



Article

A Novel Exponential-Weighted Method of the Antlion Optimization Algorithm for Improving the Convergence Rate

Szu-Chou Chen ¹, Wen-Chen Huang ^{2,*} , Ming-Hsien Hsueh ³ , Chieh-Yu Pan ⁴ and Chih-Hao Chang ⁵

- ¹ College of Management, National Kaohsiung University of Science and Technology, No. 1, University Rd., Yanchao Dist., Kaohsiung City 824005, Taiwan; 0528906@nkust.edu.tw
- ² Department of Information Management, National Kaohsiung University of Science and Technology, 2 Juoyue Rd., Nantsu, Kaohsiung 811532, Taiwan
- ³ Department of Industrial Engineering and Management, National Kaohsiung University of Science and Technology, Kaohsiung 807618, Taiwan; mhhsueh@nkust.edu.tw
- ⁴ Department and Graduate Institute of Aquaculture, National Kaohsiung University of Science and Technology, Kaohsiung 811213, Taiwan; panjade@nkust.edu.tw
- ⁵ Department of Marketing and Distribution Management, National Kaohsiung University of Science and Technology, Kaohsiung 824005, Taiwan; joechang@nkust.edu.tw
- * Correspondence: wenh@nkust.edu.tw

Abstract: The antlion optimization algorithm (ALO) is one of the most effective algorithms to solve combinatorial optimization problems, but it has some disadvantages, such as a long runtime. As a result, this problem impedes decision makers. In addition, due to the nature of the problem, the speed of convergence is a critical factor. As the size of the problem dimension grows, the convergence speed of the optimizer becomes increasingly significant. Many modified versions of the ALO have been developed in the past. Nevertheless, there are only a few research articles that discuss better boundary strategies that can increase the diversity of ants walking around an antlion to accelerate convergence. A novel exponential-weighted antlion optimization algorithm (EALO) is proposed in this paper to address slow convergence rates. The algorithm uses exponential functions and a random number in the interval 0, 1 to increase the diversity of the ant's random walks. It has been demonstrated that by optimizing twelve classical objective functions of benchmark functions, the novel method has a higher convergence rate than the ALO. This is because it has the most powerful search capability and speed. In addition, the proposed method has also been compared to other existing methods, and it has obtained superior experimental results relative to compared methods. Therefore, the proposed EALO method deserves consideration as a possible optimization tool for solving combinatorial optimization problems, due to its highly competitive results.

Keywords: metaheuristic; antlion optimization; particle swarm optimization



Citation: Chen, S.-C.; Huang, W.-C.; Hsueh, M.-H.; Pan, C.-Y.; Chang, C.-H. A Novel Exponential-Weighted Method of the Antlion Optimization Algorithm for Improving the Convergence Rate. *Processes* **2022**, *10*, 1413. <https://doi.org/10.3390/pr10071413>

Academic Editor: Philippe Bogaerts

Received: 28 June 2022

Accepted: 18 July 2022

Published: 20 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the last decade, metaheuristic optimization techniques have been introduced as primary techniques for finding an optimal solution to financial, industrial, and engineering optimization problems [1]. The issues include portfolio models, media selections, sales territory realignments, sales call scheduling, and electric and gas distribution problems. Modern optimization problems have become increasingly complex; conventional methods consume too much time solving these problems, especially as the problem dimensions increase. Increasing the size of the situation makes the convergence speed of the optimizer more significant and important.

Optimization involves finding the best value in a functional constraint using mathematical methods. However, real-world optimization problems are often nonlinear and higher order, and traditional mathematical procedures cannot provide exact theoretical solutions. There are two types of optimization problems, continuous optimization problems and combinatorial optimization problems. Continuous optimization problems have

continuous variables, while combinatorial problems have discrete variables. Discrete-space optimization problems are called combinatorial optimization (CO) problems, while continuous-space optimization problems have different solutions [2]. Three algorithms exist to solve combinatorial optimization problems, including exact, approximation, and metaheuristic [3]. Metaheuristic algorithms and approximation algorithms have a shorter runtime than exact algorithms. They may not, however, be as precise as exact algorithms. In most cases, worst-case guarantees are provided when designing approximation algorithms; therefore, the algorithms are not scalable [3]. Many CO problems are NP-hard and do not have polynomial-time solutions. Metaheuristic algorithms have accordingly been developed to solve these problems.

Metaheuristic algorithms solve combinatorial optimization problems and reasonably determine approximate optimal solutions. Metaheuristic methods are often derived from observations of nature. The goal is to find, generate, or select heuristics or partial search algorithms that can provide a satisfactory solution to an optimization problem, especially when the information is incomplete, imperfect, or analyzed with limited computing resources [4]. Unlike exact methods, metaheuristics cannot guarantee optimal solutions for specific problems. In many implementations of metaheuristic algorithms, stochastic values are added for optimization, so the solution found changes with the set of random variables generated and is slightly different every time it is run. By searching for many feasible solutions to CO problems, metaheuristics can often find solutions with less computational effort. Metaheuristics are, therefore, practical methods for solving CO problems.

Numerous metaheuristic algorithms have been developed, including the genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and gray wolf optimizer (GWO). Recent trends have shown that antlion optimization algorithms have helped optimize CO problems. In 2015, Seyedali Mirjalili proposed the ant-lion optimizer (ALO) [5], an optimization algorithm based on the hunting behavior of antlions. After using the ALO algorithm, the search space is explored better, local optima are avoided, and the convergence rate is improved. A total of six hundred articles concerning ALO studies were published in 2020 [6] and have been applied to many fields, such as computer science, medicine, engineering, mathematics, and energy. For ALO, many variant algorithms have been proposed. In addition to using basic enhancement methods, some researchers have combined their ALO algorithms with other machine learning algorithms to achieve multi-objective solutions. The ALO algorithm can be improved by replacing the roulette-wheel ant update equation with a process based on the rank of decreasing orders for all random walks [7] and by using modified decreasing boundaries, instead of step-by-step reduction to shrink the boundary [8]. A random walk, a reproduction, a sliding method, elitism, and a selection method were proposed by Kılıç and Yüzgeç [9]. However, there are only a few research papers addressing better boundary strategies that increase the diversity of ants walking around an antlion. We propose an exponential-weighted ant lion optimization (EALO) algorithm to fill this research gap, by increasing the diversity of ants sliding into the trap and accelerating convergence.

The following section reviews relevant research, including the GA, ACO, GWO, and PSO algorithms. Section 3 describes the ALO algorithm. Our EALO approach to improving the convergence rate is presented in Section 4, and Section 5 presents the experimental results on benchmark objective functions. Finally, Section 6 concludes the paper.

2. Literature Review

Since 1975, over one hundred metaheuristic algorithms [6] have been inspired by nature to solve complicated optimization problems that cover science, technology, engineering, etc. Holland proposed the genetic algorithm in 1975 [10], which is the foundation of all evolutionary algorithms and seeks to optimize exploration in space utilizing self-replication. Their evolutionary algorithms form a family, encoding a potential solution to a specific problem. This solution creates codes with a chromosome data structure and applies specific operators to recombine these structures to preserve critical information within a generation.

After many generations (iterations) of evolution, chromosomes containing meaningful information are held, and the codes of unimportant chromosomes are discarded. During the development of each generation, random values are added randomly to enhance the ability of evolution to achieve optimized results.

The genetic algorithm consists of a population, chromosomes, and genes. A gene is the basic coding unit, a chromosome is an entity, a collection of genes, and the population is a set of chromosomes. In the beginning, the genetic algorithm needs to determine the population's size (population size). The fitness function calculates the chromosomes' quality and screens out chromosomes with better fitness values. The fitness value is a numerical performance metric used to determine chromosomes' fitness level, and different fitness functions correspond to other calculation methods in different application scenarios. The genetic algorithm sets the fitness value after each iteration. When the fitness value of the group of chromosomes reaches the target fitness value, the group of chromosomes is the solution to be found. An implementation of a genetic algorithm that is problem-dependent has the following two main phases: the problem encoding and the evaluation function [11]. The first phase in implementing a genetic algorithm is to generate an initial population (typically randomly). The second phase consists of selection, crossover, and mutation. Selection is applied to create an intermediate population. According to the fitness values corresponding to different chromosomes, the group is more likely to be selected when the fitness values are higher to ensure that good gene sequences will be retained. There are many methods of selection, which have been discussed in past literature. Crossover is the mating of two genes to produce new chromosomes. The crossover of the two chromosomes is also controlled by mating probability. There are also many methods of crossover, such as single-point, multipoint, etc. Mutations make changes to specific genes on selected chromosomes. The mutation will also change according to the mutation probability. After the crossover, mutation is necessary, as the mutation prevents the result from falling into a local optimum.

After the genetic algorithm, the particle swarm algorithm [12] is also a well-known evolutionary computation algorithm. This algorithm has attracted the attention of academia, due to its advantages of easy implementation, high precision, and fast convergence and has shown its superiority in solving practical problems. Dr. Kennedy and Dr. Eberhart proposed it in 1995, which originated from research on the predation behavior of birds. The algorithm was originally inspired by the regularity of the flocking activities of flying birds and then used the wisdom of crowds to establish a simplified model. The particle swarm optimization algorithm is based on observing the behavior of animal clusters. It uses the sharing of information by individuals in the group to move the whole group to produce an evolution process from disorder to order in the problem-solving space to obtain the optimal solution. It starts from a random solution and finds the optimal solution through iteration. It evaluates the quality of the solution through fitness, but it is simpler than the genetic algorithm rules. It does not have the crossover and mutation of the genetic algorithm operations, which finds the global optimal by following the currently searched optimal value.

PSO simulates the predation behavior of a flock of birds. A flock of birds randomly searches for food. There is only one piece of food in this area. All birds do not know where the food is. However, they know how far the current location is from the food. The way to find food is to search the area around the bird closest to the food. Particle swarm optimization is a parallel algorithm. During the whole search process, the birds pass their respective information to each other so that other birds know their position. In the end, the whole flock can gather around the food source; that is, the optimal solution is found [12]. On PSO, a bird is called a "particle." All particles have a fitness value determined by an objective function, and each particle also has a velocity that determines the direction and distance they fly. Then, the particles follow the current optimal particle search in the solution space. In each iteration, the particle updates itself by tracking two extremes. The first is the optimal solution found by the particle itself, called the individual extreme value

“*pBest*”. The other extreme value is the optimal solution found by the entire population, and this extreme value is the extreme global value “*gBest*”. The particle decides the next movement through its own experience and the best experience of its peers and, in this way, finds the optimal global solution.

Dorigo et al. [13] proposed the ant colony optimization algorithm in 1991 as a novel nature-inspired metaheuristic for traveling salesman optimization problems. The foraging behavior of real ants inspires ACO. When searching for food, ants initially randomly explore the area around their nest. Once the ant finds a food source, it assesses the quantity and quality of the food and brings some back to the lair. The ants deposit a pheromone as traces on the ground on the return trip. The amount of pheromone deposited, depending on the quantity and quality of the food, guides other ants to the food source, and this pheromone will be emitted at a certain rate. Ants communicate indirectly through pheromone trajectories, enabling them to find the shortest path between the nest and the food source [14].

A significant advantage of ant colony algorithms is that they are robust in solving problems compared to other heuristic algorithms. A slight modification to the basic ant colony algorithm model can be applied to other problems and leads to improved solutions. A population-based evolutionary algorithm, or ant colony algorithm, is easily parallelized and has inherent parallelism. Various heuristic algorithms may be combined with ant colony algorithms to enhance performance. In addition, a slow convergence speed may result in falling into a local optimum.

Using swarm intelligence, the gray wolf optimizer [15] simulates the predatory behavior of gray wolves. The first algorithm based on this was proposed by the Australian scholar Mirjalili et al. in 2014. Gray wolves are assigned attack and predation duties to complete the predation process, thus contributing to global optimization. The gray wolf has an interesting social hierarchy based on social dominance. There is one male and one female leader, known as an alpha. They decide what to hunt, where to sleep, when to wake up, and more. The second tier of gray wolf authority is beta. Betas are subordinate wolves who assist the alphas in making decisions and enforcing discipline within the pack. A beta reinforces alpha commands throughout the pack and provides feedback to the alpha. Omega is the lowest ranking gray wolf. Omega wolves must always submit to all other dominant wolves to catch prey. In GWO, the process of hunting can be divided into the following three phases: encirclement, pursuit, and attack. Gray wolves represent feasible solutions in the solution space of the gray wolf algorithm. The three best-positioned gray wolves in the group are alphas and betas. These three wolves will move around and pursue prey until they find it (optimal solution) during the hunting process. During every positioning round, a new alpha and beta will be selected based on the merits of the position. In this step, each gray wolf in the wolf pack will walk a certain distance toward the three optimal gray wolves. Gray wolves will move to the three highest-ranking individuals in their group.

The GWO algorithm has been widely used in engineering optimization problems compared with GA, PSO, and ACO because of its simple structure, few parameters, and ease of operation. Even so, the convergence speed is slowed, due to the lack of diversity in optimization. When faced with complex optimization problems, it has low solution accuracy and easy early convergence to a local optimum.

ALO [5] is an algorithm for evolutionary computation. Over six hundred studies have shown the importance and superiority of the ALO algorithm, as mentioned in the first section [6]. The ant lion optimizer offers global optimization, few adjustment parameters, high convergence accuracy, robustness, and other benefits. The antlion algorithm has three roles, including ant, antlion, and elite. An ant symbolizes the solution; it walks randomly but slides toward the antlion because of the trap. Antlions represent the optimal local solution, which is updated after each random walk based on the fitness values of the ants and antlions. The elites represent the optimal global solution. Each update of all antlions will select the antlion with the highest evaluation value as the elite antlion to

prevent falling into the optimal local solution. With continuous iteration, a better solution is found near the optimal local solution, and finally, a more accurate global optimal solution is found. In the following section, we describe the antlion optimization algorithm.

The ALO algorithm has several disadvantages, including many iterative calculations and a limited range of applications. Several improvements have been made to ALO in the literature [6,16]. To solve feature selection problems, Emary et al. [17] proposed a binary ant lion optimizer (BALO). Its solutions are binary [0, 1]. In the first approach, binary solutions were obtained through random walks around the selected antlion. In this article, the second approach applies the original ALO method, which uses continuous steps as thresholds. After that, the appropriate threshold function is applied. The improved antlion optimization (IALO) algorithm proposed by Kumar et al. [7] optimizes power extraction from partially shaded solar photovoltaic panels during the rainy season by sorting all random walks of ants in decreasing order. This is carried out by orienting them toward the antlions from higher rank to lower rank to replace the original roulette wheel procedure.

Kılıç and Yüzgeç [9] improved the ALO algorithm by applying a tournament selection method to the parallel machine scheduling problem, thus improving the algorithm's performance. In the original ALO algorithm, the ant's walking mechanism is iterated as many times as possible. In this study, however, a fifth of the maximum number of iterations was used to reduce run time. Additionally, a randomly selected variable option has been used to improve accuracy and speed during the phase of sliding ants toward the antlion. Finally, the roulette wheel was replaced with a tournament. In [18], Kılıç and Yüzgeç proposed a novel approach to solving quadratic assignment problems based on tournament selection called TALO. In contrast to the original ALO algorithm, in this article, when the ant's position is outside the search space, they return to the search space again. Using this mechanism, the ants could take random walks in the search space.

A dynamic adaptive ant lion optimizer (DAALO) for route planning of unmanned aerial vehicles is presented in this paper [19]. A random walk is updated by Levy flight, making it easier to escape local optima in ALO. Additionally, ALO's performance is improved, including meeting speed, convergence accuracy, and stability, by using the population's development rate as feedback. In two different environments, including one with a city model and one with a mountain model, the proposed algorithm is superior.

For the operation parameters of an electric furnace, Ksiazek et al. [20] proposed a simulation and positioning system based on a modified ALO algorithm. With a state flipping approach, the values of the optimal model are assigned to agents calculated in parallel populations. To achieve better results, the values for lower fitness value agents are flipped. Hence, the proposed heuristic is more dynamically adaptable to the model. It also proposed some additional modifications to deploy search efficiently across the entire domain to avoid local minima. There are often optimal recitative power dispatch problems in power systems. Consequently, Rajan et al. [21] proposed improvements to ALO's search capabilities. In the elite stage of the original ALO, a new weighted elite concept was introduced to enhance the exploratory nature of the algorithm. Modified ALO (MALO) intelligently balances exploration and exploitation, enhancing ALO's hunting ability. To determine the optimal generator voltage, tap position of tap changer transformers, and volt amps reactive (VAR) output of shunt capacitors, both ALO and MALO are used. This is carried out to optimize objectives such as active power loss, total voltage deviation, and voltage stability metrics.

Jiang et al. [22] proposed a modified ant lion optimizer algorithm (IALO) to enhance the accuracy of short-term wind predictions. IALO-BP predicts models based on backward propagation neural networks with the improved ant lion optimizer algorithm (IALO). To improve the convergence speed and generalization ability of BP neural networks, this model uses IALO to optimize its weights and thresholds. IALO-BP neural networks are superior to other BP models, and their prediction values are close to the actual values. Liu et al. [23] proposed a self-adapting ALO algorithm, where A is the fitness value of the antlion participating in the roulette wheel and P is the average fitness value of all

the antlions. If $A < A * P * rand$, then A participates in the round; otherwise, it does not participate. Additionally, it is suggested to use dynamic scale coefficients when calculating ants and elite antlions. This is in addition to adaptive boundary strategies to improve and increase the diversity of ants when they walk around antlions. In this study, self-adaptive boundary strategies will be compared with each other, as explained in Section 5.

3. Antlion Optimization Algorithm

ALO is a widely used and effective optimization method proposed by Mirjalili [5]. Antlions undergo a two-phase life cycle, larval and adult, and they can live up to three years in the wild. The life cycle occurs primarily in the larval stage and only 3–5 weeks during the adult phase. The ALO mimics the behavior of the antlion, which in its larval phase digs in the sandy soil to hunt for prey. Antlions dig a cone-shaped pit in the sand as a trap for hunting their primary targets, ants, and they wait at the bottom of the pit. After discovering that the prey is in the pit, the antlion attempts to capture it. To enable prey to slide to the bottom of the pit, antlions throw sand toward the edge of the pit. When the prey is caught, it is dragged beneath the soil to be consumed. To prepare for the next hunt, the antlion will throw out the remaining prey and repair the pit after it has consumed the prey.

The hunting behavior of antlions can be formulated as a mathematical model and applied to numerous optimization problems. There are two populations used in the ALO algorithm, ants and antlions. The search space consists of two types of species, ants that represent solutions to the problem and move randomly and antlions hidden in random locations within the search space. Both are defined in the two matrices, M_A and M_{AL} , as follows:

$$M_A = \begin{pmatrix} A_{11} & \dots & A_{1d} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nd} \end{pmatrix} \quad M_{AL} = \begin{pmatrix} AL_{11} & \dots & AL_{1d} \\ \vdots & \ddots & \vdots \\ AL_{n1} & \dots & AL_{nd} \end{pmatrix} \quad (1)$$

where M_A and M_{AL} are the populations of the ants and the antlions, respectively, and both are matrices of the same size; d is the dimensions of the solution; and n is the number of the populations.

In general, ants move randomly. The random walk of ants is modeled as given in Equation (2).

$$X(t) = [0, C(2r(t_1) - 1), \dots, C(2r(t_T) - 1)] \quad (2)$$

where $X(t)$ is the random walk matrix; C is the function that calculates the cumulative sum; t is the step of random walk (iteration in the study); and T is the maximum number of iterations. Finally, $r(t)$ is a stochastic function defined as follows:

$$r(t) = \{1 \text{ if } rand > 0.5, 0 \text{ if } rand \leq 0.5\} \quad (3)$$

where $rand$ is a random number generated with a uniform distribution in $[0, 1]$. To keep the ant's position within the boundaries of the search space and prevent the ants from overshooting, the following normalization function is used for the random walk of the ants.

$$X_i^t = \frac{(X_i^t - a_i) \times (q_i^t - p_i^t)}{(b_i - a_i)} + p_i^t \quad (4)$$

where p_i^t and q_i^t are the minimum and maximum values of the lower and upper boundaries for the i -th variable at the t -th iteration, respectively; a_i is the minimum of the random walk of the i -th variable; and b_i is the maximum of the random walk in the i -th variable. In one step of the iteration process, each ant is assumed to be in the trap of only one antlion and is selected by the roulette wheel mechanism, according to its fitness value.

To simulate ant entrapment in antlion pits, the position of the chosen antlion is used to update the minimum and maximum boundaries of the ants' random walks.

$$p_i^t = p^t + Antlion_j^t \quad (5)$$

$$q_i^t = q^t + Antlion_j^t \quad (6)$$

where p^t and q^t are the minima and maximum values of all the variables at the t -th iteration, p_i^t is the minimum of all variables for the i -th ant, q_i^t is the maximum of all variables for the i -th ant, and $Antlion_j^t$ shows the position of the selected j -th antlion at the t -th iteration.

To mimic the sliding of an ant toward the bottom of the antlion pit, the boundaries of random walks should be adaptively decreased as follows:

$$p^t = \frac{p^t}{I} \quad (7)$$

$$q^t = \frac{q^t}{I} \quad (8)$$

where I is a ratio, t is the current iteration, p^t is the minimum value of all the variables at the t -th iteration, and q^t is the maximum value of all the variables at the t -th iteration. In the above equations, I is defined as follows:

$$I = 10^{\omega \frac{t}{T}} \quad (9)$$

where t is the current iteration, T is the maximum number of iterations, and ω is defined based on the current iteration ($\omega = 2$ when $t > 0.1T$, $\omega = 3$ when $t > 0.5T$, $\omega = 4$ when $t > 0.75T$, $\omega = 5$ when $t > 0.9T$, and $\omega = 6$ when $t > 0.95T$). The parameter ω is used to adjust the accuracy level of exploitation.

In the optimization process, antlions update their positions according to the fitness values of the ants. If the ant has a better fitness value than the selected antlion, then it changes its position to the latest position of the hunted ant. The following equation is presented for the antlion catching the ant and reconstructing the pit.

$$Antlion_j^t = Ant_i^t \text{ if } f(Ant_i^t) > f(Antlion_j^t) \quad (10)$$

where f is the objective function, which obtains a fitness value.

Keeping the best solution obtained at each optimization process step is necessary. Thus far, the fittest antlion has been preserved and is considered an elite. It should be capable of influencing the movement of all ants during optimal steps. This optimization process step is modeled by the following equation:

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \quad (11)$$

where Ant_i^t indicates the position of the i -th ant at the t -th iteration, R_A^t denotes the random walk around the antlion selected by the roulette wheel at the t -th iteration, and R_E^t denotes the elite antlion obtained by Equation (10).

The ALO algorithm architecture is as Figure 1,

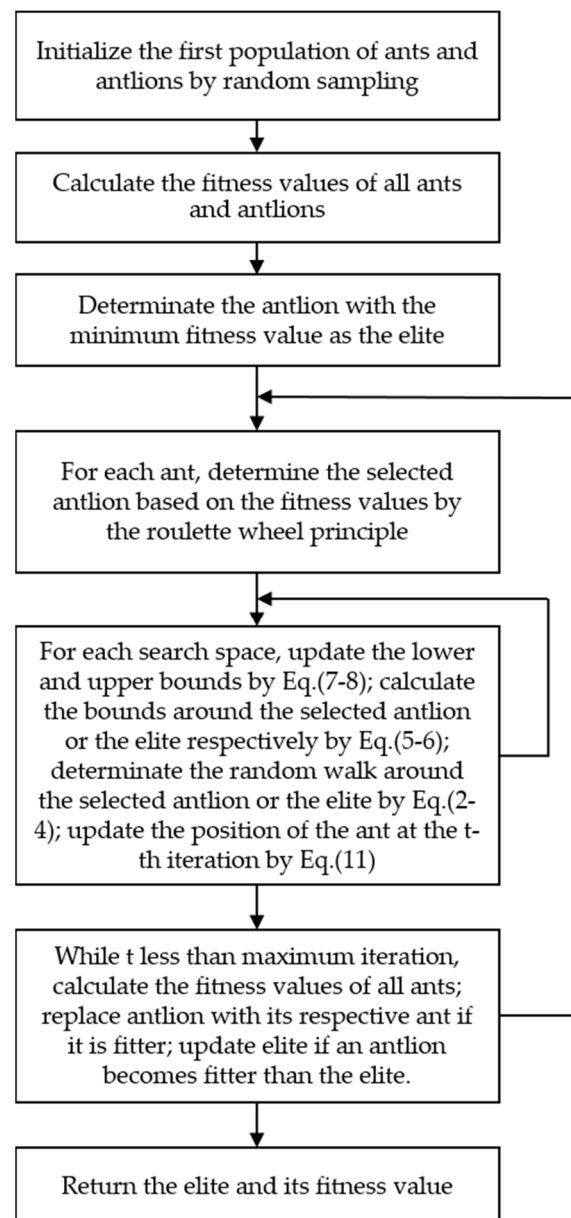


Figure 1. The architecture of the ALO algorithm.

4. The Proposed Algorithm: EALO

If we are trying to find the maximum or minimum value of a function in an optimization problem, For instance, if one considers the problem of solving $F(X_1, X_2, \dots, X_n)$, these X_s represent the parameters of the problem, which may or may not be independent; these X_s form a set of variables. Utilizing the optimization algorithm, we can find the most effective combination of variables in the shortest amount of time.

There are several things researchers look for in optimization algorithms, including the ability to explore space, ability to exploit space, obtaining optimal global solutions, not falling into local optimums, and speed of convergence. The aim of the study is to investigate how fast convergence can occur. When solving optimization problems, several variables are involved, which will profoundly affect the execution time. As the number of variables increases, the execution time will increase nonlinearly. To enhance the convergence rate, we selected the improved antlion optimization algorithm.

Compared with some other optimization algorithms, the antlion optimization algorithm can avoid falling into local optima and achieve convergence to the optimal

solution [5,18]. Despite its advantages, the antlion optimization algorithm has a drawback. Since it uses decreasing procedures for shrinking the boundary, it does not converge as fast as it should, as shown in Equation (9). The parameter ω in Equation (9) is used to adjust the accuracy level of exploitation only and the (t/T) parameter is a linear function. When the number of iterations is getting larger, the convergence speed cannot be increased as fast as it can do. To increase the diversity of ants around the antlion and accelerate the convergence rate, the following function is proposed in this study:

$$I = 10^{\omega} \frac{t}{T} \left(\exp\left(\frac{t}{T}\right) \cdot rand \cdot \eta \right) \eta = 1, 2, 3 \quad (12)$$

where *rand* is a random number between intervals of (0, 1); *t* is the current iteration; *T* is the maximum number of iterations; ω is defined based on the current iteration ($\omega = 2$ when $t > 0.1T$, $\omega = 3$ when $t > 0.5T$, $\omega = 4$ when $t > 0.75T$, $\omega = 5$ when $t > 0.9T$, and $\omega = 6$ when $t > 0.95T$); and *EXP* is the exponential function with (t/T) . η is a deviation variable used to control the degree to which the convergence speed curve accelerates as the number of iterations increases. The next section's experiments show that an η of 20 is sufficient to speed up the convergence. The exponential function significantly accelerates the convergence speed, so the enhanced ALO algorithm is called the exponential-weighted ant lion optimization (EALO) algorithm.

Hence, the pseudocode of the EALO algorithm is defined as Algorithm 1,

Algorithm 1: The exponential-weighted antlion optimization (EALO) algorithm

Input: Objective function, population size, maximum iteration number, the value of η

Output: The elite antlion and its fitness value

Step 1: Initialize positions of ants and antlions randomly

Step 2: Calculate the fitness values of ants and antlions using the objective function

Step 3: Find the best antlions and assume one is the elite

Step 4: *while* (iteration < Maximum iteration)

for every ant

Select an antlion using the roulette wheel method

Calculate *I* using Equation (12)

Update *p* and *q* using Equations (7) and (8)

Create a random walk using the equation Equation (2)

Normalize the random walk using Equation (4)

Update the position of the ant using Equation (11)

end for

Calculate the fitness values of all ants

Replace antlion with its respective ant if it is fitter using Equation (10)

Update elite if an antlion becomes fitter than the elite

end while

Step 5: Return the elite and its fitness value

5. Experimental Study on Benchmark Functions

An exponential-weighted ant lion optimization algorithm was implemented in MATLAB R2021b using a Core i5 processor at 2.8 GHz. In the following research experiments, with regard to the graphs that explore the convergence curve, the horizontal axis represents the number of iterations, and the vertical axis represents the best fitness value obtained so far. Experimental results using the proposed approach are presented in this section. In addition, we compared the proposed method with the original ant lion algorithm [5], the particle swarm optimization algorithm [12] and gray wolf optimizer [15].

5.1. Benchmark Functions

In this study, we analyzed twelve benchmark functions published in the literature [5] to investigate how the proposed algorithm performs. By comparing optimization algorithms with these functions, we were able to compare the performance of the algorithms. In

Table 1, different characteristics that are used for algorithm analysis are summarized, and in Figure 2, their 3D graphs are depicted.

Table 1. Benchmark functions.

F	Dim	Range	F _{min}	Formulations
F1	30	(−100, 100)	0	$\sum_{i=1}^n x_i^2$
F2	30	(−10, 10)	0	$\sum_{i=1}^n x_i + \prod_{i=1}^n x_i $
F3	30	(−100, 100)	0	$\sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$
F4	30	(−100, 100)	0	$\max_i \{ x_i , 1 \leq i \leq n\}$
F5	30	(−30, 30)	0	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
F6	30	(−100, 100)	0	$\sum_{i=1}^n (x_i + 0.5)^2$
F7	30	(−5.12, 5.12)	0	$\sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$
F8	30	(−1.28, 1.28)	0	$\sum_{i=1}^n iX_i^4 + \text{random}[0, 1)$
F9	30	(−600, 600)	0	$\frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F10	30	(−50, 50)	0	$\frac{\pi}{n} \{10 \sin(\pi y_i) + \sum_{i=1}^n (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]$ $+ (y_n - 1)^2\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4} u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a < x_i < a; k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$
F11	30	(−50, 50)	0	$0.1 + \{\sin^2(\beta\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)]\}$ $+ 0.1(x_n - 1)^2 [1 + \sin^2(2\pi x_n)] + \sum_{i=1}^n u(x_i, 5, 100, 4)$
F12	4	(−5, 5)	0.0003	$\sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$

5.2. The Property of Variable η

In Equation (12), a deviation variable η is used to control the amount to which the convergence speed curve accelerates with increasing iterations. η is an integer ranging from 1 to a fairly large value. To determine how large the η value should be, we conducted the following experiments: number of search agents (i.e., number of ants, equivalence to number of antlions) is 40; number of iterations is 500; benchmark function F4 is used; and the η values 1, 5, 10, 20, and 30 have been used to test which is the most suitable fit and compare it to the original ALO algorithm. Figure 3 shows the results of the experiment. As η increases, the convergence curve of the EALO deviates from the convergence curve of the original ALO algorithm, especially as the number of iterations increases.

In Figure 3, it can be observed that when the η value increases, EALO's convergence speed increases. By comparing the convergence curve with an η value equal to 30 and an η value equal to 20, we observe that the degree of deviation is almost the same, and the speed of convergence has not greatly improved. These results can also be observed in other benchmark functions, such as F6 in Figure 4. As a result, an initial parameter value of $\eta = 20$ was chosen for the following experiments.

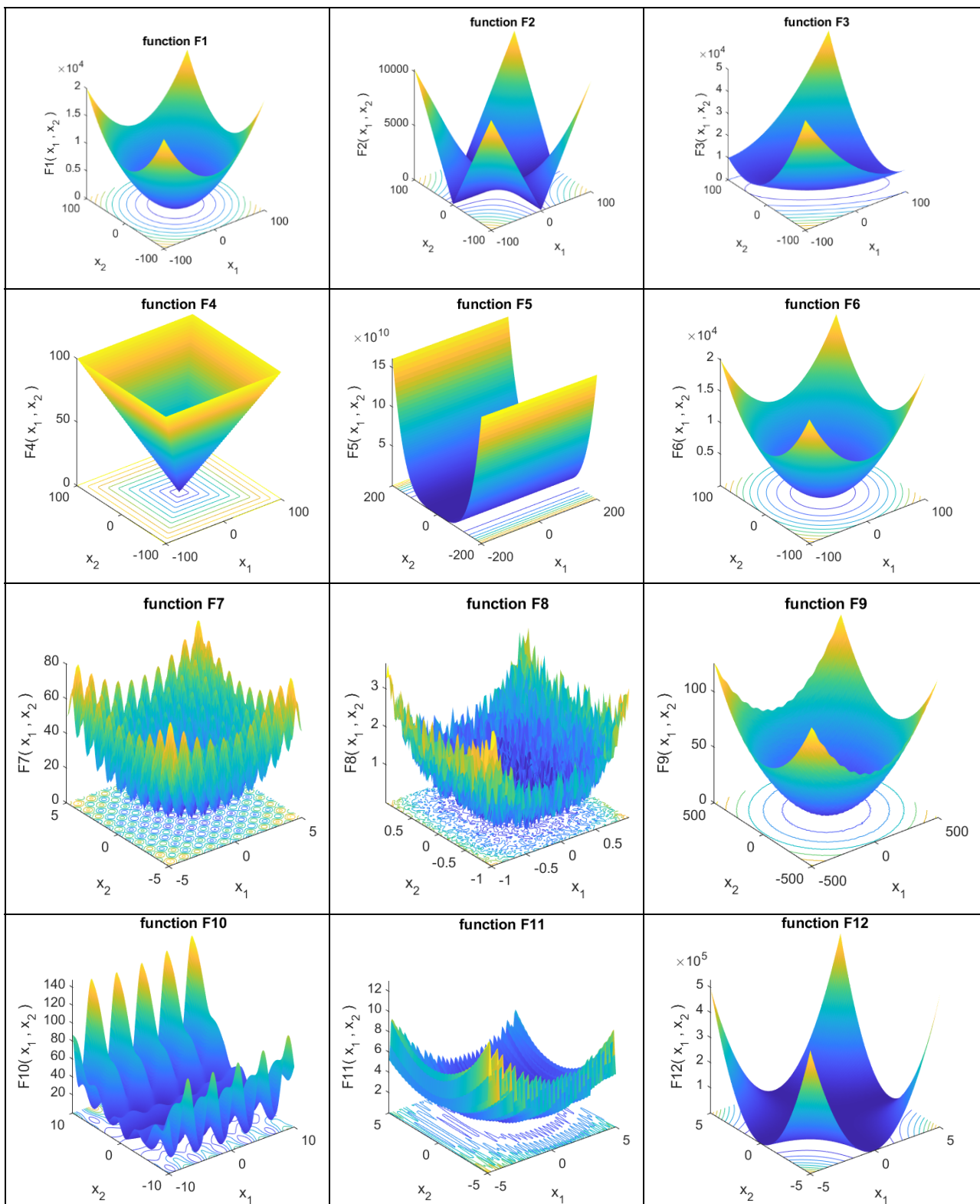


Figure 2. Benchmark function 3D graphs.

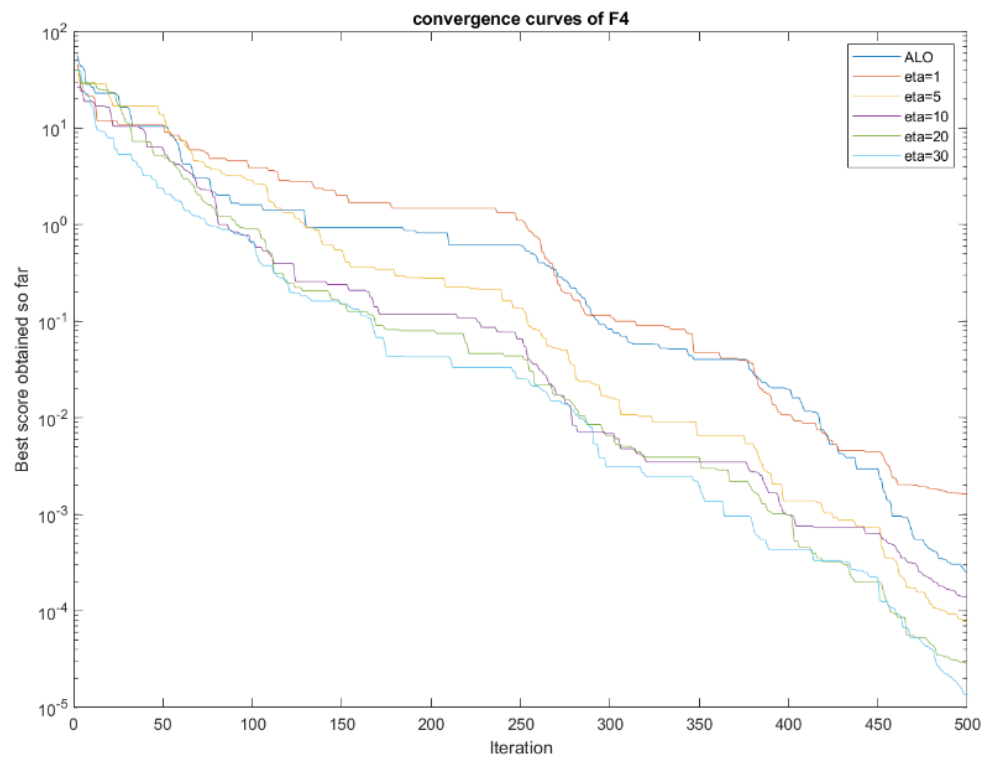


Figure 3. The results of experimentation on F4 with different η values.

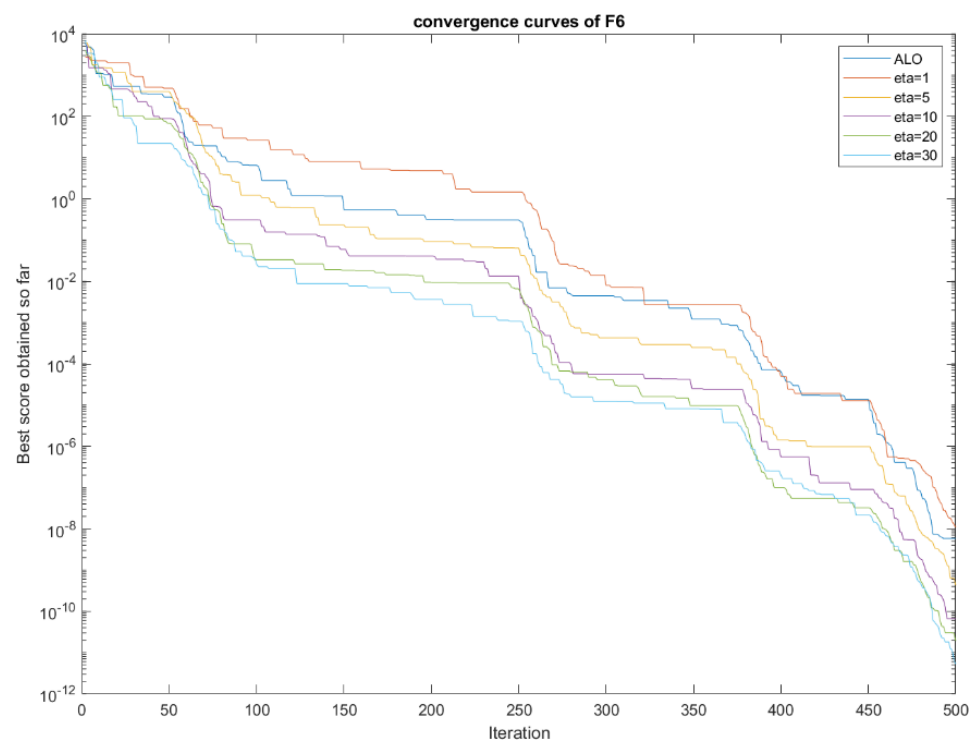


Figure 4. The results of experimentation on F6 with different η values.

In Figure 5, it illustrates the different η value ($\eta = 1, 10, 20, 30$) convergent results of EALO with benchmark functions from F1 to F12 except F4 and F6. Although $\eta = 30$ is better than $\eta = 20$ for some benchmark functions, it is reasonable to use $\eta = 20$ in this study for the following experiments.

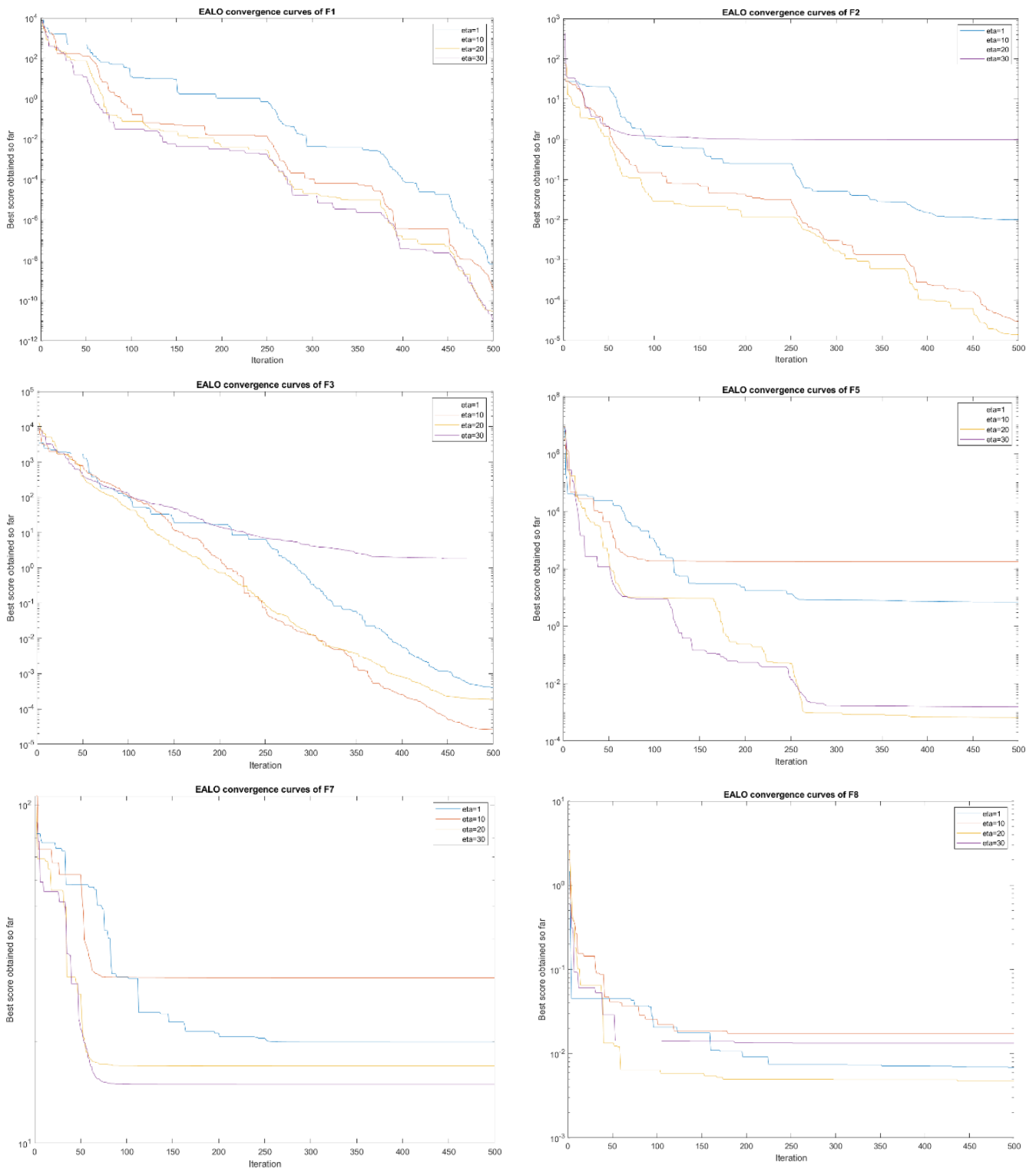


Figure 5. Cont.

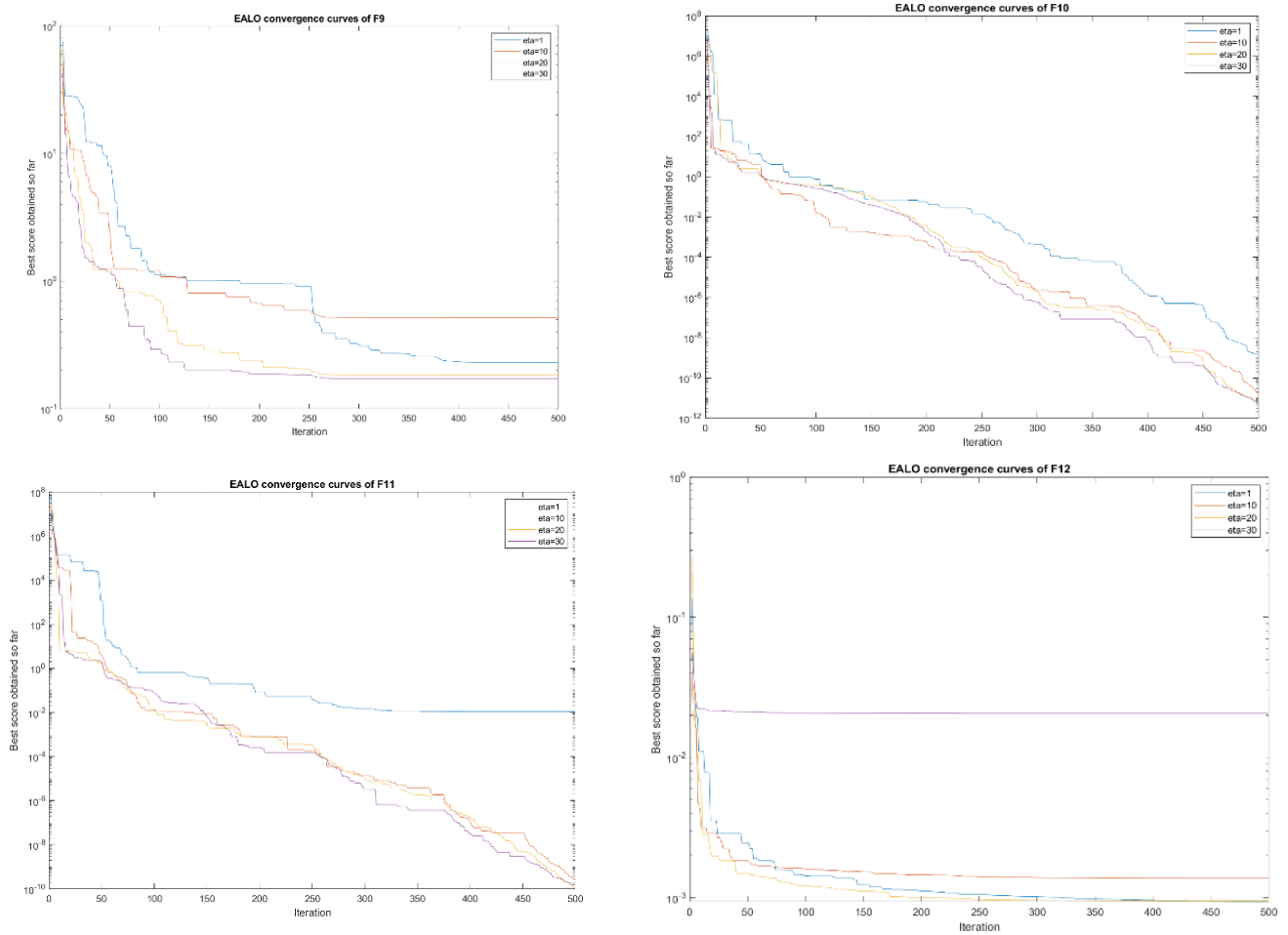


Figure 5. The different η value convergent results of F1 to F12, except F4 and F6.

5.3. Convergence Analysis

Another literature-cited method was included in this experiment to better understand the effectiveness of the proposed EALO's convergence rate. Using the trigonometric sin function for weighting, Liu et al. [23] proposed a method to speed up convergence rates. The formula is as follows:

$$I = 10^{\omega} \frac{t}{T} \left(0.5 + \sin\left(\frac{t\pi}{2T} \cdot rand\right) \right) \quad (13)$$

This method is called *saALO* here. After selecting all the benchmark functions, the optimal solutions were determined by searching 30 agents over 500 iterations with η equal to 20. Some of the experimental results are illustrated in Figure 6.

The above results demonstrate that the proposed EALO method (represented by the red line) significantly outperforms the original ALO method (represented by the blue line). Table 2 summarizes the optimal solutions for those twelve benchmark functions.

The superiority of the EALO algorithm increases with the number of iterations, which is significantly better than the original ALO algorithm and the *saALO* algorithm. In Figure 7, we show the experimental results for benchmark functions F1 and F10 at 500 and 1000 iterations.

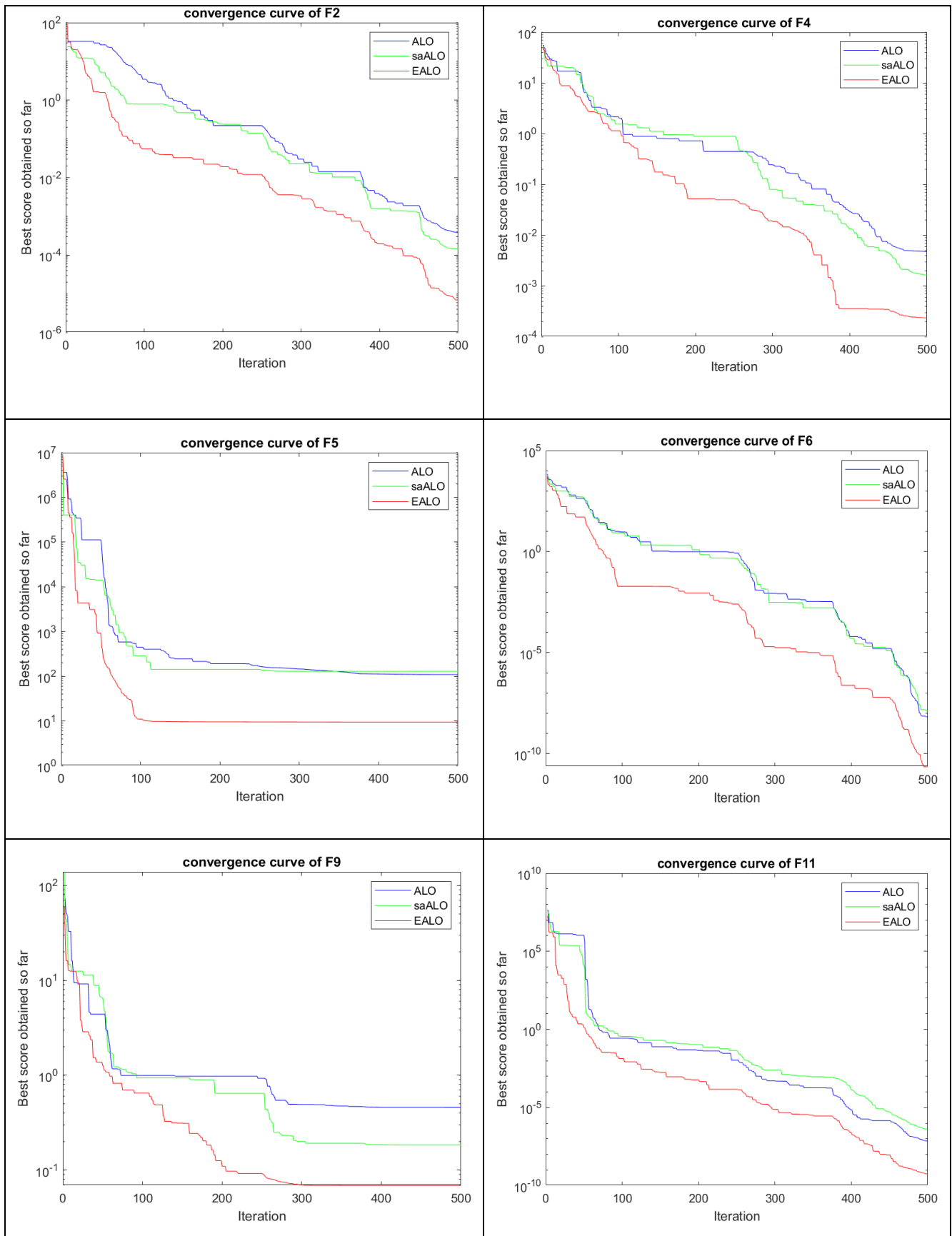
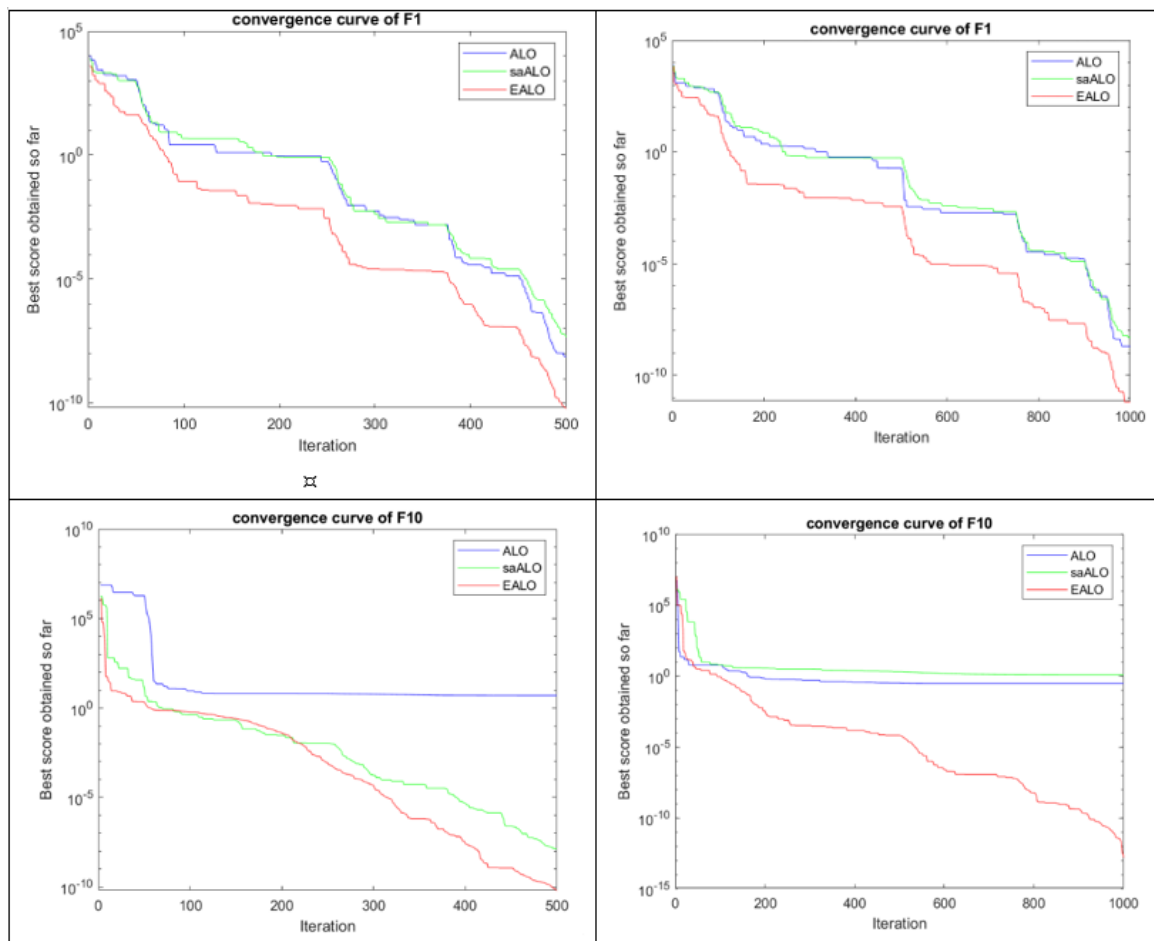


Figure 6. The results of the experiments comparing ALO, saALO, and EALO.

Table 2. The optimal solutions (i.e., elites' fitness value) of the experiment.

	ALO	saALO	EALO
F1	7.6618×10^{-9}	4.5673×10^{-8}	6.027×10^{-11}
F2	0.00037506	0.00014578	6.8061×10^{-6}
F3	0.033416	0.029191	0.013887
F4	0.0048024	0.0015434	0.00023618
F5	109.5965	128.6998	9.4584
F6	6.6575×10^{-9}	1.3089×10^{-8}	2.2545×10^{-11}
F7	27.8588	14.9244	8.9546
F8	0.029629	0.038728	0.011835
F9	0.45981	0.18441	0.068873
F10	5.0051	1.3128×10^{-8}	5.9899×10^{-11}
F11	6.8259×10^{-8}	3.8091×10^{-7}	5.5097×10^{-10}
F12	0.00076687	0.00066378	0.0005845

**Figure 7.** The experimental results of F1 and F10 at 500 and 1000 iterations.

5.4. Comparison Results with Other ALO Algorithms

In this subsection, the proposed EALO algorithm is compared to other ALO versions for benchmark functions. For comparison, the particle swarm optimization algorithm (also known as PSO) and gray wolf optimization algorithm (also known as GWO) are used. The following experiments used the same parameter values as the previous ones, meaning the number of agents is 30, the number of iterations is 500, and the value of η is 20. Some of the experimental results are illustrated in Figure 8.

The experimental results in Figure 8 show that the EALO algorithm has a much better convergence rate than the PSO algorithm.

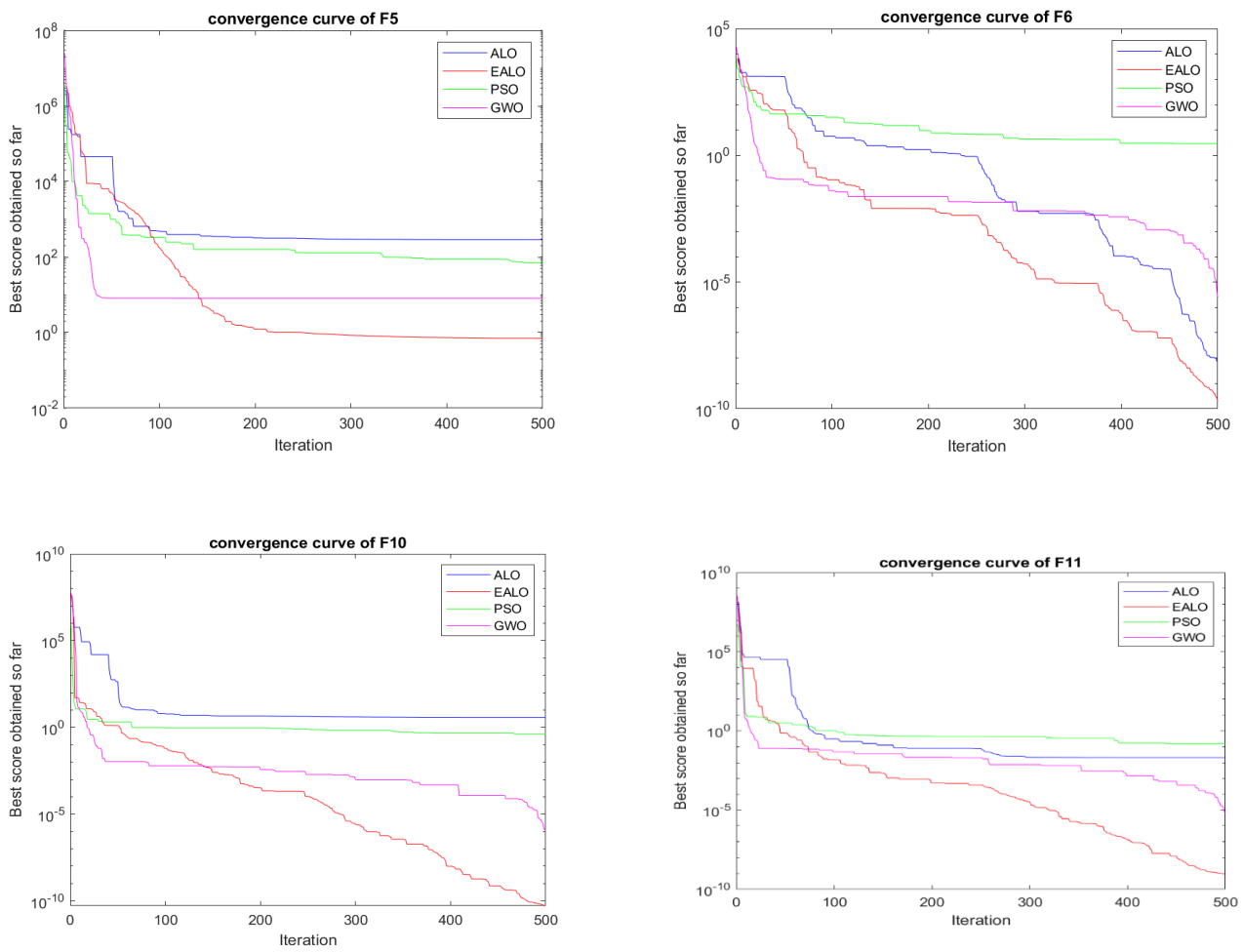


Figure 8. Comparison of EALO and PSO, GWO convergence curves for some benchmark functions.

5.5. Discussion

Figure 9 illustrates the formula as a stepped reduction using logarithms. Equation (9) is used in the original ALO method to shrink the boundary.

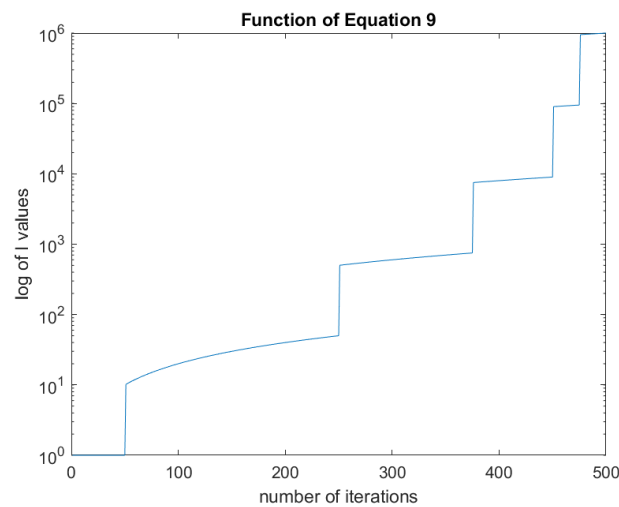


Figure 9. A stepped growth pattern for Equation (9).

In Figure 9, the I value of formula Equation (9) shows stepping growth when the iteration is increased to 500. The exponential weighting algorithm proposed in this study

accelerates the convergence rate of the original ALO algorithm and makes it run smoothly. Increasing the value of η can cause the convergence curve to deviate from the convergence curve of the original ALO algorithm, greatly speeding up convergence.

Based on the above experimental results, it is evident from the graphs that the proposed EALO algorithm has the fastest convergence rate. However, the proposed algorithm still has limitations and uncertainties. Equation (12) has a random number, and this random number is between the interval of 0, 1. This will influence the results of every experiment.

6. Conclusions

Recently, metaheuristic algorithms have become the focus of research, due to their ability to find a reasonable solution to extremely complex optimization problems within a limited time. The antlion optimization algorithm is one of the metaheuristic swarm-based approaches introduced by Mirjalili in 2015 that has attracted considerable attention. The ALO algorithm has been improved numerous times since its introduction. In addition to escaping from local optima, the ALO algorithm and its improvements can explore large search spaces, apply Laplace distributions or other stochastic functions instead of uniform distributions, and employ reproduction ant-sliding and elitism methods. It is evident that speeding up convergence is the most critical and significant improvement. Few studies have examined better boundary strategies to increase the diversity of ants on the ground and antlions to speed up convergence. Taking advantage of this research gap, a novel exponential-weighted antlion optimization algorithm is presented in this study.

An improvement in the original antlion optimization algorithm consists essentially of an adaptive shrinkage of the radius of the ants' random walk hypersphere to accelerate convergence. This study proposes the EALO algorithm, which is an improvement and an innovation over the ALO algorithm. In this paper, we present a high-level innovative method that presents a nonlinear adaptive increasing trend with a stochastic component as the evolutionary iterations increase. Therefore, the boundary size exhibits a nonlinear adaptive decreasing trend with randomness as the evolutionary iterations increase. To improve the performance of the ALO, an exponential function and a deviation variable were utilized to improve the boundary decreasing procedure. To adjust the speed of convergence, this variable is critical.

Using twelve objective function experiments, the results demonstrated that the EALO algorithm obtained very competitive results. The experimental results show that the proposed EALO algorithm has significant improvements in terms of faster convergence rates compared to the original ALO algorithm. Moreover, the proposed method has been compared with other existing methods, such as particle swarm optimization algorithms. The results showed that the EALO algorithm was significantly superior in terms of convergence rates.

In the future, EALO should be applied to practical problems in computer science, energy, medicine, and engineering design. It is also worthwhile investigating the constrained conditions that may be ameliorated by this algorithm.

Author Contributions: Conceptualization, W.-C.H. and S.-C.C.; methodology, S.-C.C.; software, S.-C.C.; validation, S.-C.C. and W.-C.H.; writing—original draft preparation, S.-C.C.; writing—review and editing, W.-C.H., S.-C.C., M.-H.H., C.-Y.P. and C.-H.C.; supervision, W.-C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, Z.; Cao, Y.; Dai, L.V.; Yang, X.; Nguyen, T.T. Finding solutions for optimal reactive power dispatch problem by a novel improved antlion optimization algorithm. *Energies* **2019**, *12*, 2968. [[CrossRef](#)]
2. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement learning for combinatorial optimization: A survey. *Comput. Oper. Res.* **2021**, *134*, 105400. [[CrossRef](#)]

3. Drori, I.; Kharkar, A.; Sickinger, W.R.; Kates, B.; Ma, Q.; Ge, S.; Dolev, E.; Dietrich, B.; Williamson, D.P.; Udell, M. Learning to solve combinatorial optimization problems on real-world graphs in linear time. In Proceedings of the 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 14–17 December 2020; pp. 19–34.
4. Bianchi, L.; Dorigo, M.; Gambardella, L.M.; Gutjahr, W.J. A survey on metaheuristics for stochastic combinatorial optimization. *Nat. Comput.* **2009**, *8*, 239–287. [[CrossRef](#)]
5. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [[CrossRef](#)]
6. Assiri, A.S.; Hussien, A.G.; Amin, M. Ant lion optimization: Variants, hybrids, and applications. *IEEE Access* **2020**, *8*, 77746–77764. [[CrossRef](#)]
7. Kumar, N.; Hussain, I.; Singh, B.; Panigrahi, B.K. Maximum power extraction from partially shaded PV panel in rainy season by using improved antlions optimization algorithm. In Proceedings of the 2016 IEEE 7th Power India International Conference (PIICON), Bikaner, Rajasthan, 1–6 November 2016.
8. Toz, M. An improved form of the ant lion optimization algorithm for image clustering problems. *Turk. J. Electr. Eng. Comput. Sci.* **2019**, *27*, 1445–1460. [[CrossRef](#)]
9. Kılıç, H.; Yüzgeç, U. Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling. *Comput. Ind. Eng.* **2019**, *132*, 66–186. [[CrossRef](#)]
10. Holland, J. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
11. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [[CrossRef](#)]
12. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
13. Dorigo, M.; Birattari, M.; Stützle, T. Ant Colony Optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [[CrossRef](#)]
14. Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278. [[CrossRef](#)]
15. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [[CrossRef](#)]
16. Abualigah, L.; Shehab, M.; Alshinwan, M.; Mirjalili, S.; Elaziz, M.A. Ant lion optimizer: A comprehensive survey of its variants and application. *Arch. Comput. Methods Eng.* **2021**, *28*, 1397–1416. [[CrossRef](#)]
17. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary ant lion approaches for feature selection. *Neurocomputing* **2016**, *213*, 54–65. [[CrossRef](#)]
18. Kılıç, H.; Yüzgeç, U. Tournament selection based antlion optimization algorithm for solving quadratic assignment problem. *Eng. Sci. Technol. Int. J.* **2019**, *22*, 573–691. [[CrossRef](#)]
19. Yao, P.; Wang, H. Dynamic adaptive ant lion optimizer applied to route planning for unmanned aerial vehicle. *Soft Comput.* **2017**, *21*, 5475–5488. [[CrossRef](#)]
20. Ksiazek, K.; Polap, D.; Woźniak, M.; Damavsevicius, R. Radiation heat transfer optimization by the use of modified ant lion optimizer. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7.
21. Rajan, A.; Jeevan, K.; Malakar, T. Weighted elitism based ant lion optimizer to solve optimum var planning problem. *Appl. Soft Comput.* **2017**, *55*, 352–370. [[CrossRef](#)]
22. Jiang, F.; He, J.; Peng, Z. Short-term wind power forecasting based on bp neural network with improved ant lion optimizer. In Proceedings of the 37th Chinese Control Conference, Wuhan, China, 25–27 July 2018; pp. 8543–8547.
23. Liu, J.; Huo, Y.; Li, Y. Preferred strategy based self-adaptive ant lion optimization algorithm. *Pattern Recognit. Artif. Intell.* **2020**, *33*, 121–132.