ISSN 2007-9737

Fuzzy Combination of Moth-Flame Optimization and Lightning Search Algorithm with Fuzzy Dynamic Parameter Adjustment

Yunkio Kawano, Fevrier Valdez, Oscar Castillo

Tijuana Institute of Technology, Computer Science Department, Mexico

monico.kawano@tectijuana.edu.mx, fevrier@tectijuana.mx

Abstract. In general, this paper is focused on creating a fuzzy combination of two optimization algorithms. In this case, the algorithms work with populations and allow us to migrate between them every certain number of iterations. On the other hand, fuzzy logic is responsible for the dynamic adjustment of parameters within each of the algorithms since the variables are different in each algorithm. In previous works, a combination between genetic algorithm and particle swarm optimization was developed, which motivated us to create this combination expecting to obtain better results when compared to the previous works. The moth-flame optimization and lightning search algorithm were combined to obtain a powerful hybrid metaheuristic combining the advantages of both individual algorithms.

Keywords. Swarm intelligence algorithms, fuzzy logic systems, migration.

1 Introduction

In this paper, a combination of two parallel optimization algorithms is made that seeks between the two a better solution to the problem with which we are working. These algorithms are the Moth-Flame Optimization and Lightning Search Algorithm. There is probably no exact relationship between these two algorithms, but as regards the results obtained individually, we can see that the moth-flame optimization is a good algorithm for exploring the search space at the moment that it is trying to fly in a straight line to his destination.

On the other hand, the lightning search algorithm focused on the creation of lightning depending on the richness in terms of the concentration of certain chemical elements that are scattered in the air of the clouds, which the shock that these elements have creates energy of lightnings. Which each small beam of light where the lightning begins and from there different strands come out to form a ray shaped like an inverted tree, which is a set of possible solutions.

The inspiration that led us to make this fuzzy combination in parallel is the previous paper in which we were working focused on the search for a good combination of two optimization algorithms at that time, where we used the optimization algorithms of GA and PSO were we shared between both algorithms a certain portion of the populations.

Now in this paper we will try to improve the results obtained in comparison with the combination of PSO and GA [1].

At the time of creating the paper, we have not found any precedent that the LSA and MFO algorithms are being used to evaluate benchmark functions as well as the use of dynamic parameter adjustment or the combination between both algorithms. We have only found that they are being used for signal optimization emitted by antennas or similar things. In the next section we present some other population-based optimization algorithms and the theory of each of the algorithms that we are going to use.

Which is the Moth-Flame optimization algorithm and the Lightning search algorithm and a bit of how our fuzzy combination is developed. In the experiments section we show the parameters that we will use as well as the fuzzy rules and the benchmark functions that we will evaluate.

There we will show some of the results obtained by the experiments. Finally, we will outline a conclusion about everything we observed within the experiments.

2 Background

There are different categories of optimization algorithms in which they may have been inspired by how an individual within the population uses certain methods to find food. In other cases, it may be that the algorithm is inspired by the movement that the individual has to reach its destination. Algorithms have also been seen that are inspired by how adolescents develop in their lives to reach the next age stage.

The search algorithms can be that of the dragonfly algorithm that is inspired by how it searches food, and we also can find another algorithm, like a grasshopper optimization algorithm.

In the case of movement, the bird swarm optimization algorithm can be seen.

There are different optimization algorithms that work with populations such as the PSO, DA, BSA, AISA, and FA algorithms, among others. But in this case, we will use the MFO and LSA algorithm that although they have nothing to do with each other, the two manage populations, only one is moths in search of food and the other seeks to create lightning. In this case, the two algorithms are dedicated to looking for something in common, which would be a good solution for any problem they are working.

Next, we will show you a brief description of each of the aforementioned algorithms.

In the case of the particle swarm optimization (PSO) is a population-based stochastic optimization technique, are inspired by social behavior of bird flocking or fish schooling in search of food. PSO has many processes similar to those that work with genetic algorithms. This algorithm initiates a swarm of random particles, where each of the contained particles could be a solution to the problem that is being worked on. These possible solutions are evaluated in each iteration that we have [2, 3, 4].

On the social spider optimization (SSO), this algorithm is based on the simulation of the cooperative behavior of social-spiders. The individuals emulate a group of spiders which interact between others spiders with the biological laws in the colony. The algorithm has two search agents are the spider males or females, each gender has different tasks [5, 6, 7].

The Dragonfly algorithm (DA) is inspired by the behavior that can occur in a static and dynamic way for dragonflies, which dynamically while in search of food can communicate with other dragonflies to find food, which for an optimization algorithm would be the part of exploration and when it is in static way it can exploit the area [8].

The Bird swarm algorithm (BSA) is inspired by social behavior and the iterations it does depend on the type of bird it is. There are three types of birds or with different tasks within the swarm, it can be foraging behavior, vigilance behavior and flight behavior [9].

The Adolescent Identity Search Algorithm (AISA) is inspired by the simulation about how the identity a teenager in a couple is formed, where all the experiences that it can lead to live improve their knowledge or behavior [10].

Finally, the firefly algorithm (FA) is a population-based optimization algorithm that mimics a firefly's attraction to flashing light. In particular it used the concept of how the brightness of individual fireflies drew them together and a randomness factor to encourage exploration of the solution space [11, 12, 13, 14].

2.1 Optimization Algorithms Based on Swarms

At the moment there are already algorithms for almost any activity which may be inspired by biology or phenomena that occur in nature, such as talking about algorithms inspired by biology, we can find the PSO algorithm that aims to schools of fish or flocks of birds move to find food, or in another case the algorithm that uses moths that focuses on how the moth makes reference to the moon to be able to fly long distances in a straight line. Or the other case, in algorithms inspired by nature, as in our case would be the lightning search algorithm, which is dedicated to trying to simulate what is known as the lightning that could be described as an inverted tree, where each thread or tip that has the lightning would be a possible solution. That is why we could say that we combine two worlds, where it is a biologically inspired algorithm and also a naturally inspired algorithm to find between the two algorithms a better solution for the problem we are dealing with.

Although there are too many algorithms of different types, we selected these algorithms

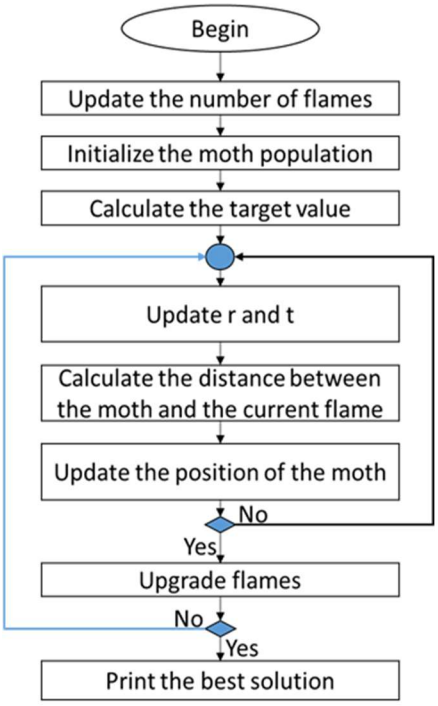

Fig. 1. Flowchart Moth-Flame Optimization

because they were interesting to us to test the combination of both, where we think that speaking of nature, moths and lightning do not get along. Apart from the fact that individually the two algorithms provide good solutions [15].

The following optimization algorithms are used to create the fuzzy combination shown in section 3 of this paper.

2.1.1 Moth Flame Optimization

This optimization algorithm is inspired by the orientation ability of the moths when flying in the middle of the night towards the moon. The moths are oriented by means of a mechanism called transverse orientation, which consists of maintaining a fixed angle in the direction of the moon to fly in a straight line over great distances, since the moon is very far away.

Although currently in nature the moths are confused in the presence of artificial lightning, which causes them to think that they see the moon, but since the distances are much shorter the moths begin to spin around the lamp without control, until unfortunately die.

The algorithm assumes that moths and flames are among the possible solutions, where moths are search agents that move around the space and flames are the best position found so far by the moth. These moths fly close to the flame in case they find a better solution. The flame is updated.

The movement of the moth while in search of a better solution is in logarithmic spiral. Where at the beginning of the spiral is a moth and at the end it must be the position of the flame, and it should be noted that the range of the spiral must not exceed the search space:

$$S(M_i, F_j) = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j \,, \tag{1}$$

where $D_i = \left|F_j - M_i\right|$ is the distance between the flame and moth where M_i is the position of the moth in i and F_j is the position of the flame in j. b it's a constant that defines the logarithmic spiral. t is a random number between $[-1,\ 1]$. In MFO, the control between exploitation and exploration is thanks to S that is the spiral movement of the moth near the flame in the search space.

At a certain point of the algorithm, the update of the number of flames is applied since it helps us to improve the exploitation of the MFO algorithm. Because the algorithm searches in various positions within the search space, which reduces the number of possibilities we have to exploit the best possible solutions.

Therefore, reducing the number of flames helps to solve this problem based on the following equation:

$$flames\ no. = \left(N - l * \frac{N-l}{T}\right), \tag{2}$$

where N is the maximum number of flames, *l* is the current number of iterations, and *T* indicates the maximum number of iterations[16, 17, 18, 19].

2.1.2 Lightning Search Algorithm

This is an optimization algorithm that is inspired by the natural phenomenon of how lightning is created in the natural environment.

A propagation mechanism is used in a staggered manner, which takes the form of an

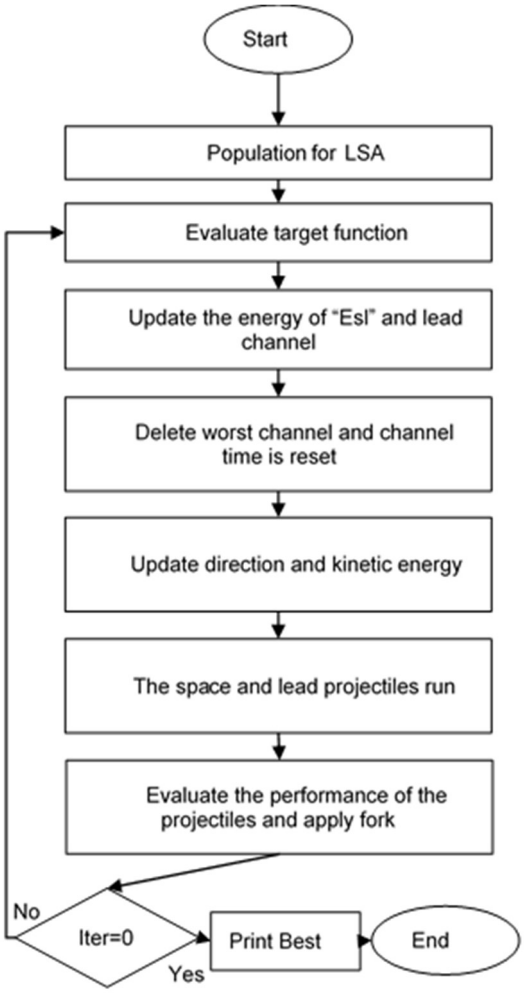

Fig. 2. Flowchart Lightning Search Algorithm

inverted tree, which is why the algorithm has three types of projectiles, where the first is a transition one which generates the first leading population, then continue the space projectiles that try to obtain a better position in the leadership range and finally we have the leading projectiles that have the best position.

The projectiles are composed of hydrogen, nitrogen and oxygen atoms that can be found near the region of the storm clouds, when the molecules of these elements travel at a great speed through the atmosphere and are ionized the produce a path or channel through the collision and transition to the step leader.

The projectiles that travel under normal conditions through the atmosphere lose kinetic energy when they collide with molecules in the air. The velocity of a projectile is obtained with the following equation:

$$v_p = \left[1 - \left(\frac{1}{\sqrt{1-\left(\frac{v_0}{c}\right)^2}} - \frac{sF_i}{mc^2}\right)^{-2}\right]^{-\frac{1}{2}}, \quad (3)$$

where v_p and v_0 are the current velocity and initial velocity, respectively, of the projectile; c is the speed of light; F_i is the constant ionization rate; m is the mass of the projectile; and s is the length of the path traveled.

The equation shows that velocity is a function of leader tip position and projectile mass. When the mass is small or when the path traveled is long, the projectile has little potential to ionize or explore a large space. Other property of a stepped leader is forking, which means that are two symmetrical branches are created because the nuclei collision of the projectile is realized by using the opposite number as in the next equation:

$$\overline{p_i} = a + b - p_i \quad (4)$$

where $\overline{p_i}$ and p_i are the opposite and original projectiles, respectively, in a one-dimensional system; *a* and *b* are the boundary limits. This adaptation may improve some of the bad solutions in the population. If forking does not improve channel propagation in the LSA, one of the channels at the forking point is illuminated to maintain the population size [20, 21, 22, 23].

2.1.3 Fuzzy Combination of MFO and LSA

The fuzzy combination of the Moth-Flame Optimization (MFO) and Lightning Search Algorithm (LSA) is to try to find a better solution, since the MFO algorithm is good to explore within the search space and the LSA algorithm is good to exploit although the algorithm converges prematurely, so that is why we combine them to try to balance the amount to exploration and exploitation necessary so that between the two algorithms better solutions are obtained when comparing the algorithms individually [21, 22, 23].

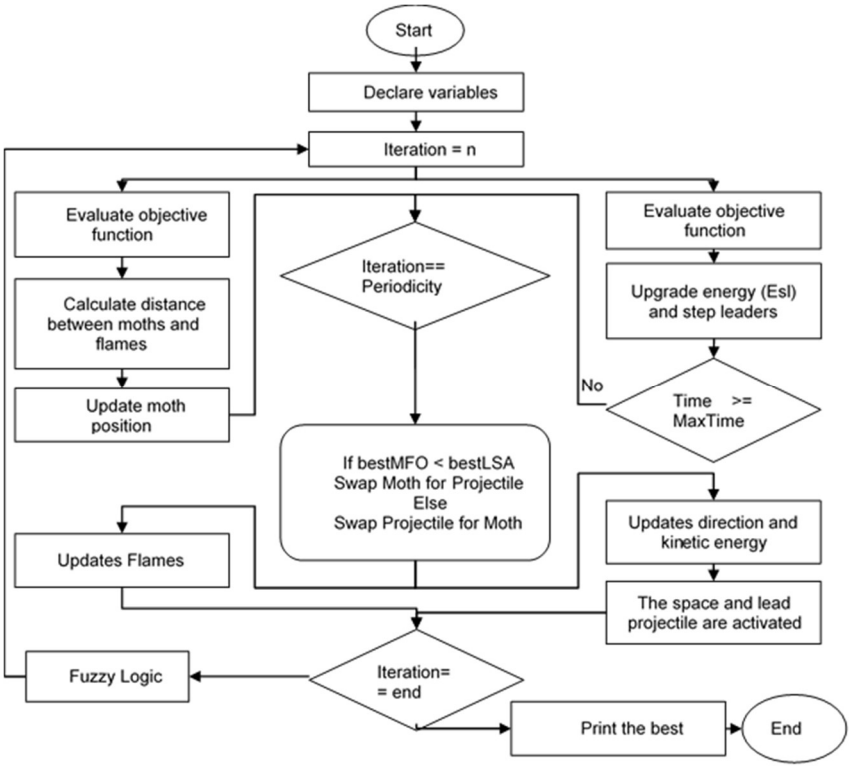

Fig. 3. Flowchart Lightning Search Algorithm

To achieve the combination of the MFO and LSA algorithms, we configure them so that the two algorithms are intertwined to run at the same time and to apply the use of migration blocks, which allow us to exchange a certain number of individuals within of each of the populations, it would be like exchanging a certain number of moths for lightning bolts within the algorithms used [27].

In Figure 3 can be appreciated how the operation of the algorithms combined into one is shown in the diagram. According to the diagram it is seen that the algorithms are executed simultaneously with each of their processes on each side of the drawing and in the central part of the diagram is what would be the migration block, it is used every certain number of individuals to be shared.

The diagram shows where the dynamic parameter adjustment is applied, but does not describe what is the condition to apply it. The condition that it is applied is from iteration 100 onwards, so that the algorithm has time to look for a shortly before starting to share the individuals and this action stops applying 100 iterations before reaching the maximum number of iterations, regardless of the maximum number of iterations that are being used. The variables that are adjusted in the two algorithms that we use, one variable is about the value of the spiral used by the moth-flame algorithm and the other is from the lightning search algorithm, which would be the probability of creating two solutions form one or dividing the lightning channel.

The results obtained are the average of 30 runs of the code, thus being able to perform a valid statistical test.

3 Experiments

The experiments were carried out on a computer with an Intel i5-9400f processor that has 6 cores and 6 threads, it is complemented by 16 gigabytes of RAM memory and a Nvidia GTX 970 video card

Table 1. Parameters for algorithms used

Parameter	Value
Populations	100
Dimensions	5, 10, 20, 40 and 80.
Iterations	500, 1000 and 2000.
Logarithmic spiral amplitude (MFO)	[0.001, 0.99]
Maximum channel time (LSA)	10
Bifurcation percentage (LSA)	[0.001, 0.99]

Table 2. First fuzzy logic system for adapt value of Spiral in Moth-Flame algorithm

Rules	Iteration	Spiral
1	Low	Low
2	Mid-Low	Mid-Low
3	Mid	Mid
4	Mid-High	Mid-High
5	High	High

Table 3. Second fuzzy logic system for adapt value of Spiral in Moth-Flame algorithm

Rules	Iteration	Spiral
1	Low	High
2	Mid-Low	Mid-High
3	Mid	Mid
4	Mid-High	Mid-Low
5	High	Low

Table 4. First fuzzy logic system to adapt the value of Fork in Lightning Search algorithm

Rules	Iteration	Spiral
1	Low	Low
2	Mid-Low	Mid-Low
3	Mid	Mid
4	Mid-High	Mid-High
5	High	High

with 4 gigabytes GDDR5. Although the latter is unnecessary since the code is not focused on having better times with the use of the GPU, in terms of storage it is a high-speed solid hard disk in which we can find MATLAB R2017b installed to run the codes.

Table 5. Second fuzzy logic system to adapt the value of Fork in Lightning Search algorithm

Rules	Iteration	Spiral
1	Low	High
2	Mid-Low	Mid-High
3	Mid	Mid
4	Mid-High	Mid-Low
5	High	Low

3.1 Fuzzy Logic Systems

Fuzzy logic allows a better analysis of the input data that a complex computational system is going to have since it gives us the ability to have several possible best solutions, which is decided by one in terms of the rules that have been specified in the system fuzzy as well as membership functions.

The following table shows the simple configuration that we use where there are only 5 rules for each of the variables with dynamic parameter adjustment, where basically all of them have one input and one output [28].

Currently there are some works that make use of fuzzy logic systems to adjust the parameters of all kinds of algorithms [23, 24, 25, 26].

Table 2 is for dynamic adjustment of the spiral variable that is used by the MFO algorithm, where the output values go increasing.

Table 3 is for dynamic adjustment of the spiral variable that is used by the MFO algorithm, where the output values go decreasing.

Table 4 is for dynamic adjustment of the fork variable that is used by the LSA algorithm, where the output values go increasing.

Table 5 is for dynamic adjustment of the fork variable that is used by the LSA algorithm, where the output values go decreasing.

4 Experiments

Table 6 shows the benchmark functions used to evaluate the performance of each of the algorithms, be they the MFO, LSA algorithms and in the fuzzy combination of both.

The following tables show some of the most relevant results obtained in the experimentation stage of each of the algorithms, already in the original version of MFO and LSA as well as in the fuzzy combined version of both, where they help each other. two for best result.

After each of the tables, an explanation of what is observed in each one is presented.

4.1 Comparison of All Algorithms Used

This section shows all the results obtained from the experiments carried out, it should be noted that each result shown is the average of 30 runs.

In Table 7 the results for 5 dimensions and 500 iterations are compared, we can see that the first four benchmark functions the fuzzy combination can obtain better results but in the other functions the results are very close in terms of the original MFO and LSA algorithms.

In Table 8 the results show that increasing the dimensions increases the complexity of the problem, which is why the results are moving away from zero, but we can see that the column of the fuzzy combination obtains better results than the columns with the original algorithms except for the F8 function.

Table 9, in the same way as in the previous table, it is shown that the more complex the problem is or that it has more dimensions, the fuzzy combination may be more profitable to use since it obtains better results.

Table 6. Benchmark functions

No.	Function	Range
F1	$\sum_{i=1}^{n} x_i^2$	[-5.12, 5.12]
F2	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]
F3	$\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_i \right)$	[-100, 100]
F4	$\max\{[x_i], 1 \le i \le n\}$	[-100, 100]
F5	$\sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$	[-30, 30]
F6	$\sum_{i=1}^{n} ([x_i + 0.5])^2$	[-100, 100]
F7	$\sum_{i=1}^{n} i x_i^4 + random[0, 1)$	[-1.28, 1.28]
F8	$\sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]
F9	$\sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12, 5.12]
F10	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$	[-32.768, 32.768]
F11	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]
F12	$\frac{\pi}{n}\left\{10\, sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 \, [1 + 10\, sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	[-50, 50]
F13	$0.1\left\{sin^2(3\pi y_i) + \sum_{i=1}^{n-1} (x_i - 1)^2 \, [1 + sin^2(3\pi y_{i+1})] + (x_n - 1)^2[1 + sin^2(2\pi y_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $+, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	[-50, 50]

Table 7. Experimental results with MFO, LSA and Fuzzy Combination for 5 dimensions and 500 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	1.649E-80	7.59E-157	3.533E-157
F2	1.12E-44	6.34E-83	6.272E-83
F3	1.95E-53	6.44E-84	7.583E-86
F4	1.794E-35	2.549E-70	2.253E-69
F5	12.46	1.586	2.275
F6	0	0	0
F7	2.186E-04	3.883E-04	1.787E-02
F8	-1890	-1996	-1989
F9	2.885	86.23	1.327
F10	8.88E-16	3.25E-15	1.243E-15
F11	0.07662	0.04762	0.04122
F12	6.62E-10	2.65E-07	0
F13	3.99E-09	1.36E-06	0

Table 8. Experimental results with MFO, LSA and Fuzzy Combination for 20 dimensions and 500 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	5.78E-13	2.964E-40	1.848E-39
F2	4.333	3.655E-13	9.012E-15
F3	2501	0.0006403	0.001162
F4	29.28	0.00000947	3.464E-06
F5	12150	20.98	20.18
F6	4.41E-13	0.7	1
F7	0.09774	0.007223	0.04333
F8	-6269	-6451	-6486
F9	67.69	27.26	32
F10	0.4659	0.9644	1.065
F11	0.03244	0.01943	0.02058
F12	0.088112	1.242846	0.15751
F13	0.005045	0.174869	0.00356

Table 9. Experimental results with MFO, LSA and Fuzzy Combination for 80 dimensions and 500 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	1.55E+04	7.05E-03	4.20E-02
F2	1.13E+02	1.02	1.34
F3	1.05E+05	1.95E+04	2.04E+04
F4	8.69E+01	36.9	3.89E+01
F5	2.85E+07	268.0	3.06E+02
F6	1.54E+04	19.1	1.14E+02
F7	27.3	0.185	2.98E-01
F8	-2.1E+04	-1.99E+04	-2.24E+04
F9	5.76E+02	1.99E+02	1.86E+02
F10	1.91E+01	3.29	5.44
F11	1.27E+02	5.41E-03	1.86E-02
F12	0.523797	21867861	0.47468
F13	3.947527	143748882	3.03139

Table 10. Experimental results with MFO, LSA and Fuzzy Combination for 5 dimensions and 1000 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	3.71E-81	0	0
F2	3.28E-44	6.48E-165	4.52E-166
F3	4.06E-52	9.14E-174	4.79E-173
F4	7.21E-35	3.45E-140	4.83E-140
F5	8.78E-01	1.24	1.27
F6	0	0	0
F7	2.79E-04	2.57E-04	1.34E-02
F8	-1.95E+03	-2.02E+03	-2.01E+03
F9	2.55	1.13	1.43
F10	8.88E-16	5.49E-02	5.49E-02
F11	7.84E-02	6.21E-02	3.37E-02
F12	0	7.38E-07	0
F13	0	2.152E-06	0

Table 11. Experimental results with MFO, LSA and Fuzzy Combination for 20 dimensions and 1000 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	6.99E-13	4.25E-81	1.925E-82
F2	2.333	2.65E-16	1.325E-24
F3	2834	2.23E-10	7.922E-11
F4	32.18	2.74E-12	1.654E-12
F5	12240	11.33	10.63
F6	6.06E-13	0.6333	0.8
F7	0.009386	0.00591	0.04692
F8	-6341	-6327	-6748
F9	71.53	31.97	30.68
F10	0.1542	0.7915	1.07
F11	0.02811	0.01739	0.02685
F12	0	0.55150	0
F13	0	0.25621	0

Table 12. Experimental results with MFO, LSA and Fuzzy Combination for 80 dimensions and 1000 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	1.40E+04	1.81E-09	8.71E-06
F2	1.08E+02	3.83E-01	8.58E-01
F3	9.68E+04	7.27E+03	8.25E+03
F4	8.79E+01	3.14E+01	3.31E+01
F5	2.50E+07	1.99E+02	1.67E+02
F6	1.43E+04	2.03E+01	1.34E+02
F7	5.44E+01	1.24E-01	2.52E-01
F8	-2.06E+04	-1.98E+04	-2.26E+04
F9	5.80E+02	1.98E+02	2.02E+02
F10	1.92E+01	3.57	5.63
F11	1.48E+02	4.11E-03	2.02E-02
F12	0.20298	3.519E+07	0.15478
F13	0.54241	7.085E+07	0.59517

The next three tables are using 1000 iterations, which allows the algorithm to have more time to find a better solution. In Table 10, when working for 5 dimensions and 1000 iterations, we can see that the fuzzy combination finds zeros in function 1 and in function 6 all algorithms reach zero.

Table 13. Experimental results with MFO, LSA and Fuzzy Combination for 5 dimensions and 2000 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	6.42E-165	0	0
F2	2.54E-89	2.27E-186	0
F3	3.93E-110	8.78E-101	0
F4	6.06E-71	3.31E-114	3.02E-282
F5	2.04	1.22	7.93E-01
F6	0	0	0
F7	1.24E-04	1.03E-03	1.12E-02
F8	-1.90E+03	-3.73E+03	-2.03E+03
F9	2.82	6.40	1.43
F10	8.88E-16	3.41E-01	1.01E-15
F11	1.17E-01	1.04E-01	4.19E-02
F12	0	9.433E-32	0.02526
F13	0	0.00037	0.03138

Table 14. Experimental results with MFO, LSA and Fuzzy Combination for 20 dimensions and 2000 iterations

Fun.	MFO	LSA	Fuzzy Combination
F1	1.339E-29	2.44E-169	8.912E-168
F2	1.667	7.032E-17	6.864E-18
F3	2667	1.845E-24	8.608E-24
F4	27.38	9.111E-27	2.211E-24
F5	6289	4.35	1.793
F6	2.275E-29	0.5667	0.3333
F7	0.004921	0.004996	0.0474
F8	-6457	-6433	-6690
F9	81.41	29.85	30.45
F10	0.1319	0.9823	1.038
F11	0.02352	0.01632	0.02384
F12	0	0.34955	0
F13	0	4.71827	0

In Table 11, 20 dimensions and 1000 iterations are used here, what we expect is that our fuzzy combination will be better when evaluating each of the different benchmark functions, but in this case, the results obtained are very similar to those obtained by the original LSA algorithm where by very little only in some of the functions wins our combination.

In Table 12, with 80 dimensions and 1000 iterations, the results show that in the same way the column of the LSA algorithm is the winner in most of the benchmark functions.

The next three tables show the results with 5 and 20 dimensions, but with 2000 iterations.

In Table 13, adding more time to the experiments, it is observed that at least it is the first three functions as well as in the sixth they show zeros as results and in the some others they are very close to zero.

In Table 14, we can observe the pattern of the results where 1000 iterations and 20 or 80 dimensions were used that the LSA column beasts the fuzzy combination column, that is why we do not show the table of 80 dimensions and 2000 iterations.

5 Conclusions

Based on the obtained results, it can be observed that the fuzzy combination can be a good idea to use when we are working with a complex problem in question in a more efficient way. The moth-flame optimization and lightning search algorithm were combined to obtain a powerful hybrid metaheuristic combining the advantages of both individual algorithms. In our case, evaluating the thirteen benchmark functions that are in a certain way arranged in an ascending level of complexity. That is why the results on some occasions are shown that the values reach zero with any of the algorithms, that is if is recommended that we work at least with one thousand iterations or more to allow the algorithm time to find the best solution.

In the future, we would like to perform tests with much more complicated benchmark functions to see if the algorithm with migration is really viable.

As future work, to improve the proposed method, we envision adding dynamic parameter adjustment using type-2 fuzzy systems to obtain better results.

On other hand, although this article is not about that, we would like to add improvements in terms of execution times, with the use of CUDA functions that we have used in other algorithms and they could help in saving time.

Acknowledgments

The authors would like to thanks CONACYT and Tijuana Institute of Technology for the support during this research work.

References

1 **Kawano, Y., Valdez, F., Castillo, O. (2018).** Performance Evaluation of Optimization Algorithms based on GPU using CUDA Architecture. IEEE Latin American Conference on Computational Intelligence (LA-CCI), pp. 1–6. DOI: 10.1109/LA-CCI.2018.8625236.

2 **Wang, D., Tan, D., Liu, L. (2018).** Particle swarm optimization algorithm: An overview. Soft Computing, Vol. 22, No. 2, pp. 387–408. DOI: 10.1007/s00500-016-2474-6.

3 **Sho, H. (2021).** Comparison of Centralized and Distributed Intelligent Particle Multi-Swarm Optimization on Search Performance. Artificial Intelligence Research, Vol. 10, No. 1, pp. 1–11. DOI: 10.5430/air.v10n1p1.

4 **Sánchez, D., Melin, P., Castillo, O. (2020).** Comparison of particle swarm optimization variants with fuzzy dynamic parameter adaptation for modular granular neural networks for human recognition. Journal of Intelligent & Fuzzy Systems, Vol. 38, No. 3, pp. 3229–3252. DOI: 10.3233/JIFS-191198.

5 **Yu, J.J.Q., Li, V.O.K. (2015).** A social spider algorithm for global optimization. Applied Soft Computing, Vol. 30, pp. 614–627. DOI: 10.1016/j.asoc.2015.02.014.

6 **Lai, Z., Feng, X., Yu, H., Luo, F. (2021).** A Parallel Social Spider Optimization Algorithm Based on Emotional Learning. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 51, No. 2, pp. 797–808. DOI: 10.1109/TSMC.2018.2883329.

7 **Priyadharshini, V., Divya, P., Preethi, D., Pazhaniraja, N., Paul, P.V. (2015).** A novel Web service publishing model based on social spider optimization technique. International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC),

pp. 0373–0387. DOI: 10.1109/ICCPEIC. 2015.7259488.

8 **Mirjalili, S. (2016).** Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, Vol. 27, No. 4, pp. 1053–1073. DOI: 10.1007/s00521-015-1920-1.

9 **Meng, X.B., Gao, X.Z., Lu, L., Liu, Y., Zhang, H. (2016).** A new bio-inspired optimisation algorithm: Bird Swarm Algorithm. Journal of Experimental & Theoretical Artificial Intelligence, Vol. 28, No. 4, pp. 673–687. DOI: 10.1080/0952813X.2015.1042530.

10 **Bogar, E., Beyhan, S. (2020).** Adolescent Identity Search Algorithm (AISA): A novel metaheuristic approach for solving optimization problems. Applied Soft Computing, Vol. 95, pp. 1–43. DOI: 10.1016/j.asoc.2020.106503.

11 **Lagunes, M.L., Castillo, O., Valdez, F., Soria, J. (2019).** Multi-Metaheuristic Competitive Model for Optimization of Fuzzy Controllers. Algorithms, Vol. 12, No. 5, pp. 1–21. DOI: 10.3390/a12050090.

12 **Ezugwu, A.E.S., Agbaje, M.B., Aljojo, N., Els, R., Chiroma, H., Elaziz, M.A. (2020).** A Comparative Performance Study of Hybrid Firefly Algorithms for Automatic Data Clustering. IEEE Access, Vol. 8, pp. 121089–121118. DOI: 10.1109/ACCESS.2020. 3006173.

13 **Mashhour, E.M., El-Houby, Wassif, K.T., Salah, A.I. (2020).** A Novel Classifier based on Firefly Algorithm. Journal of King Saud University – Computer and Information Sciences, Vol. 32, No. 10, pp. 1173–1181, DOI: 10.1016/j.jksuci.2018.11.009.

14 **Xu, C., Meng, H., Wang, Y., (2020).** A Novel Hybrid Firefly Algorithm Based on the Vector Angle Learning Mechanism. IEEE Access, Vol. 8, pp. 205741–205754. DOI: 10.1109/ACCESS.2020.3037802.

15 **Molina, D., Poyatos, J., Del Ser, J., García, S., Hussain, A., Herrera, F. (2020).** Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration versus Algorithmic Behavior, Critical Analysis and Recommendations. Cognitive Computation, Vol. 12, No. 5, pp. 897–939. DOI: 10.1007/s12559-020-09730-8.

16 **Mirjalili, S. (2015).** Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, Vol. 89, pp. 228–249. DOI: 10.1016/j.knosys.2015. 07.006.

17 **Jangir, N., Pandya, M.H., Trivedi, I.N., Bhesdadiya, R.H., Jangir, P., Kumar, A. (2016).** Moth-Flame optimization Algorithm for solving real challenging constrained engineering optimization problems. IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1–5. DOI: 10.1109/SCEECS.2016. 7509293.

18 **Li, C., Niu, Z., Song, Z., Li, B., Fan, J., Liu, P.X. (2018).** A Double Evolutionary Learning Moth-Flame Optimization for Real-Parameter Global Optimization Problems. IEEE Access, Vol. 6, pp. 76700–76727. DOI: 10.1109/ACCESS.2018.2884130.

19 **Tumar, I., Hassouneh, Y., Turabieh, H., Thaher, T. (2020).** Enhanced Binary Moth Flame Optimization as a Feature Selection Algorithm to Predict Software Fault Prediction. IEEE Access, Vol. 8, pp. 8041–8055. DOI: 10.1109/ACCESS.2020.2964321.

20 **Shareef, H., Islam, M.M., Ibrahim, A.A., Mutlag, A.H. (2015).** A Nature Inspired Heuristic Optimization Algorithm Based on Lightning. 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS), pp. 9–14. DOI: 10.1109/AIMS.2015.12.

21 **Liu, W., Huang, Y, Zong, X., Shi, H., Ye, Z., Wei, S. (2018).** Application of lightning search algorithm in localization of Wireless Sensor Networks. 4th IEEE International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), pp. 57–61. DOI: 10.1109/IDAACS-SWS.2018.8525518.

22 **Asvany, T., Amudhavel, J., Sujatha, P. (2017).** Lightning search algorithm for solving coverage problem in wireless sensor network.

Advances and Applications in Mathematical Sciences, Vol. 17, No. 1, pp. 113–127.

23 **Abualigah, L., Elaziz, M.A., Hussien, A.G., Alsalibi, B., Jalali, S.M.J., Gandomi, A.H. (2021).** Lightning search algorithm: a comprehensive survey. Applied Intelligence, Vol. 51, pp. 2353–2376. DOI: 10.1007/s10489-020-01947-2.

24 **Valdez, F., Castillo, O., Melin, P. (2021).** Bio-Inspired Algorithms and Its Applications for Optimization in Fuzzy Clustering. Algorithms, Vol. 14, No. 4, pp. 1–21. DOI: 10.3390/a14040122.

25 **Lagunes, M.L., Castillo, O., Soria, J., Valdez, F. (2021).** Optimization of a fuzzy controller for autonomous robot navigation using a new competitive multi-metaheuristic model. Soft Computing, Vol. 25, No. 17, pp. 11653–11672. DOI: 10.1007/s00500-021-06036-1.

26 **Bernal, E., Castillo, O., Soria, J., Valdez, F., Melin, P. (2018).** A variant to the dynamic adaptation of parameters in galactic swarm optimization using a fuzzy logic augmentation. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–7. DOI: 10.1109/FUZZ-IEEE.2018.8491623.

27 **Ma, G., Zhou, W., Chang, X. (2012).** A novel particle swarm optimization algorithm based on particle migration. Applied Mathematics and Computation, Vol. 218, No. 11, pp. 6620–6626. DOI: 10.1016/j.amc.2011.12.032.

28 **Valdez, F., Melin, P., Castillo, O. (2010).** Fuzzy control of parameters to dynamically adapt the PSO and GA Algorithms. International Conference on Fuzzy Systems, pp. 1–8. DOI: 10.1109/FUZZY.2010.5583934.

29 **Alalaween, W.H., Alalawin, A.H., Mahfouf, M., Abdallah, O.H. (2021).** A Dynamic Type-1 Fuzzy Logic System for the Development of a New Warehouse Assessment Scheme. IEEE Access, Vol. 9, pp. 43611–43619. DOI: 10.1109/ACCESS.2021.3060293.

30 **Avelar, E., Castillo, O., Soria, J. (2020).** Fuzzy Logic Controller with Fuzzylab Python Library and the Robot Operating System for Autonomous Mobile Robot Navigation. Journal of Automation, Mobile Robotics and Intelligent Systems, Vol. 14, No. 1, pp. 48–54. DOI: 10.14313/JAMRIS/1-2020/6.

31 **Cuevas, F., Castillo, O., Cortés-Antonio, P. (2021).** Design of a Control Strategy Based on Type-2 Fuzzy Logic for Omnidirectional Mobile Robots. Journal of Multiple-Valued Logic & Soft Computing, Vol. 37, No. 1–2, pp. 107–136.

32 **Valdez, F., Vázquez, 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.

Article received on 30/06/2021; accepted on 20/11/2021.
Corresponding author is Oscar Castillo.