iterations is 1000. The parameters of meta-heuristic algorithms used for solving the PMS problem are summarized in Table 8. The source codes of the proposed IALOT algorithm for solving PMS problem are publicly available at https://github.com/uguryuzgec/PMS-with-IALOT.

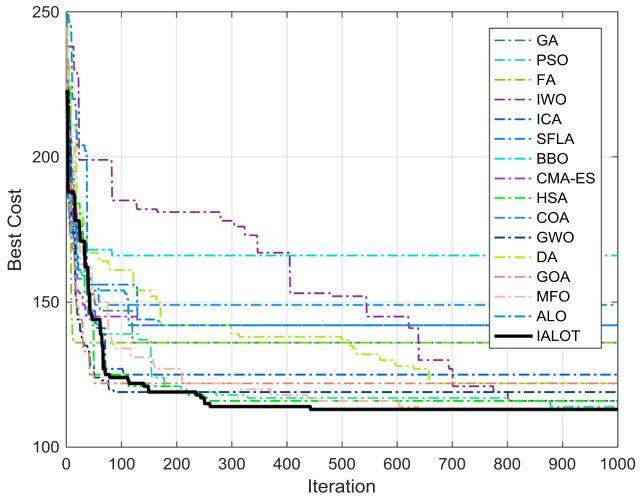

Fig. 12. Comparison result of IALOT and other meta-heuristic algorithms for Parallel Machine Scheduling Problem.

Table 9
The results with 10 runs of IALOT and other meta-heuristic algorithms for PMS.

	Mean cost	Best cost	Worst cost	Standard dev.
GA	121.4	115	130	5.232
PSO	121.3	116	134	5.143
FA	121.1	113	136	7.724
IWO	118.4	114	126	3.893
ICA	123.3	116	140	8.769
SFLA	136.8	123	162	12.691
BBO	167.1	150	198	14.617
CMA-ES	130.5	121	140	5.949
HSA	118.9	113	130	5.301
COA	137.4	130	150	5.621
GWO	127.2	119	137	6.391
DA	121.4	115	133	5.661
GOA	116.3	113	**119**	**1.636**
MFO	115.3	113	123	2.983
ALO	151.5	142	160	7.075
IALOT	**115.2**	**112**	123	3.011

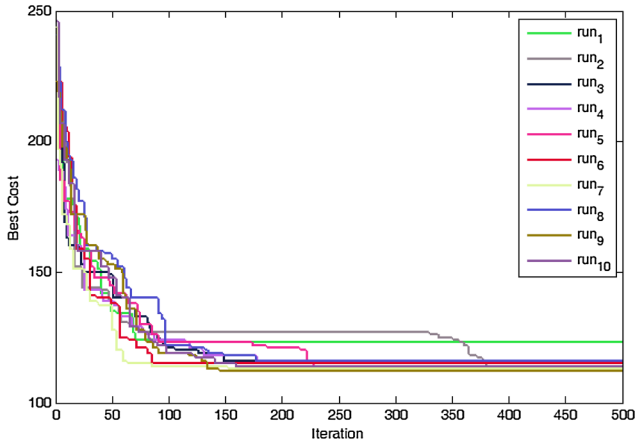

Fig. 13. Convergence curves of IALOT algorithm with 10 independent runs.

The PMS problem solution results obtained by running the IALOT and other well-known meta-heuristic algorithms are shown in Fig. 11. These figures include the final best solutions of a PMS problem with 4 parallel machines and 20 jobs. According to the PMS results obtained by meta-heuristic algorithms, the solution of each algorithm is different than the other. The results show that the proposed IALOT algorithm and MFO algorithm have the best cost value as $C_{max} = 113$. Fig. 12 shows the convergence curves obtained by the proposed IALOT algorithm and meta-heuristic algorithms during the solution of PMS problem. The IALOT algorithm's curve is emphasized in bold line. According to this figure, the best convergence curve belongs to the proposed IALOT algorithm and BBO algorithm has the worst performance among algorithms.

Table 9 gives the comparison results with 10 independent runs of IALOT and the other meta-heuristic algorithms for PMS problem. This table comprises the mean cost, best cost, worst cost and standard deviation values of the results obtained by all algorithms over 10 independent runs. The best values are shown as bold in this table. In terms of the mean cost and best cost indicators, the proposed IALOT algorithm provides the best performance (*mean cost = 115.2 and best cost = 112*) among the other meta-heuristic algorithms. GOA algorithm yields the best results for worst cost and standard deviation indicators.

Fig. 13 presents the convergence curves obtained by IALOT algorithm for 10 runs. The results show that the IALOT algorithm promotes exploration and exploitation of the search space. Fig. 14 shows the box plots concerning the PMS results including the performances of IALOT algorithm and other meta-heuristic algorithms for 10 independent runs. As it can be seen in this figure, BBO algorithm has the worst performance and IALOT algorithm has the best performance among all meta-heuristic algorithms for PMS problem.

The result of IALOT algorithm is also better than that of classic ALO algorithm. All of the results show the superiority of the IALOT algorithm depending on to the improvements suggested on the original ALO algorithm.

6. Conclusion

Antlion optimization algorithm is one of the recent meta-heuristic algorithms, but it has some drawbacks, such as long runtime during the optimization process. In this study, antlion optimization algorithm is developed based on some improvements and innovations on ALO algorithm, and a new improved ALO algorithm via tournament selection (IALOT) is proposed. The improvements on original ALO algorithm are basically about the random walking mechanism and selection methods. In the selection of antlion used for the random walk mechanism, the tournament selection method is employed instead of the roulette wheel method used in 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 movements provide that ants walk more effectively around the selected antlion in the search space.

Firstly, the convergence and behavior of the proposed IALOT algorithm were examined for the optimization benchmark problems. There were not any studies on time analysis of original ALO algorithm. To show the performance of IALOT algorithm, a comparative study was performed on 10 well known benchmark functions for different metrics. Four metric results were presented such as optimality, accuracy, mean best/standard deviation and CPU time/number of function evaluations (NFE). The results show that the proposed IALOT has the best performance in terms of optimality, accuracy and mean of best costs. Thanks to the improvements made in the original ALO algorithm, the run time of the proposed IALOT algorithm was decreased. According to the CPU-time/NFE metric results, the run time of the IALOT algorithm is lower than that of the original ALO algorithm.

Secondly, the paper considered solving proposed the Parallel Machine Scheduling (PMS) problem using the proposed IALOT algorithm. An example of PMS problem has been used which consists of 4 parallel machines and 20 jobs. In PMS tests, 15 well-known and recent meta-heuristic algorithms (GA, PSO, FA, IWO, ICA, SFLA, BBO, CMA-ES, HSA, COA, GWO, DA, GOA, MFO, and ALO) are used. PMS comparative results demonstrate that the proposed IALOT algorithm has the best performance according to the mean cost and the best cost, except

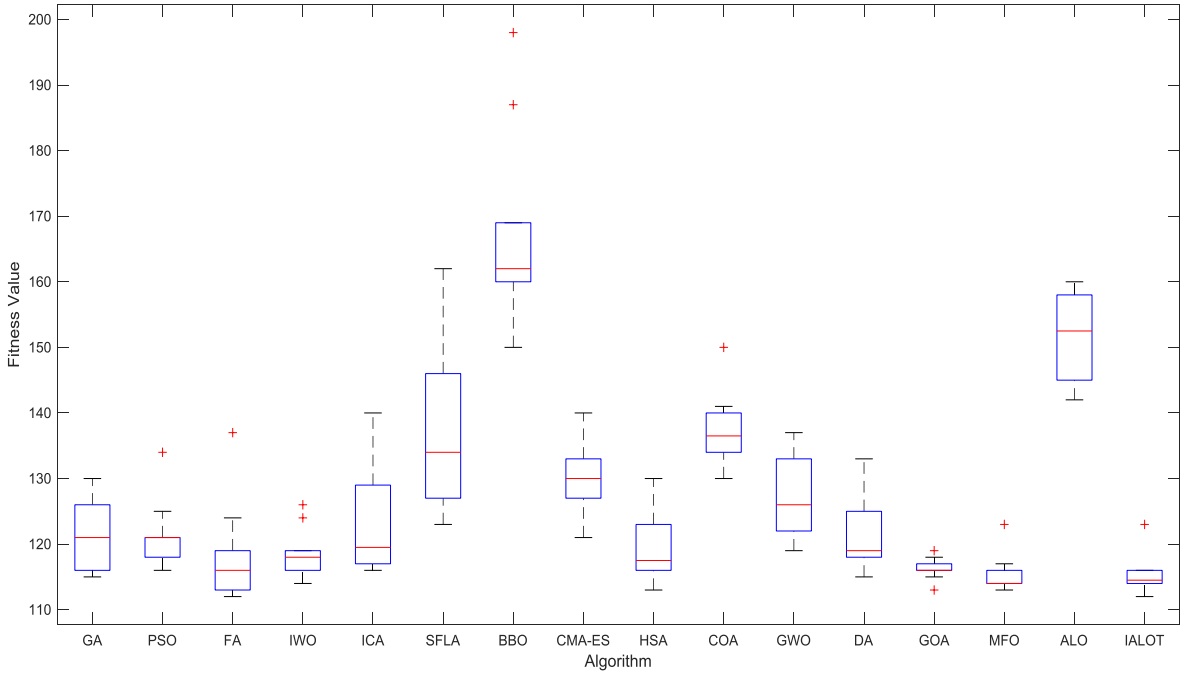

Fig. 14. Boxplots of fitness values of IALOT algorithm and other meta-heuristic algorithms.

the standard deviation and the worst cost. All results show the superiority of the IALOT algorithm depending on the improvements proposed on the original ALO algorithm.

In the future studies, to improve the performance of the ALO algorithm effective mechanisms (opposition based learning, chaotic based random number generator) can be investigated. IALOT algorithm can be implemented in real optimization problems in different fields, such as hub location allocation problem, optimal robot path planning, capacitated vehicle routing problem, minimum spanning tree etc.

Appendix A

PMS matrices

The processing time of jobs is a known matrix that can be shown as $p[20 \times 4]$. For each $k - th$ machine, the values of setup time to process job j after job i on machine have been kept on the $s_k[20 \times 20]$ matrix (Parallel Machine Scheduling using Simulated Annealing- Yarpiz, 2018). These matrices are given below:

$$p = \begin{bmatrix} 48 & 27 & 18 & 15 \\ 23 & 52 & 50 & 59 \\ 35 & 39 & 25 & 10 \\ 45 & 38 & 36 & 49 \\ 55 & 56 & 18 & 51 \\ 58 & 24 & 40 & 54 \\ 37 & 48 & 23 & 14 \\ 17 & 48 & 43 & 30 \\ 17 & 29 & 45 & 23 \\ 23 & 38 & 48 & 50 \\ 52 & 13 & 32 & 32 \\ 22 & 12 & 14 & 56 \\ 51 & 37 & 21 & 19 \\ 22 & 49 & 56 & 23 \\ 57 & 57 & 17 & 17 \\ 27 & 16 & 52 & 16 \\ 20 & 39 & 37 & 54 \\ 22 & 33 & 60 & 39 \\ 41 & 10 & 13 & 38 \\ 34 & 27 & 32 & 17 \end{bmatrix}$$

$$s_1 = \begin{bmatrix}
4 & 7 & 5 & 7 & 7 & 5 & 2 & 7 & 5 & 3 & 8 & 6 & 6 & 6 & 7 & 2 & 6 & 2 & 8 & 6 \\
3 & 5 & 8 & 5 & 6 & 6 & 5 & 2 & 7 & 4 & 2 & 2 & 5 & 2 & 4 & 7 & 5 & 2 & 3 & 4 \\
6 & 8 & 6 & 8 & 3 & 2 & 7 & 8 & 4 & 2 & 3 & 2 & 4 & 7 & 3 & 4 & 5 & 3 & 3 & 4 \\
3 & 4 & 3 & 6 & 6 & 6 & 8 & 8 & 5 & 5 & 2 & 7 & 2 & 2 & 2 & 6 & 6 & 3 & 4 & 5 \\
2 & 7 & 3 & 6 & 2 & 4 & 3 & 8 & 2 & 4 & 5 & 8 & 7 & 2 & 7 & 8 & 2 & 4 & 2 & 4 \\
7 & 4 & 4 & 7 & 6 & 2 & 3 & 8 & 3 & 3 & 2 & 5 & 4 & 6 & 3 & 5 & 4 & 4 & 6 & 4 \\
3 & 7 & 7 & 8 & 6 & 5 & 5 & 7 & 6 & 3 & 8 & 2 & 6 & 4 & 4 & 6 & 7 & 3 & 4 & 5 \\
5 & 7 & 7 & 8 & 7 & 3 & 6 & 5 & 4 & 8 & 3 & 7 & 7 & 6 & 5 & 7 & 6 & 3 & 8 & 7 \\
6 & 4 & 7 & 2 & 8 & 2 & 4 & 3 & 8 & 6 & 2 & 4 & 2 & 7 & 3 & 5 & 2 & 8 & 4 & 4 \\
4 & 3 & 4 & 8 & 8 & 3 & 3 & 4 & 2 & 5 & 4 & 4 & 2 & 6 & 6 & 6 & 2 & 6 & 6 & 5 \\
7 & 7 & 5 & 6 & 7 & 3 & 8 & 2 & 8 & 8 & 5 & 7 & 5 & 7 & 5 & 2 & 2 & 5 & 3 & 2 \\
4 & 8 & 2 & 8 & 6 & 3 & 2 & 2 & 5 & 2 & 2 & 2 & 5 & 3 & 3 & 8 & 2 & 3 & 4 & 2 \\
6 & 4 & 2 & 5 & 8 & 2 & 2 & 8 & 6 & 7 & 8 & 2 & 8 & 7 & 7 & 3 & 4 & 3 & 3 & 4 \\
6 & 6 & 2 & 5 & 6 & 6 & 2 & 4 & 8 & 7 & 4 & 6 & 7 & 8 & 2 & 3 & 6 & 2 & 7 & 4 \\
5 & 5 & 6 & 7 & 2 & 3 & 3 & 4 & 4 & 5 & 4 & 6 & 7 & 8 & 4 & 7 & 7 & 8 & 8 & 6 \\
2 & 7 & 5 & 3 & 2 & 5 & 6 & 4 & 4 & 3 & 2 & 5 & 2 & 2 & 3 & 5 & 5 & 6 & 4 & 8 \\
4 & 7 & 3 & 5 & 8 & 6 & 6 & 5 & 5 & 6 & 4 & 7 & 2 & 4 & 5 & 7 & 2 & 5 & 6 & 8 \\
4 & 3 & 5 & 8 & 5 & 5 & 2 & 6 & 7 & 4 & 2 & 6 & 2 & 4 & 2 & 4 & 6 & 4 & 4 & 5 \\
3 & 8 & 3 & 6 & 7 & 5 & 8 & 2 & 7 & 2 & 5 & 7 & 7 & 6 & 4 & 3 & 2 & 3 & 5 & 3 \\
3 & 8 & 2 & 7 & 3 & 5 & 7 & 7 & 2 & 3 & 7 & 4 & 8 & 6 & 2 & 2 & 2 & 6 & 7 & 7
\end{bmatrix}$$

$$s_2 = \begin{bmatrix}
7 & 7 & 3 & 4 & 4 & 3 & 3 & 6 & 6 & 3 & 5 & 4 & 3 & 5 & 2 & 2 & 6 & 5 & 6 & 3 \\
7 & 7 & 7 & 6 & 3 & 3 & 2 & 4 & 7 & 2 & 5 & 7 & 3 & 5 & 4 & 4 & 5 & 8 & 4 & 5 \\
7 & 2 & 2 & 8 & 2 & 5 & 3 & 7 & 2 & 2 & 8 & 5 & 6 & 8 & 3 & 3 & 4 & 7 & 8 & 8 \\
2 & 5 & 7 & 3 & 6 & 3 & 2 & 6 & 7 & 5 & 7 & 8 & 6 & 4 & 3 & 7 & 2 & 6 & 7 & 7 \\
6 & 4 & 6 & 6 & 3 & 7 & 2 & 5 & 8 & 3 & 5 & 5 & 6 & 5 & 4 & 7 & 5 & 2 & 5 & 8 \\
5 & 5 & 7 & 6 & 2 & 8 & 6 & 6 & 7 & 8 & 8 & 4 & 6 & 8 & 3 & 8 & 4 & 5 & 7 & 3 \\
3 & 4 & 6 & 4 & 7 & 2 & 8 & 5 & 2 & 2 & 2 & 6 & 2 & 2 & 4 & 6 & 7 & 6 & 4 & 6 \\
2 & 4 & 4 & 2 & 4 & 5 & 4 & 2 & 4 & 2 & 4 & 4 & 4 & 8 & 2 & 2 & 7 & 5 & 8 & 6 \\
7 & 3 & 4 & 2 & 6 & 2 & 4 & 7 & 6 & 5 & 8 & 7 & 5 & 3 & 8 & 8 & 6 & 4 & 8 & 2 \\
3 & 3 & 7 & 4 & 4 & 7 & 8 & 8 & 7 & 7 & 8 & 4 & 3 & 6 & 2 & 7 & 2 & 8 & 8 & 4 \\
3 & 2 & 4 & 3 & 6 & 8 & 8 & 4 & 3 & 4 & 6 & 5 & 7 & 6 & 8 & 4 & 2 & 7 & 4 & 3 \\
6 & 8 & 7 & 7 & 2 & 2 & 6 & 8 & 3 & 3 & 6 & 6 & 7 & 6 & 4 & 5 & 5 & 7 & 5 & 7 \\
8 & 6 & 7 & 4 & 8 & 8 & 8 & 4 & 6 & 4 & 4 & 8 & 3 & 4 & 2 & 8 & 4 & 4 & 3 & 3 \\
5 & 8 & 7 & 7 & 7 & 2 & 7 & 8 & 5 & 3 & 8 & 4 & 7 & 6 & 4 & 7 & 8 & 6 & 7 & 8 \\
6 & 3 & 5 & 7 & 7 & 6 & 4 & 5 & 6 & 5 & 2 & 7 & 2 & 7 & 7 & 7 & 8 & 8 & 8 & 7 \\
3 & 8 & 6 & 5 & 7 & 7 & 6 & 4 & 3 & 8 & 7 & 7 & 7 & 2 & 7 & 5 & 4 & 8 & 8 & 4 \\
8 & 7 & 8 & 3 & 4 & 5 & 3 & 3 & 3 & 6 & 6 & 8 & 2 & 2 & 5 & 5 & 7 & 6 & 5 & 5 \\
5 & 6 & 5 & 8 & 6 & 8 & 4 & 2 & 7 & 2 & 7 & 2 & 6 & 8 & 6 & 5 & 8 & 3 & 6 & 6 \\
6 & 5 & 2 & 3 & 6 & 8 & 6 & 4 & 7 & 4 & 4 & 4 & 4 & 6 & 8 & 3 & 6 & 6 & 3 & 7 \\
2 & 3 & 8 & 8 & 5 & 6 & 5 & 7 & 8 & 2 & 7 & 6 & 7 & 3 & 2 & 7 & 8 & 2 & 8 & 6
\end{bmatrix}$$

$$s_3 = \begin{bmatrix}
6 & 5 & 8 & 5 & 4 & 6 & 3 & 8 & 2 & 3 & 6 & 5 & 3 & 6 & 7 & 2 & 6 & 5 & 7 & 8 \\
4 & 6 & 5 & 6 & 5 & 5 & 5 & 6 & 3 & 2 & 6 & 7 & 2 & 5 & 4 & 6 & 6 & 7 & 6 & 5 \\
5 & 8 & 5 & 7 & 4 & 3 & 2 & 5 & 2 & 6 & 5 & 3 & 4 & 6 & 6 & 2 & 3 & 8 & 8 & 2 \\
6 & 7 & 4 & 5 & 7 & 6 & 7 & 7 & 5 & 8 & 3 & 4 & 6 & 3 & 2 & 6 & 2 & 7 & 2 & 2 \\
8 & 4 & 5 & 3 & 7 & 2 & 7 & 5 & 3 & 8 & 7 & 3 & 6 & 2 & 2 & 7 & 3 & 4 & 6 & 7 \\
7 & 7 & 5 & 5 & 6 & 8 & 5 & 4 & 3 & 3 & 4 & 5 & 5 & 8 & 3 & 8 & 5 & 3 & 5 & 5 \\
2 & 2 & 2 & 4 & 6 & 6 & 8 & 6 & 4 & 5 & 4 & 4 & 5 & 3 & 3 & 5 & 8 & 7 & 7 & 4 \\
6 & 2 & 8 & 8 & 8 & 2 & 5 & 4 & 2 & 4 & 8 & 5 & 4 & 8 & 6 & 5 & 6 & 2 & 3 & 7 \\
5 & 2 & 2 & 6 & 7 & 2 & 3 & 3 & 5 & 5 & 7 & 2 & 5 & 8 & 8 & 2 & 7 & 2 & 5 & 4 \\
5 & 3 & 5 & 6 & 6 & 3 & 2 & 6 & 6 & 3 & 4 & 5 & 7 & 4 & 3 & 5 & 3 & 3 & 4 & 5 \\
2 & 4 & 7 & 7 & 2 & 2 & 5 & 8 & 3 & 2 & 4 & 3 & 7 & 2 & 3 & 6 & 6 & 5 & 7 & 6 \\
7 & 4 & 4 & 4 & 4 & 5 & 6 & 4 & 7 & 5 & 6 & 3 & 6 & 6 & 4 & 3 & 7 & 8 & 6 & 8 \\
4 & 2 & 6 & 5 & 6 & 7 & 7 & 2 & 2 & 3 & 8 & 3 & 7 & 7 & 8 & 7 & 4 & 6 & 3 & 4 \\
3 & 5 & 7 & 5 & 5 & 6 & 2 & 5 & 4 & 2 & 8 & 3 & 6 & 8 & 4 & 8 & 8 & 4 & 4 & 6 \\
4 & 2 & 8 & 3 & 2 & 5 & 6 & 4 & 2 & 8 & 6 & 8 & 2 & 2 & 3 & 7 & 2 & 4 & 2 & 8 \\
4 & 3 & 8 & 5 & 3 & 8 & 5 & 4 & 3 & 5 & 4 & 8 & 5 & 5 & 3 & 5 & 4 & 7 & 6 & 2 \\
5 & 6 & 3 & 6 & 7 & 2 & 3 & 7 & 2 & 8 & 7 & 7 & 4 & 6 & 4 & 3 & 5 & 8 & 5 & 6 \\
5 & 8 & 3 & 4 & 2 & 8 & 8 & 4 & 3 & 7 & 5 & 7 & 2 & 6 & 4 & 7 & 2 & 6 & 3 & 4 \\
4 & 8 & 8 & 7 & 8 & 2 & 6 & 4 & 2 & 2 & 8 & 3 & 3 & 7 & 2 & 3 & 7 & 3 & 3 & 4 \\
4 & 5 & 6 & 7 & 2 & 5 & 5 & 4 & 3 & 6 & 2 & 4 & 3 & 6 & 5 & 8 & 5 & 2 & 5 & 3
\end{bmatrix}$$

$$s_4 = \begin{bmatrix}
7 & 7 & 8 & 3 & 8 & 2 & 5 & 2 & 3 & 8 & 2 & 5 & 7 & 7 & 3 & 4 & 7 & 6 & 8 & 7 \\
8 & 5 & 2 & 3 & 6 & 7 & 6 & 4 & 7 & 6 & 4 & 8 & 5 & 8 & 8 & 4 & 7 & 3 & 5 & 6 \\
3 & 3 & 2 & 4 & 4 & 4 & 8 & 8 & 4 & 6 & 7 & 7 & 4 & 3 & 6 & 8 & 4 & 5 & 8 & 5 \\
7 & 5 & 4 & 8 & 7 & 7 & 3 & 5 & 4 & 7 & 3 & 8 & 7 & 2 & 8 & 6 & 5 & 7 & 7 & 3 \\
3 & 5 & 6 & 5 & 8 & 5 & 7 & 4 & 3 & 2 & 7 & 3 & 5 & 3 & 5 & 8 & 8 & 3 & 5 & 8 \\
8 & 8 & 5 & 4 & 5 & 5 & 6 & 3 & 7 & 8 & 6 & 5 & 8 & 4 & 8 & 3 & 6 & 4 & 6 & 5 \\
7 & 7 & 8 & 3 & 5 & 2 & 5 & 5 & 6 & 4 & 7 & 2 & 8 & 4 & 2 & 7 & 7 & 5 & 8 & 2 \\
4 & 8 & 5 & 8 & 4 & 2 & 8 & 8 & 7 & 2 & 7 & 7 & 4 & 8 & 6 & 6 & 3 & 4 & 3 & 6 \\
7 & 6 & 5 & 4 & 2 & 2 & 4 & 2 & 7 & 7 & 4 & 6 & 5 & 2 & 7 & 3 & 6 & 7 & 4 & 5 \\
5 & 4 & 5 & 7 & 3 & 6 & 3 & 5 & 2 & 3 & 4 & 8 & 4 & 6 & 3 & 5 & 6 & 8 & 8 & 2 \\
7 & 8 & 6 & 6 & 2 & 3 & 6 & 7 & 8 & 3 & 5 & 8 & 6 & 3 & 8 & 4 & 8 & 3 & 4 & 8 \\
4 & 5 & 2 & 4 & 5 & 7 & 6 & 2 & 5 & 6 & 4 & 8 & 7 & 7 & 7 & 6 & 2 & 3 & 6 & 4 \\
2 & 3 & 7 & 8 & 2 & 8 & 4 & 6 & 7 & 3 & 7 & 4 & 7 & 3 & 7 & 7 & 5 & 6 & 8 & 3 \\
6 & 4 & 2 & 7 & 8 & 8 & 7 & 8 & 7 & 4 & 7 & 2 & 2 & 5 & 6 & 2 & 5 & 4 & 8 & 2 \\
8 & 6 & 5 & 5 & 6 & 5 & 8 & 3 & 7 & 4 & 5 & 5 & 7 & 8 & 7 & 7 & 2 & 8 & 6 & 4 \\
3 & 5 & 3 & 7 & 2 & 3 & 8 & 2 & 3 & 4 & 3 & 3 & 2 & 4 & 4 & 7 & 8 & 8 & 2 & 3 \\
5 & 7 & 4 & 8 & 2 & 3 & 2 & 6 & 5 & 4 & 6 & 3 & 4 & 2 & 3 & 4 & 8 & 6 & 2 & 6 \\
7 & 8 & 6 & 5 & 3 & 5 & 3 & 8 & 6 & 6 & 3 & 4 & 7 & 3 & 4 & 5 & 5 & 8 & 6 & 2 \\
2 & 8 & 3 & 4 & 5 & 7 & 2 & 6 & 8 & 3 & 5 & 2 & 7 & 4 & 6 & 6 & 7 & 4 & 5 & 3 \\
8 & 5 & 3 & 6 & 2 & 4 & 6 & 8 & 7 & 3 & 4 & 7 & 4 & 4 & 7 & 6 & 3 & 6 & 8 & 3
\end{bmatrix}$$

References

Ali, E. S., Abd Elazim, S. M., & Abdelaziz, A. Y. (2016). Ant Lion optimization algorithm for renewable distributed generations. *Energy, 116*, 445–458.

Ali, E. S., Abd Elazim, S. M., & Abdelaziz, A. Y. (2017). Ant Lion optimization algorithm for optimal location and sizing of renewable distributed generations. *Renewable Energy, 101*, 1311–1324.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In IEEE congress on evolutionary computation, CEC 2007, pp. 4661–4667.

Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithm. *International Journal of Advances in Soft Computing and its Applications, 5*(1), 1–35.

Bhattacharjee, K. K., & Sarmah, S. P. (2014). Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. *Applied Soft Computing Journal, 19*, 252–263.

Blickle, T., & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation, 4*(4), 361–394.

Caniyilmaz, E., Benli, B., & Ilkay, M. S. (2015). An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date. *International Journal of Advanced Manufacturing Technology, 77*(9–12), 2105–2115.

Chen, Y.-Y., Cheng, C.-Y., Wang, L.-C., & Chen, T.-L. (2012). A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems-A case study for solar cell industry. *International Journal of Production Economics, 141*(1), 66–78.

Chopra, N., & Mehta, S. (2015). Multi-objective optimum generation scheduling using ant lion optimization. Annual IEEE India conference (INDICON), New Delhi, pp. 1–6.

Coello, C. A., & Becerra, R. L. (2004). Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization, 36*(2), 219–236.

Dasgupta, D. (1999). *Artificial immune systems and their applications.* Springer-Verlag ISBN3540643907.

de Charsto, L. N., & Timmis, J. (2002). *An introduction to artificial immune systems: A new computational intelligence paradigm.* Springer-Verlag.

Dinkar, S. K., & Deep, K. (2017). Opposition based Laplacian ant lion optimizer. *Journal of Computational Science, 23*, 71–90.

Dinkar, S. K., & Deep, K. (2018). An efficient opposition based Lévy flight antlion optimizer for optimization problems. *Journal of Computational Science, 29*, 119–141.

Dinkar, S. K., & Deep, K. (2019). Accelerated opposition-based antlion optimizer with application to order reduction of linear time-invariant systems. *Arabian Journal for Science and Engineering, 44*(3), 2213–2241.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1*(4), 28–39.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26*(1), 29–41.

Dubey, H. M., Pandit, M., & Panigrahi, B. K. (2016). Ant lion optimization for short-term wind integrated hydrothermal power generation scheduling. *International Journal of Electrical Power & Energy Systems, 83*, 158–174.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In Sixth international symposium on micro machine and human science, MHS, pp. 39–43.

Emary, E., & Zawbaa, Hossam M. (2018). Feature selection via levy antlion optimization. *Pattern Analysis and Applications (PAAA)*.

Emary, E., Zawbaa, Hossam M., & Hassanien, Aboul Ella (2016). Binary ant lion approaches for feature selection. *Neurocomputing, 213*, 54–65.

Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Engineering Optimization, 38*(2), 129–154.

Geem, Zong Woo, Kim, Joong Hoon, & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation, 76*(2), 60–68.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533–549.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning, vol.* Addison-Wesley.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326.

Gupta, E., & Saxena, A. (2016). Performance evaluation of antlion optimizer based regulator in generation control of interconnected power system. *Journal of Engineering*.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* vol. Ann Arbor: Ann Arbor MI Univ. Michigan Press183.

Hosseini, S., & Al Khaled, A. (2014). A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future

research. *Applied Soft Computing Journal, 24*, 1078–1094.
Kamboj, V. K., Bhadoria, A., & Bath, S. K. (2017). Solution of non-convex economic load dispatch problem for small-scale power systems using ant lion optimizer. *Neural Computing and Applications, 28*(8), 2181–2192.
Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review, 42*(1), 21–57.
Karaboga, D., & Ozturk, C. (2011). A novel clustering approach: Artificial bee colony (ABC) algorithm. *Applied Soft Computing - Journal, 11*(1), 652–657.
Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report-TR06, Erciyes University, engineering faculty, computer engineering department, Tech. Rep.
Kaveh, A., & Ghazaan, M. I. (2017). Enhanced whale optimization algorithm for sizing optimization of skeletal structures. *Mechanics Based Design of Structures and Machines, 45*(3), 345–362.
Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Neural networks, 1995. Proceedings., IEEE Int. Conf., Vol. 4, pp. 1942–1948.
Kilic, H., Yuzgec, U., & Karakuzu, C. (2018). A novel improved antlion optimizer algorithm and its comparative performance. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-018-3871-9.
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science (80-.), 220*(4598), 671–680.
Kizilay, D., Tasgetiren, M. F., Bulut, O., & Bostan, B. (2014). A discrete artificial bee colony algorithm for the assignment and parallel machine scheduling problem in DYO paint company. In Proceedings of the 2014 IEEE congress on evolutionary computation, CEC 2014, pp. 653–660.
Li, X. (2003). *A new intelligent optimization—Artificial fish swarm algorithm (Doctor thesis)*. China: Zhejiang University of Zhejiang.
Li, X., Shao, Z., & Qian, J. (2002). An optimizing method base on autonomous animates: Fish swarm algorithm. *Systems Engineering - Theory & Practice, 22*, 32–38.
Luo, W., & Li, Y. (2016). Benchmarking heuristic search and optimization algorithms in matlab. In Automation and computing (ICAC) 22nd international conference on, IEEE, pp. 250–255.
Ma, H., & Simon, D. (2011). Blended biogeography-based optimization for constrained optimization. *Engineering Applications of Artificial Intelligence, 24*(3), 517–525.
McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science, 6*(1), 1–12.
Medjahed, S. A., Ait Saadi, T., Benyettou, A., & Ouali, M. (2016). Gray wolf optimizer for hyperspectral band selection. *Applied Soft Computing - Journal, 40*, 178–186.
Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics, 1*(4), 355–366.
Mirjalili, S. (2015). How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Applied Intelligence, 43*(1), 150–161.
Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.
Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27*(4), 1053–1073.
Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
Mishra, S. (2005). A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. *IEEE Transactions on Evolutionary Computation, 9*(1), 61–73.
Moslehi, G., & Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics, 129*(1), 14–22.
Nair, S. S., Rana, K. P. S., Kumar, V., & Chawla, A. (2016). Efficient modeling of linear discrete filters using ant lion optimizer. *Circuits, Systems, and Signal Processing*, 1–34.
Nanda, S. J., & Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation, 16*, 1–18.
Noraini, M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. *World Congress on Engineering, II*, 4–9 no. 978-988-19251-4–5.
Pakzad-Moghaddam, S. H. (2016). A Lévy flight embedded particle swarm optimization for multi-objective parallel-machine scheduling with learning and adapting considerations. *Computers & Industrial Engineering, 91*, 109–128.
Pan, W. T. (2012). A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems, 26*, 69–74.
Parallel machine scheduling using simulated annealing- Yarpiz. [Online]. Available: http://yarpiz.com/367/ypap107-parallel-machine-scheduling (visited on 17/01/2018).
Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine, 22*(3), 52–67.
Petrovic, M., Petronijevic, J., Mitic, M., Vukovic, N., Plemic, A., Miljkovic, Z., & Babic, B. (2015). The ant lion optimization algorithm for flexible process planning. *JPE, 18*(2), 65–68.
Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). Differential evolution: A practical approach to global optimization, vol. 28.
Rabadi, G., Moraga, R. J., & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing, 17*(1), 85–97.
Rajan, A., Jeevan, K., & Malakar, T. (2017). Weighted elitism based ant lion optimizer to solve optimum VAr planning problem. *Applied Soft Computing, 55*, 352–370.
Raju, M., Saikia, L. C., & Sinha, N. (2016). Automatic generation control of a multi-area system using ant lion optimizer algorithm based pid plus second order derivative controller. *International journal of electrical power and energy systems, 80*, 52–63.
Ramezanian, R., & Saidi-Mehrabad, M. (2012). Multi-product unrelated parallel machines scheduling problem with rework processes. *Scientia Iranica, 19*(6), 1887–1893.
Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Science (Ny), 179*(13), 2232–2248.
Razali, N. M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. *Int. Conf. Comput. Intell. Intell. Syst.*.
Sabri, N. M., Puteh, M., & Mahmood, M. R. (2013). A review of gravitational search algorithm. *International Journal of Advances in Soft Computing and its Applications, 5*(3).
Sang, H. Y., Duan, P. Y., & Li, J. Q. (2018). An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem. *Swarm and Evolutionary Computation, 38*, 42–53.
Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105*, 30–47.
Sayed, G. I., & Hassanien, A. E. (2017). Moth-flame swarm optimization with neutrosophic sets for automatic mitosis detection in breast cancer histology images. *Applied Intelligence, 47*(2), 397–408.
Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702–713.
Soza, C., Becerra, R. L., Riff, M. C., & Coello Coello, C. A. (2011). Solving timetabling problems using a cultural algorithm. *Applied Soft Computing - Journal, 11*(1), 337–344.
Sree Ranjini, S. R., & Murugan, S. (2017). Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications, 83*, 63–78.
Sreejith, S., Chandrasekaran, K., & Simon, S. P. (2009). Touring ant colony optimization technique for optimal power flow incorporating thyristor controlled series compensator. In World congress on nature and biologically inspired computing, NABIC 2009 - proceedings, pp. 1127–1132.
Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
Tang, W. J., Wu, Q. H., Saunders, J. R. (2006). Bacterial foraging algorithm for dynamic environments. In IEEE congress on evolutionary computation, CEC, pp. 1324–1330.
Tayarani, M. H. N., Yao, X., & Xu, H. (2015). Meta-heuristic algorithms in car engine design: A literature survey. *IEEE Transactions on Evolutionary Computation, 19*(5), 609–629.
Tharwat, A., & Hassanien, A. E. (2018). Chaotic antlion algorithm for parameter optimization of support vector machine. *Applied Intelligence, 48*(3), 670–686.
Wang, W. L., Wang, H. Y., Zhao, Y. W., Zhang, L. P., & Xu, X. L. (2013). Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm. *Computers & Operations Research, 40*(5), 1196–1206.
Wang, R. A., Zhou, Y. W., & Zheng, Y. Y. (2018). Ant lion optimizer with adaptive boundary and optimal guidance. *International conference on mechatronics and intelligent robotics* (pp. 379–386). Cham.: Springer.
Web site [Online]. Available: https://www.hq.nasa.gov/alsj/a16/a16.doodlebug.html (visited on 18/01/2018).
Willjuice Iruthayarajan, M., & Baskar, S. (2010). Covariance matrix adaptation evolution strategy based design of centralized PID controller. *Expert Systems with Applications, 37*(8), 5775–5781.
Yang, X. S. (2009). Harmony search as a metaheuristic algorithm. *Studies in Computational Intelligence, 191*, 1–14.
Yang, X. S. (2009). Firefly algorithms for multimodal optimization. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (pp. 169–178). LNCS.
Yang, X. S. (2010). Firefly algorithm, stochastic test functions and design optimization. *International Journal of Bio-Inspired Computation, 2*(2), 78–84.
Yang, X. -S., & Deb, S. (2009). Cuckoo search via Lévy flights. In IEEE world congress on nature and biologically inspired computing, pp. 210–214.
Yang, X. S., & Deb, S. (2014). Cuckoo search: Recent advances and applications. *Neural Computing and Applications, 24*(1), 169–174.
Yao, P., & Wang, H. (2016). Dynamic adaptive ant lion optimizer applied to route planning for unmanned aerial vehicle. *Soft Computing*, 1–14.
Yuan, C., Zheng, Y., Lu, Q., Li, Y., & Masood, I. (2018). ALO-DM: A smart approach based on ant lion optimizer with differential mutation operator in big data analytics. *Database systems for advanced applications: DASFAA 2018 international workshops: BDMS, BDQM, GDMA, and SeCoP, Gold Coast, QLD, Australia, May 21-24, 2018 proceedings* (pp. 64). Springer.
Zawbaa, Hossam M., Emary, E., & Grosan, Crina (March 2016). Feature selection via chaotic antlion optimization. *PLoS One, 11*(3).
Zhang, R., Song, S., & Wu, C. (2013). A simulation-based differential evolution algorithm for stochastic parallel machine scheduling with operational considerations. *International Transactions in Operational Research, 20*(4), 533–557.