Computer and Information Science; Vol. 11, No. 1; 2018
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Enhanced Firefly Algorithm Using Fuzzy Parameter Tuner

Mahdi Bidar[1], Samira Sadaoui[1], Malek Mouhoub[1] & Mohsen Bidar[2]

[1] Department of Computer Science, University of Regina, Regina, Canada

[2] Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran

Correspondence: Malek Mouhoub, Department of Computer Science, University of Regina, Regina, Saskatchewan, S4S 0A2, Canada. 1-306-585-4700. E-mail: mouhoubm@cs.uregina.ca.

Received: December 4, 2017 Accepted: December 20, 2017 Online Published: January 8, 2018

doi:10.5539/cis.v11n1p26 URL: http://dx.doi.org/10.5539/cis.v11n1p26

Abstract

Exploitation and exploration are two main search strategies of every metaheuristic algorithm. However, the ratio between exploitation and exploration has a significant impact on the performance of these algorithms when dealing with optimization problems. In this study, we introduce an entire fuzzy system to tune efficiently and dynamically the firefly algorithm parameters in order to keep the exploration and exploitation in balance in each of the searching steps. This will prevent the firefly algorithm from being stuck in local optimal, a challenge issue in metaheuristic algorithms. To evaluate the quality of the solution returned by the fuzzy-based firefly algorithm, we conduct extensive experiments on a set of high and low dimensional benchmark functions as well as two constrained engineering problems. In this regard, we compare the improved firefly algorithm with the standard one and other famous metaheuristic algorithms. The experimental results demonstrate the superiority of the fuzzy-based firefly algorithm to standard firefly and also its comparability to other metaheuristic algorithms.

Keywords: exploitation-exploration balance, firefly algorithm, fuzzy-based parameter controllers, fuzzy systems, metaheuristic algorithms, stochastic local search

1. Introduction

Optimization is the task of finding the best values for the parameters of a given function by maximizing or minimizing the output while satisfying problem constraints. The set of values that can be assigned to the parameters represent the candidate solutions, and the candidate that leads to the best output is the optimal solution. Often, the optimization process suffers from the problem of widespread (high number of solutions) and high dimensionality (high number of parameters) search space**.** Moreover most optimization problems are NP-hard i.e. with very high computational complexity. Metaheuristic algorithms are considered among the most practical approaches for solving optimization problems. These algorithms algorithms have been developed based on biological, physical processes, chemical and swarm intelligence. Metaheuristic algorithms can handle a wide variety of real-world optimization applications that are basically difficult and in some cases impractical to be solved by classical methods (Yang, X.S., 2010). Even though these algorithms find the best solution in an excellent running time, they do not always guarantee the solution optimality (Yang, X.S., 2010).

It is not always possible to go through the search space entirely due to its complexity and high-dimensionality (Yang, X.S., 2010). So to discover the optimal solution, metaheuristic algorithms utilize a mixture of exploration and exploitation strategies. With exploration, these algorithms globally search the space and generate diverse solutions. Simultaneously, metaheuristic algorithms use local search to look around for the quality solutions. Let us assume an optimization algorithm employs only exploration (with 0% contribution of exploitation). Thus, the algorithm, regardless of the previous searching steps, performs its next movement, and by considering the dimensionality of the space the algorithm faces, it cannot succeed in finding the optimal solution. Similarly, if the algorithm designed with only exploitation, making a choice for the next movement is based on the previous decisions. So the algorithm becomes a hill-climbing method, and in complex problem spaces, it will be definitely trapped in local optimum (Durkota, K., 2011). It is evident that the best approach is to take advantage of both exploration and exploitation to be able to handle the problem dimensionality. Nevertheless, to improve the performance of metaheuristic algorithms in terms of the solution quality, a critical aspect is to keep the exploration and exploitation in balance (Yang, X.S., 2010). Each algorithm has its own tuning parameters, and their values determine the contribution rates of exploration and exploitation. The behavior of metaheuristic

algorithms should be constantly monitored in run time to allocate proper values to their parameters in order to attain the right balance between local and global search (Yang, 2010).

Firefly is one of the most popular population–based algorithms inspired by the real fireflies' behavior in nature (Yang, 2010). One of the inherent features of firefly is its tendency to converge early. This is due to the fact that in each searching step, each firefly moves toward the brighter ones. This trend represents the exploitation i.e. the local search that causes the early convergence. It also increases the risk of being trapped in local optimum solutions. Moreover, in the firefly algorithm, the controlling parameters of exploration and exploitation are initialized at the beginning and then remain unchanged until the end of the searching task. Since this algorithm may encounter unwanted situations, such as being trapped in local optimums or progressing poorly in solving the optimization problem, then it needs different contribution rates of exploration and exploitation. Consequently, a parameter tuner is essential to effectively control the firefly parameters to determine the right contribution rates of exploitation and exploration considering the trend of approaching the optimal solution. For this purpose, we introduce an entire Fuzzy controlling system because Fuzzy logic is an adequate choice to adjust the exploitation-exploration trade-off in metaheuristic algorithms (Liu & Ma, 2011), Shi & Eberhart, (2001), Liu, Xu, & Abraham, 2005, Qi & Chunming, 2010). Indeed, when searching for the optimal solution, metaheuristic algorithms deal with uncertain values due to the stochastic nature of the optimization processes (Gandomi, A. H., 2014). Thus, we need to employ an approximate reasoning approach that allows making decisions with estimated values in conjunction with uncertain values. We may note that very limited studies (only two (Bidar & Kanan, 2013), Hassanzadeh & Kanan, 2014) applied Fuzzy logic to firefly. According to the progress of the optimization process, the proposed Fuzzy system allocates dynamically different values to the parameters of firefly in order to control the algorithm behavior in run-time. To this end, we utilize two measures to examine the progress trend of firefly in approaching the optimal solution.

To assess the performance of the Fuzzy-based firefly algorithm, we conduct extensive comparative experiments based on a set of 20 benchmark functions (low and high dimensional problems) as well as on two constrained engineering problems. More precisely, we apply the enhanced firefly, the standard firefly and numerous famous nature-inspired optimization algorithms on these 22 well-known problems. Additionally, we compare the performance results of these optimization methods. The results of the experiments demonstrate the superiority of our proposed Fuzzy-based firefly algorithm to standard one. The results also show that our proposed algorithm is comparable to the other algorithms in terms of quality of the solution returned.

In this article, we first discuss the significance of exploitation and exploration in metaheuristic algorithms. Section 3 presents related work on firefly extensions and fuzzy logic application to other optimization algorithms. Section 4 discusses the standard firefly algorithm. Section 5 presents an overview on fuzzy systems. Section 6 exposes the steps of the proposed fuzzy firefly algorithm. Section 7 and 8 conduct several experiments based on unconstrained benchmark functions and constrained engineering problems. Finally section 9 concludes the paper.

2. Discussion on Exploration and Exploitation

Two basic features of every metaheuristic algorithm are exploitation and exploration. Exploration or diversification enables algorithms to search globally for the optimal solution by discovering new areas in the problem space. In other words, it causes the algorithm to produce more diverse solutions (diversity refers to the difference between solutions). This is a very important task because having diversified solutions has a great impact on the performance of these algorithms, especially when dealing with multimodal problems. Additionally, exploration helps the algorithm escaping from the local optimum trap (Črepinšek, M., Liu, S. H., & Mernik, M. (2013)). One of the common problems with many metaheuristic algorithms is the premature convergence that is caused by the poor diversity of the produced solutions. On the other hand, exploitation or intensification processes the neighboring areas of already discovered points in the problem space. In hilly or unimodal problem spaces, exploitation causes the algorithm to move step by step toward the local optimum solution, which is may be the global optimum solution. However for multimodal problem spaces relying just on exploitation causes the algorithms to being trapped in local optimum.

So, all these points clearly emphasize on the importance of balancing between exploitation and exploration in different steps of solving optimization problems. Managing exploration and exploitation can be done through two different strategies, offline and online (Črepinšek, M., Liu, S. H., & Mernik, M. (2013)). In the former, the controlling parameters of the algorithm are tuned to their best in the initialization step of the problem solving. Nevertheless, in this strategy, the parameter values remain unchanged to the end of the solving process. During the optimization process, different ratios between exploitation and exploration are required. Moreover, in the situation of problems with unknown problem spaces, the appropriate amount of exploitation and exploration is

unclear. So to deal more effectively with complex optimization problems, metaheuristic algorithms require a dynamic mechanism to find the appropriate ratio between exploitation and exploration. Regarding the online strategy, there are several ways, and in this study we propose to use a fuzzy system as the parameter controller to determine the ratio between exploitation and exploration more intelligently. This system must be able to decide when the algorithm needs to produce more diversified solutions through exploration and when to emphasis more on exploitation. The challenging issue here is to measure the progress of the algorithm (as well as finding out whether the algorithm is trapped in local optimum solution or not) to be able to decide about the amount of the exploration and exploitation. To design the desired fuzzy controller, we should consider some facts. First, in the initial optimization steps, we need diverse solutions. Second, in the final steps, if we are close to the optimal solution then the emphasis should be on the exploitation. Finally, according to the feedbacks of the algorithm to the fuzzy system, appropriate ratio between exploitation and exploration must be considered (this will be discussed in detail in this paper).

The standard Firefly algorithm has controlling parameters that are set at the initial step to adjust the amount of exploitation and exploration needed for a specific problem. According to above discussion, it follows the offline strategy. To deal more effectively with optimization tasks and confer an appropriate dynamism to the algorithm, we propose a fuzzy system as a parameter controller. With such online controller, firefly algorithm will enjoy an adaptive behavior in facing different situations in the solving the problems.

3. Related Work

In this section, we investigate reliable studies about improving the firefly performance as well as the application of fuzzy systems to other metaheuristic algorithms.

3.1 Improving Firefly

The firefly algorithm is suitable for problems that have fast convergence but not for a wide range of problems (Hassanzadeh, T., & Kanan, H. R. (2014)). In the literature, various strategies have been introduced to address the issue of exploitation and exploration specifically in firefly, such as employing Fuzzy systems, the chaotic theory, the Gaussian distribution and the theory of jumper searching agents.

Fuzzy systems as parameter controllers are a known approach to adjust exploration and exploitation in metaheuristic algorithms. Nevertheless, there are very limited studies that enhanced firefly based on Fuzzy logic or Fuzzy system. Our work differs widely from these studies since we have taken different design decisions regarding the components of the Fuzzy system and Fuzzy rule base. For instance, in (Hassanzadeh, T., & Kanan, H. R. (2014)), the authors presented a Fuzzy method to improve the collective behavior and global searching task of firefly. They employed Fuzzy logic to reform the firefly equations in order to attain a proper balance. In each of the iterations, more than one firefly can affect other fireflies as opposed to the behavior of the standard firefly. The attractiveness of one firefly is a proportion to its brightness, and the brightness of K fireflies is represented by a Fuzzy variable. This method improved the performance of firefly by shifting its tendency from local search to global search (Hassanzadeh, T., & Kanan, H. R. (2014)). In our work, we have developed a Fuzzy system to allocate dynamically appropriate values to the firefly parameters. We assess the trend of approaching the optimal solution with two measures.

To achieve a proper balance, in another work (Farahani, S. M., Abshouri, A. A., Nasiri, B., & Meybodi, M. (2011)), the authors exposed a new approach to improve the global search of firefly based on the Gaussian distribution. To be able to increase the participation rate of random search, the authors introduced a randomization strategy. A significant issue of firefly is its fixed movement steps that may cause to miss good quality solutions. By allocating suitable values to the controlling parameter *Alpha*, this approach adjusts the size of the random step in each of the iterations. It also adjusts the direction and size of the random step for the stochastic movement of the brightest firefly. The main strategy in (Farahani, S. M., Abshouri, A. A., Nasiri, B., & Meybodi, M. (2011)) is to change the size of random step based only on the iteration number. To make sure that the firefly algorithm approaches the optimal solution our method utilizes two different measures, not only the iteration number but also the other measure Delta to make sure that finding the optimal solution will happen.

Another approach to adjust the exploration-exploitation balance is to apply the chaotic theory. The most important application of this theory to the firefly algorithm is to encourage diversification (exploration) to help the algorithm escape the local optimum solutions traps. Dos Santos and al. applied the chaotic theory to the controlling parameters of firefly (which are constant values in standard firefly) and replaced them with the chaotic sequences to confer more stochastic behavior to firefly (dos Santos Coelho, L., de Andrade Bernert, D. L., & Mariani, V. C. (2011, June)). This approach greatly improved the firefly efficiency. The main difference between the technique of (dos Santos Coelho, L., de Andrade Bernert, D. L., & Mariani, V. C. (2011, June)) and

our work is that our Fuzzy system takes decisions based on the Fuzzy rules. Different rules are fired according to the situations being faced by firefly. On the other hand, the chaotic-based firefly allocates a chaotic series to the controlling parameters to diversify the solutions.

3.2 Fuzzy Logic for other Metaheuristic Algorithms

Fuzzy logic has also been considered to improve the performance of other metaheuristic algorithms. For example, in the study (Liu, Y., & Ma, L. (2011, May)), the authors utilized Fuzzy logic to address the immaturity or early convergence of the PSO algorithm. The proposed method, which employs a Fuzzy system, assigns proper values to the velocity parameter of PSO in different steps. Adjusting the velocity factor prevents the early convergence of PSO and encourages the global search. The designed Fuzzy system has 2 inputs (particle value (Value) and iteration counter (NC)) and 1 output (Velocity), and employs 5 Fuzzy sets and 25 Fuzzy if-then rules. In another research, Shi and Eberhart developed a 2 inputs-1 output Fuzzy system to enhance the performance of the canonical PSO (Shi, Y., & Eberhart, R. C. (2001)). The input variables are the current best solution and current inertia weight parameter, and the output is the updated inertia weight. The authors utilized 3 Fuzzy sets and 9 Fuzzy rules to decide about the values to the inertial weight. Hongbo and Ajith in (Liu, H., Xu, Z., & Abraham, A. (2005, September)) introduced a Fuzzy system to improve the efficiency of Genetics Algorithms (GAs) by adjusting the mutation and crossover probabilities. The underlying updating strategy is based on the maximum fitness and average fitness in the population. The Fuzzy system decides on the occurrence probability of the mutation and crossover operations based on Fuzzy rules: 15 rules to update the mutation probability and 15 others for the crossover probability. In (Qi, Z., & Chunming, P. (2010, August)), the Fuzzy system is used to increase the performance of GAs. Choosing proper values for occurrence probability of mutation and crossover has a significant impact on the GAs performance. The authors also tried to strike an appropriate balance between exploitation and exploration of GAs using Fuzzy logic. So in their Fuzzy system, probabilities of mutation and crossover are adjusted to acquire better solutions and faster convergence speed. Their designed Fuzzy system contains 7 fuzzy sets and 98 rules for each input.

4. Firefly Algorithm

In this section, we first describe the behavior of the firefly algorithm, and then the issue of exploitation and exploration within firefly.

4.1 Standard Firefly

There are hundreds of firefly species in nature and most of them produce short and rhythmic flashes. The motive behind the flashing light is twofold (Yang, X. S., & Deb, S. (2009, December), Yang, X. S. (2010)): (1) to communicate with other fireflies and attract matting partners, and (2) to allure and hunt prey. The rhythm, rate and duration of the flashing light all form one part of the signal system whose purpose is to bring both sexes together. In the same specie, females respond to a male's unique pattern of flashing (Fister, I., Yang, X. S., & Brest, J. (2013)). Actually, fireflies tend to move toward those that are the brighter**.** The attractive power depends on the brightness of the firefly i.e. its light intensity, which actually attenuates with the distance. The light intensity I at a particular distance r from the light source obeys the inverse square law. In other words, I decreases as r increases in terms of I / r^2. Furthermore, the light is absorbed by air and becomes weaker and weaker. These two combined factors may make fireflies' vision limited.

The firefly algorithm, introduced by Xin-She Yang in 2008 (Yang, X. S. (2010)), is one of the popular Metaheuristic algorithms that can address a large variety of non-trivial optimization problems. The steps of firefly are exposed in Algorithm 1 where m is the number of fireflies, α the size of the random step, γ the absorption coefficient, and I the light intensity at source, which can be defined as $I = f(x_i)$ or $I=1$. We may note that $\gamma \in [0, \infty)$ and $\alpha \in [0,1)$.

Input : f(x) //fitness function
$(x_1,\ldots,x_n)$ //locations of fireflies
m,γ,α,I //problem constant
maxGen //iteration number
Output: x_{min} //location of the brightest firefly
For i=1:m
x_i= initialSolution() //distribute fireflies stochastically in the problem space

```
Endfor_i
For t=1:maxGen
min=argmin_{i={1,....m}}(f(x_i))    //update fitness of the optimal solution
For i=1:m
  For j=i+1:m
    If f(x_i)<f(x_j)
      Determine distance r_{i,j}          //calculated with equation 1
      Determine attractiveness β // calculated with equation 2
      Determine next location x_i  //calculated with equation 3
    Endif
  Endfor_j
  Endfor_i
  Update x_min  // calculated with 4
Endfor_t
```

Algorithm 1. Standard firefly algorithm (updated from (Durkota, K. (2011)).

The other features, like distance $r_{i,j}$, attractiveness β and the movement behavior of fireflies, are described in Table 1 (Yang, X. S. (2010)). *Random()* is a function that produces uniformly distributed vectors of [0,1).

Table 1. Features of firefly algorithm

Feature	Description	Equation	
Distance	Distance between two fireflies i and j at locations x_i and x_j respectively.	$r_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{n}(x_{i,k} - x_{j,k})^2}$	(1)
Attractiveness	Attractiveness of a firefly closely depends on the light intensity one can see from distance r and absorption coefficient γ.	$\beta = \frac{I_0}{1+\gamma.r^2}$	(2)
Movement	The movement of firefly i toward firefly j (at x_i and x_j) that is brighter and more attractive.	$x_i = x_i + \beta.(x_i - x_j) + \alpha.\left(random() - \frac{1}{2}\right)$	(3)
Brightest firefly movement	The brightest firefly at location x_{min} moves stochastically.	$x_{min} = x_{min} + \alpha.(random() - \frac{1}{2})$	(4)

According to the movement equation (3), each firefly utilizes random step *(α.(random()-1/2))* and local step *(x_i+β.(x_i+ x_j))* when moving toward the brighter fireflies Yang, (X. S. (2010)). The brightness of fireflies, representing the solution quality, is calculated with a fitness function that is defined according to the problem being solved (Yang, X. S. (2010)). The brightest firefly is the optimal solution.

4.2 Controlling Parameters of Firefly Algorithm and its Extreme Cases

Determining the exploitation and exploration balance has always been a challenging issue in metaheuristic algorithms. Here we investigate the movement equation of firefly to address this issue. As discussed before, every firefly when moving toward the brighter fireflies follows equation 3. The latter is composed of the movement induced by better firefly locations (their solutions) and random movement. The positions of better

fireflies are considered as hints for quality points, and the weaker firefly moves toward them by searching their neighboring areas in attempt to find higher quality solutions to enhance its status (quality). Therefore, according to the definition of the exploitation, this induced motion by better fireflies is the local search. This movement is controlled mainly by *Gamma* parameter ($\gamma \in [0, \infty)$). So, different values of *Gamma* cause different amounts of exploitation. On the other hand, fireflies also perform random movement not influenced by other fireflies. They just stochastically move through the problems space, and this results in discovering new points that are not necessarily in the neighboring areas of already discovered solutions. As stated in equation 3, the random movement part is controlled by *Alpha* ($\alpha \in [0,1)$). In fact *Alpha* controls the size of the random steps of firefly i.e. the exploration. To find out about the importance of tuning the two controlling parameters, we discuss extreme cases in firefly algorithm.

From equation (3), there are two extreme cases of the firefly algorithm as demonstrated in (Durkota, K. (2011)):

$$\textbf{Case 1: } \gamma \to \infty : lim_{\gamma \to \infty}\beta = lim_{\gamma \to \infty} I_0 . e^{-\gamma . r^2} = 0$$

As a result, equation (3) turns into equation (5):

$$x_i = x_i + \alpha . (random( ) - 1/2) \tag{5}$$

According to the equation 5, the only movement is random movement. Here there are no hints about quality solutions' positions and fireflies move without considering the previous searching steps. We already know that searching problem spaces blindly does not result in finding the optimum solution. This is specially the case when the algorithm is dealing with complex problems with high dimensional problem spaces.

$$\textbf{Case 2: } \gamma \to 0 : lim_{\gamma \to 0}\beta = lim_{\gamma \to 0} I_0 . e^{-\gamma . r^2} = I_0$$

As a result, equation (3) turns into equation (6):

$$x_i = x_i + I_0 . (x_i - x_j) + \alpha . \left(random( ) - \frac{1}{2} \right) \tag{6}$$

Equation (6) states that the distance parameter between fireflies is removed, and therefore fireflies observe each other very clearly (Durkota, K. (2011)). In this case, exploitation reaches its highest value while exploration remains normal. This reveals that the algorithm mainly does local search, and this increases the probability of being trapped in local optimum (Durkota, K. (2011)).

In regard to these two cases, the need for a parameter controller is irrefutable. Such controller will achieve a balance between exploration and exploitation under different situations that may occur in each solving step of the firefly algorithm.

4.3 Exploration and Exploitation Balance in Firefly

Controlling parameters of firefly algorithm which control the exploitation and exploration amount are *Gamma* and *Alpha*, utilized in equations (2) and (3). Gamma controls the exploitation and Alpha the exploration. The behavior of firefly should be constantly monitored to be able to allocate proper values to its controlling parameters in order to attain the right balance (online strategy) (Liu, Y., & Ma, L. (2011, May)). In the initial solving steps, there is no hint about the location of the optimal solution. So, the maximum participation rate of exploration is needed (to produce more diversity solution) to produce diverse solutions which is very important issue in metaheuristic algorithms. As discussed in previous section this diversity causes algorithm not to be trapped in local optimum solutions at the initial steps of solving problems**.** As the algorithm approaches the optimal solution, the contribution rate of exploration must be reduced and exploitation must be increased. But in some steps that progress of the solving a problem is weak, it implies that algorithm is stuck in local optimum solution and suffers from premature convergence due to lack of appropriate diversity solutions. In this situation emphasizing the exploration to produce more diversity solutions helps the algorithm to escape the trap. In the final steps, the minimum distance to the goal is expected, so a high rate of exploration cannot be fruitful as it may causes the algorithm lose the optimal solution. Consequently, in the final steps, the burden of the search must be carried out mainly by exploitation rather than exploration. General speaking, from the beginning to the final steps, the maximum rate of exploration decreases to a minimum quantity, and the minimum rate of exploitation increases to a maximum quantity. This trend, which is illustrated in Figure 1, is the basis to design a Fuzzy rule base for our parameter controller (the vertical axis indicates the range of exploitation/exploration amounts and horizontal axis the progress of problem solving, iteration simply).

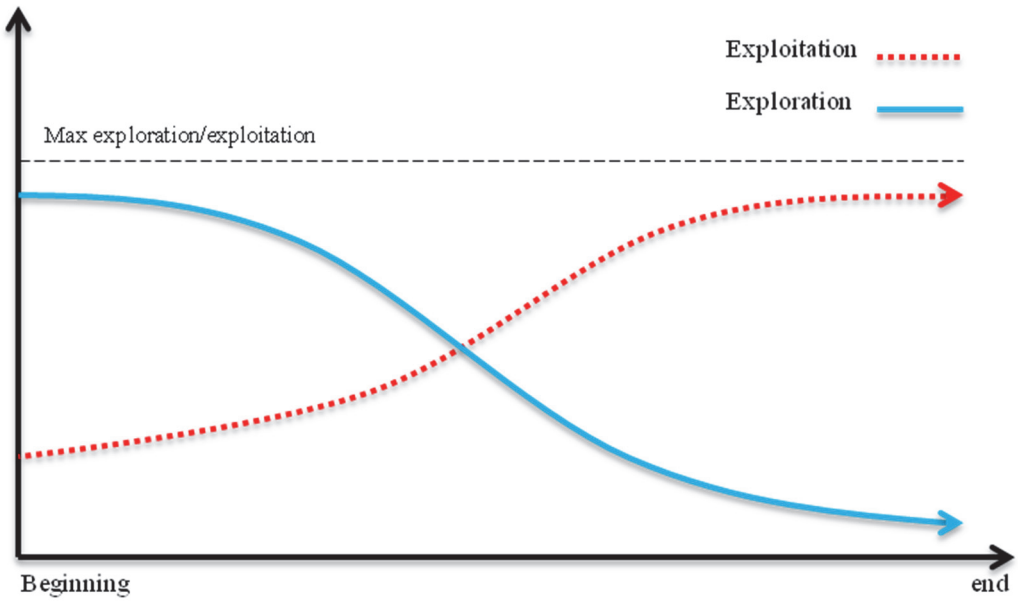

Figure 1. Change trend of exploration and exploitation from the beginning to the end

5. An Overview on Fuzzy Systems

Fuzzy logic was introduced to mathematically represent uncertainty, imprecision and vagueness of information, intrinsic features of many problems. The heart of a Fuzzy system is the rule base that contains If-Then rules designed by the experts of the given application (Wang, L. X. (1999). *A course in fuzzy systems* (pp. 258-265)).

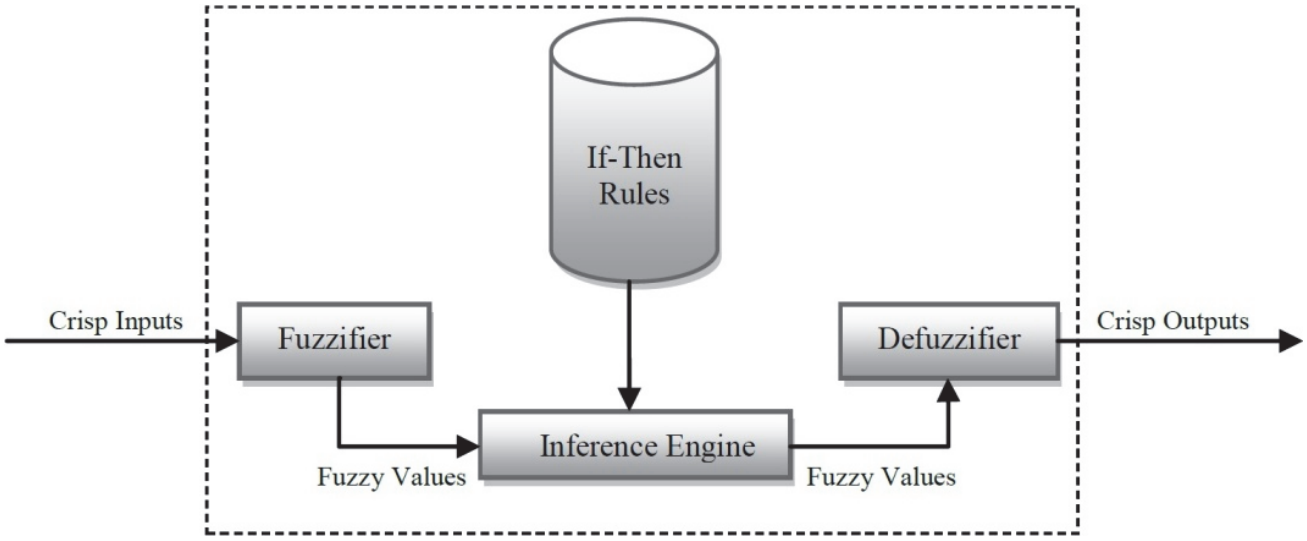

Figure 2. A general Fuzzy system (Gandomi, A. H. (2014))

A general fuzzy system is depicted in Figure 2. The responsibility of the fuzzifier is to convert a crisp input into a fuzzy value (with one or more fuzzy sets) to be processed by the inference engine (Wang, L. X. (1999)). The fuzzy sets are deployed with membership functions. Some of the most well known fuzzifiers are singleton, Gaussian and triangular. On the other hand, defuzzification is the process of converting a Fuzzy value back to the crisp domain (Wang, L. X. (1999)). There are different choices for defuzzifier, the most famous ones are center of gravity, center average and maximum defuzzifiers. When running the inference engine, *If-Then* rules are fired according to the principles of Fuzzy logic. The syntax of a Fuzzy rule is as follows (Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005)):

If x_1 *is* A_1 *and ...* x_n *is* A_n ***Then*** Y *is* B

where $\boldsymbol{A_i}$ is a Fuzzy set in $U \subset R$ (U is the input space), B is a Fuzzy set in $V \subset R$ (V is the output space), $X = (x, x, \ldots, x)^T \in U$ and $Y \in V$ input and output variables respectively (Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005)).

There are different options for the inference engine, such as Product Inference Engine, Minimum Inference Engine, Lukasiewicz Inference Engine and Dienes-Rescher Inference Engine. We refer the reader to (Wang, L. X. (1999)) for more information. Fuzzy systems can be employed as controlling components in two different ways: (1) closed-loop controller, or (2) open-looped controller. The first controller is applied when a continuous control action of a process is needed. The fuzzy system measures the outputs continuously to perform its mission. However when it is applied as an open-loop controller, the fuzzy system usually sets up the controlling parameters, and then operates with respect to these parameters. Since we need to continuously monitor the progress of the firefly algorithm to be able to allocate appropriate values to the controlling parameters, we then utilize the closed-looped Fuzzy controller.

Example: Let us assume a Fuzzy system (with 2 inputs and 1 output) is designed with the following two rules:

R1*: If (Count is L) and (Delta is LM) then Gamma is L*

R2*: If (Count is LM) and (Delta is L) then Gamma is LM*

where *L* and *LM* are two Fuzzy sets deployed with the following membership functions:

$$\mu_L = \begin{cases} \frac{0.3 - x}{0.3} & 0 < x \le 0.3 \\ 1 & x \le 0 \\ 0 & otherwise \end{cases} \tag{7}$$

$$\mu_{LM} = \begin{cases} \frac{x - 0.2}{0.15} & 0.2 < x \le 0.35 \\ 1 & .35 < x \\ 0 & otherwisw \end{cases} \tag{8}$$

We would like to apply the Singleton fuzzifier and the Product Inference Engine defined respectively as follows:

$$\mu_A(X) = \begin{cases} 1 & if X = X^* \\ 0 & otherwise \end{cases} \tag{9}$$

$$\mu_B(Y) = Sup_{x \in U} Max_{l=1}^{M}[(\mu_A(X).\prod_{i=1}^{N} \mu_{A_i}.\mu_B(Y))] \tag{10}$$

We may note that if X^* is some point in U, then the inference engine is changed to equation (11):

$$\mu_B(Y) = Max_{l=1}^{M}[(\mu_A(X).\prod_{i=1}^{N} \mu_{A_i}.\mu_B(Y))] \tag{11}$$

When the input pair *(Delta, Count) is set to (0.21, 0.23)*, *the* equation (9) will produce the Fuzzy output:

$$\mu_{B'}(y) = \max[\mu_L(0.21).\mu_{LM}(0.23).\mu_L(y), \mu_L(0.23).\mu_{LM}(0.21).\mu_{LM}(y)]$$
$$= \max[0.3 \times 0.2 \times \mu_L(y), 0.23 \times 0.067 \times \mu_{LM}(y)] = [0.06 \times \mu_L(y), 0.015 \times \mu_{LM}(y)]$$

Now by employing the Center Average Weight defuzzifier (Haynes, W., (2013)), the output Y (here Gamma) would be:

$$y = \frac{\sum_{L=1}^{M} y^{-L} \times w_L}{w_L} = \frac{y^{-1} \times w_1 + y^{-2} \times w_2}{w_1 + w_2} = \frac{0.015}{0.06 + 0.015} = 0.2$$

where y^{-1} and y^{-2} denote the center of the first and second Fuzzy sets respectively, and $y^{-1}=0$ and $y^{-2}=1$. We can see that small changes in weights (heights of the Fuzzy sets) result in small changes in the output Y.

6. Proposed Fuzzy-Based Firefly Algorithm

A decision making task needs to be performed simultaneously with firefly when processing optimization applications. A Fuzzy system as parameter controller is proposed in this section to enhance the performance of

the firefly algorithm.

6.1 Improving Firefly

We utilize two measures to examine the progress trend of firefly in approaching the optimal solution. The first one is the main loop counter (iteration) of the algorithm, renamed here *Count* and whose value is determined based on the experts' experience or by trial and error. To cover all probable cases that may occur during the searching process because of the stochastic nature of firefly, another measure is needed to warn the fuzzy controller system about unsuitable situations. The latter must be addressed by the fuzzy controller. Thus we introduce the measure, called *Delta*, which is calculated according to equation (12):

$$Delta^i = F(Best^i) - F(TBest^{i-1}) \tag{12}$$

When maximizing an optimization problem:

$$F(TBest^{i-1}) = Max(TBest^{i-2}, Best^{i-1}) \tag{12.1}$$

And when minimizing the problem:

$$F(TBest^{i-1}) = Min(TBest^{i-2}, Best^{i-1}) \tag{12.2}$$

where $Best^i$ is the best solution in iteration i and $TBest^{i-1}$ is the best solution found from the beginning to $(i-1)$-*th* iteration. In fact, the algorithm records the history of its achievements to make decision in the current step. Equation (12) calculates the difference between the best solution in iteration i and the best solution found before iteration i. It is expected that *Delta* decreases gradually from the beginning to the end. But if some undesirable situations occur, the designed Fuzzy system helps firefly overcome these situations and approach the optimal solution.

The mathematical representation of the proposed fuzzy system is given below. More precisely, we employ five Fuzzy sets with the labels of *L* (Low), *LM* (Low-Medium), *M* (Medium), *MH* (Medium-High) and *H* (High) to denote the different values of the two inputs *Delta* and *Count*.

Inputs: *{Delta,Count}=(X)*

Outputs: *{Alpha, Gamma}=(Y)*

Input and Output Fuzzy Sets: *{L, LM, M, MH, H}*

Fuzzy Rule: $R: A_1 \times \ldots \times A_n \rightarrow B$

$R: L \times LM \times M \times MH \times H \rightarrow \{L, LM, M, MH, H\}$

Triangular Membership Function:

$$\mu_{Ru}(Delta, Count, \{Alpha, Gamma\}) = \mu_{L \times LM \times M \times MH \times H \rightarrow B}(Delta, Count, \{Alpha, Gamma\})$$

Product Inference Engine (with Mamdani's product implication):

$$\mu_B(Y) = Sup_{x \in U} Max_{l=1}^{M}[(\mu_A(X).\prod_{i=1}^{N} \mu_{A_i}.\mu_B(Y))]$$

where M is the number of Fuzzy rules and N the number of Fuzzy sets, which are 25 and 5 respectively in our fuzzy system.

6.2 Fuzzy Controller

Figure 3 depicts the closed-loop relationship of the enhanced firefly algorithm with our fuzzy system. In each searching step, the fuzzy system receives two inputs from the algorithm, *Delta* and *Count,* updates *Alpha* and *Gamma* accordingly, and then sends them to firefly in order to adjust the exploration and exploitation rates. The proposed system controls the algorithm behavior in run-time. This will definitely improve the performance of the searching tasks. More precisely, our system employs the *Singleton* fuzzifier, *Mamdani* Inference Engine, *Centroid* defuzzification, *Max* Aggregation, *Min* Implication, *Max* for the Or method, *Prod* for the And method, and a rule-base with 25 *if-then* rules. As we can see in Algorithm 2, the improved firefly calculates *Delta* and then calls the fuzzy system in each resolution step.

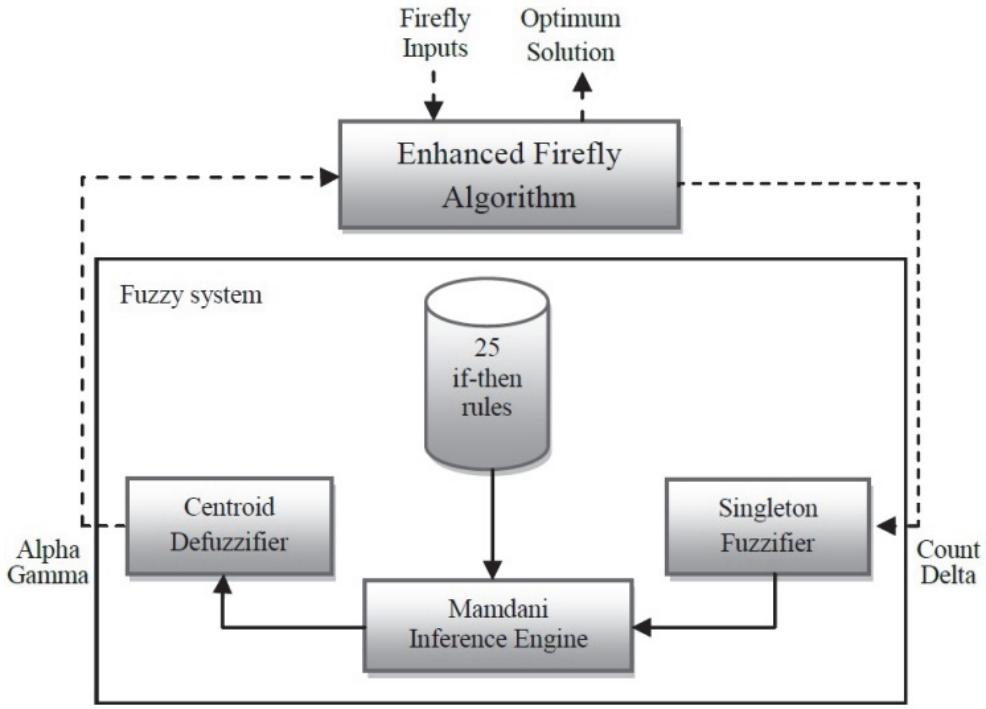

Figure 3. General Schematic of Fuzzy-based firefly

Input : *f(x)* *//fitness function*

$(x_1, \ldots, x_n)$ *//locations of fireflies*

m,γ,α,I *//problem constant*

maxGen *//iteration number*

Output: x_{min} *//location of the brightest firefly*

For i=1:m

x_i= *initialSolution() //distribute fireflies stochastically in the problem space*

Endfor$_i$

For Count=1:maxGen

min=argmin$_{i=\{1,\ldots,m\}}$***(f(x*$_i$***))*** *//update fitness of the optimal solution*

For i=1:m

For j=i+1:m

If $f(x_i)<f(x_j)$

Determine distance $r_{i,j}$ *//calculated with equation 1*

Determine attractiveness β *//calculated with equation 2*

Determine next location x_i *//calculated with equation 3*

Endif

Endfor$_j$

Endfor$_i$

Determine x_{min} *//calculated with 4*

Determine Delta *//calculated with equation 7*

Based on delta and Count, get alpha and Gamma from Fuzzy

---

controller

Endfor$_{Count}$

Algorithm 2. Fuzzy-based firefly algorithm

We have developed the entire fuzzy system by using the fuzzy toolbox MATLAB 2010 as presented in Figure 4.

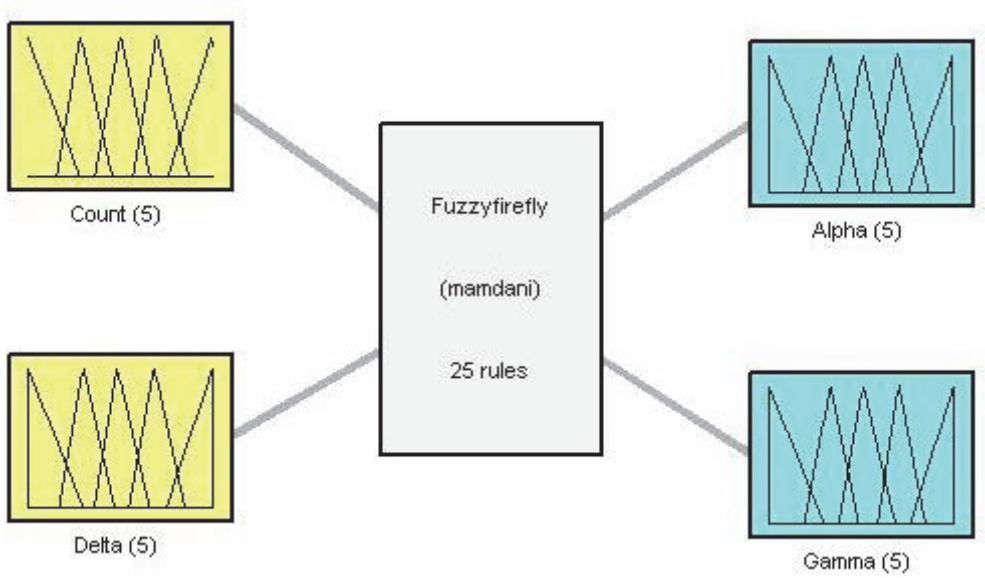

Figure 4. Fuzzy controller in MATLAB 2010

Assume that the parameter *Delta* is in the interval of [0, 100]. Considering this domain, the triangular membership function determines the degree of Delta membership to each of the five fuzzy sets. For example when *Delta=5*, its membership degree to *L* is 0.9 and 0 to the other four sets (see Figure 5).

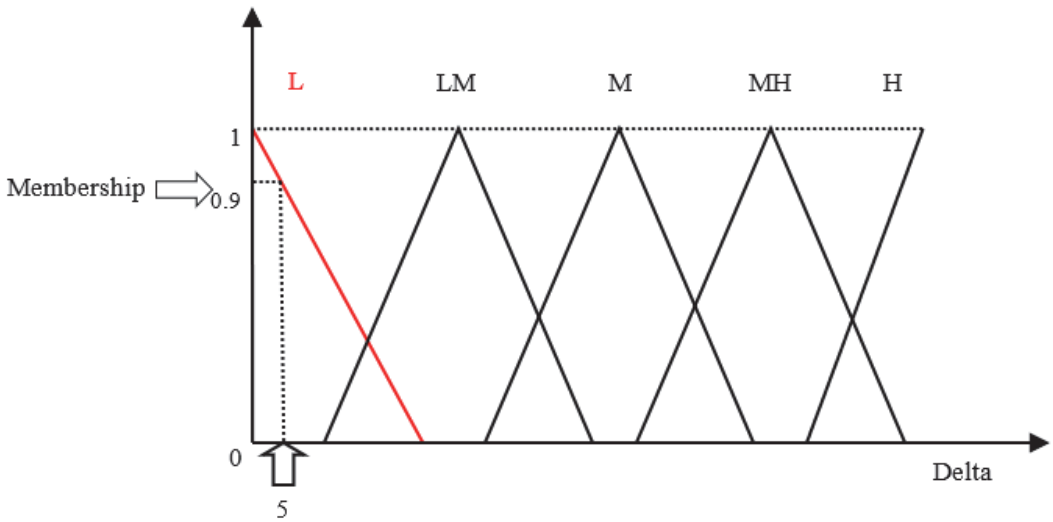

Figure 5. Membership degree of *Delta=5* to Fuzzy sets

6.3 Fuzzy Rule-Base

The rules of a fuzzy system define the logical relationship between the inputs and outputs. The rule base should be meticulously designed so that it covers all the situations firefly may encounter when searching for the optimality. Additionally, the base should contain no conflicting rules, meaning that no more than one rule fire for the inputs. We propose a base with a total of 25 rules exposed in Table 2. The main strategy behind these rules is

to emphasize on the global search when there are not much hints about the optimum solution, but more we approach the optimal solution more we emphasize on the local search.

Table 2. Rule base of the Fuzzy controller (Fuzzy If-Then rules)

1.	*If (Count is L) and (Delta is L)*	***Then*** *(Alpha is H)(Gamma is H)*
2.	*If (Count is L) and (Delta is LM)*	***Then*** *(Alpha is H)(Gamma is MH)*
3.	*If (Count is L) and (Delta is M)*	***Then*** *(Alpha is MH)(Gamma is MH)*
4.	*If (Count is L) and (Delta is MH)*	***Then*** *(Alpha is MH)(Gamma is MH)*
5.	*If (Count is L) and (Delta is H)*	***Then*** *(Alpha is M)(Gamma is MH)*
6.	*If (Count is LM) and (Delta is L)*	***Then*** *(Alpha is H)(Gamma is MH)*
7.	*If (Count is LM) and (Delta is LM)*	***Then*** *(Alpha is H)(Gamma is HM)*
8.	*If (Count is LM) and (Delta is M)*	***Then*** *(Alpha is MH)(Gamma is MH)*
9.	*If (Count is LM) and (Delta is MH)*	***Then*** *(Alpha is MH)(Gamma is MH)*
10.	*If (Count is LM) and (Delta is H)*	***Then*** *(Alpha is H)(Gamma is MH)*
11.	*If (Count is M) and (Delta is L)*	***Then*** *(Alpha is M)(Gamma is M)*
12.	*If (Count is M) and (Delta is LM)*	***Then*** *(Alpha is M)(Gamma is M)*
13.	*If (Count is M) and (Delta is M)*	***Then*** *(Alpha is MH)(Gamma is M)*
14.	*If (Count is M) and (Delta is MH)*	***Then*** *(Alpha is M)(Gamma is LM)*
15.	*If (Count is M) and (Delta is H)*	***Then*** *(Alpha is MH)(Gamma is MH)*
16.	*If (Count is MH) and (Delta is L)*	***Then*** *(Alpha is L)(Gamma is LM)*
17.	*If (Count is MH) and (Delta is LM)*	***Then*** *(Alpha is LM)(Gamma is M)*
18.	*If (Count is MH) and (Delta is M)*	***Then*** *(Alpha is M)(Gamma is MH)*
19.	*If (Count is MH) and (Delta is MH)*	***Then*** *(Alpha is MH)(Gamma is H)*
20.	*If (Count is MH) and (Delta is H)*	***Then*** *(Alpha is MH)(Gamma is MH)*
21.	*If (Count is H) and (Delta is L)*	***Then*** *(Alpha is L)(Gamma is L)*
22.	*If (Count is H) and (Delta is LM)*	***Then*** *(Alpha is LM)(Gamma is L)*
23.	*If (Count is H) and (Delta is M)*	***Then*** *(Alpha is MH)(Gamma is LM)*
24.	*If (Count is H) and (Delta is MH)*	***Then*** *(Alpha is MH)(Gamma is MH)*
25.	*If (Count is H) and (Delta is H)*	***Then*** *(Alpha is H)(Gamma is M)*

As an example, let's assume that Count is L and Delta is M. In this case, the third rule is fired:

If *Count is L* ***and*** *Delta is M* ***then*** *Alpha is MH* ***and*** *Gamma is LM*

When *Count* is *L*, it means that the searching process is still in its beginning steps and the emphasis should be on the random steps to search more globally and produce more diverse solutions. *Delta* is *M* means that the difference between the current best solution and the best solution found so far belongs to the medium set. So these values of *Count* and *Delta* fire the third rule, which sets *Alpha* to *MH* and *Gamma* to *LM.* These values put the emphasis more on exploration and less on exploitation.

The 3-D schematic of changes in variables *Alpha* and *Gamma by* considering the changes in variables *Count* and *Delta* are illustrated in Figures 6 and 7. We can see that the fuzzy controller is designed in a way that from the beginning to the end of the solving problem, by approaching the optimal solution, the emphasis of the algorithm gradually changes from exploration to exploitation.

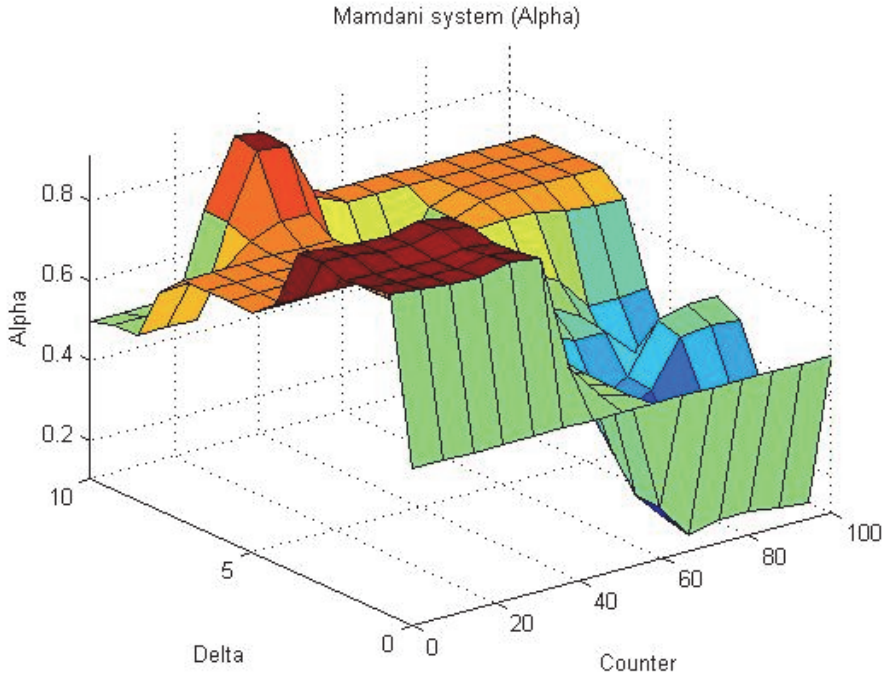

Figure 6. The 3-D surface plot of output Alpha

To get more familiar with the functionality of the new Fuzzy system, consider Figure 8. We may note for each pair of *Delta* and *Count*, there would be a rule to be fired to produce the specific outputs (Alpha and Gamma). For instance, when *Count=10* ($Count \in [0,20]$) and *Delta=2* ($Delta \in [0,4]$), the Fuzzy system would produce *Alpha=0.705* and *Gamma=0.524*.

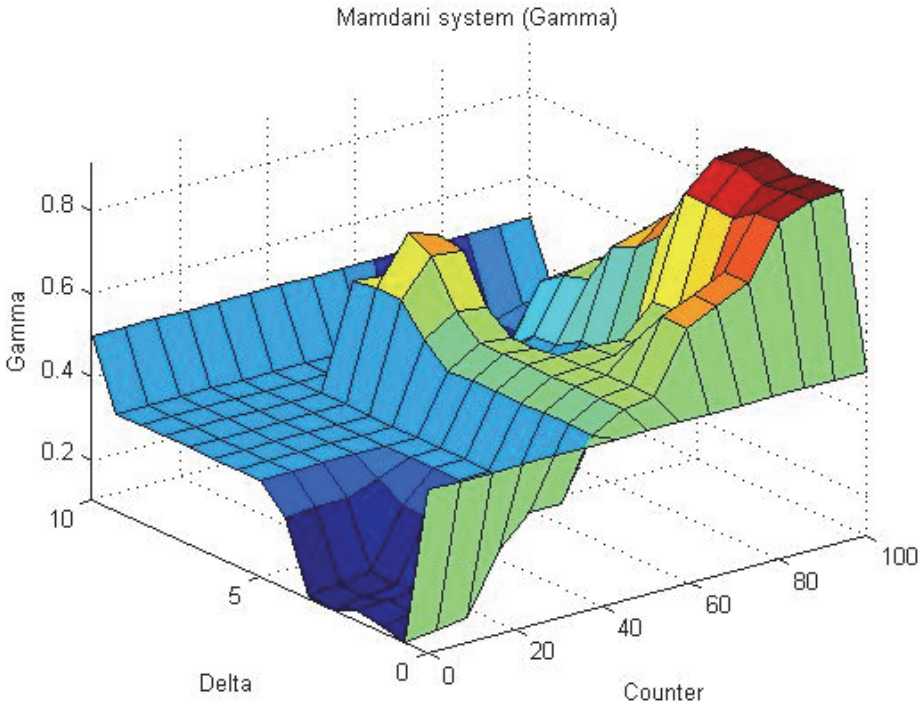

Figure 7. The 3-D surface plot of output Gamma

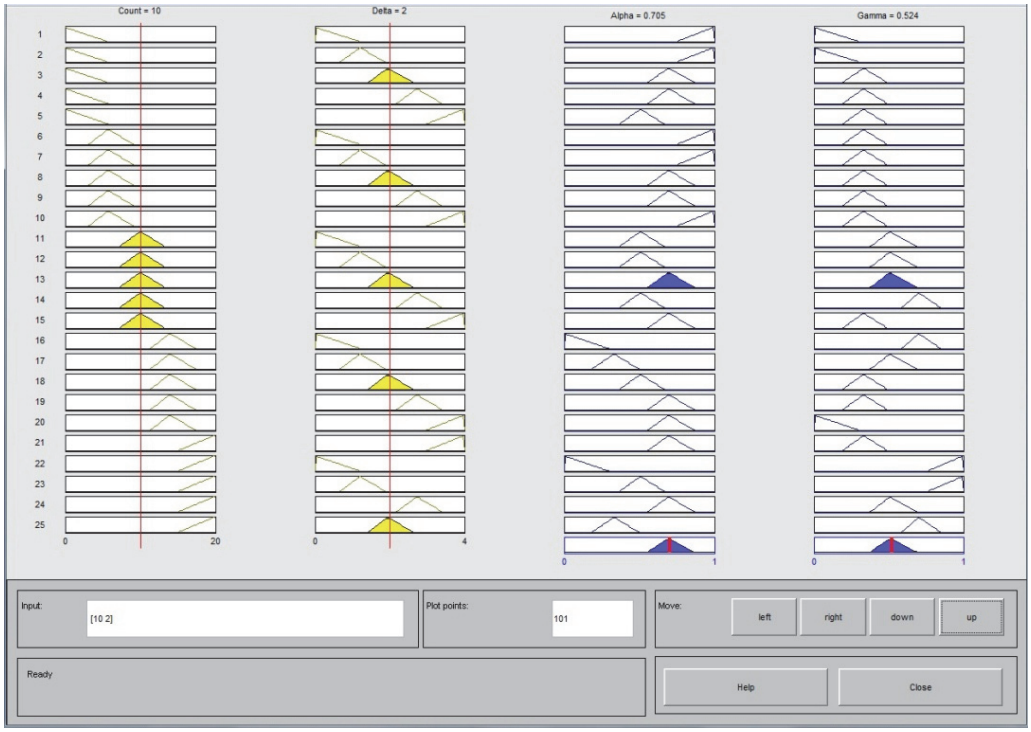

Figure 8. The rule viewer of the Fuzzy controller

7. Low and High Dimensional Problems

To properly evaluate the performance of our Fuzzy-based Firefly Algorithm (FFA), we conduct several experiments based on well-known benchmark problems. The benchmark set contains 10 low and 10 high dimensional standard functions. Moreover, we compare the results achieved by the new method with those of the standard firefly as well as with other famous nature-inspired optimization algorithms, including Differential Evolution (DE), Particle Swarm Optimization (PSO), Accelerated PSO (APSO), Biogeography-Based Optimization (BBO), Interior Search Algorithm (ISA) and Krill Herd (KH). In fact, we have applied these four algorithms to the 20 functions.

7.1 Standard Benchmark Functions

We employ high dimensional (F1-F10) and low dimensional (F11-F20) continuous functions that are difficult to solve due to their dimensionality (Bidar, M., & Kanan, H. R. (2013, August)) and are used frequently by researchers to examine the performance of different optimization methods. These functions can be divided into two categories: unimodal and multimodal (functions that have more than one local optimum and therefore it is difficult to find the optimum solution (Bidar, M., & Kanan, H. R. (2013, August), Hassanzadeh, T., & Kanan, H. R. (2014)). The goal here is to minimize these functions described in Table 3. Except for *Schwefel* 2.21, all the other functions are differentiable.

7.2 Performance Evaluation and Comparison

The population size is set to 100, *Gamma* and Alpha are in the intervals [0, 1] and [0,1) respectively. *Gamma* is set to its minimum value and *Alpha* to its maximum at the beginning of the searching process to emphasis the exploration. Due to the stochastic behavior of the metaheuristic optimization algorithms, their performance cannot be judged in one single run. So, the experimental results are the average of 50 trials. In order to facilitate the comparison, all the results are normalized according to following equation (Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005)):

$$X_{i,normalized} = 1 - \frac{(X_i - X_{\min})}{(X_{\max} - X_{\min})} \tag{13}$$

where $X_{i,normalized}$ is the normalized value of solution *i*, X_i the fitness value of solution *i*, $X_{\min}$ and $X_{\max}$ its

minimum and maximum fitness values respectively. The normalized statistical results of the proposed method along with the results of the other metaheuristic algorithms on high and low dimensional benchmark functions are given in Tables 4 and 5 respectively. Due to the random nature and stochastic behavior of the nature-inspired algorithm their real performance cannot be judged based on a single run. So in our experiments we report the results achieved in 50 runs. The results demonstrate that the proposed fuzzy controller increased the performance of the Standard Firefly Algorithm (SFA) since FFA outperforms SFA in all the experiments. Also, FFA is superior to the other metaheuristic algorithms in nine out of twenty experiments, and very close in the other experiments.

Table 3. Benchmark functions

ID	Name	Dimension	Separability Modality	Domain	Min Value/Solution
			High Dimensional		
F1	*Ackley*	20	Non-Sep./Multimodal	$-32 \le x_i \le 32$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F2	Griewank	20	Non-Sep./Multimodal	$-600 \le x_i \le 600$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F3	*Rosenbrock*	20	Non-Sep./Unimodal	$-30 \le x_i \le 30$	$f(x^*) = 0/x^* = (1, \dots, 1)$
F4	*Schwefel* 2.26	20	Sep./Multimodal	$-500 \le x_i \le 500$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F5	*Schwefel* 2.22	20	Non-Sep./Unimodal	$-100 \le x_i \le 100$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F6	*Schwefel* 2.21	20	Sep./Unimodal	$-100 \le x_i \le 100$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F7	*Schwefel* 1.2	20	Non-Sep./Unimodal	$-100 \le x_i \le 100$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F8	*Sphere*	20	Sep./Multimodal	$-100 \le x_i \le 100$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F9	*Rastrigin*	20	Sep./Multimodal	$-512 \le x_i \le 512$	$f(x^*) = 0/x^* = (0, \dots, 0)$
F10	*Quatric*	20	Sep./Unimodal	$-1.28 \le x_i \le 1.28$	$f(x^*) = 0/x^* = (0, \dots, 0)$
			Low Dimensional		
F11	*Branin*	2	Sep./Multimodal	$-5 \le x_1 \le 10$ $0 \le x_2 \le 15$	$f(x^*) = 0.39788$/(-π, 12.275), (π, 2.275), (9.424, 2.475)
F12	*Camel Back-6 Hump*	2	Non-Sep./Multimodal	$-3 \le x_1 \le 3$ $-2 \le x_2 \le 2$	$f(x^*) = -1.031628$/ (0.0898, -0.7126), (-0.0898, 07126)
F13	*Shekel's foxholes*	2	Non-Sep./Multimodal	$-65.536 \le x_i \le 65.536$	$f(x^*) = 1$/(-32,32)
F14	*Goldstein and Price*	2	Non-Sep./Multimodal	$-2 \le x_i \le 2$	$f(x^*) = 3$/(-1,0)
F15	*Hartman 1*	4	Non-Sep./Multimodal	$0 \le x_i \le 1$	$f(x^*) = -3.86$/ (0.114, 0.556, 0.582)
F16	*Hartman 2*	6	Non-Sep./Multimodal	$0 \le x_i \le 1$	$f(x^*) = -10.1532$/ (0.201, 0.150, 0.477, 0.275, 0.311, 0.657)
F17	*Kowalik*	4	Sep./Unimodal	$-5 \le x_i \le 5$	$f(x^*) = 0.000307$/ (0.1928, 0.1908, 0.1231, 0.1357)
F18	*Shekel 1*	4	Non-Sep./Multimodal	$0 \le x_i \le 10$	$f(x^*) = -10.1532$/(4, 4 , 4 , 4)
F19	*Shekel 2*	4	Non-Sep./Multimodal	$0 \le x_i \le 10$	$f(x^*) = -10.4029$/(4, 4 , 4 , 4)
F20	*Shekel 3*	4	Non-Sep./Multimodal	$0 \le x_i \le 10$	$f(x^*) = -10.5346$/(4, 4 , 4 , 4)

Table 4. Statistical results of FFA and other metaheuristic algorithms on low dimensional problems

ID	Criterion	FFA	SFA	KH	DE	PSO	APSO	BBO	ISA
F1	Best	0.987812	0.654211	0.999147	0.127942	0.856125	0.999999	0.912472	0.951744
	Mean	0.973306	0.395789	0.964644	0.000000	0.735129	0.971862	0.892411	0.932419
	Std.dev.	0.712593	0.754863	0.579421	0.987452	0.774162	0.548742	0.251441	0.514741
F2	Best	0.961567	0.881627	1.000000	0.000124	0.801594	1.000000	0.923514	0.941024
	Mean	0.932378	0.668126	0.998250	0.000000	0.684218	0.884371	0.901427	0.914781
	Std.dev.	0.910224	0.198541	0.221547	0.101268	0.558468	0.847624	0.519741	0.351478
F3	Best	0.987139	0.879512	1.000000	0.254128	0.995421	1.000000	0.967154	1.000000
	Mean	0.923885	0.784179	0.998696	0.243509	0.951247	1.000000	0.920131	1.000000
	Std.dev.	0.628442	0.612487	0.246514	0.351247	0.764851	1.000000	0.514774	0.258415
F4	Best	0.874323	0.887411	0.945217	0.621444	0.901247	1.000000	0.892141	0.880144
	Mean	0.827165	0.842713	0.992304	0.423967	0.890274	0.937512	0.831547	0.840012
	Std.dev.	0.751286	0.684527	0.754123	0.847612	0.554128	0.884571	0.514789	0.248147
F5	Best	0.900612	0.481225	0.945195	0.625874	0.754921	0.897452	0.789421	0.892491
	Mean	0.885108	0.778996	0.874231	0.368184	0.715843	0.882184	0.751478	0.860214
	Std.dev.	0.602186	0.879521	0.254158	0.984512	0.554127	0.845271	0.845712	0.454784
F6	Best	1.000000	0.845717	1.000000	0.754680	0.812765	0.871462	0.945871	0.932541
	Mean	1.000000	0.754812	1.000000	0.678919	0.741953	0.832814	0.902451	0.912457
	Std.dev.	0.963805	0.435574	1.000000	0.845219	0.351795	0.512473	0.487541	0.512475
F7	Best	0.991974	0.751243	0.421597	0.000000	0.674158	0.912842	0.984512	0.995051
	Mean	0.976963	0.612557	0.320151	0.000000	0.631542	0.799514	0.912489	0.925141
	Std.dev.	0.751632	0.847125	0.736841	0.000000	0.684219	0.421762	0.514782	0.351894
F8	Best	1.000000	0.884714	1.000000	0.602478	0.715482	0.932157	0.896514	1.000000
	Mean	0.970312	0.779141	1.000000	0.594219	0.687912	0.892347	0.822341	0.923584
	Std.dev.	0.712535	0.856412	0.254198	0.955142	0.665812	0.354199	0.512874	0.698541
F9	Best	0.961525	0.000000	0.855541	0.674519	0.972674	1.000000	0.962142	0.875691
	Mean	0.932374	0.000000	0.712949	0.642448	0.952347	1.000000	0.922154	0.829866
	Std.dev.	0.910296	0.678125	0.995124	0.884512	0.513478	0.651987	0.654147	0.584701
F10	Best	0.987152	1.000000	1.000000	0.899163	0.902147	1.000000	0.992451	0.926574
	Mean	0.923842	0.999998	1.000000	0.842163	0.899919	1.000000	0.940125	0.899999
	Std.dev.	0.628437	1.000000	0.842555	0.658411	0.351278	0.731945	0.684574	0.602144

Table 5. Statistical results of FFA and other metaheuristic algorithms on high dimensional problems

ID	Criterion	FFA	SFA	KH	DE	PSO	APSO	BBO	ISA
F11	Best	0.917241	0.907451	0.892641	0.800214	0.845768	0.974197	0.865478	0.896587
	Mean	0.841251	0.666128	0.847122	0.651244	0.817945	0.821244	0.802914	0.822571
	Std.dev.	0.362144	0.571246	0.355471	0.913514	0.215846	0.842756	0.589411	0.514781
F12	Best	0.984271	1.000000	1.000000	0.200214	0.947389	0.998427	1.000000	1.000000
	Mean	0.918141	1.000000	0.993245	0.000000	0.931795	0.973419	0.925524	0.932541
	Std.dev.	0.095181	0.999599	0.999454	0.658412	0.517942	0.332541	0.651844	0.689652
F13	Best	0.991248	0.879512	0.754120	0.254128	0.928741	0.998725	0.899854	0.900241
	Mean	0.934812	0.895371	0.912455	0.249573	0.900174	0.934817	0.829547	0.844581
	Std.dev.	0.812494	0.844712	0.665811	0.588841	0.847362	0.537469	0.421587	0.741545
F14	Best	0.955421	0.941274	1.000000	0.000000	0.891846	0.937184	0.889574	0.935748
	Mean	0.940228	0.910045	0.986214	0.000000	0.837426	0.917459	0.865741	0.898874
	Std.dev.	0.351282	0.845752	0.657143	0.354982	0.476581	0.843712	0.587412	0.458714
F15	Best	0.983279	0.981746	1.000000	0.465725	0.724891	1.000000	0.798247	0.925474
	Mean	0.963218	0.951273	1.000000	0.182414	0.710048	1.000000	0.754175	0.883625
	Std.dev.	0.658174	0.665417	0.384965	0.756182	0.665841	0.684127	0.325174	0.458712

F16	Best	1.000000	0.998912	1.000000	0.779954	0.951247	0.989271	0.951204	0.921451
	Mean	0.998567	0.990479	1.000000	0.671647	0.900145	0.951473	0.912365	0.896584
	Std.dev.	0.953147	0.900145	0.782459	0.458163	0.652184	0.798214	0.587456	0.365847
F17	Best	0.999751	0.987849	0.745281	0.664791	0.890216	0.962178	0.935887	0.955874
	Mean	0.971547	0.966847	0.315287	0.421788	0.881243	0.954712	0.900254	0.912581
	Std.dev.	0.698541	0.882845	0.942763	0.649782	0.251974	0.358941	0.298745	0.487511
F18	Best	0.910214	0.888891	1.000000	0.000000	0.854732	0.945271	0.865547	0.885471
	Mean	0.839537	0.8245115	0.884174	0.000000	0.792481	0.934812	0.823657	0.856584
	Std.dev.	0.251847	0.254761	0.631578	0.759124	0.221458	0.512798	0.401584	0.355814
F19	Best	0.998537	0.989412	0.996124	0.902458	0.988741	0.982154	0.945871	0.968745
	Mean	0.975222	0.953219	0.913942	0.821105	0.941278	0.921785	0.910547	0.912478
	Std.dev.	0.953155	0.554812	0.881451	0.658219	0.214952	0.679154	0.358741	0.568471
F20	Best	0.925612	0.902157	1.000000	0.000000	0.988167	1.000000	0.940124	0.912547
	Mean	0.89851	0.869512	0.991487	0.000000	0.931845	0.992745	0.892514	0.865471
	Std.dev.	0.558412	0.995421	0.526411	0.854122	0.479512	0.584312	0.325144	0.665874

Tables 4 and 5 present and compare results of our experiments on high and low dimensional benchmark functions. In these tables results have been reported in terms of Best (best results achieved by algorithms), Mean (mean of best results in 50 runs) and Standard Deviation. From the results we can see that in most of the experiments proposed algorithm achieved very good results. It outperformed other metaheuristic algorithms in nine (out of twenty) experiments and achieved very comparable results in the other experiments with the other algorithms.

7.3 Statistical Comparison

A useful statistical experiment to compare the efficiency of metaheuristic algorithms is the Non-parametric Wilcoxon Rank Sum Tests (Haynes, W., (2013)). We carry out this experiment with the results achieved by the proposed algorithm and also the seven mentioned metaheuristic algorithms on the 20 benchmark functions. The outcomes of this experiment are presented with *P-value* and *H* that help finding out whether there is a significant difference between the performance of FFA with another algorithm or not (see Table 6). H gets three values: 1^-, 0 and 1^+, and in which 1^- indicates that the performance of two algorithms is significantly different with 95% of confidence, *0* shows that there is no statistical difference, 1^+ indicates that FFA has a higher performance than the other one.

Table 6. Statistical comparison between FFA and other metaheuristic algorithms

ID	**SFA**		**KH**		**DE**		**PSO**		**APSO**		**BBO**		**ISA**	
	P-value	**H**	**P-value**	**H**	**P-value**	**H**	**P-value**	**H**	**P-value**	**H**	**P-value**	**H**	**P-value**	**H**
F1	5.0238E-005	1^-	0.0163	1^+	9.1795E-007	1^+	8.4294E-008	1^+	3.4064E-002	0	7.0211E-006	1^+	6.1240E-009	1^-
F2	0.0176	0	0.0843	0	4.8425E-007	1^+	6.3427E-003	1^+	5.4021E-006	1^+	3.2491E-003	1^+	4.2046E-004	1^+
F3	2.4570E-003	1^+	0.5649	0	5.2444E-007	1^+	6.0247E-004	1^+	0.0547	0	0.0547	1^+	2.0981E-004	1^+
F4	0.0943	1^+	0.1002	0	5.5239E-007	1^-	4.5297E-005	1^+	1.0842E-004	1^+	2.8406E-006	1^-	6.0028E-006	1^-
F5	4.5281E-007	1^+	5.5212E-006	1^+	6.0170E-006	1^+	5.4578E-007	1^+	9.6274E-005	1^+	6.0218E-008	1^+	7.9021E-014	1^+
F6	3.5616E-003	1^+	0.0623	1^-	6.2154E-005	1^+	8.2374E-008	1^+	3.1845E-005	1^+	5.9127E-011	0	4.9814-005	1^+
F7	4.5274E-003	1^+	5.1582E-007	1^+	2.1547E-016	1^+	1.8167E-008	1^+	0.0932	0	3.4219E-008	1^+	6.9024E-007	1^+

F8	5.2281E-006	1+	10.0223E-009	1-	5.2893E-006	1+	0.5412	1+	5.2198E-008	1-	8.6327E-007	1-	7.8920E-006	1-
F9	5.4213E-017	1+	7.5219E-006	1+	5.2878E-007	1+	0.4519	0	0.9420	0	9.2171E-004	1+	8.2174E-007	1+
F10	0.9534	1+	3.2152E-007	1+	0.1942	1+	2.1502E-004	1+	5.5713E-004	1-	3.1842E-009	1-	4.9124E-009	1-
F11	5.5113E-002	1+	0.0084	0	2.8984E-006	1+	0.1124	0	0.0041	0	4.1005E-006	1+	5.0028E-005	1+
F12	0.1962	0	0.0426	1+	6.4532E-008	1+	2.8164E-008	1+	3.1051E-009	1+	3.1207E-013	1+	5.9035E-005	0
F13	3.1845E-019	0	0.2381	1+	6.1706E-009	1+	4.0014E-012	1+	2.0023	1+	2.1812E-007	1+	6.9504E-005	1+
F14	5.2146E-011	0	0.1756	0	4.2829E-005	1+	5.8134E-009	1+	0.0025	0	6.1762E-004	1+	3.0150E-008	1+
F15	3.5129E-005	1+	5.3293E-008	1-	5.2739E-012	1-	3.3045E-011	1+	4.2442E-004	1-	3.2015E-009	1+	7.5716E-007	1-
F16	1.2646E-014	1+	0.2815	0	3.2385E-007	1+	8.2172E-011	1+	1.4076E-005	1+	1.4076E-008	1+	6.0016E-008	0
F17	9.1761E-008	1+	3.1569E-015	1+	5.1133E-012	1+	5.8519E-008	1+	1.2504E-006	1+	1.2810E-006	0	4.2984E-016	1+
F18	3.3281E-007	1+	0.1124	1+	5.0246E-005	1+	6.0293E-004	1+	0.0028	0	9.0028E-006	0	4.8001E-006	1+
F19	9.4576E-0014	1+	0.0024	0	4.1232E-003	1+	5.7154E-006	1+	5.6154E-007	1+	3.0014E-004	1+	9.2941E-004	1+
F20	5.5246E-008	1+	0.0020	1+	5.1018E-011	1+	5.5124E-007	1+	0.0127	0	2.1746E-003	0	7.8402E-005	1+

7.4 Analysis Results

From the results exposed in Tables 4 and 5, we can see that our fuzzy system significantly improves the performance of the firefly algorithm. This is due to the capability of the fuzzy firefly algorithm in recognizing different optimization situations and taking appropriate actions to handle them. The improved firefly algorithm employs a set of 25 fuzzy if-then rules. All the situations that may happen during the search are covered by these rules. For instance, when the value of parameter *Delta* is expected to change (at the beginning for example), the fuzzy system by considering the values of *Delta* and *Count* recognizes that the algorithm is being stuck in local optimum trap. So increasing the exploration amount (by changing its controlling parameters) enables the algorithm to escape from that trap. The reason for being stuck is that the algorithm cannot find better solutions. To escape from that trap, firefly must be enabled to take bigger steps. According to these tables, FFA outperforms SFA in all the experiments, and achieves very comparable results to the other metaheuristic algorithms.

To statically compare the results achieved by FFA and other metaheuristic algorithms, we perform the Wilcoxon Rank Sum test. Some analytic notions about this test are expressed below:

- F1 is *Ackley* function. The proposed method ranked third in this experiment. It has statistically significant performance over DE, KH, BBO and PSO algorithms. There is no statistically significant performance between FFA and APSO. FA and ISA have statistically significant performance over FFA algorithm.
- F3 is *Rosenbrok* function. The proposed algorithm achieved the second best result in this experiment in comparison with other metaheuristic algorithm participated in this experiment. The FFA has statistically significant performance over FA, DE, BBO, ISA and PSO algorithms. There is no statistically significant performance between FFA, KH and APSO algorithms.
- F5 is *Schwefel 2.26* function. FFA algorithm achieved the best result in this experiment. It has statistically significant performance over FA, KH, DE, PSO, BBO, ISA and APSO.
- F7 is *Schwefel 1.2* function. FFA algorithm ranked first in this experiment. It has statistically significant performance over FA, KH, DE, BBO, ISA and PSO. There is no statistically significant performance between FFA and APSO.

- F9 is *Rastrigin* function. FFA algorithm ranked first in this experiment. Statistically it has significant performance over FA, KH, BBO, ISA and DE. There is no significant performance between FFA, PSO and APSO.
- F11 is *Branin* function. FFA algorithm achieved the second best result in this experiment. Statistically it has significant performance over FA, BBO, ISA and DE. There is no statistically significant performance between FFA, KH, PSO and APSO. GSA has significant performance over FFA algorithm.
- F13 is *De Joung* function. FFA algorithm ranked second in this experiment. It has statistically significant performance over KH, DE, PSO, BBO, ISA and APSO. There is no statistically significant performance between FFA and FA.
- F15 is *Hartman 1* function. FFA achieved the second best result in this experiment. It has statistically significant performance over FA and PSO. KH, DE, BBO and APSO and ISA have significant performance over FFA.
- F17 is *Kowalik* function. FFA ranked first in this experiment. It has statistically significant performance over FA, KH, DE, PSO, ISA and APSO. There is no statistically significant performance between FFA and BBO.
- F19 is *Shekel 2* function. FFA algorithm achieved the best result in this experiment. Statistically it has significant performance over FA, DE, KH, BBO, ISA and PSO. There is no significant performance between FFA and APSO.

From Table 6, we can see that in most of the experiments, the results achieved by FFA are comparable with those of the seven metaheuristic algorithms. The high performance of FFA is mainly due to its dynamic behavior. Achieving the exploration-exploitation balance in run-time (on-line strategy) enables firefly to effectively move toward the best solution by avoiding local optimums.

To compare performance of FFA and SFA visually, convergence trends of them on four benchmark functions are shown in Figure 9. To facilitate the comparison, we have used *semilogy* function of MATLAB in plotting the convergence trends of algorithm. These plots confirm the higher performance of FFA to SFA in term of reaching the optimal solution. From the graphs, the convergence trend to the global optimum can be easily seen. This demonstrates that the proposed Fuzzy system has a great impact on the performance of firefly. By allocating proper values to the firefly parameters, the Fuzzy system keeps exploration-exploitation in balance. This will cause firefly not to be trapped in local optimum solutions while searching the problem space.

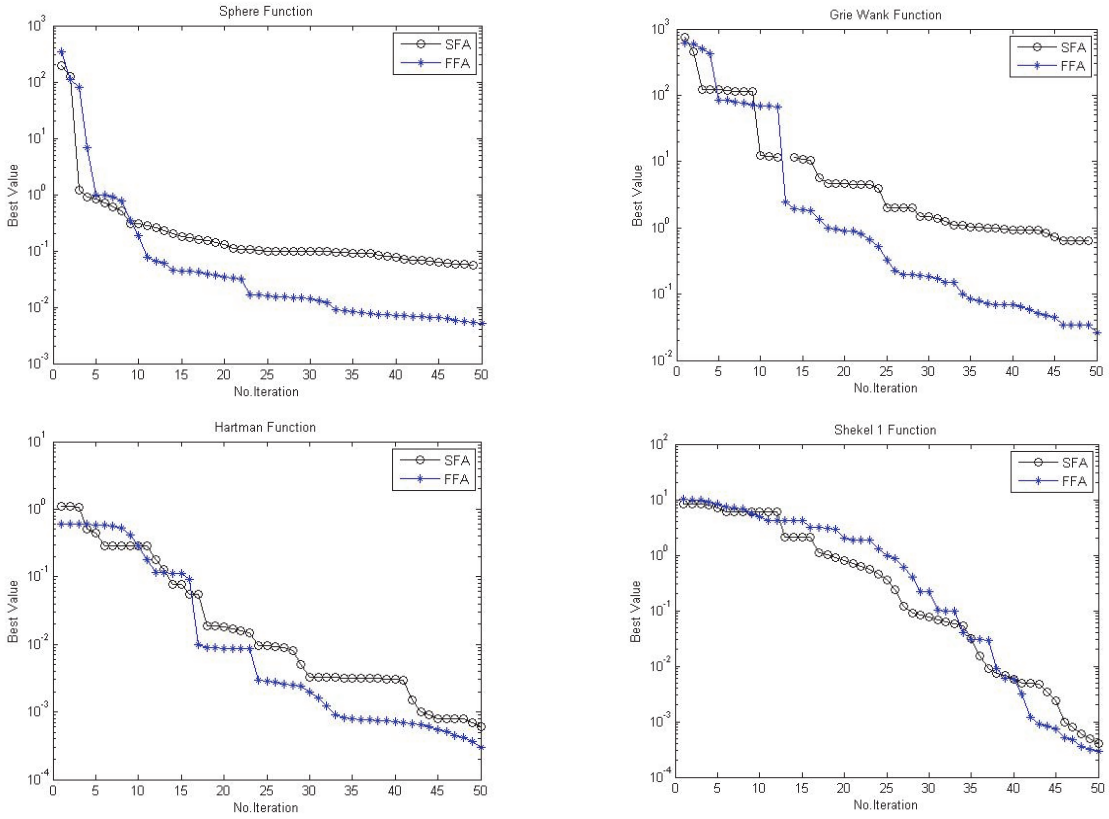

Figure 9. Convergence comparison between FFA and SFA on some benchmark functions.

8. Constrained Engineering Problems

Here, we assess the efficiency of our algorithm when applied to two constrained engineering problems, pressure vessel design and spring design, which have been utilized frequently to check the capability of different constrained optimization algorithms. Constrained optimization is the process of optimizing an objective function subject to some constraints imposed on the problem variables. The constraints divide the problem space into two subspaces: the feasible space where all the constraints are satisfied and the unfeasible space where at least one constraint is violated. The solutions of a constrained problem must be found in the feasible space. The parameters of the Fuzzy-based firefly algorithm are set in the same way as in the previous experiment.

8.1 Pressure Vessel Design

As presented in Figure 10, the pressure vessel consists of a cylinder topped with two hemispherical heads (Gandomi, A. H., & Alavi, A. H. (2012)). The objective here is to minimize the total cost by including the costs of the material, forming and welding. The involved variables of this optimization problem are four: thickness of the head (x_1), thickness of the shell (x_2), inner radius (x_3) and length of the cylinder not including the heads (x_4) (see figure 10). Table 7 exposes the constraints and cost function of this problem.

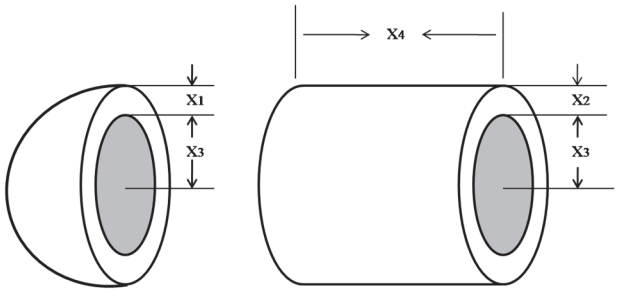

Figure 10. The pressure vessel design problem (Cagnina, Esquivel & Coello, 2008)

Table 7. Constraints and cost function of the pressure vessel design problem

Constraints	
$g_1(X) = -x_1 + 0.0193x_3 \le 0$	(14)
$g_2(X) = -x2 + 0.00954x_3 \le 0$	(15)
$g_3(X) = -\pi x_3^{\,2} x_4^{\,2} - \frac{4}{3}\pi x_3^{\,3} + 1296000 \le 0$	(16)
$g_4(X) = x_4 - 240 \le 0$	(17)
Cost Function	
$Cost(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^{\,2} + 3.1661x_1^{\,2}x_4 + 19.84x_1^{\,2}x_3$	(18)

The simple bounds of this problem are: $1\times0.0625 \le x_1$, $x_2 \ge 99\times0.0625$, $10.0 \le x_3$ and $x_4 \le 200.0$. In Table 8, we show the best solution X* found by FFA and its total cost. This solution was produced through a run of 120 000 evaluations by satisfying all constrains of the problem.

Table 8. Best solution of the pressure vessel problem found by FFA

Parameter	Best Solution
x_1	0.8131
x_2	0.4198
x_3	42.102
x_4	176.749
$g_1(X^*)$	0.00000
$g_2(X^*)$	-0.03622
$g_3(X^*)$	-1.742E+08
$g_4(X^*)$	-63.0872
Cost(X*)	6061.107

The statistical results obtained by FFA, SFA and 20 other optimization algorithms are presented in Table 9. As we can see in addition of being superior to SFA, FFA achieved comparable result in comparison with the other optimization also. This table shows that the best weight is 6059.714, obtained by the ISA algorithm. FFA produces the minimum weight of 6061.107, which is very close to the best result. We may note that FFA is better than SFA. However due to the random nature and stochastic behavior of the nature-inspired algorithm their real performance cannot be judged in a single run. So in this experiment we also include the mean and standard deviation values for each of the algorithms. In term of mean value the result achieved by FFA is much better that the result achieved by SFA and is superior to five out of fifteen optimization methods.

Table 9. Statistical results of the pressure vessel design problem obtained by different optimization algorithms (Gandomi, A. H. (2014))

Method	Mean	Best	Std.dev	No. Evaluation
TCA	7694.067	6390.554	357.000	80,000
GA-AP	8248.003	6065.821	515.000	80,000
GA-AIS	6845.496	6060.138	N/A	80,000
GA-AIS	7388.160	6059.854	124.000	80,000
GA-CP	6293.840	6288.745	7.4133	900,000
GA-SR	8012.615	6832.584	267.000	80,000
GA	6177.250	6059.946	130.930	80,000
PSO	9032.550	6544.270	995.570	100,000
CPSO	6147.133	6363.804	86.450	240,000
MOEA	6172.527	6059.926	124.000	80,000
BFO	6074.625	6060.460	15.600	48,000
DE	6085.230	6059.734	43.013	240,000
AATM	6061.988	6059.726	4.700	30,000
ISA	6410.087	6059.714	384.600	5,000
Above results have been taken from (Durkota, K. (2011))				
SFA	7321.526	6061.612	448.254	120,000
FFA	6991.271	6061.107	358.457	120,000

8.2 Tension/Compression Spring Design

The problem objective is to minimize the weight of a tension/compression spring that is subject to several constraints**,** like the minimum deflection, shear, surge frequency and limits on the outside diameter (see Figure 11) (Cagnina, L. C., Esquivel, S. C., & Coello, C. A. C. (2008)). The design variables of this problem are three: the wire diameter $(x)_1$, the mean coil diameter (x_2), and the number of active coils $(x)_3$. The constraints and cost function of the problem are shown in Table 10.

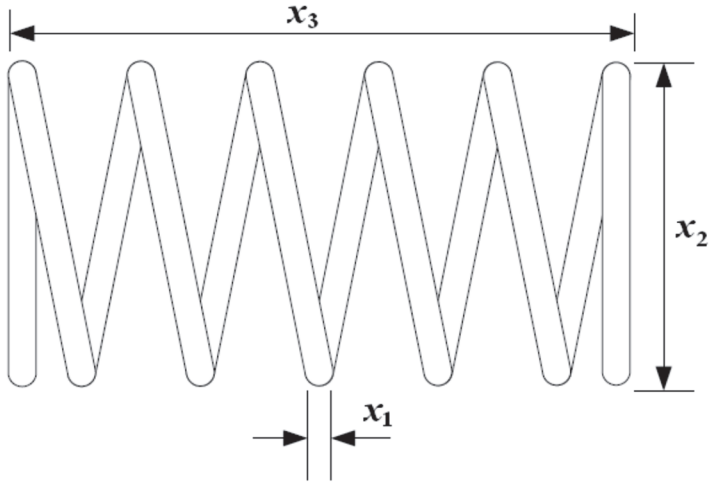

Figure 11. Tension/compression spring problem (Jamil & Yang, 2013)

Table 10. Constraints and cost function of spring design problem

Constraints	
$g_1(X) = 1 - \frac{x_2^3 x_3}{7178 x_1^4} \le 0$	(19)
$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3) - x_1^4} + \frac{1}{5108 x_1^2} - 1 \le 0$	(20)
$g_3(X) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$	(21)
$g_4(X) = \frac{x_2 + x_1}{1.5} - 1 \le 0$	(22)
Cost Function	
$\mathrm{Cost}(X) = (x_3 + 2) x_2 x_1^2$	(23)

The simple bounds of this problem are: $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$ and $2.0 \le x_3 \le 15.0$. Table 11 shows the best solution found through a run of 90 000 evaluations by respecting all the elicited constraints.

Table 11. Best solution of the tension/compression spring problem found by FFA

Parameter	Best Solution
x_1	0.05148
x_2	0.35167
x_3	11.6322
$g_1(X^*)$	-0.00209
$g_2(X^*)$	-0.00012
$g_3(X^*)$	-4.03013
$g_4(X^*)$	-4.02591
Cost(X^*)	0.012704

Table 12 compares the statistical results obtained by FFA, SFA and the 20 metaheuristic algorithms. This table shows that the best weight is 0.012665, which is obtained by TCA, SA-DS and ISA algorithms. FFA is superior to SFA and very comparable to the other algorithms. Indeed, FFA produces the minimum weight of 0.012704, which is extremely close to the best result. We may note that FFA is better than SFA. However due to the random nature and stochastic behavior of the nature-inspired algorithm their real performance cannot be judged in a single run. So in this experiment we also include the mean and standard deviation values for each of the algorithms. In term of mean value the result achieved by FFA is much better that the result achieved by SFA and is superior to seven out of fifteen optimization methods.

Table 12. Statistical results of the tension/compression spring design problem obtained by different optimization algorithm (Gandomi, A. H. (2014))

Method	Mean	Best	Std.dev	No. Evaluation
TCA	0.012732	0.012665	9.40E-05	36,000
GA-AP	0.014022	0.012679	1.47E-03	36,000
GA-SR	0.013993	0.012680	1.27E-03	36,000
GA-CP	0.012769	0.012705	3.94E-05	900,000
GA-AIS	0.013880	0.012666	N/A	36,000
GA	0.012742	0.012681	5.90E-05	30,000
PSO	0.022948	0.013120	7.21E-03	100,000
SA-DS	0.012665	0.012665	N/A	49,531
HS	N/A	0.012671	N/A	50,000
MOEA	0.012960	0.012680	3.63E-04	80,000
BFO	0.012759	0.012671	1.36E-04	48,000
SCA	0.012923	0.012669	5.92E-04	25,167
AATM	0.012708	0.012668	4.50E-05	25,000
ISA	0.013165	0.012665	1.59E-02	8,000
Above results have been taken from (Durkota, K. (2011))				
SFA	0.013842	0.012730	2.54.E-04	90,000
FFA	0.012948	0.012704	3.87E-03	90,000

9. Conclusion

The goal of this study is to tune dynamically the firefly parameters in order to achieve appropriate rates for the exploration and exploitation. To this end, we have employed Fuzzy logic and devised a Fuzzy system as a parameter controller for the firefly algorithm. In each of the optimization steps, the Fuzzy system adjusts efficiently the values of the firefly parameters based on a set of Fuzzy rules. Attaining a good balance between exploration and exploitation causes the firefly algorithm to move effectively toward the optimal solution and to avoid local optimum traps as demonstrated in this paper. To assess the efficiency of the new algorithm in terms of the solution optimality, we have conducted extensive experiments. First we have applied our Fuzzy-based firefly, the standard firefly and four other famous metaheuristic algorithms **(**Differential Evolution, Particle Swarm Optimization (PSO), Accelerated PSO and Krill Herd), Biography-Base Optimization (BBO) and Interior Search Algorithm (ISA) to ten low and ten high dimensional optimization problems. Second, we have compared the performance results of these six optimization algorithms. The experimental results proved the superiority of our proposed Fuzzy-based firefly algorithm to standard firefly. In all the experiments on standard benchmark functions, the Fuzzy firefly algorithm performed better than the standard firefly and achieved the best results in nine out of twenty benchmark functions. In the rest of the experiments, it achieved very comparable results to the other optimization algorithms. We also conducted a statistical experiment by using the Non-parametric Wilcoxon Rank Sum Tests to statistically investigate the performance of the proposed method compared to the other optimization methods. Furthermore, we have applied the improved firefly and standard firefly to two constrained engineering problems, and have demonstrated that our method is much better than standard firefly and very comparable to the best optimization algorithms in the literature.

The design of Fuzzy systems, particularly the underlying rule bases and membership functions, is based on human knowledge. Thus, we would like to enhance our Fuzzy system to improve the performance of firefly for instance by deploying more practical Fuzzy membership functions and also a better Fuzzy rule base that will cover more situations that firefly may encounter when searching for the optimal solution. Another future work would be assessing different combinations of the Fuzzy components (i.e. different fuzzifiers, defuzzifiers and inference engines) in order to find out the best arrangement that will lead to the most efficient firefly algorithm.

References

Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of global optimization, 31*(4), 635-672. https://doi.org/10.1007/s10898-004-9972-2

Bidar, M., & Kanan, H. R. (2013, August). Modified firefly algorithm using fuzzy tuned parameters. In Fuzzy Systems (IFSC), 2013 13th Iranian Conference on (pp. 1-4). IEEE. https://doi.org/10.1109/IFSC.2013.6675634

Cagnina, L. C., Esquivel, S. C., & Coello, C. A. C. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica, 32*(3).

Črepinšek, M., Liu, S. H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: a survey. *ACM Computing Surveys (CSUR), 45*(3), 35. https://doi.org/10.1145/2480741.2480752

dos Santos Coelho, L., de Andrade Bernert, D. L., & Mariani, V. C. (2011, June). A chaotic firefly algorithm applied to reliability-redundancy optimization. In Evolutionary Computation (CEC), 2011 IEEE Congress on (pp. 517-521). Ieee. https://doi.org/10.1109/CEC.2011.5949662

Durkota, K. (2011). Implementation of a discrete firefly algorithm for the QAP problem within the sage framework. Bachelor Thesis, Czech Technical University.

Farahani, S. M., Abshouri, A. A., Nasiri, B., & Meybodi, M. (2011). A Gaussian firefly algorithm. *International Journal of Machine Learning and Computing, 1*(5), 448. https://doi.org/10.7763/IJMLC.2011.V1.67

Fister, I., Yang, X. S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation, 13*, 34-46. https://doi.org/10.1016/j.swevo.2013.06.001

Gandomi, A. H. (2014). Interior search algorithm (ISA): A novel approach for global optimization. *ISA transactions, 53*(4), 1168-1183. https://doi.org/10.1016/j.isatra.2014.03.018

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*(12), 4831-4845. https://doi.org/10.1016/j.cnsns.2012.05.010

Hassanzadeh, T., & Kanan, H. R. (2014). Fuzzy FA: A modified firefly algorithm. *Applied Artificial Intelligence, 28*(1), 47-65. https://doi.org/10.1080/08839514.2014.862773

Haynes, W., (2013). Wilcoxon rank sum test. In Encyclopedia of Systems Biology (pp. 2354-2355). Springer New York. https://doi.org/10.1007/978-1-4419-9863-7_1185

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150-194. https://doi.org/10.1504/IJMMNO.2013.055204

Liu, H., Xu, Z., & Abraham, A. (2005, September). Hybrid fuzzy-genetic algorithm approach for crew grouping. In Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on (pp. 332-337). IEEE.

Liu, Y., & Ma, L. (2011, May). Solving TSP by fuzzy particle swarm algorithm. In Business Management and Electronic Information (BMEI), 2011 International Conference on (Vol. 5, pp. 202-204). IEEE.

Qi, Z., & Chunming, P. (2010, August). An improved fuzzy genetic algorithm with fuzzy adjusted crossover and mutation probabilities. In Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on (Vol. 4, pp. V4-581). IEEE. https://doi.org/10.1109/ICACTE.2010.5579287

Shi, Y., & Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. In Evolutionary Computation, 2001. Proceedings of the 2001 Congress on (Vol. 1, pp. 101-106). IEEE, https://doi.org/10.1109/CEC.2001.934377

Wang, L. X. (1999). A course in fuzzy systems (pp. 258-265). Prentice-Hall press, USA.

Yang, X. S. (2010). Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons, https://doi.org/10.1002/9780470640425

Yang, X. S. (2010). Firefly algorithm, Levy flights and global optimization. Research and development in intelligent systems XXVI, 209-218. https://doi.org/10.1007/978-1-84882-983-1_15

Yang, X. S. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation, 2*(2), 78-84, https://doi.org/10.1504/IJBIC.2010.032124

Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.

Yang, X. S., & Deb, S. (2009, December). Cuckoo search via Lévy flights. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on (pp. 210-214). IEEE.