Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 9, No. 4, 2021, 583-595.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Hybrid Particle Swarm Optimization with Ant-Lion Optimization: Experimental in Benchmarks and Applications

Zeinab Hassani[1*] and Mohsen Alambardar Meybodi[2]

1. Department of Computer Science, Kosar University of Bojnord, Iran.
2. Department of Applied Mathematics and Computer Science, University of Isfahan, P.O. Box 81746,73441, Isfahan, Iran, University.

Article Info

Article History:
Received 16 February 2021
Revised 07 May 2021
Accepted 19 September 2021

DOI:10.22044/JADM.2021.10550.2195

Keywords:
Hybrid Optimization Algorithm, Particle Swarm Optimization, Ant Lion Optimization, K-Nearest Neighbor.

**Corresponding author: Hassani@kub.ac.ir(Z. Hassani).*

Abstract

A major pitfall in the standard version of Particle Swarm Optimization (PSO) is that it might get stuck in the local optima. In order to escape this issue, a novel hybrid model based on a combination of PSO and Ant-Lion Optimization (ALO) is proposed in this work. The proposed method, called H-PSO-ALO, uses a local search strategy by employing the Ant-Lion algorithm to select the less correlated and salient feature subset. The objective is to improve the prediction accuracy and adaptability of the model in various datasets by balancing the exploration and exploitation processes. The performance of our method has is evaluated on 30 benchmark classification problems, CEC 2017 benchmark problems, and some well-known datasets.in order to verify the performance, four algorithms, including FDR-PSO, CLPSO, HFPSO, and MPSO, are elected to be compared with the efficiency of H-PSO-ALO. Considering the experimental results, the proposed method outperforms the others in many cases, so it seems that it is a desirable candidate for the optimization problems on real-world datasets.

1. Introduction

Nowadays, one of the main tools in encountering many optimization problems is using the nature-inspired optimization algorithms. These algorithms commonly attempt to find the solution closer to the optimal by converging faster. In order to achieve these purposes, the various optimization algorithms should be equipped with the exploration and exploitation to search the global optimum. Exploration and exploitation are two principal processes of the optimization algorithm. Exploration achieves a global optimal by identifying and exploring a broad search space of solutions, evaluating the candidate solutions in a more extensive search space. Exploitation obtains the local optimal solutions by identifying and examining a bounded search space. It provides the local optimal by searching in the neighborhood of the solution or solutions, while the exploration aim is to escape the local optimum. In other words, the ability that leads to the most excellent results of the functions near to the optimal results is indicated by the exploitation, and the capacity to find the best outcome in the whole search space is evaluated by exploration [20]. The main intention is to establish a trade-off between exploration and exploitation in seeking the optimal global solution. An excellent exploration keeps from getting into a local minimum, while significant exploitation proposes an effective convergence speed [22].

Many studies have been carried out to find this balance with the efficient criteria in the evolutionary algorithms like PSO. As a result, brilliant ideas are appealed to put this balance in dealing with complex optimization problems. One of the dominant tricks recently used is a combination of the heuristic algorithms and the optimization algorithms, known as the hybrid optimization technique. The hybrid optimization techniques lead to a more robust behavior and a greater flexibility against complex problems. Another technique is the local search. The algorithms used in this technique iteratively attempt to closer solution to optimal than the

available solution using a suitably determined neighborhood mechanism [3].

The rest of this paper is organized into five sections, what follows. Section 2 contains a short review of the recent related studies. Section 3 introduces the basic concepts of the PSO algorithm and the ant-lion algorithm. Section 4 discusses the detailed version of the proposed approach. Section 5 contains the experimental results compared with some other well-known algorithms.

2. Short Review of Related Works

The Particle Swarm Optimization (PSO) algorithm has been manipulated in various problems from the literature in order to find the optimal solution. Na Qu et al. [18] have used the PSO algorithm that detects series arc fault in electrical circuits. Omidinasab et al. [16] have studied the optimum design of truss structures with discrete design variables. A model based on the PSO algorithm has been presented for analyzing the dynamic behavior of the heavy-duty gas turbine plants in a real-time environment [7]. Salehpour et al. in [19] have investigates a class of optimal control problems by the PSO-SVM indirect method. A new classifier for the detection of diseases has been proposed by Sunil et al. in [21]. These and many other works show the importance of PSO in solving different optimizations in a vast area of science.

One feature that makes PSO worthwhile is that PSO could quickly converge to the optimal solution. The quick convergence occurs due to using the best experience of the particles in the process of finding the optimal solution. However, PSO is sometimes trapped in the local optima due to its non-uniform movement. In other words, PSO has a poor ability to search in the near local optimum region, which is an essential task of the optimization process [9]. Avoiding this premature convergence is a challenging problem in PSO algorithm [14]. One solution in dealing with this defect is decreasing the convergence ability in the solution space close to a globally optimal solution [3].

Hybrid methods: The first hybrid metaheuristic algorithm for feature selection was presented in 2004 [15]. After that, many researchers have contributed to this area, and investigated hybridizing different optimization algorithms. For instance, Mafarja et al. have introduced the hybrid whale optimization algorithm with simulated annealing [12]. In their approach, the local search techniques were embedded into the genetic algorithm in order to control the search process. Singh et al. have proposed a new hybrid nature-inspired algorithm called the hybrid PSO and grey wolf optimizer. Improving the ability of exploitation in PSO with the power of exploration in grey wolf optimizer is their main idea [20]. The main trick used in [3] was utilizing the seeking potential of firefly algorithm in POS. Multi-dimensional CEC 2015 and CEC 2017 computationally expensive sets of numerical and engineering, mechanical design benchmark problems are used for the experimental study. The results obtained revealed that the proposed HFPSO had outstanding performances compared to the other ones [3]. DA-PSO is the PSO that is embedded by the dragonfly algorithm (DA) to find the optimized solutions for the power system [9]. The Particle Swarm Optimization-Cuckoo Search (PSO-CS) [5] improves the success rate and convergence iteration for the PSO algorithm, and in the experiments, premier results are achieved. In 2019, S. Fan et al. introduced a cooperative multi-swarm structure in order to enhance the searching ability of the classical PSO algorithm. They proposed a variation of the original PSO algorithm for unconstrained optimization, dubbed the enhanced partial search particle swarm optimizer (EPS-PSO) using the idea of multiple cooperative swarms in an attempt to improve the convergence and efficiency of the original PSO algorithm [6], and recently, Modified PSO (MPSO) has been presented using an adaptive strategy to balance the global exploration and local exploitation to the overcoming complex optimization problems in expert systems [11]

Our contribution: As already mentioned, a primary trap in the standard version of PSO is that it might get stuck in a stagnant state if, in some consecutive iterations, the global best position cannot be improved. A combination algorithm is proposed in order to balance exploration and exploitation in dealing with this issue. PSO is combined with ALO to seek a specific search space. Briefly, the PSO algorithm is used as a global search process, and ALO is used as a local search process in our hybrid algorithm. When the fitness function value improves, compared to the previous state, it is necessary to get the best result in searching for a bounded space in the vicinity of the best solution obtained until now by the ant-lion algorithm. Otherwise, the PSO algorithm is looking for the best solution in the vast search space. The main reason for manipulating the ant-lion algorithm is its excellent results in exploitation in local optimization problems [13]. The experimental results show that the exploration

of PSO and the exploitation of ALO are fully exerted, so the hybrid algorithm has a faster convergence speed than PSO and ALO individually.

3. Literature Review

3.1. Particle Swarm Optimization Algorithm

In [8], Eberhart and Kennedy have proposed the PSO algorithm, one of the most common methods in global optimization, by an inspiring nature. Their method imitates swarm behaviors such as bird gathering and fish training. PSO looks for equilibriuzm in the search space by adjusting the position of each particle according to its own experience and that of its neighbors. Excellent properties such as faster convergence with fewer parameters and more straightforward running, lead to broad usages of PSO in different fields. Job scheduling, route planning, robotics, and wireless sensor networks, are some of these fields [26]. Briefly, the main idea of PSO is as what follows.

Initialize Parameters: We define all the essential parameters.

m: is a set of particles in the d-dimensional search space;

$X_i = [x_{i_1}, x_{i_2}, ..., x_{i_d}]$: is the position of each particle P_i, where x_{ij} is the position of the particle P_i in the dimension j;

$V_i = [v_{i_1}, v_{i_2}, ..., v_{i_d}]$: is the velocity of each particle P_i, where v_{i_j} is the velocity of the particle P_i: in the dimension j;

P_{best}: is the local optimal particle;

g_{best}: is the global optimal particle;

$x_i^{P_{best}}$: is the best position so far visited by the particle i at the swarm level;

$x_i^{g_{best}}$: is the best global position.

Update: During each generation, a particle, based on the two parameters, local and global optimal particle, updates its position and velocity according to the following formulas:

$$x_{id}(t+1) = x_{id}(t)\ (t) + v_{id}(t), \tag{1}$$

$$v_{id}(t+1) = w \times v_{id}(t) + a_1 \times (x_i^{P_{best}} - x_{id}(t)) - a_2 \times (x^{g_{best}} - x_{id}(t)), \tag{2}$$

where $x_{i_d}(t)$ and $v_{i_d}(t)$ indicate the position and velocity at the current status t, the parameter w is the coefficient of inertia, α_1 and α_2 are the learning coefficients, and r_1 and r_2 are two random numbers within the range $[0,1]$.

3.2. Ant-Lion Optimization Algorithm

In the recent years, many algorithms have been inspired by the behavior of creatures in nature. ALO, proposed by Mirjalili in 2015 [13], is one of them that applicate the hunting method of ant-lions. An ant-lion excavates a cone-shaped hole in the ground by moving along a circular path and dropping out gravel with its large prong. After drilling the trap, the ant-lion conceals under the bottom of the cone. As the edge of the cone is high slope enough to a hunting fall to the bottom of the trap, when an insect gets entered in the trap, it easily falling down. Once the larva finds out that the prey is in the trap, he attempts to get it, pulls it under the sand, and consumes it. Behind consuming the insect, the ant-lion cleans its hole by dropping the rest out of the hole, and waits for the next prey. We consider ants as the prey for ant-lions.

ALO is mathematically formalized in five main steps that consist of random walk of ants, trapping in ant-lion's holes, building trap, sliding ants towards ant-lion, consuming prey, and reconstructing traps [1, 13, 23, 29].

We summarize all the essential parameters in ALO, as follow:

n: is the number of ants;

m: is the number of ant-lions;

f: is a fitness function;

$Y_i = [y_{i_1}, y_{i_2}, ..., y_{i_d}]$: is the position of ant i, where y_{i_j} is the value of j^{th} dimension of the ant i;

$\chi_{ant} = [Y_1, Y_2, ..., Y_d]$: is the position of all ants. It is a 2D, where $\chi_{ant}[i, j]$ means the value in the j^{th} dimension of ant i;

$f(X) = [f(Y_1), f(Y_2), ...f(Y_d)]$: is the value of fitness function for the ants;

$Z_i = [z_{i_1}, z_{i_2}, ..., z_{i_d}]$: is the position of ant-lion i where z_{i_j} is the value j^{th} dimension of the ant-lion i;

$\chi_{antlion} = [Z_1, Z_2, ..., Z_d]$: is the position of all ants. It is a 2-dimensional array, where $\chi_{ant}[i, j]$ means the value in the j^{th} dimension of ant i;

$f(Z) = [f(Z_1), f(Z_2), ...f(Z_d)]$: is the value of fitness function for the ants.

Movement model of ants: The movement of ants is based on a random walk. In each iteration, each ant generates a random number in internal $[0,1]$ by uniform distribution, called rand, and then an ant moves (all dimensions) one step forward $(+1)$ or backward (-1). If the rand number is greater (or equal) than 0.5, we consider forward movement; otherwise, the backward movement occurs. thus we have a random walk in iteration like t. If we initialize the position of ant i to zero after n steps, the position of ant i is updated like the following:

$$[Y_i^0, Y_i^1, Y_i^2, ..., Y_i^n] \quad (3)$$

where Y_i^k is the accumulative sum of k step random walk in one iteration. We consider the position of ant i in iteration t equal to Y_i^n; in other words, $Y_i(t) = Y_i^k$. However, since the movement's space of ants is bounded, in other words in our problem the limited search space, the position of ants cannot be directly updated. In order to hold the position of an ant inside the search space in each step in random walk, we convert Y_i^k by $\frac{(Y_i^k - a_i)}{(b_i - a_i)}$ in the range of $[0,1]$. Then it transfers to the domain $[c_i, d_i]$ by multiplying $(d_i - c_i)$ and adding c_i, where:

a_i is the minimum value of random walk in i^{th} dimension till step k;

b_i is the maximum value of the random walk in the i^{th} dimension till step k;

c_i^t is the minimum value of the i^{th} dimension at t^{th} iteration;

d_i^t indicates the maximum of i^{th} variable at the t^{th} iteration;

The positions of all ants are saved in a 2D χ_i, as follow:

$$\chi_{\text{ant}} = [Y_1, Y_2, ..., Y_d],$$

where Y_i indicates the position vector of the i^{th} ant.

Similarly, we assume that the ant-lions are hiding somewhere in the search space. Thus their position values are maintained by $\chi_{antlion} = [Z_{k,1}, Z_{k,2}, ..., Z_{k,d}]$, where $Z_{k,j}$ indicates the value of the j^{th} dimension of ant-lion k.

For evaluating each ant, a fitness function called f is utilized during the optimization. $f(,)$ is the value of fitness function for the i^{th} ant or ant-lion.

Trapping in ant-lion's holes: Random walks of ants are affected by ant-lions' traps. After entering the ant i in the hole of the selected ant-lion j, the workspace for ant i is affected by the following equations:

$$C_i^t = Z_j^t + C^t \qquad D_i^t = Z_j^t + D^t, \quad (4)$$

where C_i^t, D_i^t are the minimum, the maximum of all variables at t^{th} iteration, C^t, D^t are the minimum, maximum of all variables for i^{th} ant, and Z_j^t presents the position of the selected j^{th} ant-lion at the t^{th} iteration. Ant i walks randomly in a hypersphere by the vectors C_i and D_i around the chosen ant-lion.

Building trap: in order to gain a high chance of catching ants, a roulette wheel is used. This mechanism recognizes the fittest ant-lions.

Sliding ants towards ant-lions: As soon as an ant enters a trap, the ant-lion starts to throw sands outwards the center of its hole. This action slides ants towards ant-lions, prevent them from escaping by decreasing C_i and D_i. For modeling this action, the radius of ants' random walks hypersphere is reduced. The following equations update these parameters as follows:

$$C^t = \frac{(C^t * T)}{(10^w * t)} \qquad D^t = \frac{(D^t * T)}{(10^w * t)}, \quad (5)$$

where t is the iteration number, C^t and D^t are the minimum, and maximum of all variables at t^{th} iteration, T is the upper bound of the iteration numbers, and w is a constant parameter that is computed on iteration number as follows:

$$w = \begin{cases} 2 & t > 0.1T, \\ 3 & t > 0.5T, \\ 4 & t > 0.75T, \\ 5 & t > 0.9T, \\ 6 & t > 0.95T \end{cases} \quad (6)$$

Catching ant and re-building the trap: After an ant reaches to the bottom of the trap, the ant-lion consumes it, and it does the final step of its hunting method. The final step is updating the position to the latest position as:

$$Z_j^t = Y_i^t \quad \text{if} \quad f(Y_i^t) > f(Z_j^t), \quad (7)$$

where t displays the current iteration number.

Elitism: How do the evolutionary algorithms preserve the best solution in iterations? Elitism is a fundamental property of these kinds of algorithms.

It leads to the ants' movements affected by the selected ant-lion. In other words, the position of the ant is calculated by the average of random walks near the chosen ant-lion. The following equation shows this effect:

$$Y_i^t = \frac{(R_A^t + R_E^t)}{2}, \quad (8)$$

where R_A^t, R_E^t are the random walks nearby the ant-lion by the roulette wheel at the i^{th} iteration.

4. Proposed H-PSO-ALO Algorithm

In this section, a hybrid optimization algorithm has is introduced that combines the search ability of PSO and ALO. The weakness of PSO, poor in searching near the optimum local, and immature convergence are covered by ALO. In other words, PSO and ALO are adapted at exploring the search space and exploiting the solution, respectively. This combination leads to a balance between exploration and exploitation, and as a consequence, the benefits of both algorithms are manipulated in the solution. Briefly, due to the ability of the fast convergence in the exploration of PSO, it is used in the global search, and due to fine-tuning in the exploitation of ALO, it is applied to the local search. A flowchart of the proposed hybrid of ALO and PSO is displayed in Figure 1. In the following, the details of our hybrid algorithm are explained.

Step 1: Initialization:

1-1: Initialize the input parameters of hybrid algorithm;

1-2: The positions and velocities of particles initialize randomly in the ranges;

1-3: Calculate fitness, global and personal best particles values (g_{best} and p_{best}).

Step 2: This step includes the exploitation and exploration phases, which are based on their local best positions and the global best of the swarm.

2-1: Exploitation phase: in this step, the algorithm compares the fitness of a particle with the best global value seen until now based on Eq.9

$$f(i,t)=\begin{cases} true & f(P_i^t) <= f(g_{best}^{t-1}), \\ false & f(P_i^t) > f(g_{best}^{t-1}), \end{cases} \quad (9)$$

where P_i^t is i^{th} particle in the current state t. If $f(i,t)$ is true, the local search proceeds, and the particle is manipulated by a simulative ALO. Then the current position is saved in x_{temp} , and the new position is considered by the position of the ant-lion algorithm, and the velocity is calculated based on Eq. 10

$$v_{i_d}(t+1)=x_{i_d}(t+1)-x_{i_{temp}}, \quad (10)$$

Otherwise, $f(i,t)$ is false, and particle will be employed by PSO and PSO continues its standard process with this particle based on Eq. 2 and Eq. 1. The minimum and maximum velocities of a particle $(V_{\min}, V_{\max})$ are applied to limit the next distance in a direction. They are randomly initialized at the beginning of the proposed hybrid algorithm in the velocity range. A linear decreasing inertia weight is applied and evaluated based on Eq. 11

$$w=w_i-((w_i-w_f)/n)*t, \quad (11)$$

Where n and t are the maximum numbers of iterations and the current iteration, respectively. w_i and w_f are the initial and final values of linear decreasing inertia weight. The value is updated dynamically to raises the global searching ability of the particle, and to avoid precocity that brings up improvements on previous personal best [24].

2-2: Exploration phase: The fitness function -is calculates, and the range limitations are investigated for all particles and ant-lions. The best solutions (g_{best} and p_{best}) are updated after calculating the fitness function.

Step 3: The hybrid algorithm will be terminated when the number of iterations(n) maximizes. Finally, the global best particles (g_{best} and its fitness value), as the output of the proposed hybrid algorithm, will be determined.

5. Experiments and Discussion

In this work, the results of two types of conducted experiments evaluate the performance of the proposed method. Computationally expensive numerical CEC 2017 has been selected in the experimental as the benchmark test sets. Also some medical and non-medical datasets are used in the second experiment as another benchmark test set. The proposed H-PSO-ALO algorithm is compared with the original PSO and remarkable recent hybrid algorithms.

5.1. Experimental setup

The results are obtained by running the proposed algorithm for 20 independent runs. Based on the dimension of the problem from the original competition of the CEC problems [4], the maximum number of function evaluations is used.

The experiments are implemented by the Matlab2017a software on a desktop computer that has Intel Core *i*7, with a 2.80 GHz processor, 16 GB memory, Windows10 OS configuration.

5.2. First Experiment: Computationally Expensive Optimization Test Problems

The CEC 2017 benchmark problems [25], are the recent collection of 30 functions that have been categorized in four groups described in Table 1. The details of these functions are in [25]. F_i^* is the optimum global value. The search range of the variables is in the interval of $[-100,100]$. In order to verify the performance of our algorithm, four algorithms including FDR-PSO [17], CLPSO [10], HFPSO [3], MPSO [11], are employed to compared with the efficiency of H-PSO-ALO. Let us initialize the necessary parameters as follows:

- $n = 50$ maximum number of iterations;
- $m = 30$ the size of the swarm;
- $D = 30$ the dimension of a particle.

We consider two different values for the pair c_1 and c_2 as the coefficient of PSO. The results in Table 2 with $c_1 = c_2 = 1.49445$ and $c_1 = c_2 = 2$ are shown. The best algorithm in each function is in bold. In order to conduct fair comparisons among algorithms, all are done with 20 dependent runs and are calculated the mean value and standard deviation (Std) are calculated.

The H-PSO-ALO has achieved the best solution than the compared algorithms the most times.

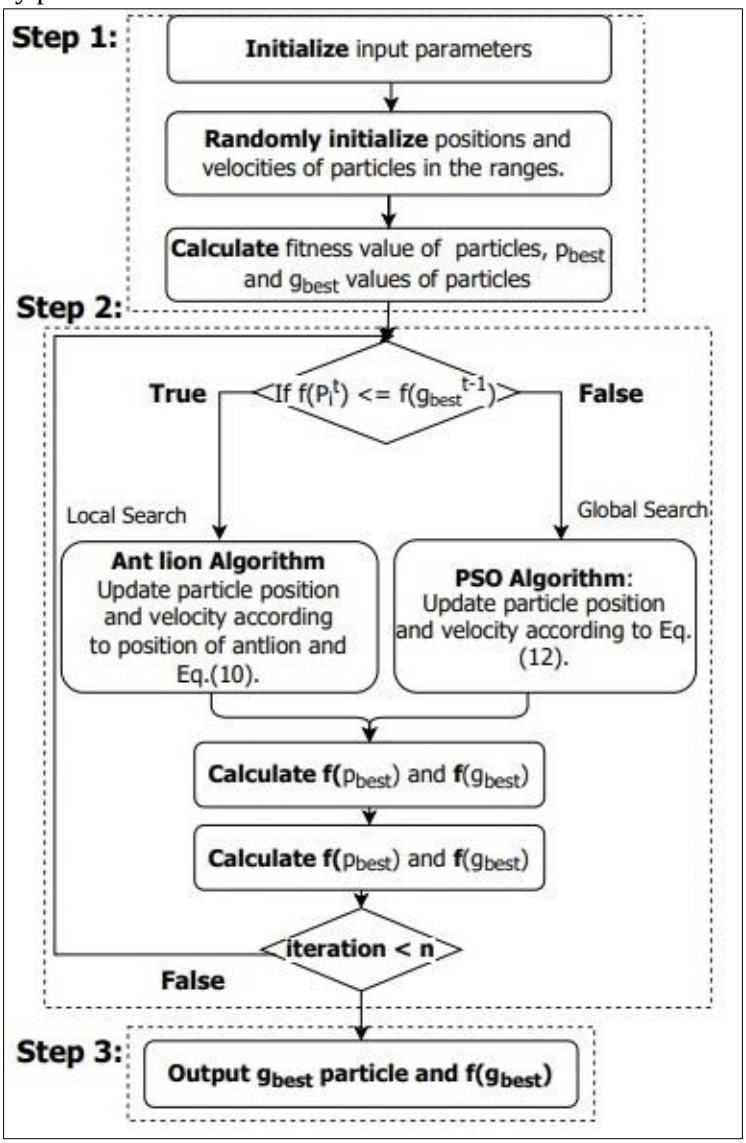

Figure 1. H-PSO-ALO Algorithm.

5.3. Convergence Performance of H-PSO-ALO

Figure 2 illustrates the convergence curves obtained using H-PSO-ALO for CEC2017. For the unimodal benchmarks like F1 and F3, the convergence function was just near a line that decreased slowly. The function F2 is not shown because it has range values different from the other functions. For the basic multi-modal benchmarks, F4 - F10, the convergence was sharp in the early stage. The curve decreases by a faster rate at first and then decreases at a shallow rate for the hybrid benchmarks. It displays the same convergence performance for the compression benchmarks, where all started with rapid convergence and then decreased slowly.

5.4. Time Complexity of H-PSO-ALO

The time complexity is determined based on the size of the population, iteration number, number of loops, and function evaluations. The running time of our hybrid algorithm can be divided into two main phases:

- Initialization of particles and velocity.
- Updating the position and velocity of particles and evaluating fitness function based on running of PSO or Ant-lion algorithms of Equation 9.

First, at the initialization phase, the most complicated computations are initializing particle position and particle velocity for each particle in swarm optimization of p particles.

Each particle is an array of D independent numbers. Thus, $O(P \times D)$ is required. In the second phase, the main computation is to update the position and velocity for each particle and evaluate of the fitness solutions at each iteration t. In PSO, the time complexity of this phase depends on the number of particles and dimensions. Thus, $O(P \times D)$ time is needed.

Table 1. Summary of CEC 2017 expensive benchmark problems.

Type	No.	Description	F_i^*
Unimodal functions	1	Shifted and Rotated(SR) Bent Cigar Function	100
	2	SR Sum of Different Power Function	200
	3	SR Zakharov Function	300
Simple Multi-modal functions	4	SR Rosenbrock's Function	400
	5	SR Rastrigin's Function	500
	6	SR Expanded Scaffer's F6 Function	600
	7	SR Lunacek Bi_Rastrigin Function	700
	8	SR NonContinuous Rastrigin's Function	800
	9	SR Levy Function	900
	10	SR Schwefel's Function	1000
Hybrid functions (HFs)	11	HF 1 (N=3)	1100
	12	HF 2 (N=3)	1200
	13	HF 3 (N=3)	1300
	14	HF 4 (N=4)	1400
	15	HF 5 (N=4)	1500
	16	HF 6 (N=4)	1600
	17	HF 6 (N=5)	1700
	18	HF 6 (N=5)	1800
	19	HF 6 (N=5)	1900
	20	HF 6 (N=6)	2000
Composition functions (CFs)	21	CF 1(N=3)	2100
	22	CF 2(N=3)	2200
	23	CF 3(N=4)	2300
	24	CF 4(N=4)	2400
	25	CF 5(N=5)	2500
	26	CF 6(N=5)	2600
	27	CF 7(N=6)	2700
	28	CF 8(N=6)	2800
	29	CF 9(N=3)	2900
	30	CF 10(N=3)	3000

In Ant-lion, the initialization of the ant-lion's position and ant's position in the population of size D, $O((antlions + ant) \times D)$ is required. Then the cost values of the ant-lions are calculated with time complexity $O\left(antlions \times D \times F\left(x\right)\right)$, where $F\left(x\right)$ represents the objective/cost function. Thus, the whole time is equal to $O(P \times D) + t \times \max\{O(P \times D), O(antlions \times O(antlions \times D \times F(x))\}$.

5.5. Second Experiment: Medical and Non-Medical Benchmark Problems

The performance of our proposed method has been evaluated with six medical datasets of several diseases such as breast cancer, heart and Parkinson's diseases, and one non-medical dataset, which is the ionosphere dataset. The datasets were picked from the online datasets from the University of California [2]. The details of datasets are shown in Table 4.

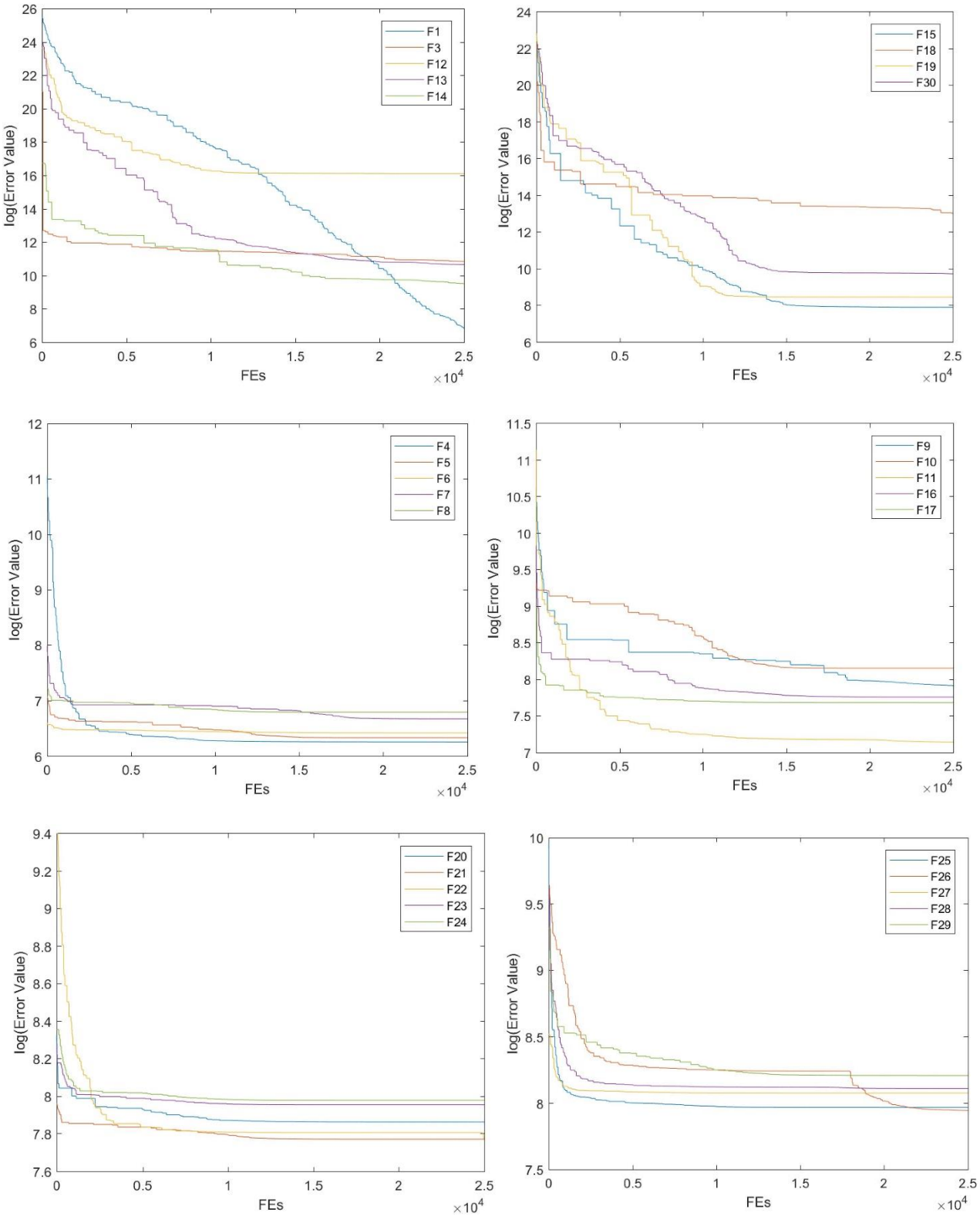

Figure 2. Convergence of H-PSO-ALO.

Table 2. Comparisons of results between H-PSO-ALO and some well-known variants of PSO.

		d=30 , c1=c2=1.49445						d=30 , c1=c2=2					
		PSO	**FDR PSO**	**CLPSO**	**HFPSO**	**MPSO**	**HPSOALO**	**PSO**	**FDR PSO**	**CLPSO**	**HFPSO**	**MPSO**	**HPSOALO**
F1	**Mean**	1.02E+10	4.25E+10	5.98E+10	1.28E+09	4.74E+10	**9.03E+08**	1.13E+10	4.21E+10	6.14E+10	1.51E+09	1.4E+10	**1.24E+09**
	Std	5.99E+09	1.61E+09	3.17E+09	7.19E+08	8.34E+09	**2.78E+08**	6.82E+09	1.23E+09	3.5E+09	5.62E+08	4.37E+09	**1.71E+08**
	Rank	3	4	6	2	5	1	3	5	6	2	4	1
F2	**Mean**	3.44E+36	1.71E+53	3.14E+56	1.34E+31	3.54E+50	**4.18E+30**	5.03E+36	1.59E+53	3.61E+56	1.13E+32	3.59E+37	**4.38E+28**
	Std	1.34E+37	2.03E+53	3.16E+56	5.26E+31	1.58E+51	**1.75E+31**	1.42E+37	8.49E+52	5.36E+56	3.61E+32	2.38E+38	**8.24E+28**
	Rank	3	5	6	2	4	1	3	5	6	2	4	1
F3	**Mean**	205039	**82670.54**	84445.45	135576.4	90065.97	106865.5	255551.3	80413.82	83077.71	131330.1	**78282.78**	111120.4
	Std	62460.7	**5316.98**	5974.059	43553.58	17915.17	28030.08	66836.03	5634.689	5589.741	33028.7	**11816.87**	29081.48
	Rank	6	1	2	5	3	4	6	2	3	5	1	4
F4	**Mean**	1773.982	12743.63	20625.38	687.2697	13958.35	**639.8918**	1566.584	12663.45	20364.86	700.9375	3126.429	**666.5102**
	Std	1111.238	474.7249	1727.459	129.0348	3526.408	**51.18574**	885.2278	444.0191	1966.419	120.5262	1099.91	**45.95006**
	Rank	3	4	6	2	5	1	3	5	6	2	4	1
F5	**Mean**	760.2078	789.3726	928.043	742.446	827.443	**720.1848**	799.778	**783.7056**	937.9646	795.3654	850.0272	785.739
	Std	65.628	26.85221	17.96531	30.95003	42.00897	**38.6668**	62.00302	**19.35728**	21.07162	39.88955	31.95861	27.90379
	Rank	3	4	6	2	5	1	4	1	6	3	5	2
F6	**Mean**	655.2522	674.1168	694.3883	652.7197	682.7383	**651.2227**	656.4102	676.6139	694.264	658.6124	683.8369	**650.04**
	Std	13.13051	4.882955	3.965887	15.5557	6.917219	**10.79812**	14.41486	5.614786	3.517341	17.9756	7.439114	**11.18119**
	Rank	3	4	6	2	5	1	2	4	6	3	5	1
F7	**Mean**	1391.861	1214.834	1379.772	1049.68	1316.497	**1027.404**	1393.357	1211.671	1396.579	1043.529	1155.033	1031.754
	Std	170.6398	51.75003	50.58902	33.39304	85.19652	**40.23209**	143.2078	35.29433	44.60647	31.53745	44.89709	31.7038
	Rank	6	3	5	2	4	1	5	4	6	2	3	1
F8	**Mean**	1022.841	997.5378	1145.57	1030.01	1066.976	1011.082	1088.01	999.8153	1139.143	1079.967	1076.847	1058.519
	Std	45.38091	15.68411	21.98358	32.9541	40.86379	34.07757	49.73875	14.65874	18.09438	27.61957	34.10152	29.5336
	Rank	3	1	6	4	5	2	5	1	6	4	3	2
F9	**Mean**	7057.4	11732.89	11030.92	9154.337	11735.49	4589.885	14244.95	11410.69	11188.63	8094.403	10298.36	4995.3
	Std	2604.79	1765.083	1103.061	3122.315	1849.619	2567.473	3547.821	1302.907	1092.743	3402.148	1750.864	2618.249
	Rank	2	5	4	3	6	1	6	5	4	2	3	1
F10	**Mean**	6236.709	5509.835	8922.614	7751.466	8313.647	7937.24	8178.814	5827.461	9056.76	8736.549	8610.6	8801.098
	Std	818.5472	575.4691	445.257	930.6287	1160.507	809.3621	1090.725	648.6855	410.7084	536.7545	669.4898	764.4995
	Rank	2	1	6	3	5	4	2	1	6	4	3	5
F11	**Mean**	5102.869	6497.095	136479.4	2450.983	6628.874	**2427.676**	10235.64	6428.993	305187.7	2622.76	3916.683	**2494.172**
	Std	3279.102	363.2002	301593.5	570.439	2121.378	**584.4686**	4717.295	508.8522	671815.6	582.7713	1165.39	**566.0884**
	Rank	3	4	6	2	5	**1**	5	4	6	2	3	**1**
F12	**Mean**	4.73E+08	1.25E+10	1.93E+10	85425406	1.08E+10	**73889966**	2.7E+08	1.25E+10	1.91E+10	1.13E+08	1.68E+09	1.2E+08
	Std	3.42E+08	3.7E+08	1.52E+09	63459428	1.95E+09	**48846100**	3.48E+08	5.02E+08	1.14E+09	72909193	9.51E+08	49367720
	Rank	3	5	6	2	4	**1**	3	5	6	1	4	2
F13	**Mean**	1.19E+08	1.75E+10	2.66E+10	4609252	9.07E+09	**2653748**	2.9E+08	1.74E+10	2.69E+10	23256817	3.29E+08	**21539459**
	Std	4.65E+08	2.96E+08	2.3E+09	15646178	4.14E+09	**4878646**	6.07E+08	2.7E+08	2.16E+09	14133287	5.6E+08	**14550969**
	Rank	3	5	6	2	4	**1**	3	5	6	2	4	**1**
F14	**Mean**	1972899	63749533	1.31E+08	491873.1	1555230	**476431.1**	2368971	63654872	1.23E+08	1079637	1024428	**884217.2**
	Std	2058131	2941555	41704226	464589.9	977195.4	**385252.2**	2868024	2266010	30367687	985175.1	658369.7	**923176.8**
	Rank	4	5	6	2	3	**1**	4	5	6	3	2	**1**
F15	**Mean**	**42745.65**	9.98E+08	1.72E+09	91478.3	1.8E+08	173297.3	66240618	9.99E+08	1.75E+09	1433240	**140702.5**	2333778
	Std	**28849.4**	2358068	4.4E+08	77158.93	1.74E+08	185461.6	2.96E+08	5684906	3.76E+08	1292324	**156376.2**	1944578
	Rank	**1**	5	6	2	4	3	4	5	6	2	**1**	3
F16	**Mean**	3428.331	9715.535	13858.2	3171.551	4729.965	**3095.398**	**3431.74**	9661.178	14325.09	3480.908	3825.351	3627.699
	Std	428.9392	584.1997	1188.298	354.4819	931.6423	**282.6888**	**474.1535**	261.5079	1019.033	377.138	389.3033	249.9368
	Rank	3	5	6	2	4	**1**	**1**	5	6	2	4	3
F17	**Mean**	2560.747	11251.24	32593.77	2292.292	3807.927	**2186.595**	2688.919	11342.22	28413.88	**2350.835**	2706.306	2375.578
	Std	246.4254	230.6423	12976.21	175.9769	2390.208	**241.7091**	328.7552	358.4232	12162.37	**217.2066**	280.9066	215.3791
	Rank	3	5	6	2	4	**1**	3	5	6	**1**	4	2
F18	**Mean**	6815657	74475119	7.21E+08	**2218129**	17408741	3510389	13568531	83564075	7.25E+08	3860892	4082552	**2864227**
	Std	8787333	10442881	2.61E+08	**1435481**	20144601	2743044	19204133	18149861	2.23E+08	2862359	2713139	**2121259**
	Rank	3	5	6	**1**	4	2	4	5	6	2	3	**1**
F19	**Mean**	11280651	1.26E+09	1.86E+09	**905206.7**	3.38E+08	1240324	6328843	1.26E+09	1.84E+09	5412352	**1351710**	5683319
	Std	40646441	9486212	3.48E+08	**845184.1**	3.59E+08	1069136	19226154	10335321	2.65E+08	4106184	**1449497**	3134869

	Rank	3	5	6	**1**	4	2	4	5	6	2	**1**	3
	Mean	2825.358	3015.796	3424.193	**2680.067**	2905.929	2765.396	2833.115	3008.658	3376.693	2768.313	2918.021	**2735.148**
F20	**Std**	261.1196	191.79	144.6556	**178.9804**	249.2348	178.0448	169.2552	171.8421	152.0247	204.1589	241.1957	**224.0068**
	Rank	3	5	6	**1**	4	2	3	5	6	2	4	**1**
	Mean	2528.987	2696.989	2876.068	2510.386	2664.29	**2503.836**	2588.27	2686.053	2873.751	2558.162	2607.647	**2557.538**
F21	**Std**	46.42121	27.6325	26.4329	35.83463	54.34697	**31.66954**	48.79353	24.64586	37.83682	31.46401	43.40994	**37.67764**
	Rank	3	5	6	2	4	**1**	3	5	6	2	4	**1**
	Mean	7074.836	7804.949	9802.406	6155.39	8365.275	**3334.993**	8176.796	7901.928	10068.07	6724.62	6073.11	**4999.791**
F22	**Std**	1953.407	310.5139	418.0474	3250.872	1211.161	**2053.506**	2608.566	336.8772	190.9968	3844.608	2339.989	**3391.72**
	Rank	3	4	6	2	5	**1**	5	4	6	3	2	**1**
	Mean	3015.15	572.202	5785.278	3000.831	3417.382	**2969.031**	**2958.698**	4559.393	5749.869	3025.545	3330.235	3016.853
F23	**Std**	96.75702	137.2579	159.8225	65.19379	134.8441	**56.37902**	**88.21256**	169.3812	148.0487	61.73343	111.6255	47.61335
	Rank	3	5	6	2	4	**1**	**1**	5	6	3	4	2
	Mean	3186.199	4472.14	4840.48	3181.648	3789.993	**3102.507**	3162.844	4431.486	4845.578	3186.276	3499.345	**3135.884**
F24	**Std**	80.34918	68.97408	43.70668	102.1247	261.1971	**66.92142**	96.14655	68.50662	59.42256	72.23671	114.8304	**35.99461**
	Rank	3	5	6	2	4	**1**	2	5	6	3	4	**1**
	Mean	3605.772	4003.548	5560.659	3053.135	4727.501	**3028.429**	3378.824	4004.536	5564.24	3069.17	3299.793	**3040.915**
F25	**Std**	423.872	64.65928	387.3282	50.54137	466.7101	**32.65372**	422.1932	59.41275	289.7106	51.06324	96.78545	**36.86505**
	Rank	3	4	6	2	5	**1**	4	5	6	2	3	**1**
	Mean	7364.933	10032.05	12294.51	**5248.453**	10447.14	5745.675	7034.143	9960.933	12348.9	5770.924	8213.369	**5409.328**
F26	**Std**	576.3024	342.3644	443.4109	**1718.761**	886.7155	1527.303	1197.84	304.3256	473.8823	1698.488	694.1069	**1727.195**
	Rank	3	4	6	**1**	5	2	3	5	6	2	4	**1**
	Mean	3465.28	6167.628	7853.255	**3329.254**	4363.094	3336.397	**3295.97**	6111.956	7670.81	3319.327	3842.12	3314.335
F27	**Std**	99.90318	285.6769	231.2631	**49.51893**	417.9529	75.10384	**41.1521**	432.7423	305.4114	55.96448	239.5948	51.1967
	Rank	3	5	6	**1**	4	2	**1**	5	6	3	4	2
	Mean	4305.346	6115.89	7868.751	3461.199	6746.598	**3421.05**	4238.78	6153.197	7783.812	3436.862	4454.025	**3431.183**
F28	**Std**	476.0519	107.8444	270.9854	72.24356	665.7676	**63.74429**	908.0946	165.1483	362.8457	42.74948	372.9678	**36.36413**
	Rank	3	4	6	2	5	**1**	3	5	6	2	4	**1**
	Mean	4721.668	10599.61	23998.81	4568.132	6794.18	**4361.444**	4580.929	10826.95	26111.92	4593.393	5599.944	**4483.123**
F29	**Std**	290.2042	528.2792	4955.154	340.8879	1173.711	**175.9411**	412.9637	754.5912	7633.618	280.0136	476.3492	**192.6005**
	Rank	3	5	6	2	4	**1**	2	5	6	3	4	**1**
	Mean	5324565	3.68E+09	5.24E+09	**5225473**	1.14E+09	8854415	**2307312**	3.69E+09	5.48E+09	9327137	22188875	11582865
F30	**Std**	4185326	1.62E+08	5.16E+08	**4320101**	8.85E+08	7529459	**2936028**	1.83E+08	5.22E+08	6813082	43055004	5327403
	Rank	2	5	6	**1**	4	3	**1**	5	6	2	4	3

Table 3. Initialize input parameters of HPSOALO algorithm.

Parameters	Description
n=50	maximum number of iterations
m=10,30	population size
D=10,30	D-dimensional
c1, c2= 1.49445	accelerating factors
EFs=500, 1500	maximum number of evaluation function($m \times D$)

Table 4. Information of datasets.

Num	Name	Instance	Feature
1	WDBC	569	31
2	WPBC	198	34
3	Caimbra 2018	116	10
4	Breast cancer	699	10
5	Heart	270	14
6	Parkinson	195	23
7	Ionosphere	351	35

Table 5. Experimental results of average of 20 independents runs.

Data	Swarm	Algorithm	Accuracy	Persian	Recall	F-measure	Num of feature
WDBC	4*30	ALO	97.06	93.78	98.33	95.92	5
		PSO	98.04	95.64	99.11	97.29	10
		HfPSO	98.04	95.49	99.25	97.27	13
		HPSOALO	**98.16**	**95.65**	**99.41**	**97.44**	11
	4*10	ALO	96.85	93.65	97.89	95..62	5
		PSO	97.86	95.21	99.06	97.04	12
		HfPSO	97.72	94.72	98.98	96.72	15
		HPSOALO	**97.93**	**95.30**	**99.16**	**97.13**	11
WPBC	4*30	ALO	78.80	93.31	81.52	86.91	5
		PSO	**82.85**	**96.13**	**84.06**	**89.54**	4
		HfPSO	82.14	95.38	83.81	89.08	6
		HPSOALO	82.21	95.60	83.7	89.13	7
	4*10	ALO	78.60	93.32	81.52	86.91	14
		PSO	82.38	96.05	83.64	89.28	6
		HfPSO	82.14	95.37	83.80	89.08	8
		HPSOALO	**82.96**	**95.91**	**84.28**	**89.58**	7
Caimbra 2018	4*30	ALO	76.33	74.55	74.86	73.21	3
		PSO	84.51	79.85	86.53	81.87	4
		HfPSO	85.64	**80.85**	88.10	82.62	4
		HPSOALO	**86.71**	80.38	**90.53**	**83.92**	4
	4*10	ALO	76.23	74.15	74.93	72.81	1
		PSO	84.73	**80.50**	86.51	82.21	4
		HfPSO	84.05	78.58	86.99	80.95	3
		HPSOALO	**85.04**	79.23	**87.91**	**81.88**	4
Breast cancer	4*30	ALO	96.91	97.24	98.08	97.63	2
		PSO	97.67	97.70	**98.76**	98.21	5
		HfPSO	97.4	97.67	98.73	98.18	5
		HPSOALO	**97.68**	**97.77**	98.70	**98.22**	5
	4*10	ALO	96.64	97	97.91	97.42	1
		PSO	97.50	97.45	98.74	98.07	5
		HfPSO	97.51	**97.56**	98.66	98.08	5
		HPSOALO	**97.54**	97.49	**98.78**	**98.11**	**4**
heart	4*30	ALO	82.39	86	83.48	84.41	2
		PSO	86.41	90.33	86.39	88.08	8
		HfPSO	86.46	90.53	86.31	88.11	6
		HPSOALO	**86.72**	**91.03**	**86.40**	**88.39**	4
	4*10	ALO	81.93	86	82.91	84.06	2
		PSO	85.15	89.13	85.39	86.91	6
		HfPSO	85.46	89.23	85.91	87.21	6
		HPSOALO	**85.93**	**89.83**	**86.09**	**87.62**	6
Parkinson	4*30	ALO	94.01	85.63	91.18	87.16	4
		PSO	97.20	91.75	97.27	93.84	7
		HfPSO	97.31	93.33	96.38	94.29	8
		HPSOALO	**97.53**	93.35	97.15	94.69	**6**
	4*10	ALO	93.08	85.45	88.82	85.77	7
		PSO	96.57	90.83	95.77	92.53	9
		HfPSO	96.84	91.45	96.40	93.22	7
		HPSOALO	**96.89**	91.08	96.68	93.24	6
Ionosphere	4*30	ALO	87.59	68.82	95.51	79.34	1
		PSO	91.62	79.55	96.81	86.84	3
		HfPSO	91.31	78.63	96.74	86.18	4
		HPSOALO	**92.22**	**80.99**	**97.12**	**87.84**	3
	4*10	ALO	87.09	67.38	95.57	78.15	5
		PSO	90.28	75.8	96.64	84.38	2
		HfPSO	90.48	76.26	96.78	84.76	3
		HPSOALO	**91.20**	**78.30**	**96.77**	**86.08**	2

In addition, different benchmark datasets of breast cancer disease are used in this work. On all datasets, the four algorithms PSO, ALO, HFPSO, and H-PSO-ALO, with different population sizes and dimensions, have been performed. The parameter settings are shown in Table 3.

First, the datasets are normalized, as follows:

$$N(x)=\frac{(x-x_{min})}{(x_{max}-x_{min})}, \quad (12)$$

where x is an original value, and x_{min} and x_{max} are the minimum and the maximum values of the feature dataset.

Then four algorithms perform in order to predict which were applied in terms of the measurement metrics of accuracy, specificity, recall, and F-measure for the classification of data. The proposed method finds the best feature with a higher accuracy performance which objective function for the classifier is the KNN algorithm.

The KNN algorithm was done in 10-fold cross-validation; 90% of the dataset was used for train and 10% for the test dataset. Regarding the size of the selected features, the average selected features are 20 runs for all algorithms. The new hybrid algorithm obtains the near-global optimum as excellent superiority, and predicts effectiveness in the classification problems. Moreover, the experimental results illustrate that the hybrid algorithm presents a balancing of the exploration, and exploitation and is a suitable methodology.

6. Conclusion

In this work, a new hybrid method for seeking the global solution was defined by combining the PSO and ant-lion algorithms. Thanks to the more effective balance between exploration and exploitation, the proposed algorithm has a convergence success in the optimization problems. The new hybrid algorithm was applied to solve the CEC 2017 benchmark problems and the seven optimization problems from UCI. The results obtained demonstrated better search ability with a combination of search strategies of both algorithms for different functions and real-world classification problems.

Acknowledgment

The work was supported by the Kosar University of Bojnord with the grant number "9704161688".

References

[1] E. Ali, S. Abd-Elazim, and A. Abdelaziz, "Ant lion optimization algorithm for renewable distributed generations". *Energy*, vol. 116, pp. 445–458, 2016.

[2] Asuncion, A. and Newman, D. Uci machine learning repository, 2017.

[3] I.B. Aydilek, "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems". *Applied Soft Computing*, vol. 66, pp. 232–249, 2018.

[4] W.-N. Chen, and D.-Z. Tan, "Set-based discrete particle swarm optimization and its applications: a survey". *Frontiers of Computer Science*, vol.12, no.2, pp.203–216, 2018.

[5] J. Ding, Q. Wang, Q. Zhang, Q. Ye, and Y. Ma, "A hybrid particle swarm optimization-cuckoo search algorithm and its engineering applications," *Mathematical Problems in Engineering*, 2019.

[6] S.-K. S. Fan, and C.-H. Jen, "An enhanced partial search to particle swarm optimization for unconstrained optimization". *Mathematics*, vol. 7, no. 4, pp. 357, 2019.

[7] M.M. Iqbal, and R. J. Xavier, "Development of optimal reduced-order model for gas turbine power plants using particle swarm optimization technique," *International Transactions on Electrical Energy Systems*, vol. 30, no. 4, 2020.

[8] J. Kennedy, and R. Eberhart, "Particle swarm optimization," In Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, pp. 1942–1948, 1995.

[9] S. Khunkitti, A. Siritaratiwat, S. Premrudeepreechacharn, R. Chatthaworn, and N. R. Watson, "A hybrid da-pso optimization algorithm for multiobjective optimal power flow problems". *Energies* vol. 11, no. 9, pp. 2270, 2018.

[10] J.J. Liang, A.K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE transactions on evolutionary computation*, vol. 10, no. 3, pp. 281–295, 2006.

[11] H. Liu, X.-W. Zhang, and L.-P. Tu, "A modified particle swarm optimization using adaptive strategy". *Expert Systems with Applications*, vol. 152, 2020.

[12] M.M. Mafarja, and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection". *Neurocomputing*, vol. 260, pp. 302–312, 2017.

[13] S. Mirjalili, "The ant lion optimizer". *Advances in engineering software*, vol. 83, pp. 80–98, 2015.

[14] P. Moradi, and M. Gholampour, "A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy," *Applied Soft Computing*, vol. 43, pp. 117–130, 2016.

[15] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, pp. 1424–1437, 2004.

[16] F. Omidinasab, and V. Goodarzimehr, "A hybrid particle swarm optimization and genetic algorithm for truss structures with discrete variables," *Journal of Applied and Computational Mechanics*, vol. 6, no. 3, pp. 593–604, 2020.

[17] Peram, T., Veeramachaneni, K., and Mohan, C.K. "Fitness-distance-ratio based particle swarm

optimization. In Proceedings of the 2003 IEEE Swarm Intelligence Symposium". SIS'03 (Cat. No. 03EX706) (2003), IEEE, pp. 174–181.

[18] N. Qu, J. Chen, J. Zuo, and J. Liu, "Pso–som neural network algorithm for series arc fault detection". *Advances in Mathematical Physics* ,2020.

[19] E. Salehpour, J. Vahidi, and H. Hossinzadeh, "Solving optimal control problems by pso-svm". *Computational Methods for Differential Equations*, vol. 6, no. 3, pp. 312–325, 2018.

[20] Singh, N. and Singh, S. "Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance," *Journal of Applied Mathematics*, 2017.

[21] N. Sunil, R. Ganesan, and B. Sankaragomathi," Analysis of osa syndrome from ppg signal using cart-pso classifier with time domain and frequency domain features," *Computer Modeling in Engineering & Sciences* vol. 118, no. 2, pp. 351–375, 2019.

[22] O. Tarkhaneh, and H. Shen, "Training of feedforward neural networks for data classification using hybrid particle swarm optimization," *mantegna lévy flight and neighborhood search.*, vol. 5, no. 4, 2019.

[23] V.R. VC, et al. "Ant-lion optimization algorithm for optimal sizing of renewable energy resources for loss reduction in distribution systems," *Journal of Electrical Systems and Information Technology*, vol.5, no. 3, pp. 663–680, 2018.

[24] L. Wang, X. Liu, M. Sun, and J. Qu, "An extended clustering membrane system based on particle swarm optimization and cell-like p system with active membranes," *Mathematical Problems in Engineering*, 2020.

[25] A. Wan, L. Jiang, C. S. Sangeeth, and C.A. Nijhuis, "Reversible soft top-contacts to yield molecular junctions with precise and reproducible electrical characteristics, " *Advanced Functional Materials*, vol. 24, no. 28, pp. 4442–4456, 2014.

[26] X. Xu, H. Rong, M. Trovati, M. Liptrott, and N. Bessis, "Cs-pso: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems, " *Soft Computing*, vol. 22, no.3, pp. 783–795, 2018.

[29] S. Hosseinirad, "Multi-layer Clustering Topology Design in Densely Deployed Wireless Sensor Network using Evolutionary Algorithms, " *Journal of AI and Data Mining*, vol.6, no.2, pp. 297-311, 2018.

ترکیب الگوریتم بهینه سازی ازدحام ذرات ترکیبی با بهینه سازی شیر مورچه‌ای: نتایج در معیارها و کاربردها

زینب حسنی[۱،*] و محسن علمبردار میبدی[۲]

[۱] دانشکده علوم پایه و فنی، دانشگاه کوثر بجنورد، ایران.

[۲] دانشکده ریاضی و آمار، دانشگاه اصفهان .

ارسال ۲۰۲۱/۰۲/۱۶؛ بازنگری ۲۰۲۱/۰۵/۰۷؛ پذیرش ۲۰۲۱/۰۹/۱۹

چکیده:

یک مسئله اساسی در الگوریتم بهینه‌سازی ازدحام ذرات (PSO) این است که ممکن است در بهینه‌ی محلی متوقف شود. در این مقاله، یک مدل ترکیبی جدید بر اساس ترکیب الگوریتم PSO و الگوریتم بهینه سازی شیرمورچه (ALO)، به نام H-PSO-ALO برای حل این مسئله پیشنهاد شده است. روش پیشنهادی از استراتژی جستجوی محلی با استفاده از الگوریتم بهینه‌سازی شیرمورچه برای انتخاب زیر مجموعه‌ای ویژگی‌های کمتر بهم مرتبط و شاخص استفاده شده است. هدف بهبود دقت پیش بینی و سازگاری مدل در مجموعه داده‌های مختلف با ایجاد تعادل بین فرایندهای اکتشاف و بهره برداری در الگوریتم بهینه‌سازی است. عملکرد روش پیشنهادی روی ۳۰ تابع معیار CEC 2017 و برخی مجموعه داده‌های مشهور ارزیابی شده است. علاوه بر این، کارایی الگوریتم پیشنهادی H-PSO-ALO با چهار الگوریتم FDR-PSO، CLPSO، HFPSO، MPSO برای ارزیابی عملکرد مقایسه شده است. با توجه به نتایج حاصل شده، روش پیشنهادی در بسیاری از موارد از دیگر روش‌ها بهتر بوده است، بنابراین روش پیشنهادی برای حل مسائل بهینه‌سازی در مجموعه داده‌های دنیای واقعی انتخاب مناسبی است.

کلمات کلیدی: الگوریتم ترکیبی بهینه سازی، بهینه سازی ازدحام ذرات، بهینه سازی شیرمورچه، K-نزدیکترین همسایه.