A Multi-strategy Shuffled Frog-leaping Algorithm for Numerical Optimization

Kezong Tang¹, Tangsen Zhan¹ and Zuoyong Li²

¹School of Information Engineering, Jingdezhen Ceramic Institute, Jingdezhen, JiangXi, 333403, China.
Email: tangkezong@126.com
²Industrial Robot Application of Fujian University Engineering Research Center, Minjiang University, Minjiang, FuJian, 350121, China.
Email: fzulzytdq@126.com

Abstract. This study proposes a multi-strategy shuffled frog-leaping algorithm for numerical optimization (MSSFLA), which combines the merits of a frog-leaping step rule, the crossover operator, and a novel recursive programming. First, the frog-leaping step rule depends on the level of attractive effect between the worst frog and other frogs in a memeplex, which utilizes the advantages of frogs around the worst frog, making the worst frog more conducive to the evolution direction of the whole population. Second, the crossover operator of the genetic algorithm is used for yielding new frogs based on the best and worst individual frog instead of the random mechanism in the original shuffled frog-leaping algorithm (SFLA). The crossover operation aims to enhance population diversity and conduciveness to the memetic evolution of each memeplex. Finally, recursive programming is presented to store the results of preceding attempts as basis for the computation of those that succeed, which will help save a large number of repeated computing resources in a local search. Experiment results show that MSSFLA has better performance than other algorithms on the convergence and searching effectivity. Therefore, it can be considered as a more competitive improved algorithm for SFLA on the efficiency and accuracy of the best solution.

1. Introduction

The shuffled frog-leaping algorithm (SFLA) was first proposed in 2003 by Eusuff and Lansey[1-2], getting inspiration on the social behavior of frogs seeking food in a pond. The characteristics of SFLA, such as stochasticity, simplicity, and easy implementation, allowed it to gain considerable importance and be widely used for solving many complex optimization problems[3-9]. In spite of its applicability, becoming trapped in a local optimum is a general problem with the SFLA and other heuristic optimization algorithms, where they may work well on one problem, but may fail on another problem. Moreover, although the SFLA can usually yield good solutions, they are still time-consuming to compute and their performances deteriorate rapidly in the increasing complexity and dimensionality of the search space.

To overcome these problems, some new improvement techniques need to be extended to obtain better performances in terms of solution quality and competitive computation complexity for local search[10-15]. To improve further the global information exchange among memeplexes and speed up convergence, we designed a recursive strategy where a recursive programming technique is presented to store the results of preceding attempts as basis for the computation of those that succeed. To store the temporary results, an external archive set is introduced to memorize the fitness values that were calculated. Furthermore, we concentrate on the operators of the GA and to improve their search
capacity while avoiding entrapment in local optima. The crossover operator of the genetic algorithm (GA) is used for yielding new frogs based on the best and worst individual frog instead of the random mechanism in the original SFLA. Moreover, experiment comparisons with other algorithms verify the efficiency and capability of the proposed algorithm.

The rest of this paper is organized as follows: Section 2 introduces the mathematical model used to describe the concepts of the standard SFLA. Section 3 describes the proposed method in details. The experimental results and some comparisons with other common methods are given in Section 4 to verify the efficiency and capability of the proposed algorithm. Finally, conclusion and future work are drawn in Section 5.

2. SFLA Scheme
SFLA is a meta-heuristic stochastic search method that mimics the behavior of frogs when seeking for food. Its basic concept of population-based approach is derived from a virtual population of frogs in which individual frogs represent a set of possible solutions. The population is sorted according to their fitness values. Next, the frogs are divided into m memeplexes. Subsequently, frogs begin to two alternating processes, namely, local exploration in a memeplex and global information exchange among all memeplexes. The former tries to evolve frogs that are fitter and, by applying a frog-leaping rule (measurement of step length and displacement) according to Eqs. (1) and (2), attempts to move to a new better location in the search space.

$$\Delta_k = \text{rand}() \times [p_k^{bm} - p_k^{wm}] \quad (1)$$

$$p_k^{new} = p_k^{old} + \Delta_k \quad (2)$$

where $p_k^{bm}$ and $p_k^{wm}$ are the best and worst frog positions in a memeplex, respectively. $\text{rand}()$ is a random number between 0 and 1. $k$ is the iteration number of each memeplex. The current location of the worst frog ($p_k^{wm}$) is improved by $\Delta_k$ to reach a new location ($p_k^{bm}$). If $p_k^{bm}$ is better than the previous position ($p_k^{wm}$), then $p_k^{wm}$ is replaced by the new location. Otherwise, the calculations in Eqs. (1) and (2) are repeated with respect to the global best frog ($p^{gw}$), that is, $p^{gw}$ replacing $p_k^{bm}$. For convenience, $p_k^{wm}$, $p_k^{bm}$, and $p^{gw}$ are the worst frog, the memeplex best individual, and the best individual of the entire population, respectively.

3. Proposed Algorithm

3.1. New Frog-leaping Step Rule
Eq. (1) shows that the improvement benefited from $p_k^{bm}$ and $p^{gw}$. However, a careful examination of the local exploration reveals that the worst frog’s leaping step is substantially influenced by all individual frogs in a memeplex, and not just $p_k^{bm}$ and $p^{gw}$. Hence, Eq. (1) can be modified as:

$$\Delta_k = \text{rand}() \times [p_k^{bm} - p_k^{wm}] + \sum_{h \in \{wm, rt\}} \varphi(h) \text{rand}() \times [p_k^{h} - p_k^{wm}]$$

$$\varphi(h) = \frac{f(p_k^{h}) - f(p_k^{wm})}{\sum_{h \in \{wm, rt\}} f(p_k^{h}) - f(p_k^{wm})} \quad (3)$$

where $s(wm, n)$ represents a submemeplex of $n$-1 individual frogs around $p_k^{wm}$ in a memeplex with $n$ individual frogs; $h$ represents the index of a particular frog. The first item in Eq. (3) considers the influence degree of the frogs surrounding the current frog $p_k^{wm}$; $\varphi(h)$ is a normalization factor that indicates the degree of impact on the worst frog while executing the leaping step rule. On the other
hand, each frog may have a tendency to remain in its current moving status while leaping to find food, which can be described by movement inertia similar to that of the standard PSO. Here, a new frog-leaping step rule is reformulated based on Eq. (3), and then expressed as:

$$\Delta_k = w_\Delta_{k-1} + \text{rand}(\cdot) \times [p_{k}^{\text{lm}} - p_{k}^{\text{wm}}] + \sum_{h \in \text{rand}(\cdot) \times [p_{k}^{h} - p_{k}^{\text{wm}}]} \phi_{h}(h)$$

(4)

where $\Delta_{k-1}$ denotes the leaping step size in the $(k-1)^{th}$ iteration, where it can be considered as the velocity of the worst frog in the $(k-1)^{th}$ iteration. The parameter $w$ is the inertia weight similar to that of the standard PSO, which helps to regulate the search process in the MSSFLA. For the local exploration in each memeplex, we choose linearly decreasing weights for $w$ in the range of $[w_{\text{min}}, w_{\text{max}}]$. $w$ is varied as:

$$w = w_{\text{max}} - (w_{\text{max}} - w_{\text{min}}) \cdot \frac{\text{iter}}{\text{iter}_{\text{max}}}$$

(5)

where $w_{\text{min}}$ and $w_{\text{max}}$ are the initial and final weight, respectively; $\text{iter}$ is the current iteration and $\text{iter}_{\text{max}}$ is the maximum number of iterations.

### 3.2. Crossover Operator

In a standard SFLA, a randomly generated frog after the predefined iteration number will substitute for the worst frog if the latter cannot be improved by using the information from and $p^{\text{pw}}$. Then, the random frog becomes a new member in the current memeplex. Consequently, this will weaken the search efficiency and affect the global performance of the memetic evolution. Given that the best frog usually carries useful information, the search space around it could be the most promising region. Hence, in the embedded crossover operator, the new frog is generated by performing one crossover operation to $p_{k}^{\text{lm}}$ or $p^{\text{pw}}$. This strategy aims to enhance the diversity of the population and conduciveness to the memetic evolution of each memeplex. The detailed steps of the crossover operator are as follows.

Step 1. First, select the two parent individual ($P_1$, $P_2$) of the crossover operation.

We randomly generate a number $rn$. If $rn<0.5$, then two random individuals are selected as the parent individuals of the crossover. Otherwise, $p^{\text{pw}}$ and $p_{k}^{\text{wm}}$ are selected for the crossover operation.

Step 2. Next, perform the crossover operation based on the parent individuals $P_1$, $P_2$.

Generate a random integer $r$, where $r$ is a position-based crossover constant used to generate two child individuals, namely, $Ch_1$, $Ch_2$ from two parent individuals. Then, exchange the gene fragments that are located at the left or right of the intersection $r$.

Step 3. Finally, make a comparison of the fitness values of $Ch_1$ and $Ch_2$. Select the better child individual to substitute for $p_{k}^{\text{lm}}$.

### 3.3. Recursive Programming

Intuitively, there is no change for the position of many frogs in the local exploration of memeplex. Hence, there is no need to reevaluate the fitness for each frog in SFLA. Here, an external archive set is introduced to memorize the fitness values that are calculated. Figure 1 depicts this recursive programming based on the standard SFLA. For convenience, the iteration number for the local exploration of a memeplex is set to 1.

The recursive region for the frogs is surrounded by a dotted box in Figure 1. Meanwhile, Figure 1 briefly illustrates the flowchart for the proposed MSSFLA algorithm to an extent. To better describe this recursive programming, we first assume that there are six individual frogs, which have been sorted according to their fitness values and divided into 2 memeplexes by executing Steps 1, 2, and 3. Next, the two worst frogs ($P_4$, $P_5$) are improved, and then two new positions of the frogs are obtained ($P_{4,\text{new}}$ and $P_{5,\text{new}}$) according to Eqs. (4) and (2). This improvement process includes the above-mentioned frog-leaping step rule and crossover operation for the local search process of a memeplex. Thereafter, the recursive programming is performed according to a predetermined procedure in the recursive
region. In this case, it consists of five basic steps, where the first step (Step4.2) is to store the fitness values of all frogs of each memeplex except for the worst frogs $P_4$ and $P_5$. $P_4_{\text{new}}$ is inserted into a specified external archive with a fixed capacity according to a specified bubble-sorting algorithm. Similarly, the above process in Steps 6 and 7 is repeated, and $P_5_{\text{new}}$ will be inserted into the external archive by the specified sorting algorithm. Finally, the external archive set would be the basis database for the next grouping of the individual frogs after completion of Step 8. However, in a standard SFLA, the individual frogs will be remixed and reevaluated after the completion of the local search of each memeplex. This will ignore the existing environments in which the position of many frogs did not change significantly. Consequently, this will result in a waste of computing resources and is not conducive to speed up the convergence of the algorithm.

![Recursive Programming](image)

**Figure 1.** Recursive Programming.

### 3.4. Proposed MSSFLA Algorithm

**Algorithm 1 - Pseudo code of the MSSLFA**

Step 1: Initialization phase. Set the algorithm parameters. Randomly generate the initial population, and evaluate each frog in the population.

Step 2: Sort the frogs according to their fitness values and divide the population into several memeplexes. For $i=1$ to $N_{\text{Global}}$

Step 3: Start a local search in each memeplex. For $j=1$ to $m$
Step 3.1: If \( j = 1 \), then store the frogs’ fitness values of the \( j \)th memeplex into the external archive set \( S_e \); otherwise, insert the frogs’ fitness value of the \( j \)th memeplex into the external archive set \( S_e \) according to a particular bubble-sorting algorithm.

For \( k = 1 \) to \( N_{local} \):

Step 3.1.1: Implement a new frog-leaping step rule. Determine the best ( \( p_{k}^{lw} \) ) and worst frog ( \( p_{k}^{wm} \) ) in a memeplex, and improve the worst frog \( p_{k}^{wm} \) with \( p_{k}^{lw} \) or \( p_{k}^{new} \) according to Eqs. (4) and (2), then obtain a new frog \( p_{k}^{new} \).

Step 3.1.2: If \( p_{k}^{new} \) has a better fitness than \( p_{k}^{wm} \), then replace \( p_{k}^{wm} \) with \( p_{k}^{new} \). Otherwise, perform a crossover operation and replace \( p_{k}^{wm} \) with a newly generated frog \( p_{k}^{new} \).

Step 3.1.3: Insert the fitness value of \( p_{k}^{new} \) into the external archive set \( S_e \) to substitute for the old value of \( p_{k}^{wm} \) using a particular sorting algorithm.

End For

End for

Step 4: Test the stop condition. Go to Step 3 if the stop condition is not met, and the external archive set will be used as the basis for the next grouping of individual frogs. Otherwise, output the best solution. End For.

4. Experiment Results

4.1. Benchmark Functions

The performance of MSSFLA is compared with three different methods, namely, standard SFLA[2], MSFLA[8] and LPSO[16]. The experimental tests utilized six widely used functions listed in Table 1, which have several local optima and saddles in their solution space. Table 1 briefly describes the function expression, dimensions \( (D) \), their search ranges, global optimal solution \( (x^*) \), the global optimization value \( (f_{min}) \), and the acceptable values \( (\varepsilon) \). If a solution found by an algorithm is better than the acceptable value, then the run is deemed successful. For a fair comparison, all functions are tested on 30 dimensions.

| Function expression | \( D \) | Search range | \( x^* \) | \( f_{min} \) | \( \varepsilon \) | Function name |
|---------------------|--------|--------------|----------|------------|----------|--------------|
| \( f_1(x) = \sum_{i=1}^{D} x_i^2 \) | 30 | \([-100,100]^D\) | \((0)^D\) | 0 | 1.0E-6 | Sphere |
| \( f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \) | 30 | \([-10,10]^D\) | \((0)^D\) | 0 | 1.0E-6 | Schwefel2.22 |
| \( f_3(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \) | 30 | \([-10,10]^D\) | \((1)^D\) | 0 | 100 | Rosenbrock |
| \( f_4(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i)) \) | 30 | \([-5.12,5.12]^D\) | \((0)^D\) | 0 | 15 | Rastigrin |
| \( f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) \) | 30 | \([-32,20]^D\) | \((0)^D\) | 0 | 1.0E-6 | Ackley |
| \( -\exp(1/D \sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + \varepsilon \) | \| | | | | |
| \( f_6(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \frac{\cos(x_i^2)+1}{\sqrt{i}} \) | 30 | \([-600,600]^D\) | \((0)^D\) | 0 | 1.0E-2 | Griewank |

4.2. Parameter Settings

For the standard SFLA, we use the parameter settings in [2]. There are 20 memeplexes, each containing 10 frogs. The local exploration in each memeplex is executed for 10 iterations. For the parameter settings for MSFLA, we follow the parameter settings in [8]. For LPSO, the cognitive and
social parameters, i.e., $c_1$ and $c_2$, are both equal to 2.0. The inertia weight $w$ linearly decreases from 0.9 to 0.4 with the increase in the iteration number. The population size is set to 30, and the maximum distance of flight is set to half the size of the range, i.e., $v_{max}=0.5 \times \text{range}$. For MSSFLA, the frogs’ number is set to $Sp=200$ and the memplex number $m$ is set to 20. To speed up convergence of the algorithm, the local evolution number $N_{Local}$ and the global iteration number $N_{Global}$ are set to 2 and 1000, respectively. For a fair comparison, the global iteration number is also set to 1000 for other compared algorithms.

4.3. Results and Discussion

Table 2 shows the results of the mean best solution and standard deviation (SD), which are used to test the convergence efficiency of the compared algorithms. In the experiments, MSSFLA and three other algorithms were applied to the optimization on six 30-dimensional functions, namely, $f_1$–$f_6$. The typical convergence curves of the mean fitness value of the 100 runs for four algorithms on six functions are illustrated in Figure 2(a)–(f). As shown in Table 2, the approach proposed in this study has very good performances in solving these problems compared with other competing heuristic algorithms.

|        | SFLA     | LPSO     | MSFLA    | MSSFLA   |
|--------|----------|----------|----------|----------|
| $f_1$  | 9.238E-10 | 3.542E-6 | 7.364E-15 | 3.453E-11 |
| $f_2$  | 4.354E-2  | 5.453E-1 | 2.102E-2 | 3.491E-1  |
| $f_3$  | 12.345    | 2.453    | 9.047    | 1.759    |
| $f_4$  | 4.367     | 5.762    | 4.012    | 7.329    |
| $f_5$  | 20.341    | 4.235    | 21.873   | 6.892    |
| $f_6$  | 8.432E-2  | 6.321E-1 | 2.321E-3 | 1.087E-2 |

Table 2. Comparison results for MSSFLA, SFLA, LPSO, and MSFLA.

Figure 2 shows that the proposed algorithm could converge to the global optimal solution in advance in terms of the iteration numbers. For instance, Figure 2(a) shows that the MSSFLA was converged around 500 iterations for the $f_1$ function, which is only half of the predefined $N_{Global}$. Although the three other algorithms can also converge to the global optimal value, the convergence speed of the MSSFLA significantly outperforms the three other algorithms according to the predefined iteration numbers. For $f_2$, the MSSFLA converged to a bounded region around 700 iterations. For $f_3$ and $f_5$, the MSSFLA converged to a bounded region around 900 iterations. For $f_6$, MSSFLA converged to a bounded region around 750 iterations. The MSSFLA was able to converge quickly to a bounded region around the best optimum in a finite number of iterations except for the $f_1$ function. For the iteration on the mean fitness value, the improved MSSFLA algorithm has faster convergence speed than the standard SFLA. From the quality of the solution perspective, the improved MSSFLA has better convergence accuracy according to Table 2.
In summary, according to above-mentioned experimental results and analysis, MSSFLA is a very promising algorithm for finding the globally optimal solution, and can also be better applied into addressing complex engineering problems for quickly finding better global stationary solutions.

5. Conclusions
This study presents a novel multi-strategy shuffled frog leap algorithm to improve the local and global search ability for complex optimization problems, where the standard SFLA is modified by integrating a variety of strategies including a new frog leap rule, a recursive programming for local exploration, and a crossover operator of GA. In our future work, we should further improve the MSSLFA by testing different crossover operators in a memeplex to make it more suitable for the discrete case, which will be discussed in detail in a future article.

6. Acknowledgments
This work is supported by the National Natural Science Foundation of China (No.61662037, 71763013), Jiangxi 2017 outstanding young talent program and the Natural Science Foundation of Jiangxi Province of China(20161BAB212042,GJJ150927), Industrial Robot Application of Fujian University Engineering Research Center, Minjiang University (MJUKF-IRA201808).

7. References
[1] Eusuff M M and Lansey K 2003 J. Water. Res. Plan. Man. 129 210-225.
[2] MEusuff M, Lansey K and Pasha F 2006 Eng Optimiz 38 129-154.
[3] Arshi S S, Zolfaghari A and Mirvakili S M 2014 Comput Phys Commun 185 2622-28.
[4] Panda S, Sarangi A and Panigrahi SP 2014 Aeu-Int J Electron C 68 1031-36.
[5] Li J Q, Pan Q K and Xie S X 2012 Appl Math Comput 218 9353-71.
[6] Fu Z, Huang F and Sun X 2016 IEEE T Serv Comput 9 1-1.
[7] Kong Y , Zhang M J and Ye D Y 2016 Knowl-Based Syst 115 123-132.
[8] Li X, Lou J P, Chen M R and Wang N 2012 Inform Sciences 192 143-151.
[9] Ngo T T, Sadollah A and Kim J H 2016 J Comput Sci-Neth 13 68-82.
[10] Teekeng W, Thammano A 2011Procedia Computer Science 6 69-75.
[11] Zhu G Y and Zhang W B 2014 Expert Syst Appl 41 6818-29.
[12] Niknam T, Farsani E A 2010 Eng Appl Artif Intell 23 1340-49.
[13] Luo J P, Li X, Chen M R and Liu H W 2015 Inform Sciences 316 266-2925.
[14] Jafari A, Bijami E, Bana H R and Sadri S 2012 Microelectron J 43 908-915.
[15] Rahimi-Vahed A and Mirzaei A H 2007 Comput Ind Eng 53 642-666.
[16] Poli R, Kennedy J, Blackwell T 2007 Swarm Intell-Us 1 33-57.