International Journal of Hybrid Information Technology
Vol. 7, No. 6 (2014), pp. 83-100
http://dx.doi.org/10.14257/ijhit.2014.7.6.07

A New Efficient Meta-Heuristic Optimization Algorithm Inspired by Wild Dog Packs

Essam Al Daoud[1], Rafat Alshorman[1] and Feras Hanandeh[2]

[1]*Computer Science Department, Zarqa University*
Zarqa, Jordan
[2]*Department of Computer Information Systems, Faculty of Prince Al-Hussein Bin Abdallah II For Information Technology, Hashemite University, Jordan.*
essamdz@zu.edu.jo; Rafat_sh@zu.edu.jo, *Feras@hu.edu.jo*

Abstract

Although meta-heuristic optimization algorithms have been used to solve many optimization problems, they still suffer from two main difficulties: What are the best parameters for a particular problem? How do we escape from the local optima? In this paper, a new, efficient meta-heuristic optimization algorithm inspired by wild dog packs is proposed. The main idea involves using three self-competitive parameters that are similar to the smell strength. The parameters are used to control the movement of the alpha dogs and, consequently, the movement of the whole pack. The rest of the pack is used to explore the neighboring area of the alpha dog, while the hoo procedure is used to escape from the local optima. The suggested method is applied to several unimodal and multimodal benchmark problems and is compared to five modern meta-heuristic algorithms. The experimental results show that the new algorithm outperforms other peer algorithms.

Keywords: *Wild dog packs, Particle swarm optimization, Harmony search, Optimization, meta-heuristic*

1. Introduction

Meta-heuristic algorithms have been successfully applied to unimodal, multimodal, continuous, and discontinuous functions. One major advantage of meta-heuristic approaches over the classical and derivative-based numerical methods is that they do not require differentiable objective functions or any condition being placed on the objective function [1]. During the last few decades, several meta-heuristic algorithms have been suggested such as simulated annealing, hill climbing, the tabu search, genetic programming, the genetic algorithm, the paddy field algorithm, ant colony optimization, particle swarm optimization (PSO), the bee colony algorithm, river formation dynamics, intelligent water drops, the gravitational search algorithm, the electromagnetism algorithm, bacterial foraging optimization, and the harmony search (HS) [2, 3]. On the other hand, meta-heuristic algorithms have been successfully applied to many real-world applications such as scheduling, vehicle routing, engineering design, the maximum clique problem, the p-median problem, image processing, 2D regular and irregular strip packing, pickup and delivery problems, and structure optimization [4–9]. In recent years, the optimization problems have become very complex and finding a global solution has become more challenging, especially for multimodal, hybrid, and combinatorial problems. Therefore, more effective meta-heuristic approaches are always needed.

ISSN: 1738-9968 IJHIT

The rest of this paper is organized as follows. Section 2 introduces the related work and focuses on two relatively new and important groups of meta-heuristic algorithms and their variants; namely, PSO and the HS. Section 3 illustrates the natural behavior of the wild dog pack and outlines the main three strategies that will be exploited in this paper. Section 4 converts the strategies of the wild dog pack into programmable steps and highlights the role of the alpha dogs. Section 4 describes a set of widely used test functions, which has several features such as regularity, multimodality, continuity, reparability, and difficulty. Section 5 presents the numerical results and compares the wild dog pack optimization (WDPO) algorithm against five relatively new meta-heuristic algorithms. Conclusions are drawn in Section 6.

2. Related Work

In the literature, most of the meta-heuristic algorithms have been derived from life: from physics or nature. For example, evolution algorithms, swarm-based algorithms, and ecology algorithms are three families inspired by bio behavior, which include genetic programming, the genetic algorithm, the paddy field algorithm, ant colony optimization, PSO, and the bee colony algorithm [10]. Algorithms derived from physics are those such as simulated annealing and the electromagnetism algorithm [11]. However, we will discuss here two relatively new and popular algorithms and their variants; namely, PSO and the HS.

PSO is inspired by the social behaviors of animals and insects such as bird flocking, insect swarming, fish schooling, etc. The first version was proposed by Kenned and Eberhart in 1995 [12]. The standard PSO (SPSO) is a modified version of PSO and was suggested by Shi and Eberhart [13]. In SPSO, the solution vector (the position vector) x_i and the velocity vector v_i are updated in the iteration t as follows:

$$v_i(t+1) = wv_i(t) + c_1 r_1(p_i - x_i(t)) + c_2 r_1(g - x_i(t))$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

where w, c_1, and c_2 are constants, r_1 and r_2 are uniform random numbers in the range [0,1], g is the global best position that is discovered by the whole swarm, and p_i is the best position of the i^{th} particle. Clerc and Kennedy modified the velocity equation by multiplying the previous velocity by a constriction factor χ as follows [14]:

$$\chi = 2k / (|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|)$$

$$v_i(t+1) = \chi(v_i(t) + \varphi_1 rand()(p_i - x_i(t)) + \varphi_2 rand()(g - x_i(t)))$$

where $\varphi > 4$, $\varphi = \varphi_1 + \varphi_2$, and k is in the range [0, 1]. Two main problems faced the early versions of PSO; namely, slow convergence and stagnation of the population. Thus, several variants have been suggested in the last few years to solve these problems. PSO variants can be divided into three main categories. The first uses adaptive strategies for selecting appropriate parameters [15–19]. The second involves improving the topological structure of the population, or organizing and clustering the population [20, 21]. The third category involves detecting the stagnation and jumping out from the local optima, such as with regrouping particle swarm optimization (RegPSO) [22, 23]. However, these methods do not introduce a suitable connection between the parameters and the fitness function; moreover, avoiding the local optima and parameter selection are not flexible enough.

Another popular meta-heuristic optimizer is the HS, which was introduced by Geem *et al.* in 2001 [24, 25]. The main steps in the HS are constructing a new vector from the previous vectors and replacing the worst one. After initializing the harmony memory, the HS algorithm can be described as follows:

Repeat until termination condition is fulfilled
1- for each component *i* do
if $HMCR \geq rand$
$x_{new}^{i} = x_{j}^{i}$
if $PAR \geq rand$
$x_{new}^{i} = x_{new}^{i} \pm rand \times bw$
else
$x_{new}^{i} = rand$
2- if the new vector is better than the worst, replace the worst vector

where $j \in 1$ is the size of the harmony memory (*HMS*), *HMCR* is the harmony memory considering rate, *PAR* is the pitch adjusting rate, and *bw* is the bandwidth. Mahdavi *et al.* introduced an improved version of the HS where the *bw* and the *PAR* are updated as follows [26]:

$$bw(t) = bw_{\max} e^{\left(\frac{h}{MaxIter} iter\right)}$$

where

$$h = \ln\left(\frac{bw_{\min}}{bw_{\max}}\right)$$

and

$$PAR(t) = PAR_{\min} + \frac{PAR_{\max} - PAR_{\min}}{MaxIter} iter$$

MaxIter is the maximum number of iterations, bw_{min} and bw_{max} are the minimum and maximum bandwidths, PAR_{min} and PAR_{max} are the minimum and maximum *PARs*. Another development was introduced by Omran and Mahdavi, where a global best pitch was used to enhance the i^{th} component in the pitch adjustment step instead of the random bandwidth [27]. Wang and Huang updated the pitch adjustment by removing *bw* and using the maximum and the minimum values of the harmony memory [28]. Most of the other HS variants attempt to find a dynamic solution for parameter selection [29–36]. However, the same situation arises as for PSO, as there is no conscious connection between the selection of the parameters and the progress in the fitness function.

3. Wild Dog Packs

Wild dog packs (also referred to as *Lycaon pictus*) are the most efficient hunters from among all of the canines. The percentage of successful hunts is nearly 80, while the success rate of lions is only 30%. Their high hunting success rate is due to their especially cooperative

and social behavior. Every morning the adult wild dogs will group together and go hunting. During the hunts, they use high-pitched squeaks to indicate directions, and chase the prey for 3 to 5 km, tirelessly until exhausting it. The territory of a wild dog pack is about 900 square miles. Each pack consists of alpha dogs and between 10 to 30 dogs from several generations. Alpha dogs are the dominant male and female; one of them is usually more dominant. The dominant alpha is in charge of making decisions such as the pack territory, the den location, the hunting time, and directions. The other pack members usually follow the alpha dogs and avoid conflict. Unlike other canine species, there is a strict ranking system, the rank being determined by posturing, and they do not act aggressively—they do not even fight over food. Instead of fighting, the hunter dogs will bring a share of meat to the sick and injured dogs. In some cases, a non-alpha dog will temporarily lead the dogs to the prey. If one of the alpha dogs dies, then the oldest member will gain the alpha status over the others. Alpha dogs use smell to make decisions and to lead the pack. They can smell several locations or directions, and then they select a suitable one [36–37]. By using smell, the alpha dogs can interpret the surrounding environment. The part of the dog's brain that is devoted to analyzing smell is about 40 times larger than the equivalent part in humans, and they can identify any smell one million times better than humans can. Actually, no sniffing device can compete with the precision of the dog's nose. Another wonderful character that can be noted in the wild dog pack is their communication method. They can communicate by using body language, sound, and scent. They use low-pitched and high-pitched sounds, deep barks, growls, and a bell-like "hoo" sound to call the other members of the pack [38]. In this study, we will exploit three strategies that can be described as follows:

1- The alpha decision: each time the dominant alpha dog will smell several locations, evaluate them, adopt the best one, and adjust the direction.
2- The pack decision: the other dogs follow the dominant alpha dog. In some cases, a non-alpha dog well temporarily lead the dogs, or has the alpha status over the others.
3- Hoo call: any dog can call the other members with a bell-like "hoo" sound.

4. Wild Dog Pack Optimization

Inspired by the three strategies listed at end of the previous section, WDPO consists of three main steps. The first is the alpha decision, which will be used for exploiting good locations (related to intensification), the second is the pack decision, which will be used to explore the search area (related to diversification), and the third is the "hoo" call, which can be used to escape from the local optima when there is no improvement. Three self-adaptive parameters are used that play an important role in controlling the alpha's movement. These parameters can be considered as the smell strength, such that, if the smell becomes stronger, then the search area and the alpha dog's movement becomes smaller, and the target closer. Although in this study only three parameters are used to control the movement, it can be generalized to any number. In the alpha decision step, we will allow the alpha dog to have about 50% of the pack decisions, which means that the number for the function evaluation accomplished by the alpha dog is nearly equal to the number for the function evaluation accomplished by the rest of the pack. For example, if the pack size is 25 dogs, then the number for the function evaluation in each iteration is 27 for the alpha dog (it must be dividable by the number of categories) and 23 for the rest of the pack (pack size minus the two alpha dogs). The following procedure summarizes the main WDPO steps.

Repeat the following steps for *m* iterations
1. The alpha decision:

1.1. The alpha dog senses the potential *n* locations (directions) remotely (by its nose), which are categorized into three categories; each category can be controlled by one parameter.
1.2. The alpha dog evaluates all the locations, and memorizes the enhancement that is gained from each category (parameter).
1.3. If a new location is better than the current location, the alpha dog will update the current location.
1.4. After a few iterations, the alpha dog compares the accumulated enhancements that are gained from each category (parameter), and finds the winner parameter.
1.5. The winner parameter will not be changed, while the other two parameters will be updated to become closer to the winner parameter, such that the first parameter becomes less than the winner, and the second becomes more than the winner parameter.

2. The pack decision:
2.1. The location of the rest of the dogs in the pack will be updated according to the location of the alpha dog with a random increment to reflect each dog's opinion.
2.2. If the location of a dog in the pack is better than the alpha dog's location, then it will have the alpha status over the others.

3. Hoo call:
3.1. If there is no enhancement after a few iterations, re-randomize the locations of the dogs in the pack, add a perturbation to the last best alpha location, and call the pack by implementing step 2 until a new, better location is found.

Procedure 1 describes the WDPO algorithm where the following notations are used in this procedure and the rest of this paper:

m: denotes the maximum number of iterations
n: denotes the size of the dog pack
q: denotes the number of iterations needed to update the parameters
v: denotes the number of iterations needed after stagnation to the *hoo* call procedure
ub: is the upper bound
lb: is the lower bound.
x_{best}: is the best location that is discovered by the pack
b: is used to control the distance from x_{best}
c: is used to control the diversity from alpha
loc: an array that contains the locations that are generated by the alpha
dog: an array that contains the locations of the wild dogs
α : is the alpha dog
gain: an array that contains the accumulated enhancement of each category
CheckBounds: a function that is used to correct the locations if it is out of bounds

Procedure 1: Wild dog pack optimization (WDPO)
Generate *n* dogs randomly and choose the best as *alpha*
For j =1 to m
% Alpha decision
For i=1 to n
Call Build_Locations
Call Evaluate_Locations
Every q iterations

```
            Call  Update_Parameters
End
                % pack decision
For i=1 to n-2
     Call Dog_Locations
     Call Evaluate_Dogs
End

If no improvement for v iterations
      Call Hoo

End
```

To illustrate the alpha decision, consider the parameters p_1, p_2, and p_3, which are initialized as

$$ub > p_1 > p_2 > p_3 > lb$$

then, the generated locations are:

$$loc_{c_j}(t+1) = \alpha(t) + rand(1,d) * p_j$$

where j = 1, 2, and 3, $c_j \in \{i : i - j \text{ is dividable by } 3\}$ and i = 1,2..., n. The parameters p_1, p_2, and p_3 will be used to improve the alpha's capability of exploiting the good locations. This can be done by letting the three parameters compete with each other, which will lead to finding a suitable step size dynamically, as described in Procedure 2.

Procedure 2: *Build_Locations*

```
if mod(i,3)==0
     Loc(i,:)=  alpha+(rand(1,d)*2-1 )* p1;
elseif mod(i,3)==1
     Loc(i,:)=  alpha+(rand(1,d)*2-1  )* p2;
else
     Loc(i,:)=  alpha+(rand(1,d)*2-1 )* p3;
end

Loc(i,:)=CheckBounds(Loc(i,:), lb, ub);
```

To calculate the winner parameter, we first measure the accumulated enhancement of each category as follows:

$$gain_j = gain_j + en_{c_j}$$

where en_{c_j} is the enhancement in a location that belongs to the category c_j (or the gained enhancement by using the parameter p_j). For more details, see Procedure 3.

Procedure 3: *Evaluate_Locations*

```
NewVal=Evaluate(Loc(i,:));
```



```
if NewVal<Min
    alpha=Loc(i,:);
    if mod(i,3)==0
        gain1=gain1+(Min-NewVal);
    elseif mod(i,3)==1
        gain2=gain2+(Min-NewVal);
    else
        gain3=gain3+(Min-NewVal);
    end

    Min=NewVal;
end
```

For every *q* iterations, the parameters will be updated to $p_j = t_j$ such that:

if gain$_1$ is the winner then

$t_1 = (p_1 + ub) / 2$

$t_2 = p_1$

$t_3 = (p_1 + p_2) / 2$

if gain$_2$ is the winner then

$t_1 = (p_1 + p_2) / 2$

$t_2 = p_2$

$t_3 = (p_2 + p_3) / 2$

if gain$_3$ is the winner then

$t_1 = (p_3 + p_2) / 2$

$t_2 = p_3$

$t_3 = p_3 / 2$

In this way, the winner value will be reused and the other values will become surrounded by and closer to the winner. More details are introduced in Procedure 4.

Procedure 4: *Update_Parameter*s

```
if mod(t, q)==0          % q=15
    if gain1>gain2 && gain1>gain3
        t1= (p1+para(5))/2;
        t2=  p1;
        t3 =(p2+p1)/2;
        p1=t1; p2=t2; p3=t3;
    end
    if gain2>gain1 && gain2>gain3
        p1= (p2+p1)/2;
        p3= (p2+p3)/2;
    end
    if gain3>gain2 && gain3>gain1
        t1= (p2+p3)/2;
```

```
    t2=p3;
    t3 =p3/2;
    p1=t1; p2=t2; p3=t3;
  end
  if gain1==gain2 && gain2==gain3
    p1=p1/2; p2=p2/2; p3=p3/2;
  end
  gain1=0;
  gain2=0;
  gain3=0;

end
```

Table 1 and Table 2 show that using the alpha decision (parameter competition) is much better than using fixed parameters where the minimum point can be discovered very rapidly. For example, the minimum value in the sphere and the Griewank functions can be found in less than 300 iterations with acceptable accuracy.

Table 1. Comparison of Algorithm Performance with and without Parameters Competition for the Sphere Function and D=100

Iteration	Accumulated enhancement			Parameters			Minimum value	
	$gain_1$	$gain_1$	$gain_1$	p_1	p_2	p_3	With Competition	With Fixed Parameter
0	0	0	0	50.00	10.00	5.00	60931.65	43532.96
15	0.0000	28054.118	34108.821	7.500	5.000	2.500	18345.17	10289.24
30	1467.3472	1744.5512	3023.1561	3.750	2.500	1.250	3298.729	4183.817
45	100.0673	77.6925	378.1331	1.875	1.250	0.625	1614.602	1176.565
…	…	…	…	…	…	…	…	..
180	0.0338	0.0172	0.0784	0.018	0.012	0.006	0.117363	52.38631
195	0.0002	0.0063	0.0152	0.009	0.006	0.003	0.058182	38.33545
210	0.0038	0.0078	0.0096	0.005	0.003	0.002	0.012510	31.09056
225	0.0005	0.0030	0.0006	50.00	0.005	0.004	0.005736	22.73557

Table 2. Comparison of Algorithm Performance with and without Parameters Competition for the Griewank Function and D=100

Iteration	Accumulated enhancement			Parameters			Minimum value	
	$gain_1$	$gain_1$	$gain_1$	p_1	p_2	p_3	With Competition	With Fixed Parameter
0	0	0	0	50.00	10.00	5.000	1042.4922	965.188
15	0.0000	20441.825	29554.796	7.500	5.000	2.500	174.47933	484.873
30	817.4576	2419.9122	3851.2134	3.750	2.500	1.250	32.806285	221.760
45	123.6147	438.4252	892.6131	1.875	1.250	0.625	6.9144180	85.3848
…	…	…	…	…	…	…	…	…
180	0.1229	0.2240	0.3014	0.018	0.012	0.006	0.0069660	0.60010
195	0.0319	0.0953	0.0925	0.009	0.006	0.003	0.0015833	0.59550
210	0.0044	0.0006	0.0119	0.005	0.003	0.002	0.0005472	0.59550
225	0.0000	0.0000	0.0064	0.004	0.002	0.001	0.0002494	0.58310

The pack decision is used to explore the surrounding area of the alpha dog; therefore, the locations of the dogs can be described as follows:

$$dog_i(t+1) = dog_i(t) + c\ rand\ (\alpha(t) - dog_i(t)) + c\ rand\ (1+d)(\alpha(t) - dog_i(t))$$

The random value is used to reflect the diversity of the dogs' opinions, while the random vector is used to reflect the diversity of the landscape. The parameter *c* is initialized to 1, but if the *hoo* procedure is called then the value of *c* becomes 2, which will increase the probability of escaping from a local optima. Procedures 5 and 6 describe the pack decision.

Procedure 5: *Dog_Locations*

```
dog(i,:)=dog(i,:)+ c* rand* (alpha-dog(i,:))+c* (rand(1,d).*(alpha-dog(i,:)));
dog(i,:)=CheckBounds(dog(i,:), lb, ub);
```

Procedure 6: *Evaluate _Dogs*

```
NewVal=Evaluate(dog(i,:));
if NewVal<Min
  alpha=dog(i,:);
  Min=NewVal;
end
```

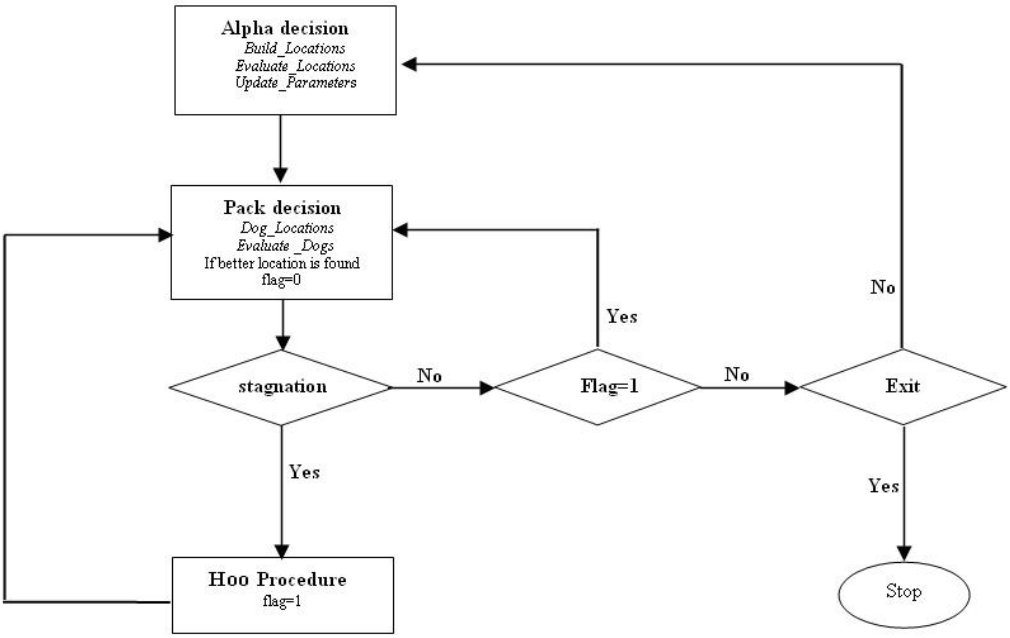

Figure 1. Flowchart of the Wild Dog Pack Procedure

During the implementation, if there is no improvement after *v* iterations, then this means that the dogs fall in an optima and stagnation is detected; thus, the *hoo* procedure will be called. We will consider the source of the sound x_{best} as the best location that was discovered by the pack; therefore, the locations of the dogs are reinitialized randomly to be closer to the source of the sound x_{best} as follows:

$$dog_i(t+1) = x_{best} + b\ rand\ (1, d)$$

The best location among the new locations is considered as a new alpha dog (most of the time, the new location of the alpha is worse than the x_{best} location, but this action is necessary to disturb the previous local optima); however, the alpha decision procedure will not be called

until a better location is found (better than the new *alpha* location). Figure 1 introduces the flowchart of the wild dog pack procedure.

5. Benchmark Problem

Although the "No Free Lunch" theorem states that the searching algorithms will be, on average, the same, fortunately, most of the real applications have some important characteristics that can be exploited during the development of the search algorithms, such as, a small change in the value of the variables will not lead to a huge change in the target functions, and the global optima tends to be close to the local optima. Therefore, in this study, a set of widely used test functions is selected, which has several features such as regularity, multimodality, continuity, reparability, and difficulty. Eight different functions are listed below that can be extended to arbitrary dimensions *d* with a known global minimum [39, 40]:

1- The ***sphere*** function is unimodal, continuous, smooth, separable, convex, and one of the simplest test functions. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to *ub* = 100 and *lb* = −100.

$$f_1(x) = \sum_{i=1}^{d} x_i^2$$

2- The ***Rosenbrock*** function is unimodal, continuous, non-separable, and difficult. Most of the search algorithms are not able to find the correct direction for the global minimum. The global optimum is $f_1(x^*) = 0$ and lies inside a very narrow valley. The test area is restricted to *ub* = 2.048 and *lb* = −2.048. The definition of this function is:

$$f_2(x) = \sum_{i=1}^{d-1} [100\ (x_{i+1} - x_i^2)^2 + (1 + x_i)^2]$$

3- The ***Ackley*** function is multimodal, continuous, non-separable, and regular. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to *ub* = 32.768 and *lb* = −32.768. The definition of this function is:

$$f_3(x) = -20\ \exp\left(-2.0\sqrt{\frac{1}{30}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{30}\sqrt{\sum_{i=1}^{d}\cos(2\pi x_i)}\right) + 20 + e$$

4- The ***Griewank*** function is multimodal, continuous, non-separable, and has numerous local minima that are regularly distributed. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to *ub* = 600 and *lb* = −600. The definition of this function is:

$$f_5(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

5- The ***Schwefel*** function is unimodal, continuous, and non-separable. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to *ub* = 10 and *lb* = −10. The definition of this function is:

$$f_6(x) = \sum_{i=1}^{d}\left|x_i\right| + \prod_{i=1}^{d}\left|x_i\right|$$

6- The ***step*** function is unimodal, discontinuous, and separable. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to *ub* = 100 and *lb* = −100. The definition of this function is:

$$f_4(x) = \sum_{i=1}^{d} (\lfloor x_i^2 + 0.5 \rfloor)^2$$

7- The ***rotated hyper-ellipsoid*** function is unimodal, continuous, and separable. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to $ub = 100$ and $lb = -100$. The definition of this function is:

$$f_7(x) = \sum_{j=1}^{d} \sum_{i=1}^{j} x_i^2$$

8- The ***Rastrigin*** function is highly multimodal, continuous, and separable, and the local minimum are regularly distributed. The global optimum is $f_1(x^*) = 0$ and the test area is restricted to $ub = 5.12$ and $lb = -5.12$. The definition of this function is:

$$f_8(x) = 10\,d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i))$$

6. Experimental Results

All the results in this section are obtained by averaging 30 runs. The suggested algorithm is compared with five meta-heuristic algorithms; namely, the HS [24], global-best harmony search (GHS) [27], self-adaptive harmony search (SAHS) [28], SPSO [13], and RegPSO [22]. Table 3 shows the used parameters for each of the tested algorithms. In contrast to other methods, the new method is not parametrically sensitive, and the average results will be nearly the same for any value of q between 10 and 25, and for any value of v between 30 and 80, while a small change in w, c_1, c_2, *HCMR*, *PAR*, *bw*, ε , or ρ will lead to different results. The performance of the sensitive algorithms cannot be predictable when a new application is considered.

Table 3. The Parameters of the Tested Algorithms

Algorithm	Parameters
HS	HMS=10, HMCR=0.92, PAR=0.35 and bw=0.01
GHS	HMS=10, HMCR=0.92 and $0.01 \le$ PAR ≤ 0.99
SAHS	HMS=50, HMCR=0.99 and $0 \le$ PAR ≤ 1
SPSO	n=40, W=0.73, c_1=1.49 and c_2=1.49
RegPSO	n=20, W=0.73, c_1=1.49, c_2=1.49, ε =1.1e-004 and ρ =1.2/ε
WDPO	n=25, q=15, v=50 and b=0.5

Table 4 shows the importance of using the three strategies of WDPO together. For example, when the new algorithm is applied to Ackley D = 30, the accuracy after 10^5 function evaluations will be 2.9310×10^{-14}, while the accuracy will be reduced to 6.7351, 2.9603×10^{-8} or 2.9603×10^{-6} when the hoo, pack, or alpha procedures are not used, respectively.

Table 4. The Effect of WDPO Strategies by using 100000 Function Evaluations

Function	Dimension	without HOO	without Pack	without alpha	ALL
Rosenbrock	100	**7.6547e+001**	9.4323e+001	8.8839e+001	**7.5318e+001**
	30	**7.9100e-003**	1.7221e+001	9.7075e+000	**7.9451e-003**
Ackley	100	1.6766e+001	1.7800e+001	1.2879e+001	**4.0186 e+000**
	30	6.7351e+000	2.9603e-008	2.9603e-006	**2.9310e-014**
Step	100	5.9720e+002	3.5500 e+001	1.7100e+001	**9.1000e+000**
	30	2.2320e+001	1.2600e+001	**0**	**0**

In Tables 5–8, the benchmark functions are tested by using two different dimensions of 30 and 100, and by using two different numbers for the function evaluations of 5×10^4 and 5×10^5. Several themes can be seen in the tables. First, the new algorithm significantly outperforms all the tested meta-heuristic algorithms for all benchmark functions except for the Rastrigin function, where RegPSO shows a better performance. Second, the new algorithm can be used for unimodal and multimodal scenarios due to its ability to jump out from the local minima when stagnation is detected by using the *hoo* procedure. Third, an accuracy of 10^{-3} can be satisfied by using the new algorithm for most of the benchmark functions of dimension 30 with less than 5×10^5 function evaluations. However, this accuracy can be obtained for all functions up to the dimension of 100 with less than 8×10^5 function evaluations. Fourth, the HS algorithm and its variants are better than PSO and its variants for the Rosenbrock, Ackley, Griewank, Schwefel, and step functions, while PSO and its variants are better than the HS algorithm and its variants for the sphere, rotated-h-e, and Rastrigin functions.

Table 5. Comparison of Five Algorithms and WDPO by using 500000 Function Evaluations and D=30

function	HS	GHS	SAHS	SPSO	RegPSO	WDPO
Rosenbrock	3.3099e-001 (1.5959e-001)	7.0763e-001 (5.1249e-002)	6.3643e-002 (1.9570e-002)	8.6155e+000 (9.4399e-001)	5.2040e-001 (4.2241e-002)	**4.2135e-028** (4.1298e-026)
Sphere	7.3855e-001 (9.3880e-002)	5.0836e-001 (5.2526e-002)	7.6724e-012 (1.7946e-012)	3.9627e-050 (2.0727e-048)	2.6813e-015 (9.4491e-016)	**2.6963e-315** (7.6182e-316)
Ackley	5.0173e-002 (3.8611e-003)	5.4041e-002 (3.1124e-003)	2.0308e-004 (5.7756e-005)	7.1769e-001 (5.5129e-001)	8.8399e-002 (5.2253e-002)	**1.5099e-014** (3.7810e-015)
Griewank	4.9675e-005 (8.1259e-006)	1.1189e-003 (3.8829e-004)	4.2386e-010 (1.9681e-010)	5.0307e-002 (6.8827e-003)	2.2718e-002 (3.2402e-003)	**0** (0)
Schwefel	3.7089e-005 (2.2943e-006)	4.4902e-007 (4.4570e-008)	5.6850e-006 (2.6129e-006)	3.0358e-005 (9.8407e-006)	6.5908e-006 (1.5911e-006)	**4.8970e-139** (1.0203e-138)
Step	**0** (0)	**0** (0)	**0** (0)	4.200e+000 (9.1515e-001)	1.4200e+000 (4.1958e-001)	**0** (0)
Rotated-h-e	1.6955e+001 (1.7992e-000)	1.5609e+001 (4.4549e-001)	1.1240e+001 (8.0782e-000)	6.4821e-010 (2.1824e-010)	5.4502e-008 (1.8352e-007)	**9.5172e-320** (1.3780e-320)
Rastrigin	9.5395e+000 (7.1824e-001)	4.6053e+000 (8.9957e-001)	7.1162e+000 (5.2029e-001)	6.2590e+001 (5.5109 e+001)	**8.3092e-010** (1.2680e-008)	6.2341e+000 (4.9925e-000)

Table 6. Comparison of Five Algorithms and WDPO using 500,000 Function Evaluations and D = 100

function	HS	GHS	SAHS	SPSO	RegPSO	WDPO
Rosenbrock	9.4369e+001 (3.6837e+001)	4.1439e+001 (5.1667e+000)	2.9130e+001 (7.4963 e+000)	6.1359e+001 (1.5878e+001)	2.8585e+001 (5.2026e+000)	**2.8220e+001** (3.8701e+000)
Sphere	2.7614e+000 (2.4391e+000)	7.6280e+000 (3.9352e-001)	2.2626e-001 (9.5596e-002)	6.4617e-030 (4.1978e-031)	2.6805e-010 (7.0694e-011)	**6.4645e-306** (1.3992e-304)
Ackley	7.2550e+000 (1.2915e+000)	1.9868e-001 (3.0582e-002)	8.2945e-003 (3.4291e-003)	5.9062e+001 (4.1226e+001)	4.7216e+000 (5.4716e+000)	**9.3259e-014** (7.2402e-013)
Griewank	6.1744e-002 (7.9980e-003)	1.3734 e+000 (3.0219e-001)	9.4160e-005 (1.1619e-005)	1.4608e+000 (4.3095e-001)	7.5091e+000 (9.4062e-001)	**1.1102e-016** (5.4497e-013)
Schwefel	4.3133e-002 (1.4385e-002)	2.2362e-003 (7.0903e-004)	1.7854e-002 (3.2936e-003)	5.6822e-003 (4.0933e-004)	5.7614e-003 (5.1386e-004)	**1.3011e-058** (2.1560e-057)
Step	5.1000e+001 (8.5762e+000)	5.6533e+001 (7.1944e+000)	4.1200e+001 (7.4370e+000)	6.6500e+001 (1.8133e+001)	6.2500e+001 (1.0904e+001)	**4.1333e+000** (1.0023e+000)
Rotated-h-e	9.4022e+001 (1.4578e+001)	8.0442e+001 (1.8269e+001)	7.6734e+001 (9.7645e+000)	4.1302e-004 (6.7442e-004)	9.2711e-004 (4.4394e-004)	**7.2618e-045** (2.2514e-041)
Rastrigin	2.7076e+001 (1.6308e+001)	2.2458e+001 (2.0266e+001)	3.9200e+001 (1.8324e+001)	3.1940e+001 (1.1708e+001)	**1.7459e-001** (6.5896e-002)	5.1810e+001 (8.4351e+000)

Table 7. Comparison of Five Algorithms and WDPO by using 50000 Function Evaluations and D=30

function	HS	GHS	SAHS	SPSO	RegPSO	WDPO
Rosenbrock	3.9455e+001 (7.3647e+000)	4.1367e+001 (5.0152e+000)	3.6953e+001 (8.9989 e+000)	2.6260e+001 (5.0839e+000)	9.5734e+001 (3.4700 e+000)	**6.7795e+000** (6.8722e-001)
Sphere	5.1066e+001 (4.5269e+000)	4.9520e+001 (1.0853e+001)	6.7442e+000 (1.2769e+000)	7.7033e-004 (9.6081e-005)	1.5073e-004 (1.8376e-005)	**2.7434e-037** (6.0890e-036)
Ackley	1.3304e+001 (6.5810e+000)	1.7441e+001 (6.2152e+000)	1.8987e+000 (1.1000e+000)	5.8575e+000 (3.8057e-001)	3.5713e+000 (3.5923e-001)	**2.6544e-005** (7.9923e-006)
Griewank	1.1747e+001 (3.7578e+000)	1.7107e+002 (5.0723e+001)	9.5488e-001 (1.2459e-001)	7.5165e+001 (1.6418e+001)	5.1529e+001 (5.3233e+000)	**1.3870e-014** (8.3260e-013)
Schwefel	9.5172e-001 (4.6220e-001)	1.0489e-002 (8.5586e-003)	1.0534e-001 (4.8074e-002)	5.6079e-001 (6.5940e-002)	8.6103e-001 (6.1982e-002)	**2.4618e-014** (1.8713e-014)
Step	1.6000e+000 (3.3962e+000)	1.3333e-001 (2.7824e-002)	1.6000e-001 (5.7910e-002)	5.0000e+000 (8.6665e-001)	1.3333e+000 (9.3645e-001)	**0** (0)
Rotated-h-e	1.4827e+002 (2.0395e+001)	5.1880e+002 (1.1778e+001)	1.1002e+002 (4.2402e+001)	8.9763e-001 (1.4418e-001)	8.9809e-001 (8.6678e-002)	**5.7546e-024** (5.2541e-022)
Rastrigin	5.4339e-001 (5.0704e-001)	5.3503e+001 (2.1977e+001)	7.6706e+001 (8.0748e+000)	7.8798e+002 (7.0019e+001)	**3.0872e+000** (2.6729e+000)	1.1369e+002 (4.5061e+001)

Table 8. Comparison of Five Algorithms and WDPO by using 50000 Function Evaluations and D=100

function	HS	GHS	SAHS	SPSO	RegPSO	WDPO
Rosenbrock	3.5480e+002 (7.5071e+001)	3.1685e+002 (9.1771e+001)	8.5773e+001 (4.8637e+000)	9.4893e+001 (3.4378e+001)	9.0658e+001 (2.8031e+001)	**8.5009e+001** (9.4571e+000)
Sphere	2.1259e+002 (4.5310e+001)	4.5618e+002 (5.6909e+001)	5.0306e+001 (1.9145e+001)	9.8969e+001 (1.1728e+001)	3.4192e+001 (9.7323e+000)	**1.5448e-032** (4.0118e-031)
Ackley	2.1799e+001 (4.2384e+000)	2.8337e+001 (7.5522e+000)	**1.4494e+001** (9.8972e-001)	2.3189e+002 (1.6986e+001)	5.6085e+001 (4.3506e+000)	1.5230e+001 (1.8922e+000)
Griewank	4.5626e+002 (2.4610e+001)	4.2314e+002 (4.8993e+001)	7.6855e+000 (1.5228e+000)	4.9168e+002 (3.2678e+001)	1.0810e+001 (5.6477e+000)	**2.8208e-05** (6.7527e-005)
Schwefel	9.3548e+001 (1.2985e+001)	6.0212e+000 (2.0491e+000)	3.2853e+001 (4.9568e+000)	5.4276e+001 (7.0886e+000)	2.2272e+001 (1.9448 e+00)	**7.7974e-001** 1.9016e-001
Step	5.5200e+003 (8.5044e+001)	8.3733e+003 (1.5157e+002)	3.7633e+002 (2.7528e+001)	4.2520e+002 (3.0979e+001)	6.9400e+002 (7.5901e+001)	**1.6333e+001** (2.8654e+000)
Rotated-h-e	9.1561e+003 (2.2566e+002)	3.0004e+004 (1.8022e+003)	7.6699e+003 (5.6183e+002)	1.3963e+002 (7.8786e+001)	3.5721e+002 (9.2153e+001)	**3.8721e+001** (5.9632e+000)
Rastrigin	2.4528e+002 (4.1866e+001)	1.6438e+002 (3.7609 e+01)	7.9321e+002 (3.6942e+001)	7.8688e+002 (4.6359e+001)	**2.7511e+000** (1.4674 e+00)	3.1823e+002 (7.3341e+001)

Figure 2 and Figure 3 show the convergence curve of the Rosenbrock and Griewank functions when using the new algorithm where no stagnation is detected; therefore, the *hoo* procedure was not called. While Figure 4 and Figure 5 show the enhancement of the convergence curve when using the *hoo* procedure when the stagnation is detected in the Ackley and Step functions.

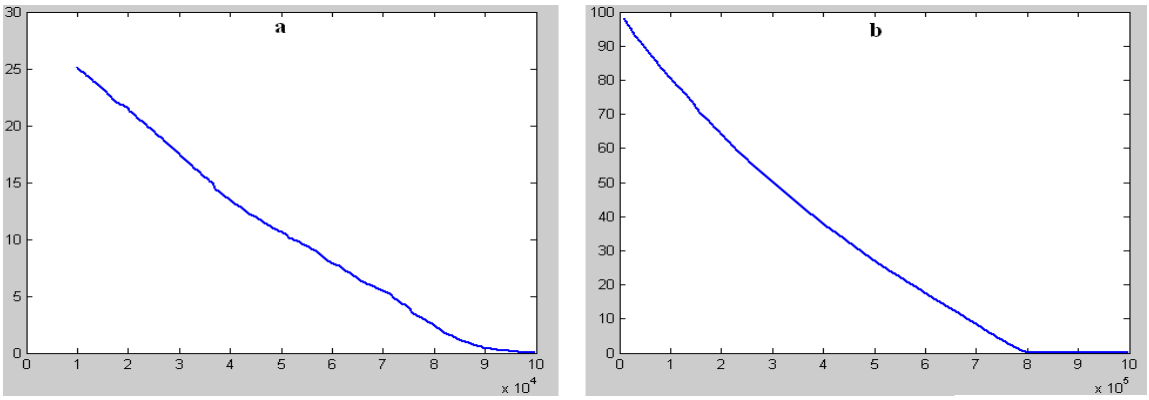

Figure 2. The Convergence Curve of the Rosenbrock Function: (a) D = 30; (b) D = 100

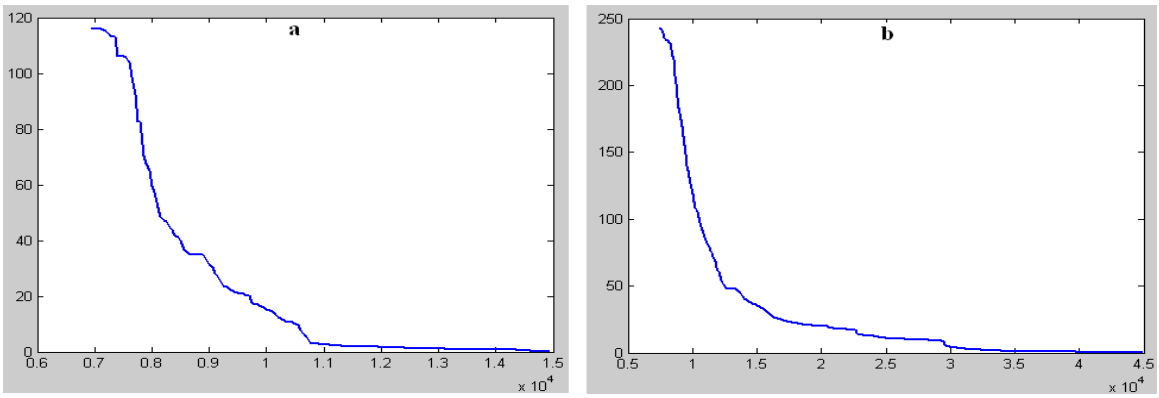

Figure 3. The Convergence Curve of the Griewank Function: (a) D = 30; (b) D = 100

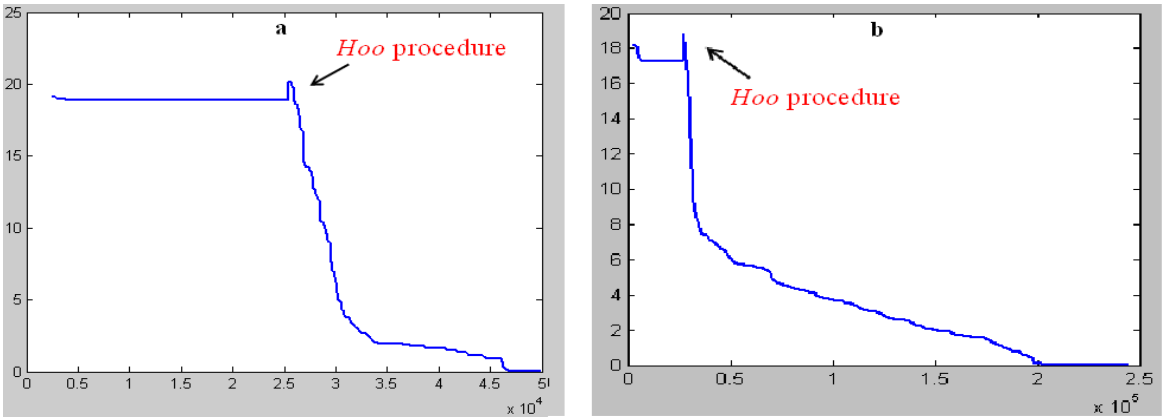

Figure 4. The Enhancement of the Convergence Curve using the *hoo* Procedure in the Ackley Function: (a) D = 30; (b) D = 100

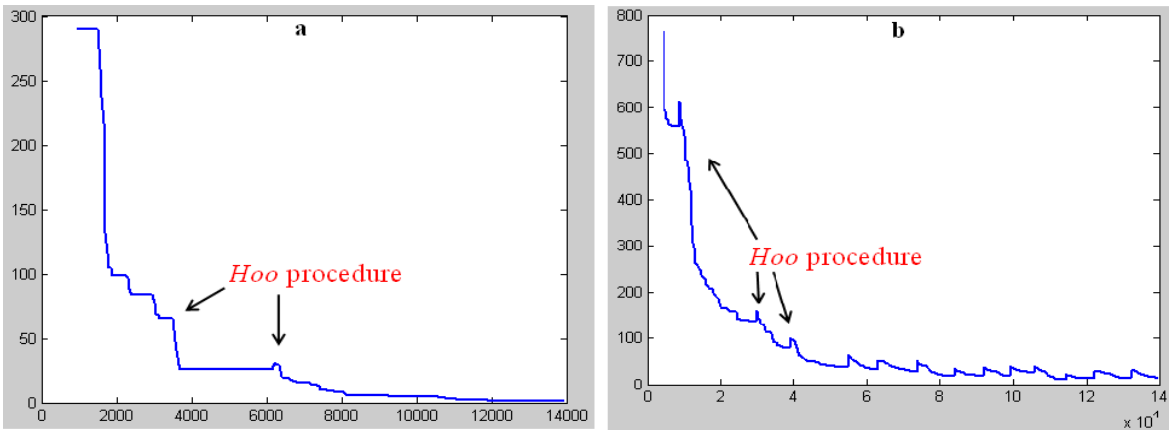

Figure 5. The Enhancement of the Convergence Curve using the *hoo* Procedure in the Step Function: (a) D = 30; (b) D = 100

7. Conclusion

To trade off between exploring and exploiting, three main procedures are used. The first is to exploit any potential position by using the alpha dog decision, which will reach an extreme point very rapidly. This strategy leaves enough time to try another potential route by using the second procedure (the *hoo* procedure*)*. The third procedure calls up the pack decision, which can be used to exploit and explore the neighboring landscape. The results indicate that the new algorithm outperforms other state-of-the-art algorithms. An important direction for future work would be to extend this WDPO algorithm to discrete cases and combinatorial problems. Moreover, the suggested algorithm could be hybridized with other meta-heuristic algorithms and operations.

References

[1] I. Boussaïd, J. Lepagnot and P. Siarry, "A survey on optimization metaheuristics, Information Sciences", vol. 237, **(2013),** pp. 82-117.
[2] H. Afaq and S. Saini, "On the Solutions to the Travelling Salesman Problem using Nature Inspired Computing Techniques", International Journal of Computer Science, vol. 8, **(2011),** pp. 326-334.
[3] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Computing Surveys, vol. 35, **(2003),** pp. 268–308.
[4] S. Parragh, K. Doerner and R. Hart, "A survey on pickup and delivery problems", *Journal für Betriebswirtschaft*, vol. 58, **(2008),** pp. 81–117.
[5] Z. W. Geem, "Optimal scheduling of multiple dam system using harmony search algorithm", Lecture Notes in Computer Science, vol. 4507, **(2007),** pp. 316-323.
[6] A. Vasebi, M. Fesanghary and S. Bathaeea, "Combined heat and power economic dispatch by harmony search algorithm", International Journal of Electrical Power and Energy Systems, vol. 29, (**2007),** pp. 713-719.
[7] Z. W. Geem, K. S. Lee and Y. Park, "Application of harmony search to vehicle routing", American Journal of Applied Sciences, vol. 2, **(2005),** pp. 1552-1557.
[8] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm", Computers and Structures, vol. 82, **(2004),** pp. 781–798.
[9] Z. W. Geem, J. H. Kim, and G.V. "Loganathan, Harmony search optimization: application to pipe network design", Int. J. Model. Simul. vol. 22, **(2002),** pp. 125–133.
[10] S. Binitha and S. Sathya, "A Survey of Bio inspired Optimization Algorithms", International Journal of Soft Computing and Engineering, vol. 2, **(2012),** pp. 2231-2307.

[11] B. Naderi, R. Tavakkoli-Moghaddam and M. Khalili, "Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan", Knowl.-Based Syst. 23, **(2010),** pp. 77-85.
[12] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", Proceedings of the sixth international symposium on micro machine and human science, Nagoya, Japan, **(1995)**, pp. 39–43.
[13] J. Kennedy and R. Eberhart, "Particle swarm optimization, Proceedings of IEEE International Conference on Neural Network", Piscataway, NJ, **(1995)**, pp. 1942–1948.
[14] M. Jiang, Y.P. Luo and S.Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm", Inform. Process. Lett. vol. 102, **(2007)**, pp. 8–16.
[15] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients", IEEE Trans. Evol. Comput. vol. 8, **(2004),** pp. 240–255.
[16] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung, "Adaptive particle swarm optimization", IEEE Trans. Syst., Man, Cybern. B: Cybern. vol. 39, **(2009),** pp. 1362–1381.
[17] X. Yang, J. Yuan and H. Mao, "A modified particle swarm optimizer with dynamic adaptation", Appl. Math. Comput. vol. 189, **(2007),** pp. 1205–1213.
[18] Y. Tang, Z. D. Wang and J. A. Fang, "Feedback learning particle swarm optimization", Appl. Soft Comput. vol. 11, **(2011),** pp. 4713–4725.
[19] Z. Wu and J. Zhou, "A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment", Conf. Comput. Intell. and Security, **(2007)**, pp.133–136.
[20] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology", IEEE Transactions on Evolutionary Computation. vol. 14, **(2010),** pp. 150–169.
[21] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments", IEEE Trans. Evol. Comput. vol. 14, **(2010),** pp. 959–974.
[22] G. Evers and M. B. Ghalia, "Regrouping Particle Swarm Optimization: A new Global Optimization Algorithm with Improved Performance Consistency Across Benchmarks", IEEE International Conference on Systems, Man and Cybernetics, San Antonio, **(2009)**, pp. 3901–3908.
[23] X. C. Zhao, "A perturbed particle swarm algorithm for numerical optimization", Appl. Soft Comput. 10, **(2010),** pp. 119–124.
[24] Z. W. Geem, J. H. Kim and G.V. Loganathan, "A new heuristic optimization algorithm: harmony search", Simulation, vol. 76, **(2011),** pp. 60–68.
[25] Z. W. Geem, "Music-Inspired Harmony Search Algorithm: Theory and Applications", Studies in Computational Intelligence, Springer, **(2009).**
[26] O. M. Alia and M. Rajeswari, "The variants of the harmony search algorithm: an Overview", Artif. Intell. Rev. vol. 36, **(2011),** pp. 49–68.
[27] M. G. Omran and M. Mahdavi, "Global-best harmony search. Applied Mathematics and Computation 198, **(2008),** pp. 643–656.
[28] C. M. Wang and Y. F. Huang, "Self adaptive harmony search algorithm for optimization", Expert Systems with Applications, vol. 37, **(2010),** pp. 2826–2837.
[29] P. Chakraborty, G. G. Roy, S. Das, D. Jain and A. Abraham, "An improved harmony search algorithm with differential mutation operator", Fundam. Inform., vol. 95, **(2009)**, pp. 1–26.
[30] S. O. Degertekin, "Improved harmony search algorithms for sizing optimization of truss structures", Computers and Structures, vol. 92, **(2012),** pp. 229–241.
[31] S. Afshari, B. Aminshahidy and M. R. Pishvaie, "Application of an improved harmony search algorithm in well placement optimization using streamline simulation, Journal of Petroleum Science and Engineering**,** vol. 78, **(2011),** pp. 664–678.
[32] D. Zou, L. Gao, J. Wu and S. Li, "Novel global harmony search algorithm for unconstrained problems", Neurocomputing, vol. 73, **(2010),** pp. 3308-3318.
[33] M. Al-Betar, A. Khader and I. Liao, "A harmony search with multi-pitch adjusting rate for the university course timetabling", Recent advances in Harmony search algorithm, **(2010)**, pp 147–161.
[34] M. Mahdavi, M. Fesanghary and E. Damangir, "An improved harmony search algorithm for solving optimization problems", Applied Mathematics and Computation, vol. 188, **(2007),** pp. 1567–1579.
[35] S. Hadi and Z. Kamran, "Improvement of harmony search algorithm by using statistical analysis", Artif. Intell., vol. 37, **(2011),** pp. 181–215.
[36] S. R. Creel and N. M. Creel, "Communal hunting and pack size in African wild dogs", Lycaon pictus. Animal Behaviour, vol. 50, **(1995),** pp.1325-1339.
[37] J. H. Fanshawe and C. D. Fitzgibbon, "Factors influencing the hunting success of an African wild dog pack.", Animal Behaviour, vol. 45, **(1993),** pp. 479-490.

[38] A. Pole, I. J. Gordon, M. L. Gorman and M. MacAskill, "Prey selection by the African wild dog (Lycaon pictus) in southern Zimbabwe", Journal of Zoology, vol. 262, **(2004),** pp. 207-215.
[39] N. Andrei, "An Unconstrained Optimization Test Functions Collection", Advanced Modeling and Optimization, vol. 10, **(2008),** pp. 147-161.
[40] M. O. Ali, S. P. Koh, K. H. Chong and F. D.Yap, "Hybrid Artificial Immune System-Genetic Algorithm optimization based on mathematical test functions", IEEE Student Conference on Research and Development, **(2010)**, pp. 256 – 261.