ISSN 2007-9737

Dragonfly Algorithm for Benchmark Mathematical Functions Optimization

Hector M. Guajardo, Fevrier Valdez*

Tecnológico Nacional de México, Campus Tijuana, Mexico

fevrier@tectijuana.mx

Abstract. In this paper, we study the dragonfly algorithm, an optimization method derived from the observation of nature and mathematical modeled after the swarming behavior of dragonflies. Xin-She Yang devised this approach, which has been applied to various optimization challenges. [1]. The algorithm effectively explores the search space by imitating dragonfly behaviors like hunting for prey, fleeing from predators, and swarming. The method consists in apply Type-1 Fuzzy Logic to some of the parameters of the algorithm, in this case, W and Betha to analyze the results applied to the mathematical functions F1 through F10 included in this paper, once that we have applied the adaptation of parameters, we will review the results compared with the rest of the papers that implement the same mathematical functions, so we can have a general idea if this method can be reliable.

Keywords: Optimization, dragonfly algorithm, bio-inspired, type-1 fuzzy logic.

1 Introduction

Search and optimization algorithms such as Particle Swarm Optimization (PSO), Differential Evolution (DE), Genetic Algorithm (GA), Firefly Algorithm (FA), and Dragonfly Algorithm (DA) have proven to be efficient in terms of speed and convergence for certain types of problems, so we expect the optimization algorithm to work efficiently applied to intelligent computing optimization problems. These algorithms are based on bioinspired principles and have been extensively studied and tested in scientific literature.

The choice of a specific algorithm depends on the optimization problem in question and its characteristics. However, combining different search and optimization algorithms in an algorithm can further improve efficiency in terms of convergence speed and the quality of the solution obtained. Algorithms that combine different search and optimization strategies can take advantage of the strengths of each of the algorithms and overcome their limitations.

In summary, an optimization algorithm combining PSO, DE, GA, and FA are expected to perform efficiently when applied to optimization problems due to its proven effectiveness in terms of speed of convergence in solving optimization problems. Where there are insects like dragonflies, fireflies, and damselflies, there are many more insects with the same similarities for hunting, reproduction, or in matters of movement on the entire ecosystem.

Observing the behaviors and structures of organisms in nature often suggests they perform their functions exceptionally well. Numerous studies have been conducted on this subject, including one by Xin-She Yang that began with publications pertaining to the firefly algorithm. The classification of many species of related insects will be reviewed in the following study [2].

Metaheuristics encompass broad strategies that skillfully blend different methodologies to navigate through the solution space. In design of optimization, the design objective can be as simple as maximizing production efficiency or minimizing production costs. An optimization algorithm is a technique that compares several solutions repeatedly until an ideal or feasible answer is identified. Currently, two types of optimization methods are frequently employed.

To transition from one solution to the next, deterministic algorithms employ a set of rules.

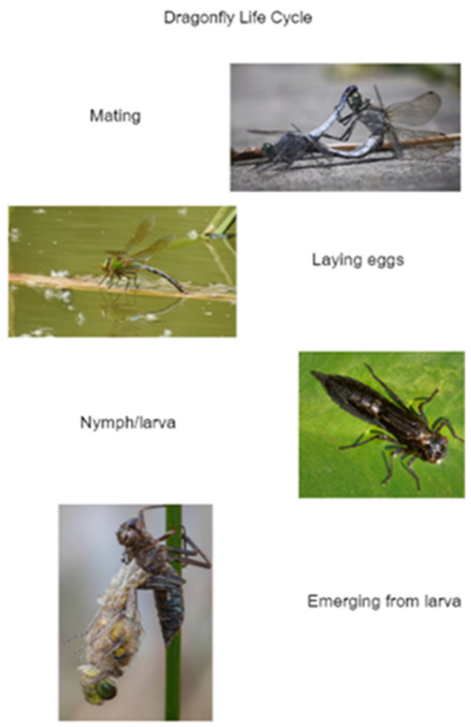


Fig. 1. The dragonfly cycle represents the steps the dragonfly must go through to become an adult dragonfly

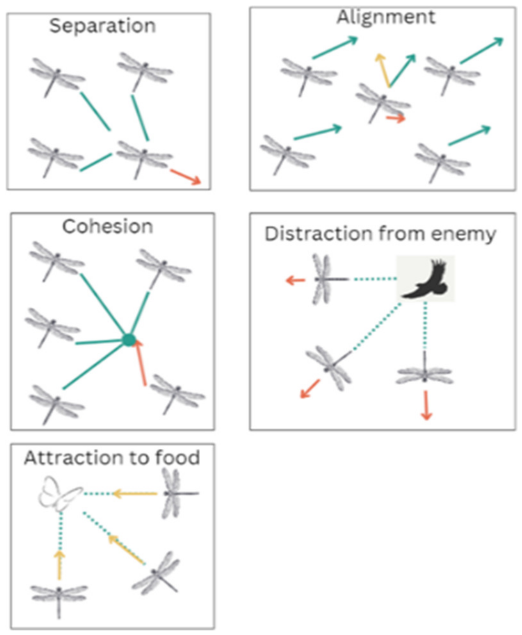


Fig. 2. The dragonfly patterns between individuals in the swarm

These algorithms have been effectively employed to solve a variety of engineering design challenges [3].

Since stochastic algorithms include probabilistic translation rules and a random nature, they may execute in a different sequence or produce a different outcome each time they are run with the same input.

The layout of this article is outlined as follows: 1. Introduction, where we provide a summary of the article's content; 2. Nature Inspiration the new optimization technique we offer is based on organic inspiration; 3. Literature review, which includes all the studies relevant to the topics of this article; 4. The presentation and implementation of the Dragonfly Algorithm (DA); 5. Type-1 fuzzy logic explanation, 6. Results, analysis, and comparison of the experiments performed; 7. Analytical conclusions and a summary of the investigation are described in this article.

2 Nature Inspiration

In some cultures, dragonflies were called kachi-mushi (victorious insects) because they only fly forward, which gave them the character of those who never retreat and always move forward, whatever the circumstances.

Thus, it became a symbol of strength, courage and determination. About 5,000 species of dragonflies are known and the algorithm is inspired by static behavior and dynamic behavior all dragonflies can move together to the same place for example a migration.

In the following, we can review the life cycle of the dragonfly.

The adult dragonfly starts to mate, after that, they will lay their eggs in or around the water eggs in the water, after the eggs will become a larva and after that the dragonfly will emerge. This can be appreciated in Fig 1.

Based on these factors, it was suggested an algorithm that follows the model of insects and how they adjust to physiological changes by sharing resources and communicating to survive and grow. This algorithm was called "dragonfly".

Natural optimization methods have proven to be adaptable, flexible, and efficient in handling practical problems.

ISSN 2007-9737

Algorithm 1 Dragonfly algorithm pseudocode

```
Initialize Dragonfly population randomly
Initialize Step vector/Size for dragonfly
While (current iteration < maximum iteration)
    Calculate fitness values for each dragonfly.
    Update food sources and enemy.
    Update parameters $w$, $s$, $a$, $c$, $f$ and $e$.
    Calculate $S$, $A$, $C$ and $F$.
    Update the neighboring radius.
    if dragonfly has at least one neighboring dragonfly.
        Update velocity and position.
    else
        Update position vector
    else if
    Check and correct the new positions based on the
    boundaries of variables.
End While
```

The fact that there is currently no optimization technique that can handle all problems is well recognized [5]. There are everyday issues that can be solved in business, economics, research, and other fields. There is a wide variety of optimization techniques, and some are more effective than others in solving specific problems. There are various metaheuristic optimization techniques, for example, Particle Swarm Optimization (PSO) [10], Artificial Colony Optimization (ACO) [8, 11], Artificial Bee Algorithm (ABC) [9], Firefly Algorithm (FA) [12], as well as other algorithms based on Hill climbing swarms [13], genetic algorithms (GA) [6], and different techniques based on trajectories. Differential evolution (DE) [14] and genetic programming (GP) [16] are examples of evolutionary algorithms.

Table 1. Description of the algorithm parameters

Parameter	Description	Values
Population	Population size	40
Boundaries	Number of boundaries	2
Dimensions	Dimension's size	8, 10, 16, 32
Iterations	Iterations size	500
S	Separation weight	
A	Alignment weight	
C	Cohesion	
F	Food factor	
E	Enemy factor	
w	Inertia weight	
S_i	Separation of the i=th individual	
A_i	Alignment of the i=th individual	
C_i	Cohesion of the i=th individual	
F_i	Food source of the i=th individual	
E_i	Enemy position of the i=th individual	
t	Current iteration	

3 Study of Literature

There are several social behaviors used in nature to carry out various tasks. Although survival is the goal of all individuals and collective actions, organisms collaborate and interact in groups for a variety of purposes, including hunting, defending, navigating, and foraging.

Wolf packs, for example, have some of the best-structured social interactions for hunting. Wolves often follow a social hierarchy to pursue prey in various ways: chasing, circling, tormenting, and attacking. [17]. Holland authored a book detailing the development of genetic algorithms (GAs). De Jong concluded his research by showcasing the considerable potential and robustness of evolutionary algorithms across various objective functions, including those that are noisy, multimodal, or discontinuous [6]. To minimize their learning and prediction errors through iterative trial and error, artificial neural networks, support vector machines, and other machine learning approaches are genetic algorithms and can be considered a heuristic optimization methodology.

In 1961 Van Bergeijk, W. A, Harmon, L. D. and Levinson, J. Z., and Harmon, L. D. proposed artificial neurons as simple information processing units [19, 20, 21].

Particle Swarm Optimization (PSO), an optimization method inspired by the collective intelligence of fish, birds, and even human beings, was created in 1995 by James Kennedy and Russell C. Eberhart [22, 23].

Table 2. CEC2013 math functions

Fun	Name	Range	Nature
F1	Sphere	[-5.12, 5.12]	U
F2	Rosenbrock	[-5, 10]	U
F3	Griewank	[-600, 600]	M
F4	Rastrigin	[-5.12, 5.12]	M
F5	Ackley	[-32.768, 32.768]	M

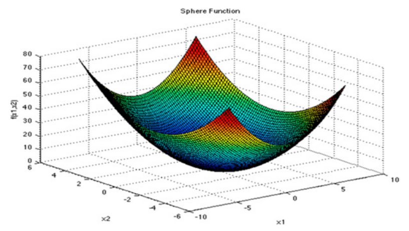
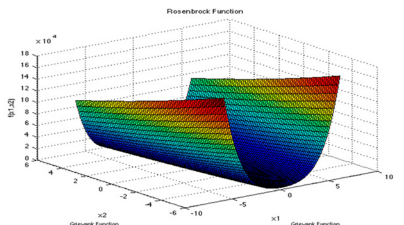
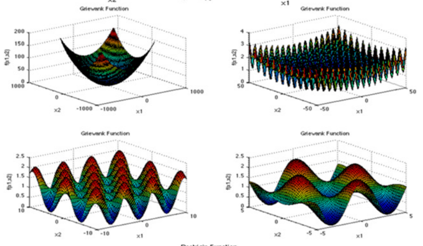
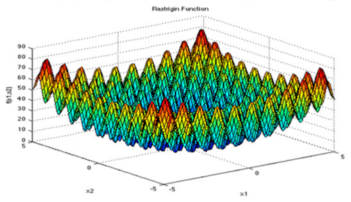
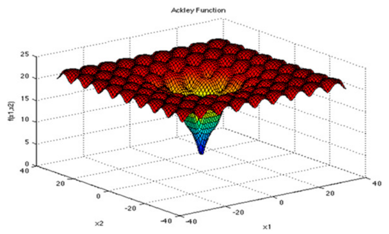
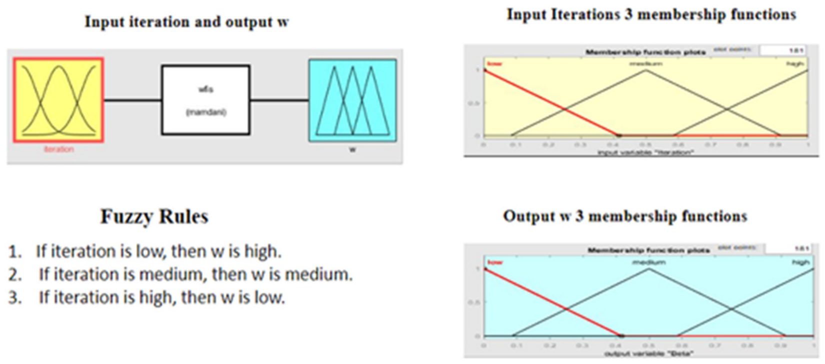
Fun	Graph	Equation
1		$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2$
2		$f(\boldsymbol{x}) = \sum_{i=1}^{d-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$
3		$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
4		$f(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$
5		$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sqrt{\sum_{i=1}^{d}\cos(cx_i)}\right) + a + \exp(1)$

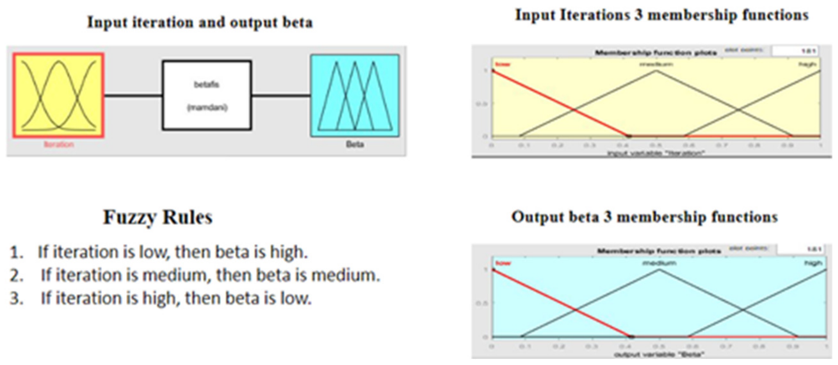
Fig. 4. Plots of five CEC2013 mathematical functions


Input iteration and output w
iteration
wfis
(mamdani)
w
Input Iterations 3 membership functions
Membership function plots
low
medium
high
input variable "iteration"
Output w 3 membership functions
output variable "Beta"

Fuzzy Rules

1. If iteration is low, then w is high.
2. If iteration is medium, then w is medium.
3. If iteration is high, then w is low.

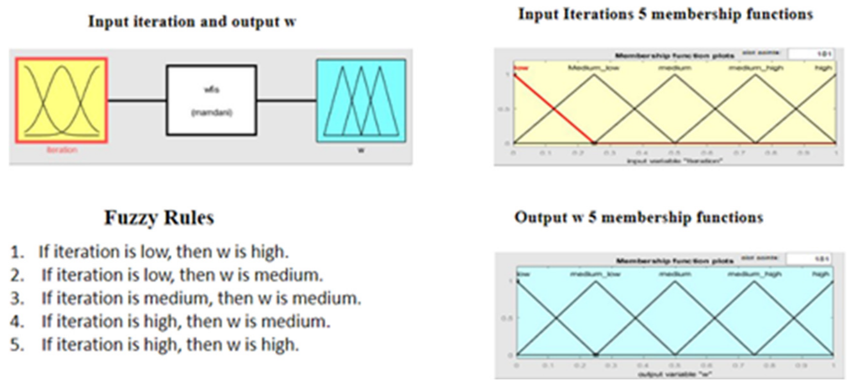
Fig. 5 Type-1 fuzzy inference systems w parameter


Input iteration and output beta
iteration
betafis
(mamdani)
Beta
Input Iterations 3 membership functions
Membership function plots
low
medium
high
input variable "iteration"
Output beta 3 membership functions
output variable "Beta"

Fuzzy Rules

1. If iteration is low, then beta is high.
2. If iteration is medium, then beta is medium.
3. If iteration is high, then beta is low.

Fig. 6 Type-1 fuzzy inference systems beta parameter


Input iteration and output w
iteration
wfis
(mamdani)
w
Input Iterations 5 membership functions
Membership function plots
low
Medium_low
medium
medium_high
high
input variable "iteration"
Output w 5 membership functions
output variable "w"

Fuzzy Rules

1. If iteration is low, then w is high.
2. If iteration is low, then w is medium.
3. If iteration is medium, then w is medium.
4. If iteration is high, then w is medium.
5. If iteration is high, then w is high.

Fig. 7. Type-1 fuzzy inference systems beta parameter

With time, the PSO method has demonstrated its superiority over conventional algorithms and genetic algorithms in specific problem domains, though it may not be suitable for every scenario. There isn't a universal algorithm that excels in all optimization problems; hence, current research aims to identify the most effective and efficient algorithm(s) for particular tasks. D. H. Wolpert and W. G. Macready introduced the No-Free Lunch theorems to caution the scientific community that if algorithm A outperforms algorithm B for certain optimization functions, then B is likely to outperform A for other functions. [24]. Over time, researchers S. Nakrani and C. Tovey suggested the honey bee algorithm and its use as a foraging algorithm for problems including multimodal and dynamic optimization. [25].

Table 3. Comparison results for 30 dimensions of CMOA, DA and DA with Type-1

Fun	DA		CMOA		DA Type-1	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
F1	2.85E-18	7.16E-01	8.11E-09	4.79E-09	1.25E+00	2.88E+0
F2	7.60E+00	6.79E+0	6.58E-09	3.32E-09	4.41E-01	4.79E-01
F3	1.03E-02	4.69E-03	1.64E-09	1.39E-09	1.00E+01	5.15E+01
F4	1.60E+01	9.48E+0	8.13E-09	4.42E-09	1.25E+00	1.55E+0
F5	2.31E-01	4.87E-01	1.44E-09	1.21E-09	2.11E+02	3.21E+02

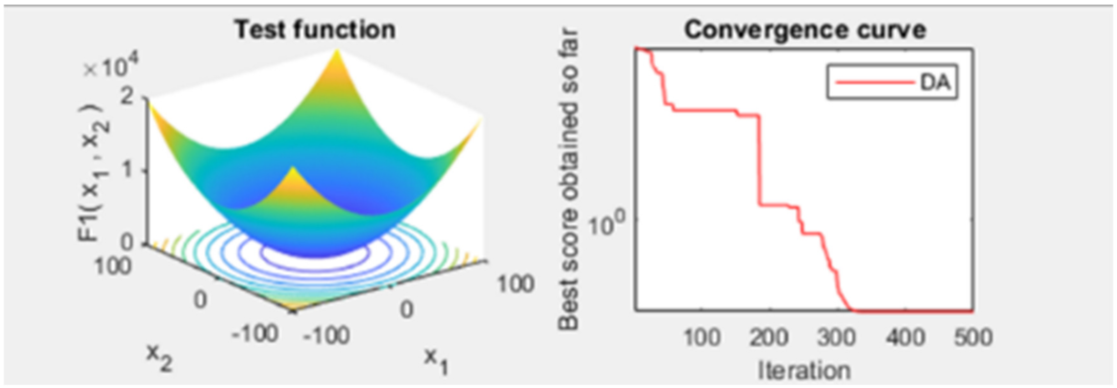


Fig. 8. Display and convergence for function 1

Table 4. Results for function 1 experiments

500 Iterations				
40 Dragonflies				
Exp	**8 Dim**	**10 Dim**	**16 Dim**	**32 Dim**
Ave	1.25E+00	4.09E+00	1.30E+02	1.53E+03
Std	2.88E+00	1.19E+01	1.70E+02	6.96E+02

The techniques were inspired by how actual bees feed in the nature. To identify spreaders utilizing a variety of targets, Amir S. and Ahman Z. developed the artificial bee colony (ABC) algorithm in 2020. [9]. Particle Swarm Optimization (PSO) and the Fire-fly Algorithm (FA), inspired by the flashing patterns of fireflies, were combined in a practical project undertaken by Khennak, I., Drias, H., and Drias, Y. [12].

Seyedali Mirjalili et al. proposed the Grey Wolf Optimizer (GWO) in 2013 [25], Inspired by grey wolves (Canis lupus), it imitates the natural leadership structure and hunting strategy of these canines. The Coyote Optimization Algorithm, which Juliano Pierezan and Leandro dos Santos Coelho created in 2018, is a population-based metaheuristic for optimization that draws inspiration from the canis latrans species [26].

In 2020, Abdolkarim Mohammadi-Balani and his team introduced their innovative Golden Eagle Optimizer algorithm, designed to adjust speed at different points along a spiral trajectory, mimicking the hunting behavior of golden eagles. [27].

Additionally, new metaheuristic algorithms that are better than others at solving a particular kind of problem will continue to be developed.

4 Dragonfly Algorithm (DA)

In nature, practically all other little insects are preyed upon by dragonflies, which are thought of as small predators. Additionally, nymph dragonflies eat other maritime insects and even small fish. The intriguing characteristic of dragonflies is their uncommon and unusual swarming behavior.

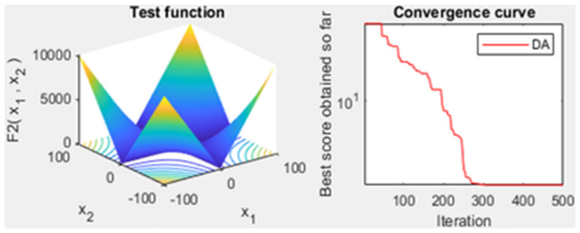


Fig. 9. Display and convergence for function 2

Table 5. Results for function 2 experiments

500 Iterations				
40 Dragonflies				
Exp	**8 Dim**	**10 Dim**	**16 Dim**	**32 Dim**
Ave	4.43E-01	1.00E-07	3.79E+00	1.48E+01
Std	4.71E-01	1.00E-07	1.55E+00	7.64E+00

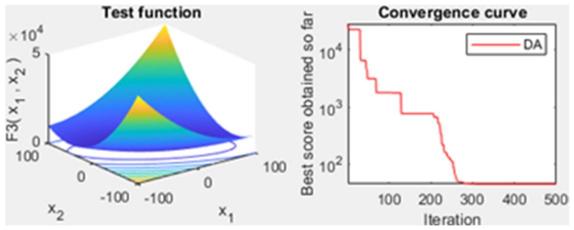


Fig. 10. Display and convergence for function 3

Table 6. Results for function 3 experiments

500 Iterations				
40 Dragonflies				
Exp	**8 Dim**	**10 Dim**	**16 Dim**	**32 Dim**
Ave	2.69E+01	1.80E+02	1.46E+03	1.30E+04
Std	5.08E+01	2.43E+02	1.70E+03	9.96E+03

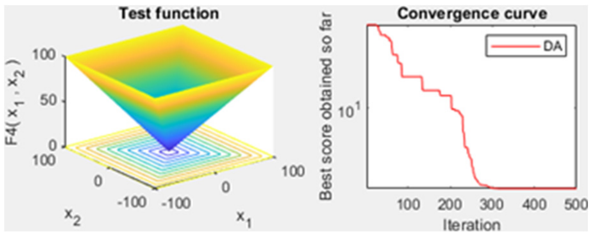


Fig. 11. Display and convergence for function 4

Table 7. Results for function 4 experiments

500 Iterations				
40 Dragonflies				
Exp	**8 Dim**	**10 Dim**	**16 Dim**	**32 Dim**
Ave	1.24E+00	1.85E+00	9.17E+00	2.93E+01
Std	1.52E+00	1.36E+00	5.81E+00	1.06E+01

Only two things cause dragonflies to swarm: migration and hunting. Both are referred to as swarms—the former as a static (feeding) swarm and the latter as a dynamic (migratory) swarm.

As already mentioned, the DA is an optimization technique that draws inspiration from the same-named bug [28]. The static and dynamic characteristics of swarms serve as the primary source of inspiration for the DA algorithm.

These two are extremely like the exploration and exploitation phases of metaheuristic optimization. The main goal of the exploration phase is for dragonflies to organize into sub-swarms and fly in a static swarm over numerous locations.

Certainly, during times of static swarming, it is observed that dragonflies tend to fly together in larger groups, all aligning their flight paths—a behavior that is notably advantageous, particularly during the exploitation phase.

The primary goal of any swarm is survival, every member should be drawn to food sources and vigilant against external threats.

As demonstrated on Fig. 2 the five essential factors that affect how individuals in swarms update their positions considering these two behaviors. Swarm behavior follows three important principles:

1. Separation: Individual avoid static collision with neighbor:

$$S_j = \sum_{j=1}^{N} X - X_j. \quad (1)$$

2. Alignment: Individual velocity matched with neighbor individuals:

$$A_i = \sum_{j=1}^{N} V_j. \quad (2)$$

3. Cohesion: Individual tendency toward center of the herd:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X. \quad (3)$$

X: This represents the position of an individual, which typically denotes its current location or coordinates in each space.

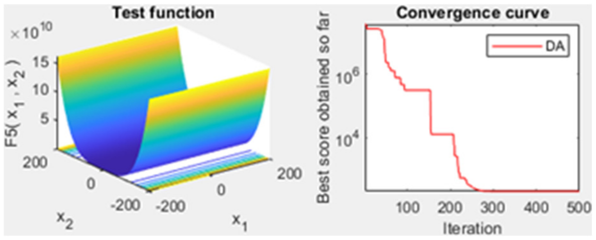


Fig. 12. Display and convergence for function 5

Table 8. Results for function 5 experiments

500 Iterations				
40 Dragonflies				
Exp	8 Dim	10 Dim	16 Dim	32 Dim
Ave	2.05E+02	3.51E+03	1.25E+04	1.88E+05
Std	3.17E+02	1.67E+04	2.86E+04	1.88E+05

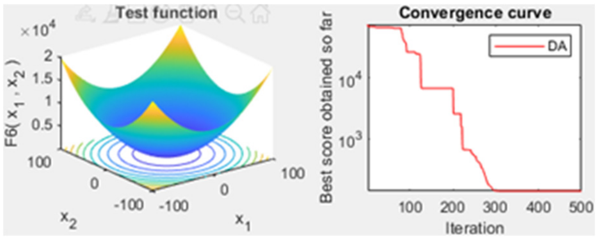


Fig. 13. Display and convergence for function 6

Table 9. Results for function 6 experiments

500 Iterations				
40 Dragonflies				
Exp	8 Dim	10 Dim	16 Dim	32 Dim
Ave	1.25E+00	2.76E+00	7.44E+01	1.50E+03
Std	5.38E+00	4.87E+00	1.17E+02	1.22E+03

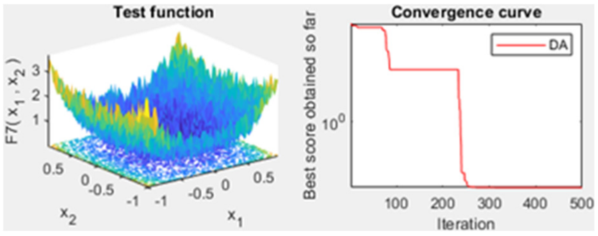


Fig. 14. Display and convergence for function 7

Table 10. Results for function 7 experiments

500 Iterations				
40 Dragonflies				
Exp	8 Dim	10 Dim	16 Dim	32 Dim
Ave	1.39E-02	2.73E-02	6.21E-02	5.83E-01
Std	1.13E-02	2.13E-02	5.26E-02	3.27E-01

V_j: This represents the velocity of an individual, which typically refers to the speed and direction at which it is moving.

N: This represents the number of neighborhoods or groups of individuals in your system. It is essentially a parameter that determines how individuals are grouped or organized into neighborhoods. Attraction to food source is calculated:

$$F_i = X^+ - X, \tag{4}$$

where:

X : Is the position of the current individual,

X^+: Is the area of the food.

Distraction from enemy is calculated:

$$E_i = X^- + X, \tag{5}$$

where:

X : Is the position of the current individual,

X^-: Is the area of the enemy.

In this research, dragonfly behavior is supposed to be a combination of these five corrective patterns. Two vectors are used to update the position of artificial dragonflies in a search space and replicate their movements: step (ΔX) and position (X). The step vector represents the direction of the dragonfly movement and is described as follows:

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_i, \tag{6}$$

where s is the separation weight S_i representing $i = th$ individual's separation, a represents the alignment weight, A represents $i = th$ the individual's alignment, and c represents the cohesion weight. C_i is the $i = th$ individual's cohesiveness, f is the food factor, and F_i is the $i = th$ individual's food supply, e is the enemy factor, E_i is the $i = th$ individual's position of enemy, w is the inertia weight, and t is the iteration timer, in Algorithm 1 we can see with more detail the pseudocode from dragonfly algorithm.

Table 1 presents the names and concise descriptions of all the parameters applied in the Dragonfly algorithm. In the subsequent section of this paper, we provide the outcomes of the experimentation conducted in this study.

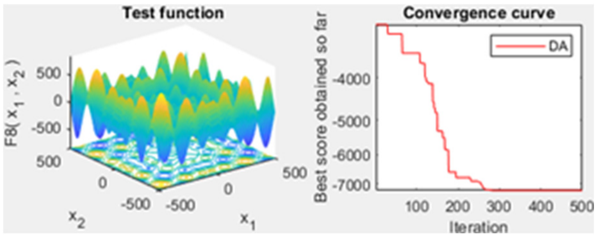


Fig. 15. Display and convergence for function 8

Table 11. Results for function 8 experiments

500 Iterations				
40 Dragonflies				
Exp	8 Dim	10 Dim	16 Dim	32 Dim
Ave	-2.48E+03	-2.89E+03	-3.78E+03	-5.78E+03
Std	3.47E+02	3.19E+02	4.74E+02	7.51E+02

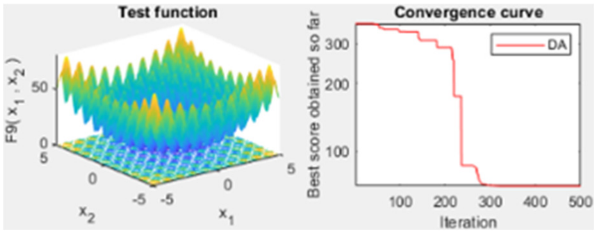


Fig. 16. Display and convergence for function 9

Table 12. Results for function 9 experiments

500 Iterations				
40 Dragonflies				
Exp	8 Dim	10 Dim	16 Dim	32 Dim
Ave	1.86E+01	2.77E+01	5.79E+01	1.56E+02
Std	1.03E+01	1.29E+01	2.05E+01	3.53E+01

Table 2 outlines the titles of five mathematical functions from the CEC2013 dataset, along with their corresponding scopes and characteristics, which were utilized in our investigation. The Fig. 4 presents graphical representations and mathematical equations for five out of the ten mathematical functions utilized in this study.

5 Type-1 Fuzzy Logic

Fuzzy logic, alternatively referred to as fuzzy sets theory, provides a mathematical framework for addressing reasoning and decision-making in scenarios characterized by uncertainty and imprecision. Unlike traditional binary logic where statements are either true or false, fuzzy logic allows for degrees of truth between 0 and 1, representing degrees of membership or truthfulness.

This allows for more nuanced modeling and analysis, particularly in areas where precise boundaries are difficult to define. [29] Key Concepts of Fuzzy Logic: Fuzzy Sets: Fuzzy sets are a fundamental concept in fuzzy logic, introduced by Lotfi Zadeh in 1965. Unlike classical sets where an element either belongs to a set or does not, fuzzy sets allow for degrees of membership.

In a fuzzy set, each element has a membership value that represents the degree to which the element belongs to the set. These membership values range between 0 and 1, where 0 indicates no membership, 1 indicates full membership, and values in between represent degrees of partial membership.

Fuzzy sets are especially useful for modeling uncertainties and vagueness present in many real-world systems. Membership Functions: Membership functions are mathematical functions that define the degree of membership of each element in a fuzzy set.

These functions map each element from the universal set to a real number in the interval [0, 1]. There are various types of membership functions, such as triangular, trapezoidal, Gaussian, and sigmoidal, each suited for different applications and interpretations.

The choice of membership function depends on the specific characteristics of the problem domain and the preferences of the modeler. Membership functions play a crucial role in fuzzy logic systems as they determine the degree to which fuzzy sets represent real-world phenomena. Fuzzy Operators: Fuzzy operators are mathematical operations defined on fuzzy sets that allow for combining and manipulating fuzzy information.

These operators extend classical set operations such as union, intersection, and complement to accommodate the degrees of membership associated with fuzzy sets. The basic fuzzy operators include union (OR), intersection (AND), and complement (NOT). Additionally, there are other operators like algebraic product, bounded sum, and drastic sum, each serving different purposes in fuzzy logic systems.

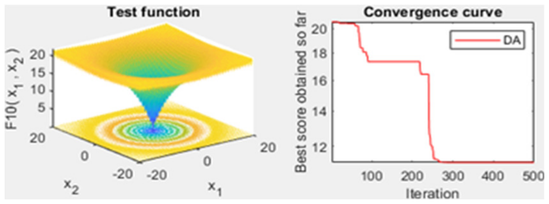


Fig. 17. Display and convergence for function 10

Table 13. Results for function 10 experiments

500 Iterations				
40 Dragonflies				
Exp	8 Dim	10 Dim	16 Dim	32 Dim
Ave	1.52E+00	2.57E+00	4.42E+00	9.53E+00
Std	1.14E+00	1.54E+00	1.41E+00	1.72E+00

Fuzzy operators are essential for performing reasoning and making decisions in fuzzy logic-based control systems, pattern recognition, and other applications. [32, 33, 34]. Example: Consider the concept of "temperature" in a room. Instead of categorizing it simply as "hot" or "cold," fuzzy logic allows for a more nuanced approach.

We might define a fuzzy set for "comfortable temperature" with a membership function that peaks around 22 degrees Celsius. Then, if the room is at 20 degrees, it might have a membership value of 0.8 in the "comfortable temperature" set, indicating it is somewhat comfortable but not perfect.

Adaptation of Type-1 Fuzzy Logic

We will present Type-1 fuzzy inference systems implemented for the parameters w and beta of the Dragonfly algorithm in the following table. Here, in Fig. 5, we can observe the inference functions with the iteration parameter as input and the parameter was output, with their three fuzzy rules:

Fuzzy Rules for the Parameter w:

1. If iteration is low, then w is high.
2. If iteration is medium, then w is medium.
3. If iteration is high, then w is low.

Here, in Fig. 6 we can observe the inference functions with the iteration parameter as input and the beta parameter as output, with their three fuzzy rules.

Fuzzy Rules for the Parameter Beta:

1. If iteration is low, then beta is high.
2. If iteration is medium, then beta is medium.
3. If iteration is high, then beta is low.

In Fig. 7, we can observe the inference functions with the iteration parameter as input and the parameter was output, with its five fuzzy rules:

Fuzzy Rules for the Parameter w:

1. If iteration is low, then w is high.
2. If iteration is low, then w is medium.
3. If iteration is medium, then w is medium.
4. If iteration is high, then w is medium.
5. If iteration is high, then w is high.

6 Results and Comparison

In Table 3, we present a comparison of outcomes derived from two distinct algorithms: The Dragonfly Algorithm (DA), where the Dragonfly algorithm is integrated with type-1 application to parameters such as Beta and w, and the Continuous Mycorrhiza Optimization Algorithm (CMOA). The table showcases the most favorable mean values and standard deviations obtained from each experiment across different functions.

The bio-inspired Dragonfly Algorithm serves as a remarkable example of how nature's mechanisms can provide creative and effective solutions to modern technological obstacles. Table 4 showcases the best results acquired from 30 experiments conducted across dimensions of 8, 10, 16, and 32.

These experiments employed a population of 40 dragonflies and a maximum of 500 iterations, tailored specifically for Function 1. Fig 8 depicts the graphical representation and convergence curve for Function 1.

Table 5 displays the best results achieved from 30 experiments carried out across dimensions of 8, 10, 16, and 32. These experiments employed a population of 40 dragonflies and a maximum of 500 iterations, all tailored for Function 2. Fig 9 visually represents the performance and convergence curve for Function 2.

Table 6 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32.

These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 3. In Fig. 10, it is illustrated the display and convergence curve for Function 3. Table 7 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32.

These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 4. In Fig. 11, we illustrate the display and convergence curve for Function 4.

Table 8 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32. These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 5. In Fig. 12, we illustrate the display and convergence curve for Function 5.

Table 9 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32. These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 6.

In Fig. 13, it is illustrated the display and convergence curve for Function 6. Table 10 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32. These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 7.

In Fig. 14, we illustrate the display and convergence curve for Function 7. Table 11 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32. These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 8.

In Fig. 15, we illustrate the display and convergence curve for Function 8. Table 12 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32.

These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 9. In Fig. 16, it is illustrated the display and convergence curve for Function 9.

Table 13 presents the optimal values obtained from 30 experiments conducted across dimensions of 8, 10, 16, and 32. These experiments utilized a population of 40 dragonflies and a maximum number of 500 iterations, all specifically designed for Function 10. Fig. 17 illustrates the display and convergence curve for Function 10.

7 Conclusions

In conclusion, when comparing the Dragonfly method with itself, utilizing a population of 40 Dragonflies and an iteration value of 500, but with the incorporation of Type-1 fuzzy logic adaptation for parameters w and beta, we can confidently state that much better results are obtained.

Furthermore, when compared to the CMOA algorithm in 50 dimensions, the Dragonfly method continues to yield superior outcomes, as evidenced in more detail in Table 3, significantly enhancing the results when mean and standard deviation are applied across all experiments conducted.

We will continue to generate results, with future work focusing on adapting Type-2 fuzzy logic to further compare outcomes with other algorithms. The findings will be shared with the community to support fellow researchers.

It is worth mentioning that while the DA method may not represent the ultimate optimization technique presently accessible, it does demonstrate potential and can be beneficial in particular optimization problem contexts.

As future work, we will consider other metaheuristics for the same approach, like in [35-40]. Later, we expect to utilize type-2 fuzzy logic

[41-44] for parameter adaptation in the dragon fly algorithm, as in [45-46].

References

1. **Yang, X. (2010).** A new metaheuristic bat-inspired algorithm. Studies in Computational Intelligence, Vol. 284, pp. 65–74. DOI: 10.1007/978-3-642-12538-6_6.
2. **Bybee, S. M., Kalkman, V. J., Erickson, R. J., Frandsen, P. B., Breinholt, J. W., Suvorov, A., Dijkstra, K. B., Cordero-Rivera, A., Skevington, J. H., Abbott, J. C., Herrera, M. S., Lemmon, A. R., Lemmon, E. M., Ware, J. L. (2021).** Phylogeny and classification of odonata using targeted genomics. Molecular Phylogenetics and Evolution, Vol. 160, pp. 107115. DOI: 10.1016/j.ympev.2021.107115.
3. **Osman, I. H., Kelly, J. P. (1996).** Meta-heuristics: An overview. Meta-Heuristics, pp. 1–21. DOI: 10.1007/978-1-4613-1361-8_1.
4. **Glover, F., Mulvey, J. M., Hoyland, K. (1996).** Solving dynamic stochastic control problems in finance using tabu search with variable scaling. Meta-Heuristics, pp. 429–448. DOI: 10.1007/978-1-4613-1361-8_26.
5. **Carreres-Prieto, D., Ybarra-Moreno, J., García, J. T., Cerdán-Cartagena, J. F. (2023).** A comparative analysis of neural networks and genetic algorithms to characterize wastewater from led spectrophotometry. Journal of Environmental Chemical Engineering, Vol. 11, No. 3, pp. 110219. DOI: 10.1016/j.jece.2023.110219.
6. **Holland, J. H. (1992).** Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. MIT Press. DOI: 10.7551/mitpress/1090.001.0001.
7. **Dorigo, M., Gambardella, L. C., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (2006).** Ant colony optimization and swarm intelligence. 5th international workshop, ANTS 2006, Brussels, Springer.
8. **Gutjahr, W. J. (2002).** Aco algorithms with guaranteed convergence to the optimal solution. Information Processing Letters, Vol. 82, No. 3, pp. 145–153. DOI: 10.1016/s0020-0190(01)00258-7.
9. **Sheikhahmadi, A., Zareie, A. (2020).** Identifying influential spreaders using multi-objective artificial bee colony optimization. Applied Soft Computing, Vol. 94, pp. 106436. DOI: 10.1016/j.asoc.2020.106436.
10. **Khennak, I., Drias, H., Drias, Y., Bendakir, F., Hamdi, S. (2022).** I/f-race tuned firefly algorithm and particle swarm optimization for k-medoids-based clustering. Evolutionary Intelligence, Vol. 16, No. 1, pp. 351–373. DOI: 10.1007/s12065-022-00794-z.
11. **Yang, X. (2013).** Cuckoo search and firefly algorithm: overview and analysis. Studies in Computational Intelligence, Vol. 516, pp. 1–26. DOI: 10.1007/978-3-319-02141-6_1.
12. **Jacobson, S. H., Yücesan, E. (2004).** Analyzing the performance of generalized hill climbing algorithms. Journal of Heuristics, Vol. 10, No. 4, pp. 387–405. DOI: 10.1023/b:heur.0000034712.48917.a9.
13. **Storn, R., Price, K. (1997).** Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, Vol. 11, No. 4, pp. 341–359. DOI: 10.1023/a:1008202821328.
14. **Cordes, K., Rosenhahn, B., Ostermann, J. (2011).** Increasing the accuracy of feature evaluation benchmarks using differential evolution. IEEE Symposium on Differential Evolution, pp. 1–8 DOI: 10.1109/sde.2011.5952056.
15. **Nicolau, M., Krawiec, K., Heywood, M. I., Castelli, M., García-Sánchez, P., Merelo, J. J., Rivas-Santos, V. M., Sim, K. (2014).** Genetic Programming. Lecture Notes in Computer Science, 17th European Conference, EuroGP 2014, Granada, Spain.
16. **Saif, F. A., Latip, R., Hanapi, Z. M., Shafinah, K. (2023).** Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. IEEE Access, Vol. 11, pp. 20635–20646. DOI: 10.1109/access.2023.3241240.

17. **Saophan, P., Pannakkong, W., Singhaphandu, R., Huynh, V. (2023).** Rapid production rescheduling for flow shop under machine failure disturbance using hybrid perturbation population genetic algorithm-artificial neural networks (PPGA-ANNS). IEEE Access, Vol. 11, pp. 75794–75817. DOI: 10.1109/access.2023.3294573.

18. **Bergeijk, W. A. V. (1961).** Studies with artificial neurons, II: analog of the external spiral innervation of the cochlea. Kybernetik, Vol. 1, No. 3, pp. 102–107. DOI: 10.1007/bf00 290180.

19. **Harmon, L. D. (1961).** Studies with artificial neurons, I: properties and functions of an artificial neuron. Kybernetik, Vol. 1, No. 3, pp. 89–101. DOI: 10.1007/bf00290179.

20. **Levinson, J., Harmon, L. D. (1961).** Studies with artificial neurons, III: Mechanisms of flicker-fusion. Kybernetik, Vol. 1, No. 3, pp. 107–117. DOI: 10.1007/bf00290181.

21. **Kennedy, J., Eberhart, R. (1995).** Particle swarm optimization. Proceedings of the International Conference on Neural Networks, Vol. 4, pp. 1942-1948. DOI: 10.1109/ICNN. 1995.488968.

22. **Sengupta, S., Basak, S., Peters, R. (2018).** Particle swarm optimization: a survey of historical and recent developments with hybridization perspectives. Machine Learning and Knowledge Extraction, Vol. 1, No. 1, pp. 157–191. DOI: 10.3390/make1010010.

23. **Wolpert, D., Macready, W. (1997).** No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 67–82. DOI: 10.1109/4235. 585893.

24. **Baig, A. R., Rashid, M. (2007).** Honey bee foraging algorithm for multimodal and dynamic optimization problems. Proceedings of the 9th annual conference on Genetic and evolutionary computation pp. 169. DOI: 10.11 45/1276958.1276983.

25. **Pierezan, J., Coelho, L. D. S. (2018).** Coyote optimization algorithm: A new metaheuristic for global optimization problems. IEEE Congress on Evolutionary Computation, pp. 1-8. DOI: 10.1109/CEC.2018.8477769.

26. **Mohammadi-Balani, A., Nayeri, M. D., Azar, A., Taghizadeh-Yazdi, M. (2021).** Golden eagle optimizer: a nature-inspired metaheuristic algorithm. Computers and Industrial Engineering, Vol. 152, pp. 107050. DOI: 10.1016/j.cie.2020.107050.

27. **Wikelski, M., Moskowitz, D., Adelman, J. S., Cochran, J., Wilcove, D. S., May, M. L. (2006).** Simple rules guide dragonfly migration. Biology Letters, Vol. 2, No. 3, pp. 325–329. DOI: 10.1098/rsbl.2006.0487.

28. **Zadeh, L. (1965).** Fuzzy sets. Information and Control, Vol. 8, No. 3, pp. 338–353. DOI: 10.1016/s0019-9958(65)90241-x.

29. **Kosko, B. (1992).** Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence. Prentice-Hall.

30. **Klir, G. J., Yuan, B. (1995).** Fuzzy sets and fuzzy logic: Theory and applications. Prentice Hall P T R.

31. **Roger-Jang, J. S., Sun, C. T., Mizutani, E. (1997).** Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence. Prentice Hall.

32. **Yager, R. R., Filev, D. P. (1994).** Essentials of fuzzy modeling and control. 1st Edition, Wiley.

33. **Dubois, D., Prade, H. (1980).** Fuzzy sets and systems: Theory and applications. Academic Press, Vol. 144, pp. 1–393.

34. **Amador-Angulo, L., Ochoa, P., Peraza, C., Castillo, O. (2023).** Fuzzy dynamic adaptation of an artificial fish swarm algorithm for the optimization of benchmark functions. Studies in Computational Intelligence, pp. 99–114. DOI: 10.1007/978-3-031-28999-6_6.

35. **Castillo, O., Lizárraga, E., Soria, J., Melin, P., Valdez, F. (2015).** New approach using ant colony optimization with ant set partition for fuzzy control design applied to the ball and beam system. Information Sciences, Vol. 294, pp. 203–215. DOI: 10.1016/j.ins.2014.09.040.

36. **Amador-Angulo, L., Mendoza, O., Castro, J., Rodríguez-Díaz, A., Melin, P., Castillo, O. (2016).** Fuzzy sets in dynamic adaptation of

ISSN 2007-9737

parameters of a bee colony optimization for controlling the trajectory of an autonomous mobile robot. Sensors, Vol. 16, No. 9, pp. 1458. DOI: 10.3390/s16091458.

37. **Valdez, F., Vazquez, J. C., Melin, P., Castillo, O. (2017).** Comparative study of the use of fuzzy logic in improving particle swarm optimization variants for mathematical functions using co-evolution. Applied Soft Computing, Vol. 52, pp. 1070–1083. DOI: 10.1016/j.asoc.2016.09.024.
38. **Sánchez, D., Melin, P., Castillo, O. (2017).** A grey wolf optimizer for modular granular neural networks for human recognition. Computational Intelligence and Neuroscience, Vol. 2017, pp. 1–26. DOI: 10.1155/2017/4180510.
39. **González, B., Valdez, F., Melin, P., Prado-Arechiga, G. (2015).** Fuzzy logic in the gravitational search algorithm for the optimization of modular neural networks in pattern recognition. Expert Systems with Applications, Vol. 42, No. 14, pp. 5839–5847. DOI: 10.1016/j.eswa.2015.03.034.
40. **Tai, K., El-Sayed, A., Biglarbegian, M., Gonzalez, C., Castillo, O., Mahmud, S. (2016).** Review of recent type-2 fuzzy controller applications. Algorithms, Vol. 9, No. 2, pp. 39. DOI: 10.3390/a9020039.
41. **Ontiveros, E., Melin, P., Castillo, O. (2020).** Comparative study of interval type-2 and general type-2 fuzzy systems in medical diagnosis. Information Sciences, Vol. 525, pp. 37–53. DOI: 10.1016/j.ins.2020.03.059.
42. **Melin, P., Castillo, O. (2004).** A new method for adaptive control of non-linear plants using type-2 fuzzy logic and neural networks. International Journal of General Systems, Vol. 33, No. 2-3, pp. 289–304. DOI: 10.1080/03081070310001633608.
43. **Moreno, J. E., Sanchez, M. A., Mendoza, O., Rodríguez-Díaz, A., Castillo, O., Melin, P., Castro, J. R. (2020).** Design of an interval type-2 fuzzy model with justifiable uncertainty. Information Sciences, Vol. 513, pp. 206–221. DOI: 10.1016/j.ins.2019.10.042.
44. **Guerrero, M., Valdez, F., Castillo, O. (2022).** Comparative study between type-1 and interval type-2 fuzzy systems in parameter adaptation for the cuckoo search algorithm. Symmetry, Vol. 14, No. 11, pp. 2289. DOI: 10.3390/sym14112289.
45. **Cuevas, F., Castillo, O., Cortés-Antonio, P. (2022).** Generalized type-2 fuzzy parameter adaptation in the marine predator algorithm for fuzzy controller parameterization in mobile robots. Symmetry, Vol. 14, No. 5, pp. 859. DOI: 10.3390/sym14050859.

Article received on 02/02/2024; accepted on 23/04/2024.
Corresponding author is Fevrier Valdez.