The CSI Journal on Computer Science & Engineering

The CSI Journal on
Computer Science and Engineering
Vol. 15, No. 2, 2018
Pages 18-30
Regular Paper

The Construction of Fuzzy Classification Systems Using the Shuffled Frog Leaping Algorithm

Seyyed Mohsen Mirhosseini[1] Hassan Haghighi[1] Jamshid Bahrami[1]

[1] Faculty of Computer Science and Engineering, Shahid Beheshti University G. C., Tehran, Iran

Abstract

Various methods have been proposed for constructing and optimizing fuzzy inference systems. This paper first proposes a new method to create zero-order Sugeno fuzzy inference systems using the Shuffled Frog Leaping Algorithm (SFLA). As the second contribution, the paper introduces four improvements over SFLA. The resulting version of SFLA, called ISFLA (Improved SFLA), is also applied to create zero-order Sugeno fuzzy inference systems. We conducted experiments to assess ISFLA and compare it with the original SFLA and three well-known evolutionary algorithms over five standard classification data sets from the UCI machine learning repository. The experimental results show that ISFLA creates fuzzy systems more efficiently than the standard SFLA and some other evolutionary algorithms, i.e., GA, ACO and PSO. Moreover, with respect to the accuracy and the convergence speed criteria, ISFLA and PSO outperform other evolutionary algorithms, while their performance is comparable to each other.

Keywords: Metaheuristic, shuffled frog leaping algorithm, fuzzy inference system, memetic algorithm, classification.

1. Introduction

In classification problems, a set of samples with specific features should be assigned to different classes. The diversity of problems that can be addressed by classification algorithms is significant. Customer Target Marketing, Medical Disease Diagnosis, Supervised Event Detection, Multimedia Data Analysis, Biological Data Analysis, Document Categorization and Filtering and Social Network Analysis are some common engineering application domains in which classification problems arise.

Fuzzy Inference Systems (FISs) are widely used to solve classification problems. These systems are well known methods within soft computing, based on fuzzy concepts to address complex real-world problems. FISs have been deployed in a number of engineering and science areas, e.g., bioinformatics [1], data mining [2], control engineering [3,4], finance [5], robotics [6], and pattern recognition [7].

There has been much research on the creation and optimization of fuzzy inference systems. Various methods have been proposed that use metaheuristic algorithms, including Genetic Algorithm (GA) [8-11], Particle Swarm Intelligence (PSO) [12-13], Ant Colony Optimization (ACO) [14], and differential evolution [15], to extract and optimize fuzzy rules. Metaheuristic algorithms, which are often inspired by nature, use heuristics to find a global or near-global optimal solution.

Memetic Algorithms (MAs) represent one of the recent growing areas of research in metaheuristic algorithms. The term MA is now widely used as a synergy of a global search method with a local search method [16]. SFLA [17-18] is a memetic metaheuristic in which a set of frogs (or initial solutions) cooperate to find the largest source of food. The frogs act as hosts or carriers of memes. At first, the frogs are randomly distributed over the search space. Then, the population is partitioned into smaller communities called memeplexes. A random subset, called a submemeplex, is selected from each memeplex. In each submemeplex, the worst frog tries to get to the location that has the maximum amount of available food by leaping toward the best frog. To ensure the global exploration, the frogs are periodically shuffled and reorganized into new memeplexes by a technique similar to that used in the shuffled complex evolution

algorithm [19]. Throughout the paper, the frog that has the best fitness value is called the best frog and the frog that has the worst fitness value called the worst frog in short.

By combining features from the genetic-based memetic algorithm and PSO, and by using local and global searches simultaneously, SFLA is effective for different sorts of optimization problems. This algorithm has been successfully used in various problem domains such as the water distribution network design [20], parameter identification [21], unit commitment [22], classification [23], robot optimal controller design [24], project management [25], job shop scheduling [26-27], hybrid flow shop scheduling [28] and so on. In [29], different application types of SFLA were presented. Authors in this paper show that SFLA has been mostly employed in solving the flexible job/flow shop scheduling problems, electrical power flow optimization problems and classification problems.

Some researches were conducted in order to compare SFLA with other evolutionary algorithms. For example, in [30], the results of comparing SFLA with GA, PSO, and ACO reveal that SFLA has similar performance to PSO, while it outperforms other mentioned algorithms with respect to the success rate, solution quality and processing time criteria. Based on the results of [30], the most distinguished advantage of SFLA lies in its fast convergence speed. Sarkheyli et al. [29] investigated the previous research to apply SFLA for solving optimization problems in various applications. This investigation demonstrated the advantages of SFLA over other evolutionary algorithms, such as GA, PSO, ACO, Simulated Annealing (SA), and Artificial Bee Colony.

Motivated by these experiences and findings (specially, the successful application of SFLA on classification problems), and considering the fact that the effectiveness of SFLA for optimizing fuzzy systems has not been examined yet, this paper investigates the application of SFLA to construct and optimize fuzzy inference systems. Since Sugeno fuzzy inference systems are the most widely used FISs [31], we examine the ability of SFLA to optimize fuzzy rules of a Sugeno fuzzy inference system. In these systems, input values are provided in the form of “if-then” rules, and outputs are combined using the weighted average to determine the class of a given sample.

The local search in SFLA involves the creation of memeplexes and evolution of memes/frogs within them, while the global search involves combining the evolved memeplexes. Therefore, methods employed in creating memeplexes can play an important role in the quality of the final solution. Moreover, the way the worst frog leaps toward the best frog for memes evolution can affect the exploration and exploitation capabilities of the algorithm, and consequently can affect the final solution. At last, the success level and the convergence speed of SFLA largely depend on the initial population used in the algorithm.

Regarding the above facts, as the second contribution, this paper introduces an improved version of SFLA. For this purpose, we propose a new method for creating memeplexes, a new method for memes evolution (based on a new form of frog leap), and a new method for generating the initial population. Our method to generate the initial population preserves randomness of the population individuals and improves their quality.

The new SFLA based algorithm, called ISFLA (Improved SFLA), is applied to create FISs. Our experimental results indicate that ISFLA creates fuzzy systems more efficiently than the standard SFLA and some other evolutionary algorithms, i.e., GA, ACO and PSO. In addition, with respect to the accuracy and convergence speed criteria, ISFLA and PSO outperform other evolutionary algorithms, while their performance is comparable to each other.

The rest of this paper is organized as follows. Section 2 provides an overview on the concept of SFLA, and fuzzy inference systems. Section 3 explains our method of encoding fuzzy systems in SFLA. Section 4 presents the proposed improvements over SFLA. Section 5 demonstrates the experiments and their results. Finally, we present our conclusions and outline the future lines of research in Section 6.

2. Preliminaries

In this section, two prerequisite concepts, i.e., SFLA and fuzzy inference systems, are reviewed.

2.1. Shuffled Frog Leaping Algorithm

SFLA is a memetic metaheuristic algorithm designed to seek the global optimum in complex problems [17]. SFLA is based on the evolution of memes carried by individuals (through a local search) and a global exchange of information among the population. It involves a population of possible solutions defined by a set of frogs partitioned into subsets called memeplexes. Each memeplex performs a local search. The goal of the frogs is to improve their meme and move toward the global optimum. Like any other evolutionary algorithm, SFLA starts with a random initial population, and evolution takes places after several iterations.

An overview of different stages of this algorithm is shown in Fig. 1. The first stage involves setting the initial parameters of SFLA, including the number of iterations (*nIteration*), the number of memeplexes (*m*), the number of frogs in each memeplex or memeplex size (*n*), the number of frogs in each submemeplex or submemeplex size (*q*), and the number of evolutions in each memeplex (*N*). In the second stage, an initial population of frogs is generated in the search space. For random generation of each frog within the population, the upper and lower bounds are set for each memotype of the frog's meme; then, random numbers between 0 and 1 are generated and scaled to be within these two bounds. In SFLA, the population is seen as hosts of memes, i.e. a memetic vector. Each host carries a single meme. The third stage involves evaluating the fitness of frogs using a fitness function. This function takes the parameters of the problem as a set of memes and determines the fitness of each solution. Since this paper applies SFLA for creating a fuzzy inference system, the accuracy of the system is taken as the fitness. Thus, the fitness function takes the parameters of the fuzzy system and returns its accuracy as the fitness value.

Stages 4 to 7 will be repeated until the convergence condition of the algorithm is satisfied. In the fourth stage, the frogs are sorted in descending order according to their fitness. In the fifth stage, the frogs are divided into *m* memeplexes. The *ith* frog goes to the (*i mod m*)*th* memeplex, such that all the frogs

are distributed in all the memeplexes [17]. The sixth stage is memetic evolution within each memeplex. In other words, it corresponds to the local search where the frogs evolve within each memeplex. This stage is described in more detail in the next paragraph. The seventh stage involves shuffling the memeplexes. At this point, the stopping condition based on the convergence criterion is examined. If this condition is satisfied, then the best frog found so far is selected as the final solution. Otherwise, the algorithm is repeated from stage 4 until this condition is satisfied.

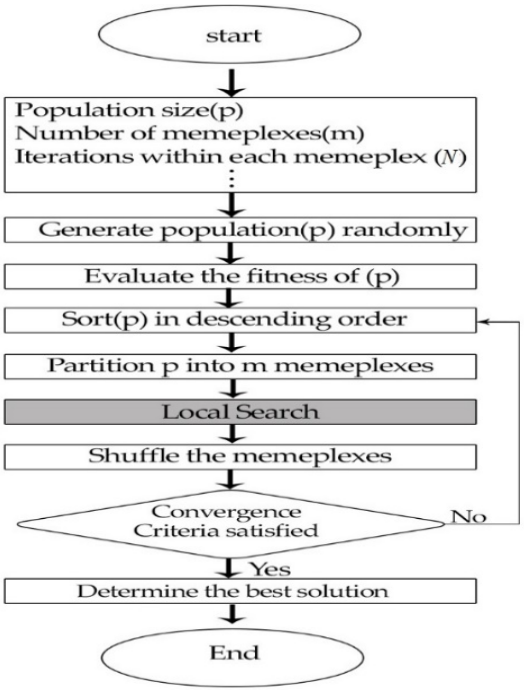

Fig. 1. The SFLA flowchart [17]

The local search in stage 6 is applied independently to each memeplex. First, an arbitrary memeplex is selected and a submemeplex is created in the selected memeplex. As mentioned before, the number of frogs in each memeplex (q) is set in the first stage. To create submemeplexes, all the frogs in a memeplex are first sorted in descending order according to their fitness value. Then, based on their rank in the ordered list, their chance of selection (for presence in the submemeplex) is determined using a triangular probability distribution. The probability of selecting the jth frog in the sorted memeplex with n frogs is calculated from Eq. (1):

$$p_j = \frac{2(n+1-j)}{n(n+1)} \qquad (1)$$

Regarding the selection probability calculated for each frog, q number of frogs with the highest fitness values are selected to form a submemeplex. Within each submemeplex, the position of the worst frog (shown as X_w) is adjusted toward the best frog (shown as X_b) using Eq. (2):

$$X_w^{(new)} = X_w^{(old)} + rand \times \left(X_b^{(old)} - X_w^{(old)}\right) \qquad (2)$$

where *rand* is a random number between 0 and 1, $X_w^{(old)}$ is the old position of the worst frog, and $X_w^{(new)}$ is its new position. If this procedure does not produce a better solution, the calculation in Eq. (2) is repeated with respect to the global best frog in the entire population. In other words, X_g replaces X_b, where X_g represents the position of the global best frog. If there is still no improvement, a new frog (solution) is randomly generated to replace the worst frog. After upgrading the worst frog of a submemeplex, all the frogs are returned to the memeplex. The process of allocating weights (i.e., probabilities of selecting each frog in the memeplex), creating submemeplexes, and upgrading the worst frog are repeated *it* times. After finishing the local search for the first selected memeplex, a similar process is done for the other memeplexes.

2.2. Fuzzy Inference System

Fuzzy if-then rules are very close to human reasoning. Using these rules and approximate inference algorithms, fuzzy inference systems provide a powerful computational framework that can even work with ambiguous or incomplete data. These systems are widely applied in different fields such as data classification, automatic control, expert systems, decision-making, robotics, time series analysis, pattern classification, process planning, and system identification [1,4,32]. A fuzzy inference system consists of three principal components: (1) a rule base, comprising of the selected fuzzy rules, (2) a database, maintaining the membership functions of the fuzzy rules, and (3) a reasoning mechanism, performing a fuzzy inference procedure upon the rules to derive a reasonable output or conclusion [33].

Each fuzzy rule consists of antecedent and consequent parts. The antecedent part usually contains a combination of fuzzy statements, while the consequent part can have different forms. With respect to the form of the consequent part, FISs can be divided into three categories: Mamdani, Sugeno, and Tsukamoto fuzzy inference systems. Due to the difference in the consequent part of these systems, each system has its own unique inference method [33]. The Tsukamoto fuzzy model is not often used since it is not as transparent as either the Mamdani or Sugeno fuzzy models. The most fundamental difference between Mamdani-type FISs and Sugeno-type FISs is the way the crisp output is generated from the fuzzy inputs. The expressive power and interpretability of the Mamdani output are lost in the Sugeno FIS since the consequents of the rules are not fuzzy [34]. It can be considered as a disadvantage of Sugeno-type FISs in comparison with Mamdani-type FISs. However, there are some advantages for Sugeno-type inference systems as follows:

- Mamdani-type FISs use the time consuming technique of defuzzification of a fuzzy output, while Sugeno-type FISs use the weighted average to compute the crisp output. Thus, Sugeno-type FISs have less processing time.
- Due to the dependency on defuzzification, Mamdani-type FISs are less flexible in system design in comparison with Sugeno-type FISs.
- The Sugeno method is computationally efficient and works well with optimization and adaptive techniques. This makes it very attractive for control problems, particularly for dynamic nonlinear systems. The adaptive techniques can be used to customize the membership functions so that the fuzzy system can model the data in the best way [31].

According to the mentioned strengths, and since Sugeno fuzzy inference systems are the most widely used FISs [31], we concentrate on these systems in this paper.

A typical fuzzy rule in a Sugeno fuzzy model has the following form:

$$\boldsymbol{If\ (x\ is\ A\ and\ y\ is\ B)\ then\ (z = f(x,y))}$$

In Sugeno systems, the antecedent includes a set of fuzzy statements that are usually connected using the AND operator. $\boldsymbol{z = f(x,y)}$ is a crisp function in the consequent. Usually, $\boldsymbol{f(x,y)}$ is a polynomial over the input variables x and y. In an n-order Sugeno fuzzy system, $\boldsymbol{f(x,y)}$ is an n-order polynomial function. When $\boldsymbol{f(x,y)}$ is a constant, we have a zero-order Sugeno fuzzy system as follows:

$$\boldsymbol{If\ (x\ is\ A\ and\ y\ is\ B) then\ (z = c)}$$

where $\boldsymbol{x\ is\ A}$ and $\boldsymbol{y\ is\ B}$ are fuzzy statements, $\boldsymbol{A}$ and $\boldsymbol{B}$ are linguistic variables that are determined by a fuzzy membership function, $\boldsymbol{c}$ is a constant, and $\boldsymbol{z}$ is the output of the fuzzy system.

The fuzzy inference is done in two stages. Suppose that the rule base of the Sugeno fuzzy system with inputs $\boldsymbol{x}$ and $\boldsymbol{y}$ and output $\boldsymbol{z}$ has the following rules:

$$\boldsymbol{If\ x\ is\ A_1\ and\ y\ is\ B_1\ then\ z = c_1}$$
$$\boldsymbol{If\ x\ is\ A_2\ and\ y\ is\ B_2\ then\ z = c_2,}$$

where $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ are two membership functions defined for the input variable $\boldsymbol{x}$; $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ are two membership functions defined for the input variable $\boldsymbol{y}$; $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are two real values; and $\boldsymbol{z}$ is the output. In the first inference stage, the output of each rule (i.e., the weights) is calculated according to the input values, membership functions, and antecedent part of the fuzzy rules. Fig. 2 illustrates the first stage of the inference for producing weights $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$. As seen, first the membership degree of the input values in membership functions used in the fuzzy statements of the antecedent part is calculated. Then, these membership degrees are combined using a T-norm operator (e.g., Min or Product operators) to produce weights $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$ for the first and second rules, respectively.

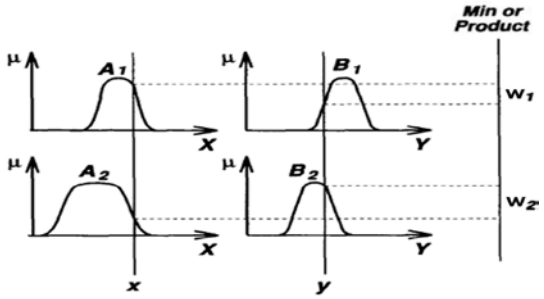

Fig. 2. The weight calculation in Sugeno fuzzy systems

In the second stage of inference, given the weights produced in the first stage, the constant values $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$, and the consequent part of the rules, the final output of the fuzzy system is calculated using the mean weight according to Eq. (3):

$$z = \frac{w_1 c_1 + w_2 c_2}{w_1 + w_2} \qquad (3)$$

Methods used to create fuzzy systems take membership functions and a set of data from functions' behavior as input and returns fuzzy if-then rules as output. The complexity of the fuzzy system and the computation needed to produce the output tend to increase the number of membership functions and if-then rules. However, this can also increase the accuracy of the fuzzy system. Based on the Occam's razor principle [35], the best model is the one with the highest accuracy and the least complexity. Thus, methods used to create fuzzy systems first make a simple model, and then optimize it to achieve higher accuracy with less complexity. Different optimization techniques are used for this purpose, including GA [8-11], PSO [12,13], differential evolution [15], propagation algorithm [36], extended Kalman filter algorithm [37], and the weighted fuzzy rules generation method [33]. In this paper, we improve SFLA, and then apply it to create a Sugeno fuzzy system.

3. Encoding Fuzzy Systems in SFLA

This section explains encoding a zero-order Sugeno fuzzy system in SFLA. Each fuzzy system is modeled as a meme of frog, and SFLA tries to maximize the system's accuracy by adjusting the input parameters. Each fuzzy system contains a set of inputs, a set of fuzzy if-then rules, and an output. A FIS generation algorithm determines fuzzy if-then rules. It takes the training dataset as input.

The data is stored as tables in the training dataset. Each dataset consists of rows and columns of data. Each row represents a training sample, and each column represents an attribute that describes a characteristic of the studied space. The last column of each table represents the output. This column provides the class membership for each point of the search space described by other attribute values. The FIS generation algorithm determines its parameters in such a way that enables it to accurately specify the class of each point from a limited number of training samples (each sample specifies one point in the search space and its class). In other words, a good FIS can accurately describe the points of a search space using the training dataset.

Increasing the number of parameters and over-training the model can result in overfitting to the training data. In this way, the model is very accurate for the training data but less accurate for the test data. In this research, the number of learned rules is reduced and considered as a constant value to have low complexity while trying to achieve high accuracy. Then, the optimal number of parameters of the FIS is determined using SFLA in a continuous space.

The type of all membership functions defined on the variables is Gaussian. These functions have two adjustable parameters: (1) center of the Gaussian function (C), and (2) width of the Gaussian function (σ). SFLA determines the center and width of each membership function defined on the interval of an input variable.

If the fuzzy system has two input variables, each if-then rule in the zero-order fuzzy system will be expressed as:

$$\boldsymbol{If\ x\ is\ A\ and\ y\ is\ B\ then\ z = r}$$

Here $\boldsymbol{x}$ and $\boldsymbol{y}$ are input variables; $\boldsymbol{z}$ is the output of the fuzzy system; $\boldsymbol{A}$ and $\boldsymbol{B}$ are membership functions defined on $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively; and $\boldsymbol{r}$ is a constant. In the encoded fuzzy system, five Gaussian membership functions are defined on each variable. Let mf_i^j represent the *j*th membership function

used for the *i*th input. The five rules for optimizing the fuzzy system are as follows:

$$\textbf{if } (x_1 \textbf{is } mf_1^1) \textbf{ and } (x_2 \textbf{ is } mf_2^1) \textbf{ and } \dots \textbf{and } (x_n \textbf{ is } mf_n^1) \textbf{ then } z_1$$
$$\textbf{if } (x_1 \textbf{is } mf_1^2) \textbf{ and } (x_2 \textbf{ is } mf_2^2) \textbf{ and } \dots \textbf{and } (x_n \textbf{ is } mf_n^2) \textbf{ then } z_2$$
$$\textbf{if } (x_1 \textbf{is } mf_1^3) \textbf{ and } (x_2 \textbf{ is } mf_2^3) \textbf{ and } \dots \textbf{and } (x_n \textbf{ is } mf_n^3) \textbf{ then } z_3$$
$$\textbf{if } (x_1 \textbf{is } mf_1^4) \textbf{ and } (x_2 \textbf{ is } mf_2^4) \textbf{ and } \dots \textbf{and } (x_n \textbf{ is } mf_n^4) \textbf{ then } z_4$$
$$\textbf{if } (x_1 \textbf{is } mf_1^5) \textbf{ and } (x_2 \textbf{ is } mf_2^5) \textbf{ and } \dots \textbf{and } (x_n \textbf{ is } mf_n^5) \textbf{ then } z_5$$

where z_1 to z_5 are real numbers, *n* denotes the total number of attributes, and x_i denotes the value of the *i*th attribute. By keeping at a constant value the number of rules and membership functions type, the center and width of the Gaussian membership functions as well as the values of z_1 to z_5 are optimized. In other words, SFLA determines the center (C) and width (σ) of Gaussian membership functions as well as the constant values $(z_1 - z_5)$ in the search process. If the total number of attributes is *n*, then there are $(n \times 5 \times 2) + 5$ parameters that should be optimized using our algorithm. The reason is that we have 5 constant values $(z_1 - z_5)$ and $n \times 5$ membership functions each of which has two variable parameters (center and width).

Fig. 3 shows the meme structure of a frog, i.e., the encoded fuzzy system. $c_{mf_i^j}$ and $\sigma_{mf_i^j}$ show the center and width of the Gaussian membership function mf_i^j, respectively. Each frog carries one meme. Each meme consists of $(n \times 5 \times 2) + 5$ memotypes, and each memotype corresponds to a parameter of the fuzzy system.

$c_{mf_1^1}$	$\sigma_{mf_1^1}$	$c_{mf_1^2}$	$\sigma_{mf_1^2}$	…	$c_{mf_1^5}$	$\sigma_{mf_1^5}$	…	$c_{mf_n^5}$	$\sigma_{mf_n^5}$	z_1	z_2	z_3	z_4	z_5

Fig. 3. The encoded fuzzy system as memes in SFLA.

To determine the upper and lower bounds of the center of membership functions, the interval of input variables is divided into 5 equal parts as illustrated in Fig. 4. Let *Ci* denote the *i*-th part in an interval, and Interval_Width indicate the length of each part. The center of the first membership function can only be changed in the first part (C_1); the center of the second membership function can only be changed in the second part (C_2), and so on.

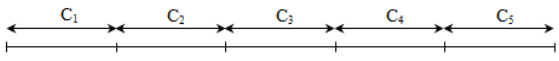

Fig. 4. The interval of the center of membership functions defined on a variable.

In our implementation, the width of a Gaussian membership function can vary within [Interval_Width/10, Interval_Width]. Fig. 5 illustrates a sample in which the minimum width of membership functions is 10 and the center of the Gaussian functions is located on 5. Therefore, by changing the value of σ, we can create different membership functions as shown in Fig. 5.

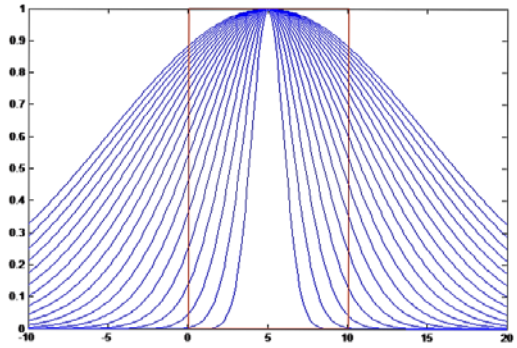

Fig. 5. Different membership functions created by changing σ and keeping C constant.

4. Improvement of SFLA

In this section, we present four modifications to the standard SFLA. The experimental results in Section 5 demonstrate that these modifications improve the accuracy of the standard SFLA while decreasing the execution time.

4.1. Removing the Process of Submemeplex Creation

For performing the local search in the standard SFLA, a set of frogs is randomly selected with a triangular probability distribution. During the local exploration, the worst frog leaps toward the best frog in a submemeplex. If no improvement becomes possible in this case, the worst frog leaps toward the global optimum, and if this procedure fails, the frog is replaced with a randomly generated one. This process is repeated for a user-defined number of times (i.e., *it* times). The first problem with this approach is that the frogs in a memeplex are selected for creating a submemeplex based on the triangular probability distribution. As Eq. (1) shows, the worst frogs have a small chance of being present in the submemeplex (and being evolved), while they require evolution more than other frogs. In other words, by evolving just in a submemeplex, a frog that is already good, leaps toward a better frog, and thus, the frog's quality may not change significantly.

To resolve the above problem, our proposed algorithm does not include the process of creating submemeplexes at all. In the new algorithm, first, the worst frog leaps toward the best frog in the memeplex itself. If no improvement is seen in its fitness, the worst frog leaps toward the global optimum. If there is still no improvement, a new frog is randomly generated to replace the worst frog. This process is repeated *it* times.

4.2. New Method of Memeplex Creation

In the standard SFLA, memeplexes are not balanced with respect to the quality of their frogs. That is, the average quality of frogs in the first memeplex is better than that of other memeplexes. Thus, the last memeplex has the least quality. This happens more often when there is a significant difference in the frogs' fitness.

To overcome this problem, our proposed algorithm probabilistically creates memeplexes. To create *m*

memeplexes with *n* frogs in each memeplex, the frogs are first sorted in descending order based on their fitness. Then, it should be decided to which memeplexes the best and worst frogs are assigned. Initially, the first frog goes to the first memeplex, the second frog goes to the second memeplex, and frog *m* goes to the *m*th memeplex. Before the next iteration, based on the fitness of the frogs in each memeplex, the probability of being in a given memeplex is calculated. For this purpose, the memeplex with the weakest frogs takes the highest probability of selection. Based on these probabilities, a memeplex is randomly selected for the next frog. There is a hope that the next frog places in the weakest memeplex, i.e., a memeplex in which the total quality of frogs is less than that of other memeplexes. In this way, more balanced memeplexes are generated. The same process is done in each iteration, i.e., the probabilities are calculated again, and a memeplex with the least fitness takes the highest probability of selection.

The process of randomly selecting the memeplexes and updating the probabilities continues until all the frogs are placed in memeplexes. When *n* frogs are placed in a memeplex, the probability of the selection of that memeplex in next iterations becomes zero.

4.3. New Method of Memetic Evolution in Each Memeplex

Another problem with the standard SFLA is that if the worst frog always leaps toward the best frog, the ability for the global search decreases, and the possibility of getting stuck in a local optima increases. We use a similar procedure to that of [38] to overcome this problem. Let X_b and X_w, represented as vectors, be the position of the best and worst frogs, respectively. The leap vector is calculated from Eq. (4):

$$S = rand \times (X_b - X_w) + rand(d) \times \frac{maxStep}{k} \qquad (4)$$

where *S* is the leap vector along which the frog moves; $rand$ is a random number between 0 and 1; $rand(d)$ is a *d*-dimensional vector of random numbers between -1 and 1, where *d* indicates the number of frog memes (i.e., problem parameters). $maxStep$ is also a *d*-dimensional vector that denotes the largest leap in each dimension. The multiplication of two vectors $rand(d)$ and $maxStep$ results in a vector whose elements are computed by the multiplication of the corresponding elements in these vectors. Therefore, the leap vector *S* is calculated by adding two vectors. The first vector is a vector with a random length along the path between the worst and the best frogs. The second vector adds a slight deviation to the first vector to prevent it from being stuck in a local optimal. The maximum length of this vector is the length of the largest possible leap divided by k to not cause a significant deviation and not affect the convergence of the algorithm. According to the performed sensitivity analysis which will be reported in subsection 5.1, we set *k=15* in the implementation of our algorithm.

4.4. Partial Opposition-Based Initial Population Generation

The success level and the convergence speed of SFLA largely depend on the initial population. Selecting an appropriate initial population can lead to find the global optimum. Otherwise, the algorithm will probably be trapped in a local optimum. To help the algorithm in finding the global optimum, the initial population should have two characteristics: randomness of the initial population and appropriate quality of the population individuals. Randomized spread of the population individuals in the search space may increase the probability of placing some of them in the vicinity of the global optimum. On the other hand, the lack of such spread may lead the algorithm to miss some parts of the search space. In addition, the appropriate quality of the initial population can raise the convergence speed of the algorithm. With more qualified population, the algorithm converges more quickly, and the search process often happens in the vicinity of the global optimum.

Rahnemayan et al. in [39] presented the method of opposition-based initial population generation for the Differential Evolution (DE) algorithm. Experiments in [39] show that selecting the more qualified individuals from the population itself and from the opposite group leads to better results and can raise the convergence speed. To perform this process, for each individual of the population, the opposite point will be calculated. Let $X = (x_1, x_2, \dots, x_d)$ represent an individual in a *d*-dimensional search space. Each element of the opposite individual will be calculated using the following equation:

$$\breve{x}_i = a_i + b_i - x_i \qquad (5)$$

where x_i shows the ith element of the individual X, $\breve{x}_i$ shows the ith element of the opposite individual of X; a_i and b_i show the lower and upper bounds of the ith dimension of the search space, respectively.

half of the individuals with more quality will be selected as the initial population. The problem of this method is that selecting individuals with the highest quality (from the initial population and the opposite population) decreases the randomness degree of the population. This will decrease the ability of the algorithm in finding the optimal solution, although the convergence speed will be increased.

In this paper, instead of calculating all elements of the opposite individual, only a subset of the elements will be randomly selected and computed. In other words, a partial opposite point will be determined by applying Eq. (5) only on some randomly selected elements and keeping the other elements unchanged. In this way, the opposition-based initial population generation method [39] is a particular form of our proposed approach, in which the selected subset includes all the elements.

In Section 5, we conduct an experiment to compare our proposed method for generating the initial population with the "Randomized" method used in the standard SFLA, and the "Opposition-Based" method [39]. The results show that our method achieves higher accuracy in comparison with the other two methods.

5. Experimental Results

To evaluate ISFLA, we compare this algorithm with four evolutionary algorithms, including the standard SFLA, GA, ACO, and PSO. The comparison is performed with respect to the following evaluation criteria:

1- Execution time

2- Accuracy, i.e., the best fitness of all generations
3- Convergence pattern to the optimal solution, which is determined in this paper by calculating the average of the best fitness values in each generation over all runs.

All the under comparison algorithms have been applied to six benchmark datasets taken from UCI [40]. The characteristics of these datasets are provided in Table 1.

Table 1. The characteristics of the benchmark datasets

Dataset	Number of Samples	Number of Attributes
Breast Cancer	699	10
Ecoli	336	8
Glass	214	10
Wine	178	13
MAGIC Gamma Telescope	19020	11

For each algorithm, the population size and the number of generations are constantly set to 100 and 50, respectively. Other specific parameters of each algorithm are listed in Table 2. The setting of the parameters of SFLA and ISFLA is supported by the sensitivity analysis results reported in the next subsection. All the experiments have been implemented in Matlab R2014a, performed in the environment of Windows 7, and run on a 2.50 GHz Intel (R) Core (TM) i7 processor with 8.00 GB RAM.

Table 2. The parameter settings

Algorithm	Parameter	Value
PSO	w: Inertia Weight	Fine Grained Inertia Weight (FGIW), Nonlinear Adaptive [41]
	Vmax	Equal to maxstep in SFLA
	c1: Cognitive Parameter	2.05
	c2: Scaling Parameter	2.05
	Mutation Operator	Adaptive Mutation Selection (AMS) [42]
	Initialization	Generalized Opposition-Based Learning (GOBL) [43]
ACO	Alpha: Pheromone Decay Parameter	0.3
	T: Pre-set Slope	1.0
	Phi: Adjustment Coef.	0.5
	p0: Threshold of the global random search control	0.5
	q0: Threshold of the neighborhood transfer control	0.5
	Rmax	Equal to Maxstep in SFLA
GA	Mutation Function	@mutationadaptfeasiblew
	Elitism Rate	0.05
	Other Parameters	default values from the GA Toolbox Matlab Software
SFLA And ISFLA	Number of Memeplexes	10
	Memeplex Size	10
	Submemeplex Size (in SFLA)	8
	Number of memeplex evolutions in each iteration	9
	Number of generations	50
	Maxstep	0.2

5.1. Sensitivity Analysis

Like all the heuristics, parameter selection is critical to the ISFLA performance. ISFLA has four main parameters: the number of memeplexes (m), the number of frogs in a memeplex (n), the number of evolution steps between two successive shufflings in a memeplex (N), and parameter k in Eq. (4). At this point, no clear theoretical basis is available to dictate parameter value selection for ISFLA. Therefore, we use sensitivity analysis to find the optimal values of these parameters.

In our sensitivity analysis, first, m was varied in the range from 5 to 50 with an increment of 5, where n=10, N=5, k=10 (as the default values of these parameters). Then, the best fitness value of ISFLA for all the benchmark datasets was calculated. The results are shown in Fig. 6. Based on these results, we select m=10 since this value leads to relatively better accuracy.

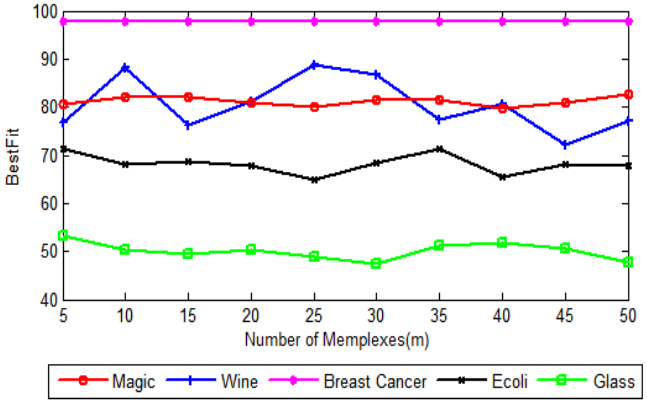

Fig. 6. Sensitivity analysis on the number of memeplexes

To perform sensitivity analysis for n, this parameter was varied in the range from 5 to 50 with an increment of 5, where m=10, N=5, k=10 (N and k were assigned with their default values, while m was assigned with its best value determined in the previous analysis). The resulting best fitness values per each benchmark are shown in Fig. 7.

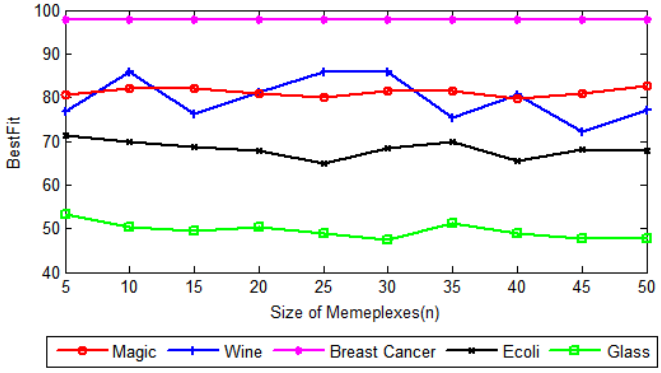

Fig. 7. Sensitivity analysis on the size of memeplex

According to these results, we set n=10 from this point on. As Fig. 7 shows, n=35 is almost more appropriate for some datasets. However, choosing a larger value of n increases the population size, and thus, increases the algorithm execution time (because the number of function evaluations to reach the goal increases).

Next, parameter N was varied from 3 to 27 with an increment of 3, while m=10, n=10, k=10.

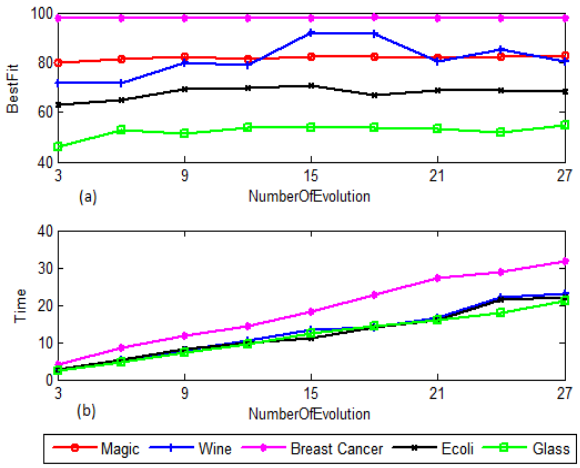

Fig. 8. Sensitivity analysis on the number of evolutions

According to Fig. 8(a), N=9 results in an acceptable accuracy. However, for some datasets, such as Wine, there are more appropriate values for N. Nevertheless, as Fig. 8(b) illustrates, increasing N would result in a considerable increase in the execution time. Thus, we prefer to choose a small value of N, which at the same time leads to an acceptable accuracy. The execution time for the "MAGIC Gamma Telescope" dataset could not be depicted in Fig. 8(b) since it is considerably larger than the execution time for the other datasets.

To perform sensitivity analysis on parameter k (refer to Eq. (4)), it was varied from 5 to 40 with an increment of 5, while m=10, n=10, N=9. The results of applying ISFLA on the benchmark datasets for different values of k are shown in Fig. 9. As seen, different datasets imply different best values for k. As a solution, we could select a value of this parameter that provides relatively good results for all the datasets. Another solution could be to choose and apply the best value of this parameter per each dataset, separately. In this paper, we rely on the first solution and choose k=15 for evaluating the performance of our algorithm on different datasets.

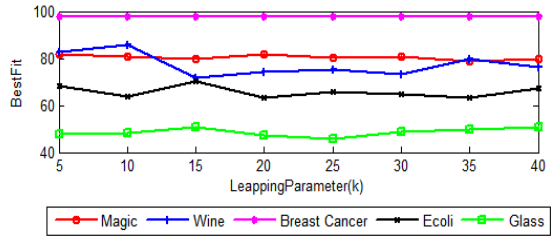

Fig. 9. Sensitivity analysis on the leaping parameter

5.2. Performance Evaluation

To increase the reliability of the results, each algorithm (SFLA, ISFLA, GA, ACO, and PSO) was run 50 times (each run included 50 generations), and three mentioned evaluation criteria were calculated for each dataset.

Fig. 10 represents the best fitness of the population in 50 runs of the under comparison algorithms over five datasets. The horizontal axis represents the number of runs of the algorithms. The best fitness shows the best accuracy gained by the algorithms. As it can be found from this figure, in all cases, ISFLA and PSO had better accuracy than the other algorithms. In addition, the performances of ISFLA and PSO are comparable to each other. For some datasets, PSO outperformed ISFLA and vice versa. Table 3 provides the average best fitness for all the runs of five algorithms. This table confirms the above findings. To provide a standardized way of displaying the distribution of the resulting data based on the five number summary, i.e., minimum, first quartile, median, third quartile, and maximum, we use the box plot representation in Fig. 11.

Table 3. Comparison of the average best fitness in all runs

Algorithm	Dataset Name				
	Wine	Breast Cancer	Ecoli	Glass	Magic
ACO	77.517	97.190	65.042	49.514	78.602
GA	86.427	97.785	67.541	52.972	80.438
SFLA	78.842	97.728	66.867	50.654	79.112
PSO	96.314	97.816	71.759	54.336	81.534
ISFLA	95.595	97.871	71.962	53.841	81.440

To examine the effectiveness of our proposed method for generating the initial population, we conducted an experiment over five datasets to compare this method with "Randomized" and "Opposition-Based" methods. The results in Table 4 show that our method achieved higher accuracy in comparison with the other methods over different datasets.

Table 4. Comparison of the initial population generation methods

Dataset	ISFLA Best Fitness		
	Random	Opposition-Based	Partial Opposition-Based
Wine	95.225	95.376	95.595
BreastCancer	97.834	97.854	97.871
Ecoli	71.706	71.753	71.962
Glass	53.787	53.798	53.841
Magic	81.419	81.426	81.440

Fig. 12 shows the time taken for each run of the algorithms over five datasets. As it can be found from this figure, ISFLA had the least execution time among all the algorithms for all the datasets. The order of the other algorithms in terms of the execution time is ⟨GA, SFLA, PSO, ACO⟩. For small datasets, there is not much difference in the execution time of the various algorithms. However, for large datasets (e.g., "MAGIC Gamma Telescope"), the difference is noticeable.

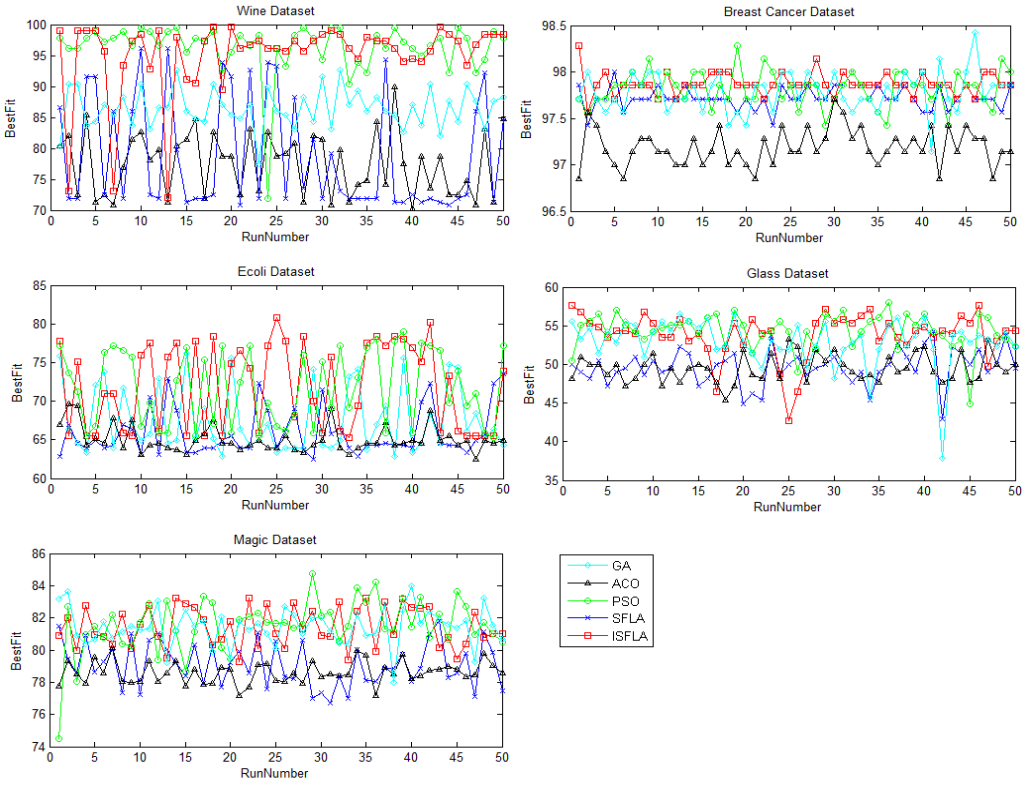

Fig. 10. The best fitness of each algorithm

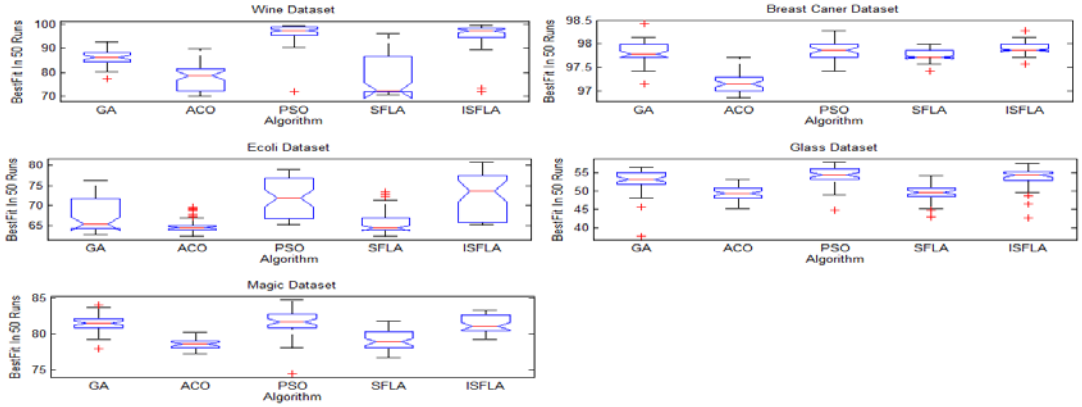

Fig. 11. The best fitness representation in the box plot

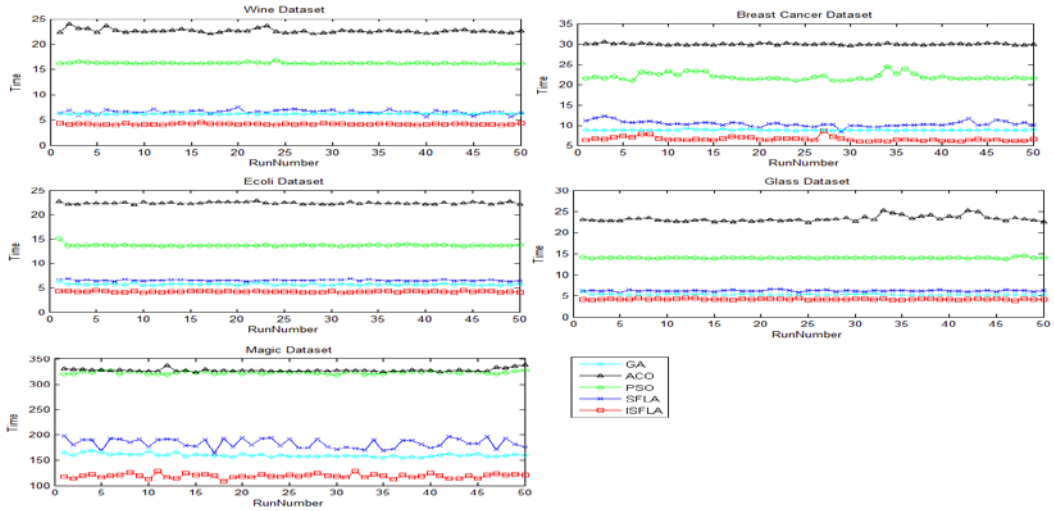

Fig. 12. The execution time of each algorithm

For example, the execution time of PSO over "MAGIC Gamma Telescope" is almost three times more than the execution time of ISFLA.
One major reason for the results in Fig. 12 is that ISFLA has a lower number of fitness function evaluations than other algorithms. As an example, Fig. 13 shows the number of fitness function evaluations for the "Wine" dataset. Due to the removal of the submemeplex creation process in ISFLA, it has less execution time than SFLA

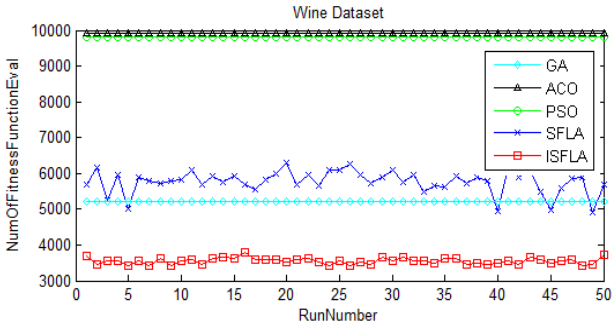

Fig. 13. The number of fitness function evaluations for the Wine dataset

Fig. 14 depicts the "average of the best fitness values in each generation over all runs" as a metric of evaluating the convergence speed to the optimal solution. For the "Ecoli" dataset, ISFLA has the best convergence speed, while for the "Wine" and "Glass" datasets, PSO outperforms ISFLA. In cases of "Breast Cancer" and "Magic" datasets, the convergence behavior of PSO and ISFLA is almost similar.

5.3. Comparison through Statistical Analysis

In order to draw conclusions in much higher confidence, we performed the ANOVA (ANalysis Of VAriance) test on the best fitness and the execution time results of each algorithm (see Tables 5 and 6).
From the ANOVA test results shown in Table 5, we can see that ISFLA's execution times are always lower than the execution times of SFLA, PSO, ACO, and GA for all the benchmarks. Considering the significance, the p-values of ISFLA against other algorithms are far less than 0.05 for all the benchmarks. It means that the mean difference of the algorithms is significant at the 0.05 level.
For another metric, "best fitness", the relations between five algorithms (refer to Table 6) are slightly different compared to the results of the "execution time" metric. Considering the significance, the p-values of ISFLA vs. SFLA, ISFLA vs. ACO, and ISFLA vs. GA are less than 0.05 for all the benchmarks.
The p-values of ISFLA vs. PSO are greater than 0.05 for three benchmarks. That is to say, these two algorithms have no significant difference in most cases. For the "Wine" dataset, the significance between ISFLA and PSO is 0 and (x-z)% is negative, meaning that PSO is better than ISFLA. In contrast, for the "Ecoli" dataset, the significance value is 0.0253 and (x-z)% is positive, meaning that ISFLA is better than PSO in this case.

According to the above statistical analysis, we can conclude that ISFLA has a significant advantage over other algorithms in terms of the execution time. With respect to the best fitness, ISFLA and PSO are comparable, while they outperform other algorithms.

6. Concluding Remarks and Future Work

SFLA is a relatively recent evolutionary algorithm that has been proven to be efficient in solving various optimization problems. The problem of automatic creation of a Sugeno fuzzy system can be considered as an optimization problem where the optimal values of the system's parameters need to be determined. In this paper, we first applied the standard SFLA to create a Sugeno fuzzy system. Then, we improved this algorithm by presenting a new method for creating memeplexes, a new mechanism for frog leaping, and a new method for generating the initial population. We also eliminated submemeplex creation from the original algorithm. The experimental results showed that the new proposed algorithm can create fuzzy systems more efficiently than the standard SFLA and some other evolutionary algorithms (i.e., GA, ACO and PSO). In addition, with respect to the accuracy and convergence speed criteria, ISFLA and PSO outperformed other evolutionary algorithms, while their performance was comparable to each other.
In this paper, we eliminated the process of constructing submemeplexes from the standard SFLA. In future works, we plan to propose a new solution to improve the construction of submemeplexes without removing them. Another future work could be the hybridization of SFLA with other effective evolutionary algorithms in order to utilize the benefit of each individual algorithm for improving the convergence rate, and balancing two antagonist notions, exploration and exploitation.

Table 5. The ANOVA test on the execution time of each algorithm at the 0.05 significant level.

Dataset	ISFLA(x) vs SFLA(y)		ISFLA(x) vs PSO(z)		ISFLA(x) vs GA(w)		ISFLA(x) vs ACO(q)	
	(x-y)%	p-value	(x-z)%	p-value	(x-w)%	p-value	(x-q)%	p-value
Wine	-2.39	2.94E-121	-12.02	1.30E-287	-18.38	0.0E0	-1.96	4.96E-103
Breast Cancer	-3.79	8.97E-102	-15.33	3.32E-242	-23.39	7.91E-287	-2.14	6.03E-56
Ecoli	-2.25	1.37E-163	-9.50	0.0E0	-18.16	0.0E0	-1.43	9.05E-119
Glass	-2.06	1.71E-87	-9.82	2.40E-242	-19.07	0.0E0	-1.16	2.54E-45
Magic	-64.27	3.29E-78	-203.55	4.69E-188	-208.65	1.32E-190	-43.70	3.89E-50

Table 6. The ANOVA test on the best fitness of each algorithm at the 0.05 significant level.

Dataset	ISFLA(x) vs SFLA(y)		ISFLA(x) vs PSO(z)		ISFLA(x) vs GA(w)		ISFLA(x) vs ACO(q)	
	(x-y)%	p-value	(x-z)%	p-value	(x-w)%	p-value	(x-q)%	p-value
Wine	16.75	2.22E-30	-0.72	0.0E0	18.07	3.65E-34	9.17	7.98E-11
Breast Cancer	0.14	0.0E0	0.05	0.132	0.68	1.79E-51	0.08	1.51E-2
Ecoli	5.09	3.24E-10	0.20	2.53E-2	6.92	6.79E-13	4.42	8.91E-6
Glass	3.187	3.42E-12	-0.49	0.053	4.32	6.14E-13	0.86	0.0E0
Magic	2.33	2.14E-17	-0.09	0.102	2.84	1.16E-23	1.002	0.0E0

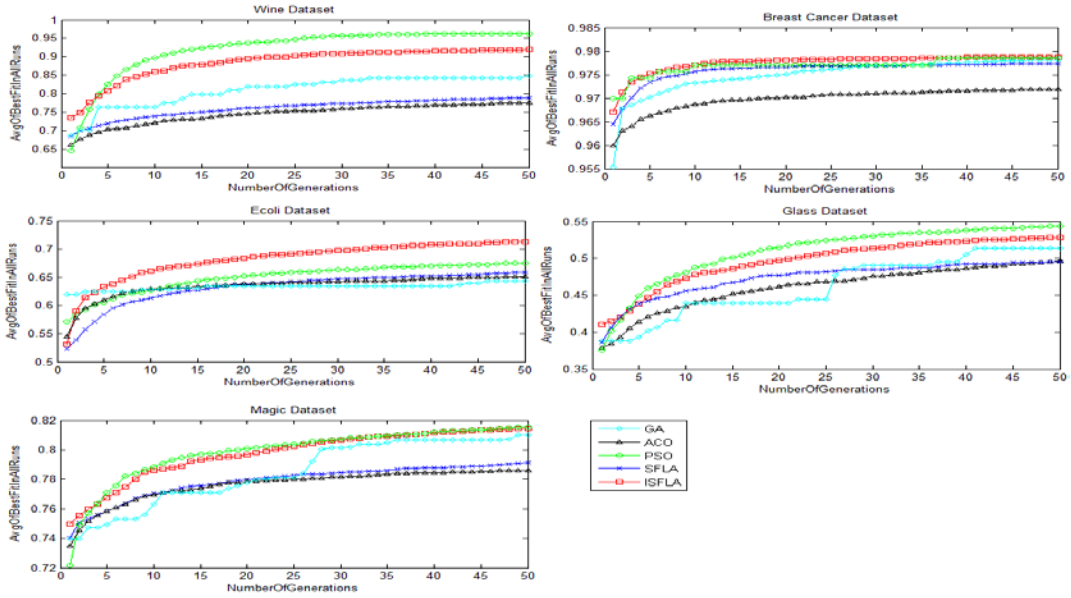

Fig. 14. Convergence speed of each algorithm

References

[1] S. Zhou, R. Lyons, S. Brophy, and M. Gravenor, "Constructing Compact Takagi-Sugeno Rule Systems: Identification of Complex Interactions in Epidemiological Data," *PLoS ONE*, vol. 1, no. 12, pp.1–14, 2012.

[2] H. Ishibuchi, T. Nakashima, and M. Nii, "Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining," *Springer-Verlag*, 2005.

[3] R. A. Carreras, D. k. Marker, and B. J. Lutz, "Fuzzy Logic Control for Optical Membrane Mirror," *Intelligent Automation & Soft Computing*, vol. 11, no. 1, pp. 59-67, 2013.

[4] M. H. Kazeminezhad, A. Etemad-Shahidi, and S. J. Mousavi, "Application of fuzzy inference system in the prediction of wave parameters," *Ocean Engineering*, vol.32, no.14, pp. 1709–1725, 2005.

[5] M. A. Boyacioglu, and D. Avci, "An Adaptive Network-Based Fuzzy Inference System (ANFIS) for the Prediction of Stock Market Return: The Case of the Istanbul Stock Exchange," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7908–7912, 2010.

[6] Y. Bai, H. Zhuang, and Z. S. Roth, "Fuzzy Logic Control to Suppress Noises and Coupling Effects in a Laser Tracking System," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 1, pp.113–121, 2005.

[7] Z. Chi, H. Yan, and T. Pham, "Fuzzy Algorithms with Applications to Image Processing and Pattern Recognition" *World Scientific*, 1996.

[8] Z. Huai-xiang, W. Feng, and Z. Bo, "Genetic optimization of fuzzy membership functions," *International Conference on Wavelet Analysis and Pattern Recognition*, pp. 465-470, 2009.

[9] M. Setnes, and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," I*EEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp.509-522, 2000.

[10] S. E. Papadakis, and J. B. Theocharis, "A GA-based fuzzy modeling approach for generating TSK models," Fuzzy Sets and Systems, vol. 131, no. 2, pp.121-152, 2002.

[11] A. Zafari, "Developing a fuzzy inference system by using genetic algorithms and expert knowledge," MsC

Thesis in Geo-Information Science and Earth Observation, University of Twente, Netherlands, 2014.
[12] H. M. Elragal, "Improving accuracy of fuzzy classifiers using swarm intelligence," *International Conference on Communication Software and Networks (ICCSN)*, pp. 170-174, 2011.
[13] F. Gu, K. Ngai Ming, and H. Quang, "Automatic fuzzy membership function tuning using the particle swarm optimization," *Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pp. 324-328, 2008.
[14] Z. Yongsheng, and L. Baoying, "A new method for optimizing fuzzy membership function," *International Conference on Mechatronics and Automation*, pp. 674-678, 2007.
[15] M. Bodur, A. Acan, and T. Akyol, "Fuzzy system modeling with the genetic and differential evolutionary optimization," *International Conference on Computational Intelligence for Modelling, Control and Automation*, pp. 432-438, 2005.
[16] N. Krasnogor, and J. Smith, "A tutorial for competent memetic algorithms: model taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, pp. 474–488, 2005.
[17] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization," *Engineering Optimization*, vol. 38, no. 2, pp. 129-154, 2006.
[18] Z. Ziyang, W. Daobo, and L. Yuanyuan, "Improved shuffled frog leaping algorithm for continuous optimization problem," *IEEE Congress on Evolutionary Computation*, pp. 2992-2995, 2009.
[19] Q. Duan, S. Sorooshian, and V. Gupta, "Effective and efficient global optimization for conceptual rainfall-runoff models," *Water Resources Res*, vol. 28, no. 4, pp. 1015-1031, 1992.
[20] M. M. Eusuff, and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *J Water Resour Plan Manag*, vol. 129, no. 3, pp. 210-225, 2003.
[21] M. A. Ahandani, "A diversified shuffled frog leaping: an application for parameter identification," *Appl Math Comput*, vol. 239, pp. 1–16, 2014.
[22] M. Barati, and M. M. Farsangi, "Solving unit commitment problem by a binary shuffled frog leaping algorithm," *IET Gener Transm Dis*, vol. 8, no. 6, pp. 1050–1060, 2014.
[23] N. Yuvaraj, and A. Sabari, "Twitter Sentiment Classification Using Binary Shuffled Frog Algorithm," *Intelligent Automation & Soft Computing*, vol. 23, no. 2, pp. 373-381, 2016.
[24] M. P. Aghababa, M. E. Akbari, A. M., Shotorani, and R.M., Shotorbani, "Application of modified shuffled frog leaping algorithm for robot optimal controller design," *International Journal of Scientific & Engineering Research*, vol. 11, no. 2, pp. 1-6, 2011.
[25] E. Elbeltagi, T. Hegazy, and D. Grierson, "A modified shuffled frog-leaping optimization algorithm: applications to project management," *Structure and Infrastructure Engineering*, pp. 53-60, 2007.
[26] D. Lei, and X. Gou, "A shuffled frog-leaping algorithm for job shop scheduling with outsourcing options," *International Journal of Production Research*, vol. 54, no. 16, 2016.
[27] X. Xie, R. Liu, X. Cheng, X. Hu, and J. Ni, "Trust Driven and PSO-SFLA based job scheduling algorithm on cloud," *Intelligent Automation & Soft Computing*, vol. 22, no. 4, pp. 561-566, 2016.
[28] L. Deming, and G. Xiuping, "A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents," *Expert Systems with Applications*, vol. 42, no. 23, pp. 9333–9339, 2015.
[29] A. Sarkheyli, A. Zain, and S. Sharif, "The role of basic, modified and hybrid shuffled frog leaping algorithm on optimization problems: a review," *Soft Computing*, vol. 19, pp. 2011–2038, 2015.
[30] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 9, pp. 43–53, 2005.
[31] A. Kaur, and A. Kaur, "Comparison of mamdani-type and sugeno-type fuzzy inference systems for air conditioning system," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, 2012.
[32] A. P. Jacquin, and A.Y., Shamseldin, "Review of the application of fuzzy inference systems in river flow forecasting," *Journal of Hydro Informatic*, pp. 202-210, 2009.
[33] T. J. Ross, "Fuzzy logic with engineering applications," *third ed., Wiley*, 2009.
[34] A. Haman, and N. D. Geogranas, "Comparison of mamdani and sugeno fuzzy inference systems for evaluating the quality of experience of hapto-audio-visual applications, *HAVE 2008 – IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2008.
[35] A. Baker, "Simplicity. Stanford Encyclopedia of Philosophy," *California: Stanford University. ISSN 1095-5054*, 2010[2004].
[36] I. Silva, and R., Flauzino, "Efficient parametric adjustment of fuzzy inference system using unconstrained optimization," *Computational and Ambient Intelligence*, Springer Berlin Heidelberg, pp. 399-406, 2007.
[37] S. Kosanam, and D. Simon, "Fuzzy membership function optimization for system identification using an extended kalman filter," *Annual meeting of the North American Fuzzy Information Processing Society*, pp. 459-462, 2006.
[38] T. H. Huynh, "A modified shuffled frog leaping algorithm for optimal tuning of multivariable PID controllers," *ICIT 2008. IEEE International Conference*, pp. 1-6, 2008.
[39] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-Based Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64-79, 2008.
[40] M. Lichman, "UCI Machine Learning Repository" [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.
[41] P. Chauhan, K. Deep, and M. Pant, "Novel inertia weight strategies for particle swarm optimization," *Memetic Computing Journal*, vol. 5, no. 3, pp. 229-251, 2013.
[42] W. Dong, L. Kang, and W. Zhang, "Opposition-based particle swarm optimization with adaptive mutation strategy," *Memetic Computing Journal*, vol. 5, no. 3, pp. 229-251, 2012.

[43] H. Wang, Z. Wu, and S. Rahnamayan, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Science*, vol. 181, no. 20, pp. 4699–4714, 2011.

Seyyed Mohsen MirHosseini is working towards a doctoral degree in the area of concurrent programs testing at the Shahid Beheshti University, Tehran, Iran. His areas of interest include Software Testing and Metaheuristic algorithms.

Email: m_mirhosseini@sbu.ac.ir

Hassan Haghighi received his Ph.D. in computer engineering from Sharif University of Technology. He is an associate professor at the faculty of Computer Science and Engineering in Shahid Beheshti University, Tehran, Iran. His research focus is on Software Testing, Formal Methods, and Software Architecture.

Email: h_haghighi@sbu.ac.ir

Jamshid Bahrami is a Ph.D. student in Computer Engineering at Sanandaj Branch of Islamic Azad University. Currently, he collaborates with the software testing lab of Shahid Beheshti University.

Email: bahramij@yahoo.com

Paper Handling Data:

Submitted: 05.23.2018
Received in revised form: 08.01.2018
Accepted:08.04.2018
Corresponding author: Hassan Haghighi,
Faculty of Computer Science and Engineering,
Shahid Beheshti University G. C., Tehran, Iran