

## Performance Comparison of Biology based Metaheuristics Optimization Algorithms using Unimodal and Multimodal Benchmark Functions

Fatma BELLI<sup>1</sup>, Harun BINGOL<sup>1\*</sup>

<sup>1</sup>Software Engineering, Faculty of Engineering and Natural Sciences, Malatya Turgut Ozal University, Malatya, Turkey  
<sup>1</sup>fatmacubukcu@outlook.com, <sup>1\*</sup>harun.bingol@ozal.edu.tr

(Geliş/Received: 05/12/2022;

Kabul/Accepted: 25/02/2023)

**Abstract:** Optimization is used in almost every aspect of our lives today and makes our lives easier. Optimization is generally studied as classical and heuristic optimization techniques. Classical optimization methods are not effective in real-world engineering problems. These methods, by their nature, require a mathematical model. Metaheuristic optimization methods have started to be used frequently today in the solution of these problems when a mathematical model cannot be created or a solution cannot be produced in an effective time even if it is created. These methods, by their nature, cannot produce effective results in all engineering problems. Therefore, new metaheuristic optimization methods are constantly being researched. In this study, quality test functions have been used to compare the performances of five algorithms that have been developed in recent years and produce effective results. The results obtained from these functions are shared in this study. It has been observed that the Artificial Hummingbird Optimization Algorithm (AHA) gives better results than other metaheuristic algorithms.

**Keywords:** AHA, optimization, artificial intelligence, metaheuristic optimization

### Tek Modlu ve Çok Modlu Kıyaslama Fonksiyonlarını Kullanan Biyoloji Tabanlı Metasezgisel Optimizasyon Algoritmalarının Performans Karşılaştırması

**Öz:** Optimizasyon günümüzde hayatımızın hemen her alanında kullanılmakta ve hayatımızı kolaylaştırmaktadır. Optimizasyon genellikle klasik ve sezgisel optimizasyon teknikleri olarak incelenmektedir. Klasik optimizasyon yöntemleri gerçek dünya mühendislik problemlerinde etkili değildir. Bu yöntemler doğaları gereği matematiksel bir modele ihtiyaç duyarlar. Matematiksel modelin oluşturulamadığı yada oluşturulsa bile etkili bir zamanda çözüm üretilemeyeceği anlarda bu problemlerin çözümünde metasezgisel optimizasyon yöntemleri günümüzde sıklıkla kullanılmaya başlanmıştır. Bu yöntemler tüm mühendislik problemlerinde etkili sonuçları doğaları gereği üretmezler. Bundan dolayı sürekli olarak yeni metasezgisel optimizasyon yöntemleri araştırılmaktadır. Bu çalışmada da son yıllarda geliştirilen ve etkili sonuçlar üreten beş adet algoritmanın başarımlarını karşılaştırmak amacıyla kalite test fonksiyonları kullanılmıştır. Bu fonksiyonlardan elde edilen sonuçlar bu çalışmada paylaşılmıştır. Yapay Sinek Kuşu Optimizasyon Algoritması'nın (YSA) diğer metasezgisel algoritmalarından daha iyi sonuçlar verdiği gözlemlenmiştir.

**Anahtar kelimeler:** YSA, optimizasyon, yapay zeka, metasezgisel optimizasyon

#### 1. Introduction

Optimization literally means best improvement. It is the task of obtaining the best solution from all the solutions of the problem within certain limits. Optimization problem is the solution of unknown parameter values can be defined as any problem involving the presence of in such a way as to meet the constraints of the problem [1,2]. In addition to using optimization techniques to speed up decision-making and improve decision quality, it is also used to solve problems in the real world in a way that is efficient, accurate, and timely [3,4].

The element that affects the performance of the optimization is called the decision variable. The objective function is formed by the analytical display of the decision element on the objective. The values of the decision variables affect what the value of the objective function will be. Some restrictions need to be met in this process. Any solution in the solution space that satisfies the constraints of the problem is called a feasible solution. Optimum solution refers to the best solution among suitable solutions according to the determined purpose.

We can examine the methods used in the optimization process under two main headings. These are Deterministic (mathematical modeling) and Stochastic (random) [5]. Stochastic methods are examined under two sub-headings as heuristic and metaheuristic. In the deterministic method, it is tried to reach the result within the

tolerances. This method takes a long time to solve large-scale problems and no exact solution can be found. Heuristic methods also produce problem-specific solutions and do not guarantee the optimum solution. But it is faster than analytical methods. Metaheuristic methods, on the other hand, obtain a solution by adapting certain algorithms to the problem to be solved. In this study, we tested using benchmark functions that AHA, which is one of the biological-based metaheuristic algorithms based on swarm intelligence, gives faster results and performs better against the gray wolf optimization (GWO), whale algorithm (WHO), dragonfly algorithm (DA), ant lion algorithm (ALO). All of these algorithms are biology-based algorithms inspired by nature.

The organization of this article is as follows. In the first chapter, general information about optimization methods is given. In the second part, information about the working principle of the AHA optimization algorithm is given. In the third chapter, information about the working principle of the GWO algorithm is given. In the fourth chapter, information about the working principle of the WHO optimization algorithm is given. In the fifth chapter, information about the working principle of the DA optimization algorithm is given. In the sixth chapter, information about the working principle of the ALO optimization algorithm is given. In the seventh chapter, the mathematical expressions of the quality test functions used in the experiments and the experimental results are given. Finally, the results obtained in the eighth chapter are interpreted and the study is summarized.

## 2. AHA Algorithm

The artificial hummingbird algorithm (AHA) is inspired by hummingbirds, considered the smallest birds in the world. This algorithm was designed by Seyedali Mirjalili, Weiguo Zhao, Liying Wang in 2021. AHA is a biological-based metaheuristic algorithm based on swarm intelligence.

The artificial hummingbird algorithm (AHA) is designed by modeling hummingbirds' special flight skills and intelligent foraging strategies. This algorithm was created by modeling strategies such as guided foraging, regional foraging, foraging on the migration route, and axial, cross and omnidirectional flight skills. To guide hummingbirds in the algorithm, a visitation chart has been created that mimics hummingbirds' extraordinary memory abilities [6].

### 2.1. Main Components of AHA

The AHA algorithm has 3 basic components. These are as follows;

**Food sources:** A hummingbird frequently assesses the properties of the sources, such as the nectar quality and content of flowers, the nectar fill rate, and the most recent visit. In the AHA, the nectar filling rate of each food source is represented by the function fitness value, and it is assumed that each food source has the same number and kind of flowers. A food source is a solution vector. The rate of nectar replenishment from the food source increases in direct proportion to fitness value.

**Hummingbirds:** Every hummingbird has a specific food source that it may eat from, and both the hummingbird and its food source are in the same location. A hummingbird can remember where this specific food source is and how quickly it fills up on nectar. It may also communicate this knowledge to other hummingbirds in the community. Each hummingbird can recall how long it has been since it last visited a particular food source.

**Visit table:** The visits chart tracks the frequency with which various hummingbirds visit each food source and displays the amount of time since the same hummingbird last visited a certain food source. A hummingbird will prioritize visits to a food source that receives a lot of visits from that hummingbird. Among the food sources with the same greatest visitation level, a hummingbird prefers to visit the one with the highest ratio of nectar filling in order to get more nectar. Each hummingbird can use the visit chart to locate its preferred food source. Every iteration often involves updating the visit table.

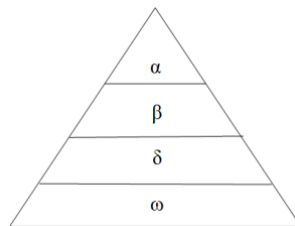
The mathematical expression of the components of the AHA in the algorithm is shown in Table 1. The following algorithm  $i$ .location of food source,  $j$ .food source,  $X_i(t)$  candidate food source,  $V_i$  regional foraging,  $X_{wor}$  worst nectar rate.

**Table 1.** Pseudo Code Artificial Hummingbird Algorithm [6]

<p><b>Input:</b> <math>n, d, f, \text{Max\_It}, \text{Up}</math>  <b>Output:</b> Glo-min, Glo-minimizer  <b>Initialization</b>  <b>For</b> ith h-bird from 1 to <math>n</math>, <b>Do</b>  <b>if</b> <math>i \neq j</math>  <b>End For</b>  <b>Then</b> <math>V\_tbl_{ij} = 1</math>,  <b>Else</b> <math>V\_tbl_{ij} = \text{null}</math>,  <b>End If</b>  <b>End For</b>  <b>While</b> <math>t \leq \text{Max\_It}</math> <b>Do</b>  <b>For</b> ith h-bird from 1 to <math>n</math>, <b>Do</b>  <b>If</b> <math>\text{rand} \leq 0.5</math> <b>Then</b>  <b>If</b> <math>r &lt; 1/3</math> <b>Then</b>  <math>D^{(i)} = \begin{cases} 1, &amp; \text{if } i = \text{randi}([1, d]) \\ 0, &amp; \text{else} \end{cases} \quad i = 1, \dots, d</math>  <b>Else If</b> <math>r &gt; 2/3</math> <b>Then</b>  <math>D^{(i)} = \begin{cases} 1, &amp; \text{if } i = P(j), j \in [1, k], P = \text{randperm}(k), k \in [2, \lceil r_1 \cdot (d - 2) + 1 \rceil] \\ 0, &amp; \text{else} \end{cases}</math>  <b>Else</b> <math>D^{(i)} = 1 \quad i = 1, \dots, d</math>  <b>End If</b>  <b>End If</b>  <math>V_i(t + 1) = x_{i, \text{tar}}(t) + a \cdot D \cdot (x_i(t) - x_{i, \text{tar}}(t))</math>  <b>If</b>  <math>f(V_i(t + 1)) &lt; f(x_i(t))</math>  <b>Then</b>  <math>x_i(t + 1) = V_i(t + 1)</math>  <b>For</b> jth fd_src from 1 to <math>n</math> (<math>j \neq \text{tar}, i</math>), <b>Do</b>  <math>V\_tbl(i, j) = V\_tbl(i, j) + 1</math>,  <b>End for</b>  <math>V\_tbl(i, \text{tar}) = 0</math>,  <b>For</b> jth fd_src from 1 to <math>n</math> (<math>j \neq \text{tar}, i</math>), <b>Do</b>  <math>V\_tbl(j, i) = \max(V\_tbl(j, l)) + 1</math>  <math>l \in n</math> and <math>l \neq j</math>  <b>End For</b></p>	<p><b>Else</b>  <b>For</b> jth fd_src from 1 to <math>n</math> (<math>j \neq \text{tar}, i</math>) <b>Do</b>  <math>V\_tbl(i, j) = V\_tbl(i, j) + 1</math>,  <b>End For</b>  <math>V\_tbl(i, \text{tar}) = 0</math>,  <b>End</b>  <b>Else</b> <math>V_i(t + 1) = x_i(t) + b \cdot D \cdot x_i(t)</math>  <b>If</b> <math>f(V_i(t + 1)) &lt; f(x_i(t))</math>  <b>Then</b> <math>x_i(t + 1) = V_i(t + 1)</math>  <b>For</b> jth fd_src from 1 to <math>n</math> (<math>j \neq i</math>), <b>Do</b>  <math>V\_tbl(i, j) = V\_tbl(i, j) + 1</math>,  <b>End For</b>  <b>For</b> jth fd_src from 1 to <math>n</math>, <b>Do</b>  <math>V\_tbl(i, j) = \max(V\_tbl(j, l)) + 1</math>  <math>l \in n</math> and <math>l \neq j</math>  <b>End For</b>  <b>Else</b>  <b>For</b> jth fd_src from 1 to <math>n</math> (<math>j \neq i</math>), <b>Do</b>  <math>V\_tbl(i, j) = V\_tbl(i, j) + 1</math>,  <b>End For</b>  <b>End If</b>  <b>End If</b>  <b>End For</b>  <b>If</b> <math>\text{mod}(t, 2n) == 0</math>, <b>Then</b>  <math>x_{\text{wor}}(t + 1) = \text{Low} + r \cdot (\text{Up} - \text{Low})</math>  <b>For</b> jth fd_src from 1 to <math>n</math> (<math>j \neq \text{wor}</math>), <b>Do</b>  <math>V\_tbl(\text{wor}, j) = V\_tbl(\text{wor}, j) + 1</math>,  <b>End For</b>  <b>For</b> jth fd_src from 1 to <math>n</math>, <b>Do</b>  <math>\text{Visit\_table}(j, \text{wor}) = \max(V\_tbl(j, l)) + 1</math>  <math>l \in n</math> and <math>l \neq j</math>  <b>End For</b>  <b>End If</b>  <b>End While</b></p>
--	---

### 3. GWO Algorithm

Gray wolf Optimization (GWO) gray wolves; It was inspired by the fact that they lived in packs and hunted together, forming a strong social hierarchy [7].



**Figure 1.** Hierarchy in gray wolves [7]

The hierarchical structure of the gray wolf algorithm is shown in Figure 1. Alpha is the leader in Gray Wolf Optimization. Alpha; It is responsible for making decisions such as hunting, sleeping place, and time to get up and communicating these decisions to the swarm. Beta is at the second level in the hierarchy. The beta assists the alpha in decision making or other swarm-related activities. The wolf at the third level is the delta. While Delta takes orders from alpha and beta, he gives orders to those below him in the hierarchy pyramid. Scouts, lookouts, elders, hunters and rangers make up the delta. In the hierarchy pyramid, omega is at the lowest level and omega receives orders from wolves at the higher level [8].

GWO simulates the hunting strategies of gray wolves apart from the social hierarchy of wolves. In GWO, in order to model the social hierarchy mathematically, the best individual is taken as alpha ( $\alpha$ ), second individual as beta ( $\beta$ ), and third individual as delta ( $\delta$ ). It is expected that the remaining solutions are omega ( $\omega$ ) [8].

### 3.1. Components of the GWO Algorithm

The GWO algorithm was developed, inspired by the hunting and feeding behavior of gray wolves living in packs. The hunting behavior of wolves is shown in Figure 2.

**Encircling prey:** Gray wolves have the ability to find and surround their prey. . Hunting behavior of gray wolves; chasing, encircling the prey, and attacking the prey. [9]

**Hunting:** Although beta and delta sometimes take place in hunting, it usually takes place as a result of alpha's guidance. To model this mathematically, it is assumed that alpha, beta, and delta are better solutions than others for locating prey. For this reason, new locations of search agents, including omega, are updated according to the three best solutions.

**Search for prey (exploration):** When the A value is greater than 1 in the gray wolf algorithm, a better prey search is provided by moving away from its prey and thereby improving the global research ability of the gray wolf optimization algorithm.

**Attacking prey (exploitation):** The basic exploitation phase of the gray wolf algorithm is just like in the real world, these wolves first attack their prey and then eat it until it is finished. This corresponds to the exploitation phase of the algorithm. Flowchart of gray wolf algorithm Figure 3 is also shown.



Figure 2. Hunting in gray wolves [7]

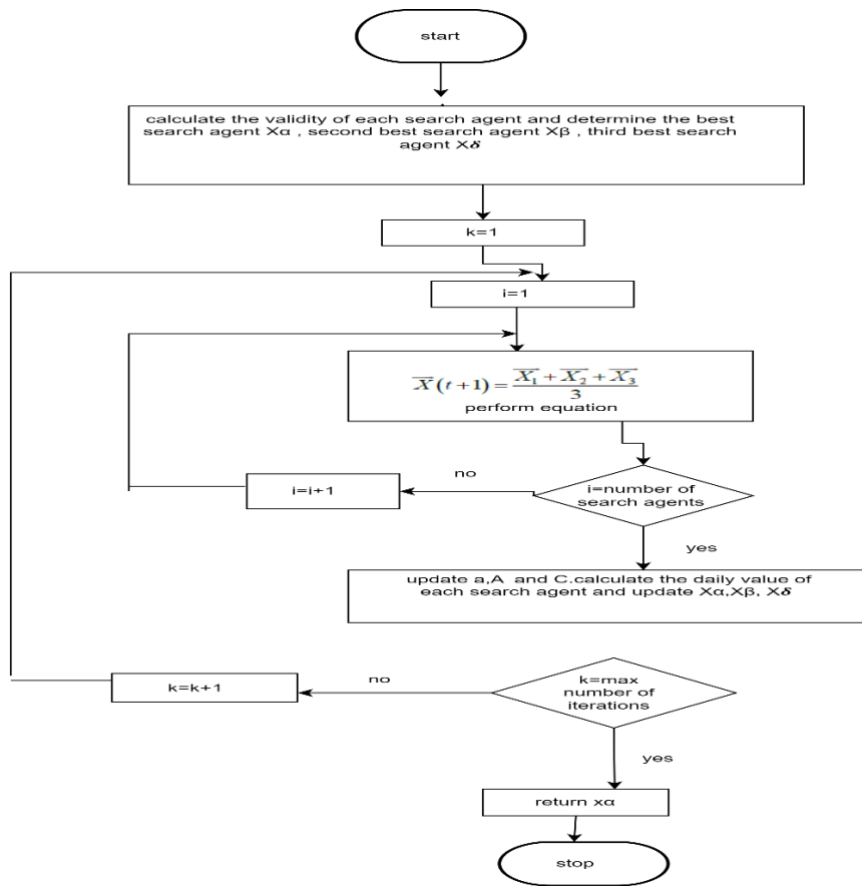


Figure 3. Flow Chart of the Gray Wolf Algorithm

#### 4. WOA Algorithm

Humpback whales are the largest of the whale species. Humpback whales have a great hunting method for finding small and krill fish. WOA was developed inspired by this hunting behavior of humpback whales [10]. The distinctive bubble behavior of humpback whales, which often graze on shoals of tiny fish, allows them to breathe underwater and produce bubble clouds. As seen in Figure 4, these huge, interconnecting air bubble masses are quite successful at rallying victims. The whale then starts to ascend towards the surface in the newly generated bubbles. It continues to bubble as it ascends, and as it gets closer to its intended prey, the bubble circle gets smaller and the target gets smaller. This behavior allows the hunter to remain hidden from the prey while also locating the prey, immobilizing it, and taking it by surprise [11].



Figure 4. Bubble Hunting of Humpback Whales [10]

#### 4.1. WOA Components

The three components of the whale optimization algorithm's hunting strategy are circling the prey, approaching the prey, and looking for prey [12].

**Encircling prey:** Since the optimum solution to optimization issues is unknown, the algorithm's best result or a point very close to it is taken to be the ideal answer. According to the best solution value found, the positions of the other solutions are updated.

**Don't move towards the prey:** This stage is modeled in two parts as constricting the circle around the prey and spiral movement. Humpback whales in nature perform the constricting containment mechanism and the spiral movement at the same time. That is, a humpback whale can choose a constricting motion or a spiraling motion when updating its position relative to the best humpback whale. Therefore, WOA uses these two behaviors with a 50 percent probability.

**Search for prey (exploration):** The global solution determines the new positions of the solution candidates around a randomly chosen solution candidate instead of calculating the best known solution candidate.

The flowchart of the whale algorithm is shown Figure 5. In the flowchart  $X^*$  is the best available position vector,  $\vec{X}$  the location vector of the relevant search agent,  $\vec{A}$  and  $\vec{C}$  coefficient vector,  $r$  [0 1] takes a random value in the range,  $\vec{a}$  decreases linearly from 2 to 0 during the iteration.  $D$  where  $p$  is the distance between *whale* and *prey*, spiral motion.

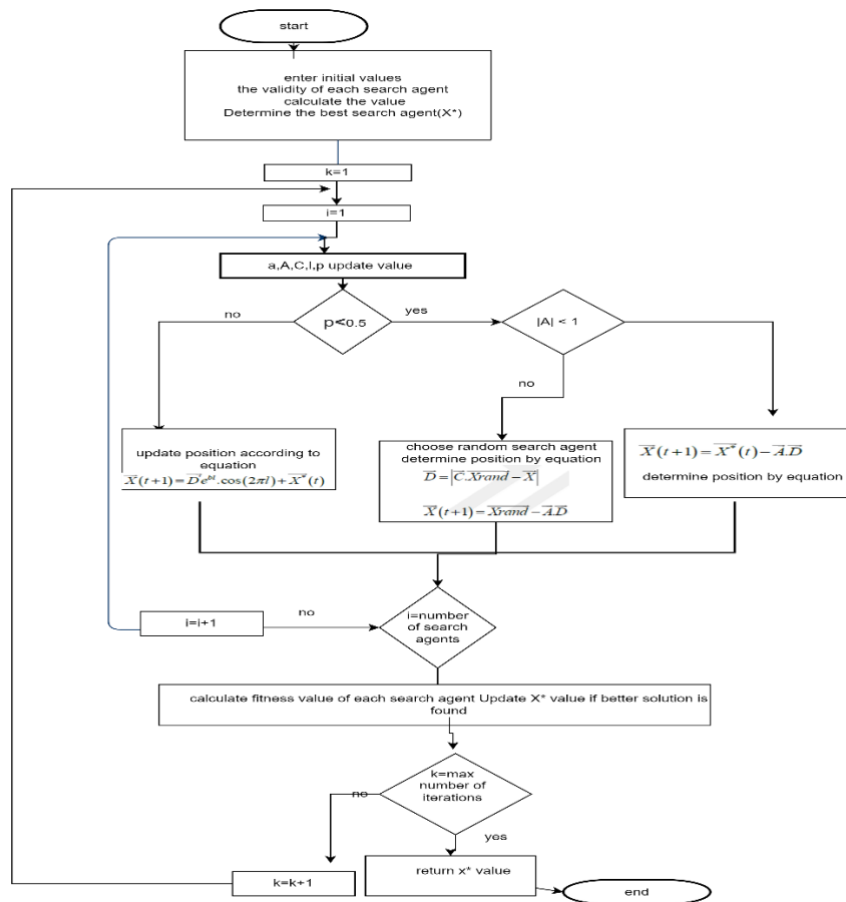


Figure 5. Flowchart of Whale Optimization Algorithm

## 5. ALO Algorithm

Ant lions construct their cone-shaped traps by constructing a circular trail to the ant colonies. They then wait for the ants to fall into the trap by burrowing themselves into the bottom of the cone, which serves as the trap's pointy end. The ant lions begin tossing sand as soon as the ants enter the trap to keep them from escaping and to help them fall to the bottom of the trap. Finally, they swallow the ants, which they slide to the bottom of the trap, with their large jaws. After each hunting job developed in this way, ant lions prepare their traps for a new hunt. The hunting tactic of the ant lion is shown in Figure 6 [13].

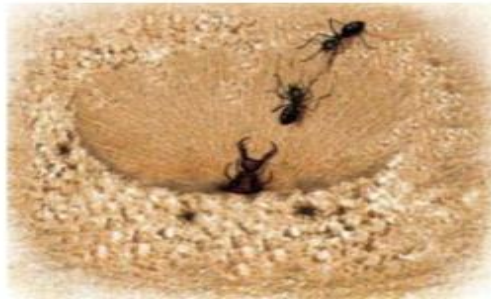


Figure 6. Ant lion hunting [12]

### 5.1. Components of the ALO

The ALO algorithm replicates the five essential phases of hunting: ant random wander, trap setting, ant capture in a trap, prey capture, and trap rebuilding [14]. The flow diagram of the ALO algorithm is given in Figure 7.

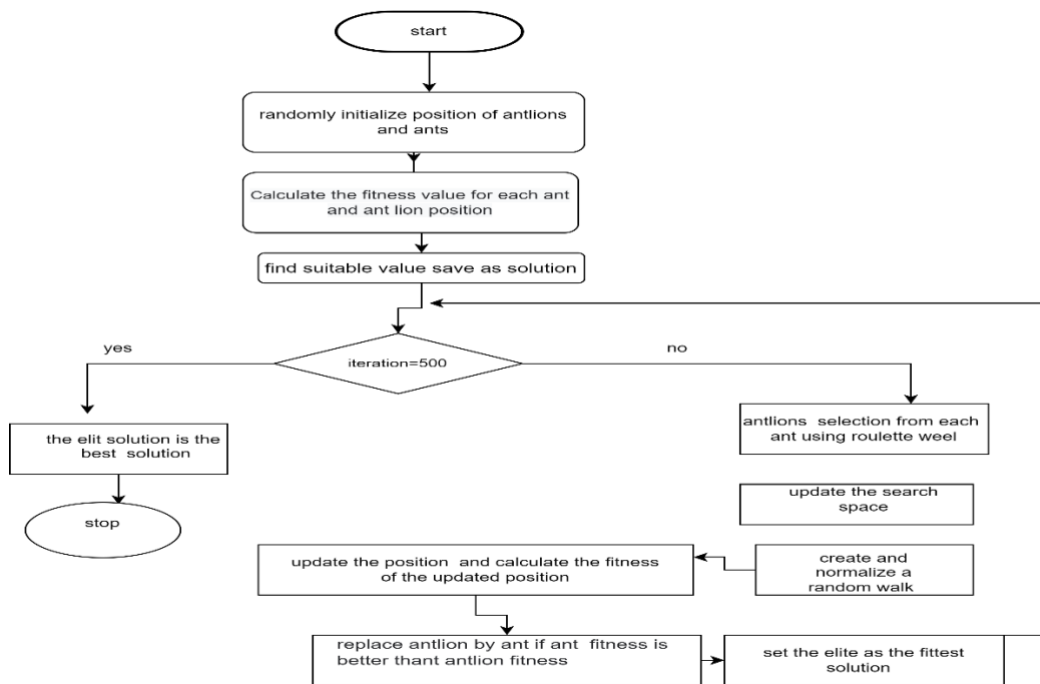


Figure 7. Flowchart of Ant Lion Algorithm

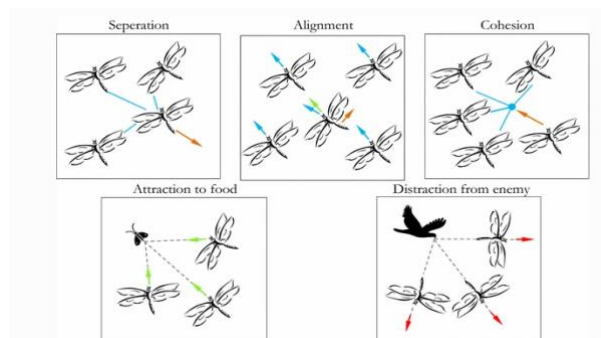
## 6. DA Algorithm

The main inspiration for the creation of the dragonfly algorithm is the static and dynamic swarming behavior of dragonflies. These two characteristics resemble the two primary stages of optimization where metaheuristics are employed: research and exploitation. The primary goal of the exploration phase is to see how dragonflies organize into smaller swarms and fly over various places in a static swarm. The exploitation phase benefits from dragonfly static swarms because they fly in larger flocks and in a single direction [15].

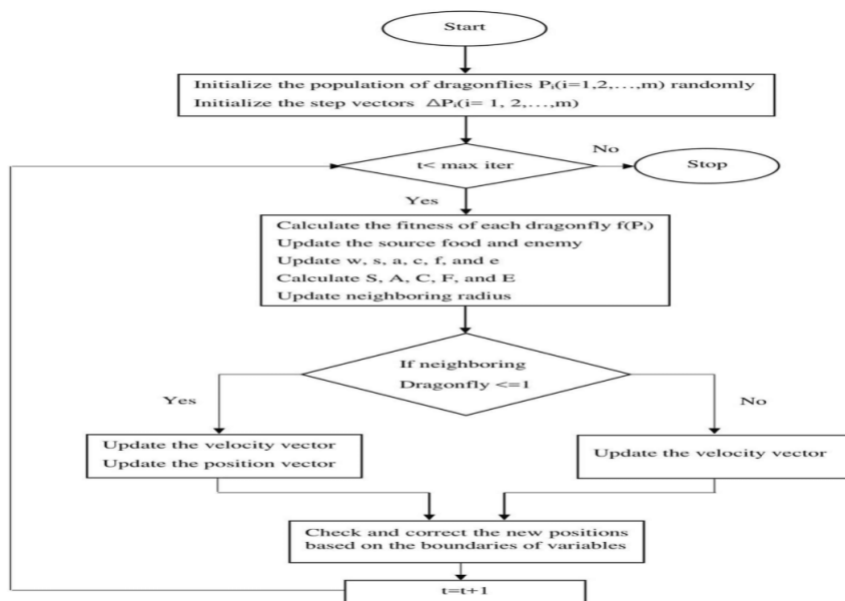
### 6.1. Components of the DA

According to Reynold, the swarming behavior of dragonflies falls within three basic principles.

- a- **Separation**; means to avoid collision with other people in the neighbourhood.
- b- **Alignment**; shows the speed compatibility with other individuals in the neighborhood.
- c- **Cohesion**; refers to the tendency of individuals towards the center of the neighborhood mass.
- d- **Attraction for food**; the main purpose of any swarm is to survive and therefore individuals must tend towards their food source.
- e- **Enemy**; the flock can be disturbed by outside enemies. These components are shown in Figure 8. The flow diagram of the dragonfly algorithm is shown in Figure 9.



**Figure 8.** Modeling the behavior of dragonflies ((a) Separation, (b) alignment, (c) cohesion, (d) attraction for food, (e) escape from the enemy) [16]



**Figure 9.** Flow diagram of the DA



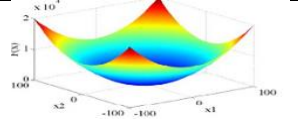
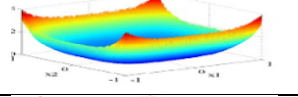
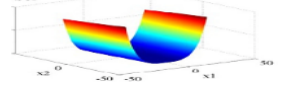
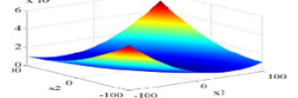
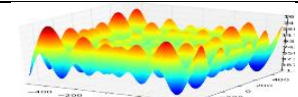
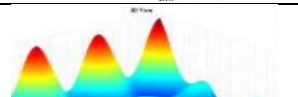
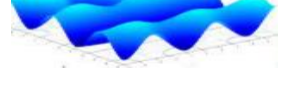
In the flowchart of the dragonfly algorithm Separation behavior  $S_i$ , Alignment behavior  $A_i$ , Association behavior  $C_i$ , Food source  $FI$ , Distracting enemy  $E_i$ , Position of dragonflies  $X_i$ , Direction of dragonflies movement

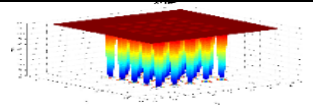
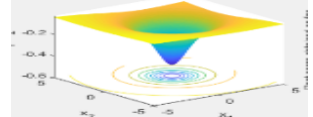
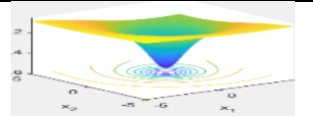
**7. Benchmark Functions and Experimental Results**

All of the experiments were carried out on a Windows 10 operating system computer with Intel i5 processor, 4 GB graphics card, and 16 GB RAM. All algorithms were run under equal conditions. The number of candidate solutions was determined as 10 and the number of iterations was determined as 100. During the experiments, each algorithm was run independently 10 times. During the experiments, the best performance of the algorithms as "min", the worst performance as "max", the mean values "mean" and the standard deviation "std" are shown in Table 3.

The performance of metaheuristic optimization algorithms has been tested in 10 unimodal and multimodal quality test functions. The mathematical expression, size, low-up, and 3-D graphics of these functions are given in Table 2. The performances of metaheuristic algorithms are shown in Table 3.

**Table 2.** Benchmark function that we used in our study

	Mathematical expression	Function	Size	Low-Up	3-D Graphic
f1	$\sum_{i=1}^n X_i^n$	Sphere	30	[-100, 100]	
f2	$\sum_{i=1}^n iX_i^4 + random[0,1]$	Quartic	30	[-1.28, 1.28]	
f3	$\sum_{i=1}^{n-1} [100(X_{i+1} - X_i^2) + (X_i - 1)^2]$	Rosenbrock	30	[-30, 30]	
f4	$\sum_{i=1}^n (\sum_{j=1}^i X_j)^2$	Schwefel1.2	30	[-100, 100]	
f5	$\sum_{i=1}^n -X_i \sin(\sqrt{ x_i })$	Schwefel	30	[-500, 500]	
f6	$\frac{\pi}{\mu} \left\{ (10 \sin(\pi y_1)) + \sum_{i=1}^{n-1} (y_{i-1})^2 [1 + 10 \sin^2(\pi y_{i+1}) + (y_n - 1)^2] \right\}$ $+ \sum_{i=1}^n \mu(X_i, 10, 100, 4)$ $+ \frac{X_i + 1}{4} \mu(X_i, a, k, m)$ $= \begin{cases} k(X_i - a)^m & X_i > a \\ 0 & -a < X_i < a \\ k(-X_i - a)^m & X_i < -a \end{cases}$	Penalized	30	[-50, 50]	
f7	$\sum_{i=1}^{11} [a_i - \frac{X_1 (b_i^2 + b_i X_2)}{b_i^2 + b_i X_3 + X_4}]^2$	Kowalik	4	[-5, 5]	

f8	$\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (X_i - a_i)^6}$	Foxholes	2	[-65536, 65536]	
F9	$\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	Shekel5	4	[0, 10]	
f10	$\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	Shekel10	4	[0, 10]	

**Table 3.** Performance ranking of each algorithm for each function

		AHA	DA	WOA	ALO	GWO
<b>f1</b>	Min	9.12E-152	1.1907609	1.35E-82	0.0000966	3.86E-25
	Max	1.73E-138	9.844576	2.84E-70	0.00047257	8.91E-23
	Mean	2.53E-139	5.07500656	3.45E-71	0.000315656	2.20E-23
	Std	5.4949E-139	3.377311435	8.86385E-71	0.000149881	3.07612E-23
<b>f2</b>	Min	0.0001003	0.10041	0.00014927	0.071756	0.0004015
	Max	0.76047	0.76356	0.017281	0.27236	0.0038547
	Mean	0.120642553	0.475331	0.00293937	0.1680026	0.001711651
	Std	0.264469479	0.238954597	0.005207293	0.062277744	0.001037898
<b>f3</b>	Min	2.62052	10.8272623	27.4626	21.359	26.0935
	Max	2.73431	48.8701355	28.7498	2041.9985	28.7489
	Mean	2.660833	20.97074773	27.82521	461.82734	27.51782
	Std	0.035851183	12.11520826	0.362612421	685.0012458	1.119730584
<b>f4</b>	Min	2.71E-139	4984.86	23598.49	99.67	2.17E-03
	Max	8.82E-126	86499.82	68700.99	3274.40	4.34E-01
	Mean	8.83E-127	23052.09	47879.37	2256.25	7.97E-02
	Std	2.7897E-126	24303.27	14954.70	885.64	0.137797611
<b>f5</b>	Min	-7204.36	-65337.08	-120688.57	-7585.36	-7650.84
	Max	-5562.74	-6762.20	-8307.41	-5117.15	-5150.69
	Mean	-6558.14	-50808.21	-21278.14	-6355.05	-6203.37
	Std	570.53	16281660.96	34971.94	846.92	735.62
<b>f6</b>	Min	0.00010648	1.60612	0.0054591	4.7364	0.022058
	Max	0.79688	9.9621	0.15795	22.2252	0.090913
	Mean	0.08077894	3.9121777	0.02751851	10.70978	0.0448123
	Std	0.251620616	2.623666763	0.046326745	5.217910697	0.019629011
<b>f7</b>	Min	0.0003404	0.00065729	0.00032828	0.0003111	0.0003075
	Max	0.0012634	0.0016555	0.0022519	0.0012446	0.020363
	Mean	0.000603179	0.001035532	0.001099359	0.000896353	0.004451198
	Std	0.00028031	0.000344255	0.000861785	0.000261617	0.008390604
<b>f8</b>	Min	0.998	0.998	0.998	0.998	0.998
	Max	10.001	1.992	10.7632	5.9288	12.6705
	Mean	2.380002	1.3956	5.79437	1.88829	4.81292
	Std	2.759779274	0.513299393	4.492096588	1.574072941	4.645515959
<b>f9</b>	Min	-10.15	-10.15	-10.15	-10.15	-26.30
	Max	-5.04	-5.05	-2.62	-2.6305	-10.15
	Mean	-7.47	-7.11	-7.76	-6.11858	-11.76
	Std	2.56	5.18	2.95	2.94	5.10
<b>f10</b>	Min	-10.5303	-10.5364	-10.53	-10.5364	-10.5358
	Max	-5.1253	-5.1285	-2.42	-1.8595	-10.5333
	Mean	-8.29517	-8.38704	-6.20	-5.53283	-10.53457
	Std	2.728681234	2.77428543	3.170572096	3.623704507	0.000938142

## 8. Conclusion

Various optimization algorithms have been created by inspiring different living things in nature. These are used in solving today's complex engineering problems. In previous studies, AHA's performance has been tested against many algorithms such as Particle swarm algorithm, Salp swarm algorithm, Differential convolution algorithm, Butterfly algorithm, Shade algorithm, which are the first optimization algorithms of AHA, and it has been proven that it gives faster results than the analyzed algorithms and provides the best convergence to the global optimum [6]. In this study, we tested the performance of AHA with current optimization algorithms. We observed that AHA is faster and gives more successful results. In f5, f7, and f9 functions, AHA generally achieved a worse min value than other algorithms. AHA algorithm obtained similar min values with other algorithms in f8 and f10 functions. In addition, the AHA algorithm obtained better results than other algorithms in f1, f2, f3, f4, and f6 functions. In the future, it is planned to test the performance of the metaheuristic algorithms examined in this study on limited real-world problems.

## References

- [1] Murty KG. Optimization models for decision making: Volume. University of Michigan, Ann Arbor, USA, 2003.
- [2] Baydoğan C. Sentiment Analysis in Social Networks Using Social Spider Optimization Algorithm. *Tehnički vjesnik* 2021; 28(6): 1943-1951.
- [3] Winston WL. Operations research: applications and algorithms. Cengage Learning, USA, 2022.
- [4] Baydoğan C, Alatas B. Metaheuristic ant lion and moth flame optimization-based novel approach for automatic detection of hate speech in online social networks. *IEEE Access* 2021; 9, 110047-110062.
- [5] Ehlers S. A procedure to optimize ship side structures for crashworthiness. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 2010; 224(1): 1-11.
- [6] Zhao W, Wang L, Mirjalili S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Comput. Methods Appl Mech Eng* 2022; 388, 114194.
- [7] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Adv Eng Software* 2014; 69: 46-61.
- [8] Asghari K, Masdari M, Gharehchopogh, FS, Saneifard R. A chaotic and hybrid gray wolf-whale algorithm for solving continuous optimization problems. *Prog Artif Intell* 2021; 10(3): 349-374.
- [9] Şenel FA, Gökçe F, Yüksel AS, Yiğit T. A novel hybrid PSO–GWO algorithm for optimization problems. *Eng Comput* 2019; 35(4): 1359-1373.
- [10] Mirjalili S, Lewis A. The whale optimization algorithm. *Adv Eng Software* 2016; 95: 51-67.
- [11] Gharehchopogh FS, Gholizadeh H. A comprehensive survey: Whale Optimization Algorithm and its applications. *Swarm Evol Comput* 2019; 48: 1-24.
- [12] Deepa R, Venkataraman R. Enhancing Whale Optimization Algorithm with Levy Flight for coverage optimization in wireless sensor networks. *Computers & Electrical Engineering* 2021; 94, 107359.
- [13] Kılıç H, Yüzgeç U. Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling. *Comput Ind Eng* 2019; 132: 166-186.
- [14] Yue X, Zhang H. A novel industrial image contrast enhancement technique based on an improved ant lion optimizer. *Arabian J Sci Eng* 2021; 46(4): 3235-3246.
- [15] Acı Çİ, Gülcan H. A modified dragonfly optimization algorithm for single-and multiobjective problems using Brownian motion. *Comput Intell Neurosci* 2019; 2019, 6871298: 1-17.
- [16] Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput Appl* 2016; 27: 1053-1073.