International Journal of Hybrid Information Technology
Vol.8, No. 5 (2015), pp. 195-206
http://dx.doi.org/10.14257/ijhit.2015.8.5.22

Two-Phases Learning Shuffled Frog Leaping Algorithm

Jia Zhao and Li Lv

School of Information Engineering, Nanchang Institute of Technology, Nan Chang 330099, China
zhaojia925@163.com; lvli623@163.com

Abstract

In order to overcome the drawbacks of standard shuffled frog leaping algorithm that converges slowly at the last stage and easily falls into local minima, this paper proposed two-phases learning shuffled frog leaping algorithm. The modified algorithm added the elite Gaussian learning strategy in the global information exchange phase, updated frog leaping rule and added the learning capability that the worst frog of current swarm learned from the best frog of other swarm. The learning capability of two-stage on the one hand increased the search range, on the other hand enhanced the diversity of population. Experiments were conducted on 13 classical benchmark functions, the simulation results demonstrated that the proposed approach improved the convergence rate and solution accuracy, when compared with common swarm intelligence algorithm and the latest improved shuffled frog leaping algorithm.

Keywords: *shuffled frog leaping algorithm; elite Gaussian learning; frog leaping rule, phase*

1. Introduction

Shuffled frog leaping algorithm (SFLA) [1], originally developed by Eusuff and Lansey in 2003, is a meta-heuristic optimization method for solving discrete optimization problems, SFLA, which mimics the mimetic evolution of a group of frogs while seeking for the maximum amount of available food, is based on evolution of memes carried by interactive individuals and a global exchange of information among the population. SFLA, basically combines the benefits of the mimetic algorithm (MA)[2] based on genetic evolution as well as particle swarm optimization(PSO) based the social behavior[3], has many advantages[4] which are simple structure, fast evolution velocity, a few parameters and easy implementation. it has been applied in these fields successfully, such as water Distribution[1], distribution feeder reconfiguration[5], assembly line sequencing problem[6], function optimization[7], Wireless Sensor Network Coverage Optimization[8], Wind Power Integrated System[9].

SFLA has been shown that it can yield good performance for solving various optimization problems. However, it tends to suffer from falling into local optimal, and low precise solution when solving complex and high-dimension problems with many extreme points. So many researchers pay more attention to improve the global convergence, [10] presented meme triangular probability distribution shuffled frog-leaping algorithm which chose meme strategy with several frogs. Taher Niknam proposed chaotic shuffled frog leaping algorithm using chaotic local search [11]. Improved shuffled frog leaping algorithm based on molecular dynamics simulations was proposed in [12]. [13] Put forward improved SFLA using power law external optimization, Jiang proposed a method a shuffled frog leaping algorithm using niche technology [14]. The above improved algorithms improved the performance in some extent, however, it is difficult to accelerate the convergence velocity when they avoid premature.

As to avoid premature and improve the convergence velocity, this paper presents a novel approach, called two-phases learning shuffled frog leaping algorithm (TLSFLA), where the elite Gauss learning is introduced to accelerate convergence velocity in the global information exchange, and improved SFLA rule is used in the local search for avoiding premature and keeping the diversity. The rule is helpful for increasing the search range and improving the learning ability of the worst frog from the best frog in the same memeplex and other meme lexes. A comprehensive set of 13 complex benchmark functions is employed for experimental verification. The results demonstrate promising performance of the new method TLSFLA on convergence velocity and global optimum, comparing with swarm intelligent algorithms and SFLA variants.

Organization of the rest of this paper is as follows. The standard SFLA is shortly reviewed in Section 2. In Section 3, the proposed algorithm is presented and a comprehensive experimental comparisons and results are provided in Section 4. Finally, the work is concluded in Section 5.

2. Shuffled Frog Leaping Algorithm (SFLA)

SFLA contains elements of local search and global information exchange, an independent local search allows the transference of a meme among local individuals, and element of global information exchange enable frogs to interchange meme messages among memeplexes. And two elements alternate until convergence conditions are met.

2.1. Memeplex Division

The SFLA is described as follows. In the first step of algorithm, the initial population is formed by F randomly generated frogs X_i , each possible solution is presented as $X_i = (x_i^1, x_i^2, ..., x_i^N)$ where N is the number of the variables. The fitness value for the ^{i}th frog can be evaluated and sorted in descending order to find the global best frog (solution) P_g . Then the frogs are divided into m meme lexes each holding n frogs, obviously, it is that $F = m \times n$. Division process is that the first frog goes to the first memeplex, the second one goes to the second memeplex, the mth one goes to the mth memeplex, and the (m+1) th frog goes into the first memeplex. This process continues until all the frogs go into the appropriate memeplex.

2.2. Local Search

Influencing by the best frog, the worst frog leaps towards the best frog to attain more food. So the location of the worst frog should be adjusted and updated. According to the frog leaping rule, the worst frog is updated as follows.

$$D_k = r \times (P_{k \cdot b} - P_{k \cdot w}) \tag{1}$$

$$P_{k \cdot new\_w} = P_{k \cdot w} + D_k \text{ , } -D_{\max} \le D_i \le D_{\max} \tag{2}$$

where k is the kth memeplex with $1 \le k \le m$; r is a random number with uniform distribution between 0 and 1; $P_{k \cdot b}$ and $P_{k \cdot w}$ stand for the best frog position and the worst frog position; D_k denotes the leaping step size for the worst frog for the kth memeplex; $P_{k \cdot new\_w}$ represents a new position of the worst frog after updating and iteration, $D_{\max}$ is the maximum leaping step size.

After updating according to Equations. (1) and (2), if the fitness value of $P_{k \cdot new\_w}$ is better the value of $P_{k \cdot w}$, the $P_{k \cdot new\_w}$ replaces $P_{k \cdot w}$, otherwise, $P_{k \cdot b}$ is taken place of P_g to update newly. The location of frog is updated by the Eqs.(2) the leaping step size for the worst frog is updated as follows.

$$D_k = r \times (P_g - P_{kw}) \tag{3}$$

and after renewing the worst frog by formula (3) and (2), the $P_{k \cdot new\_w}$ is better than $P_{k \cdot w}$, it takes $P_{k \cdot new\_w}$ instead of $P_{k \cdot w}$, otherwise, the rule generates new frog instead of P_{kw} randomly by formula (4).

$$P_{k \cdot w} = r \times (O_{max} - O_{min}) + O_{\min} \tag{4}$$

where O_{max} and O_{min} represent the maximum and minimum search range.

The process continues repeatedly until a pre-specified criterion (e.g., the number of mutations) is satisfied. Then they exchange global information after local explorations of all memeplexes are finished.

2.3. Global Information Exchange

After local search with a predefined number evolutionary in each memplexes, the memeplex is sorted in descending order; repartition the frog into new memeplexes, then progress local search repeatedly until the convergence condition is met.

3. Two-Phases Learning Shuffled Frog Leaping Algorithm (TLSFLA)

In the local search of SFLA, the worst frog learns from the best frog in the same memeplex, the worst frog's possible position is substantially limited to the range between the current position and the best position [15], which restrains the search area of meme evolution, and decreases the convergence velocity; As the iteration increases, the worst frog become nearer to the best one in the same memeplex, it makes the algorithm to be premature, and the diversity becomes worse. In global information exchange, frogs of each memeplex are fixed and sorted in descending order, then algorithm repartitions memeplexex. However, the advantages of global optimal frog are not fully utilized, which influences the evolution effect. Based on the above reasons, the paper presents two-phases learning shuffled frog leaping algorithm using learning in the local search and in the global information exchange.

3.1. Local Search

The best frog of each memeplex has better evolution advantage in local search. If the worst frog can learn from the best frog of other memeplex in the evolution process, it can bring 3 benefits. The first is to increase the possibility of the worst frog towards to the global optimal point; the second is to enlarge the search scope of the worst frog according frog rule; finally, it can ensure the diversity of each memeplex in the evolution process. The location is updated by formula (2), the leaping step size of the worst frog is updated according to the following rule:

$$D_k = r_1 \times (P_{kb} - P_{kw}) + r_2 \times (P_{jb} - P_{kw}) \tag{5}$$

where $j \in 1, 2, 3, \cdots, m$ and $j \neq k$; r_1 and r_2 generate a random number between (0, 1).

After executing the updating strategies (5) and (2), if the fitness value of $P_{k \cdot new\_w}$ is better than that of $P_{k \cdot w}$, $P_{k \cdot new\_w}$ will replace $P_{k \cdot w}$, otherwise, the leaping step size of the worst frog is updated according to the following rule:

$$D_k = r_1 \times (P_g - P_{k \cdot w}) + r_2 \times (P_{j \cdot b} - P_{j \cdot w}) \tag{6}$$

After executing the updating strategies (6) and (2), if the fitness value of $P_{k \cdot new\_w}$ is better than that of $P_{k \cdot w}$, $P_{k \cdot new\_w}$ will replace $P_{k \cdot w}$, otherwise, the strategy generates a new population position instead of the previous $P_{k \cdot w}$ according to formula (4).

3.2. Global Information Exchange

Global optimal frog (*i.e.* elite frog) is different from other frogs. It cannot learn from other frogs, once fell into local optimum, algorithm would get into premature state. There is a new technology called Gaussian Learning which has been applied to a lot of optimization algorithm. In Gaussian learning, it add Gaussian perturbations to candidate solution, where it chooses the best solution as the generational individual. So we introduce Gaussian learning to the global optimal frog in global information exchange, which help to find better solution, lead frogs towards the better location and jump out the local optimum. The paper proposed elitist learning strategy, where we choose several dimensions in the location of global optimal frog randomly according to learning probability, a Gaussian random number is added in a dimension and information of other dimensions remains unchanged. The strategy gets a new optimal frog, if the fitness value is better than initial best value, the fitness value will replace the initial best value, and otherwise, it keeps the same. The definition of elitist learning strategy is given as follows.

Definition **(Elitist Learning Strategy) ---** Assume that is the global best location (*i.e.*, elite particle) in a D-dimensional space, then elitist learning strategy of the dth dimension $(1 \le d \le D)$ is computed as:

$$P_g^d = \begin{cases} P_g^d + (O_{max} - O_{min}) \times Gaussian(\mu, \sigma^2) & , \quad if\ (r \le p_r) \\ P_g^d & , \quad otherwise \end{cases} \tag{7}$$

where $Gaussian(\mu, \sigma^2)$ is Gaussian random number function in which $\mu = 0$ stands for mean value of Gaussian random number; σ^2 is variance, Gaussian learning probability p_r is constant; and r is a random number between 0 and 1.

Exploration and exploitation capability [16] are a pair of contradiction of swarm intelligence algorithm. Exploration is that particle deviates from the original optimization trajectory to the new direction for search; exploitation means particle continues to track the original optimization process for detailed search. As to balance the exploration and exploitation, Gaussian learning dynamically adjusts variance, σ^2 is large, the search space is also large with powerful global search in the evolution period. With the evolution into in-depth, σ^2 becomes smaller and smaller, the search space also becomes smaller, but the local exploitation ability becomes strong. Therefore, Variance adjustment strategy is defined as follows.

$$\sigma = \sigma_{max} - (\sigma_{max} - \sigma_{min}) \times \frac{g}{G} \tag{8}$$

where, σ_{max} and σ_{min} mean the maximum and minimum variance respectively; g is the number of global information exchange; and G stands for Pre-set the total number of global information exchange.

3.3. Algorithm Flow

The main steps of TLSFLA are as follows.

The main steps of EOLSFLA
1.Initialize parameters: F , m , n , G , I , p_r , P_g ,etc;
2.Generate population (represented by F frogs) randomly;
3.Evaluate fitness of each frog;
4.For $i = 1, \cdots, G$;
5.Sort F frogs in descending order;
6. Construct m memeplexes and submemeplexes;
7. Update P_g according to Eq. (7) and (8);

```
8.  For $j = 1,\cdots,I$
9.    For $k = 1,\cdots,m$
10.       Local search learning for every frog according to Eq.(2),(5) and(6);
11.     End;
12.   End;
13.   Shuffled all frogs;
14.End;
```

4. Experimental Verifications

4.1. Test Functions

To verify the performance our proposed approach, we employ a set of 13 benchmark functions from the literature [17]. The main characteristics of these benchmark functions are summarized in Table 1. They are high-dimensional functions, where functions f_1 to f_7 are unimodal functions, functions f_8 to f_{13} are multimodal functions. Most optimization algorithms fall into local optimal in the process of searching the best point.

Table 1. Benchmark Functions

Function symbol	Function	Test Function	Search space	Optimum coordinate	optimum
f_1	Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	$[-100,100]$	$(0,\cdots,0)$	0
f_2	Schwefel.2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]$	$(0,\cdots,0)$	0
f_3	Schwefel.1.2	$f(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$[-100,100]$	$(0,\cdots,0)$	0
f_4	Schwefel.2.21	$f(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	$[-100,100]$	$(0,\cdots,0)$	
f_5	Rosenbrock	$f(x)=\sum_{i=1}^{n}[100(x_{i+1}-x_i^2)^2+(1-x_i^2)^2$	$[-30,30]$	$(1,\cdots,1)$	0
f_6	Step	$f(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]$	$(0,\cdots,0)$	0
f_7	Quadric Noise	$f(x)=\sum_{i=1}^{n} i\cdot x_i^4+random[0.1]$	$[-1.28,1.28]$	$(0,\cdots,0)$	0
f_8	Schwefel	$f(x)=\sum_{i=1}^{n}-x_i*\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]$	$(-420.9687,\cdots,-420.9687)$	-12569.5
f_9	Rastrigin	$f(x)=\sum_{i=1}^{n}[x_i^2-10\cos 2\pi x_i+10]$	$[-5.12,5.12]$	$(0,\cdots,0)$	0
f_{10}	Ackley	$f(x)=-20\exp(-0.2\sqrt{1/n\sum_{i=1}^{n}x_i^2})-\exp(1/n\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	$[-32,32]$	$(0,\cdots,0)$	0
f_{11}	Griewank	$f(x)=1/4000\sum_{i=1}^{n}(x_i)^2-\prod_{i=1}^{n}\cos(x_i/\sqrt{i})+1$	$[-600,600]$	$(0,\cdots,0)$	0
f_{12}	Penalized. 1	$f(x)=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)\quad y_i=1+0.25(x_i+1)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a,\\ 0, & -a\le x_i\le a,\\ k(-x_i-a)^m, & x_i<-a.\end{cases}$	$[-50,50]$	$(1,\cdots,1)$	0

f_{13}	Penalized. 2	$f(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1+\sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1+\sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	$[-50, 50]$	$(1, \cdots, 1)$	0

4.2. Comparison of TLSFLA with Other Standard Intelligent Algorithms

In this section, we compare the performance of TLSFLA with other standard intelligent algorithms on the test suite. The involved algorithms and parameter settings are listed as follows.

(i) Particle Swarm Optimization (PSO)[18].
(ii) Shuffled Frog Leaping Algorithm (SFLA) [1].
(iii) Artificial Bee Colony (ABC) [19].
(iv) Intelligent Single Particle Optimizer (ISPO) [20].

In Experiment, the dimension of all test functions is 30, the iteration number is 2.0×10^5. For PSO, the population size is set to 200, the inertia weight w is set to 0.6, The acceleration coefficients c_1 and c_2 are set to 2. For ABC, the total population size is also 200, which is divided into 100 employed bees and 100 scouts; for the remaining parameters, please refer to [19]. For ISPO, the population size is only one, ISPO has many parameters, which has different results with different settings, so in order to calculate, this paper adopts the following settings: $a = 4000$, $b = 2$, $p = 40$, $l = 1$, $s = 4$, $\varepsilon = 1$; for SFLA, the population size is 200, the number of frogs is 200, there are 20 memeplexes, each containing 10 frogs, The local exploration in each submemeplex is executed for 10 iterations, the maximum leaping step size is 0.4 times of the maximum search area. The partition of memeplex and the evolution times in TLSFLA are set as the same of SFLA,and $p_r = 0.4$,$\sigma_{max} = 1.0$,$\sigma_{min} = 0.1$.

Table 2 presents' results of each test function over 50 runs randomly, where "Mean" represents the mean function fitness value and "Std.Dev" stands for the standard deviation by averaging over 100 independent experiments. To judge whether performance is different significantly between TLSFLA and other 4 algorithms, this paper employs 2-tailed t test to analyze, where significance level is set to 0.05, degree of freedom is set to 29. Table 2 shows the experimental results of 5 algorithms, where $w/t/l$ means that TLSFLA wins in w function, ties in t functions, and loses in l functions, compared with 4 standard intelligent algorithms. The best values are shown in bold.

From the results of Table 2, TLSFLA has better solution and stability. TLSFLA performs better than PSO on 3 problems, while both of them obtain the same results for the rest 10 problems; for the comparison of SFLA and TLSFLA, both of them achieve the same results on 7 problem, TLSFLA wins the rest 6 problems; TLSFLA outperforms ABC on 5 problems, while ABC achieves better results on f_5 , f_6 , f_{13} , for the rest 5 problems, both TLSFLA and ABC can find the global optimum; TLSFLA performs the same as ISPO on 6 problems, for the rest, TLSFLA wins 6, while ISPO wins only f_8 problem. Because there are many parameters of ISPO whose settings influence the solution, ISPO achieves the better solution on f_8 problem.

In order to compare the performance of 5 algorithms in the statistical sense, Friedman test is employed to analyze results. Table 3 presents the average ranking of PSO, SFLA, ABC, ISPO and TLSFLA on 13 benchmark functions. The smaller is value of ranking, the better the performance, the higher the rank. From the results in Table 3, the 5 algorithms can be sorted by average ranking into the following order:TLSFLA, ABC, PSO, SFLA, ISPO.

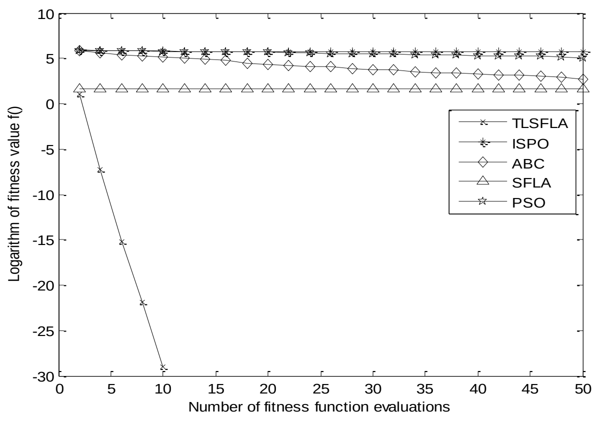
To illustrate changes of the function fitness value, this paper chooses 4 functions from 13 benchmark functions which include f_1 , f_9 , f_{10} and f_{11} . In fig.1, abscissa means function evaluations (FEs) from 0 to 5*10^4, ordinate shows logarithm of fitness value.

Table 2. The Comparison Results with EOLSFLA and 4 Intelligent Algorithms on 13 Benchmark Functions

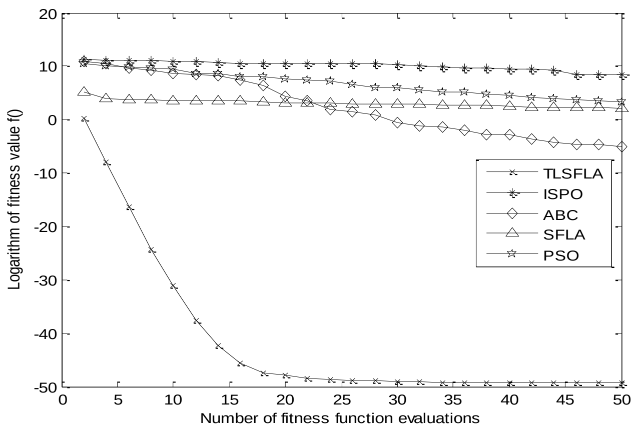
Function	Mean+ Std.Dev			
	Sphere	**Griewank**	**Ackley**	**Rastrgin**
[21]	7.5100e-018±7.5300e-018	1.1900e-002±1.29e-002	1.2700e-008±7.7500e-009	9.4521e+000±3.4243e+000
[22]	7.5000e-003±1.3000e-003	1.1100e-002±1.5200e-002	7.3800e-002±1.6400e-002	6.6463e+000±3.4642e+000
[23]	7.4950e-017±1.4838e-16	1.2107e-007±2.8321e-006	2.0263e-004±3.2010e-004	7.709e-002±1.1230e-002
[24]	7.5600e-011±2.2670e-010	7.4100e-004±1.4500e-002	2.8680e-008±4.9700e-002	3.1743e+001±6.3130e+000
[25]	2.9490e-003±7.1700e-004	8.5587e-002±8.4717e-002	4.0741e-002±1.0185e-002	9.7575 e+000±3.2458e+000
[26]	9.0000e-006±1.7000e-005	4.3305e-002±3.2415 e-002	2.606e-003±4.265e-003	9.3870e+000±3.1769e+000
[27]	5.1272e-024±8.8207e-024	**0.0000e+000±0.0000e+000**	5.1337e-014±6.0403e-014	**0.0000e+000±0.0000e+000**
Our approach	**9.7187e-031±1.3359e-029**	0.0000e+000±0.0000e+000	**5.8872e-016±0.0000e+000**	0.0000e+000±0.0000e+000

Table 3. Results of 5 Algorithms Using Friedman Test

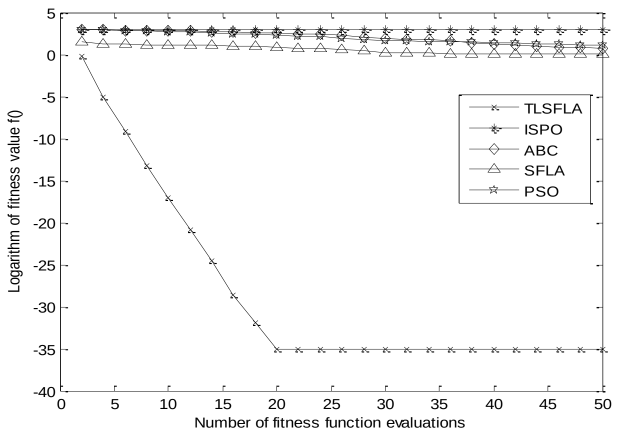
algorithm	ranking
TLSFLA	**1.54**
ABC	2.23
PSO	3.38
SFLA	3.54
ISPO	4.31



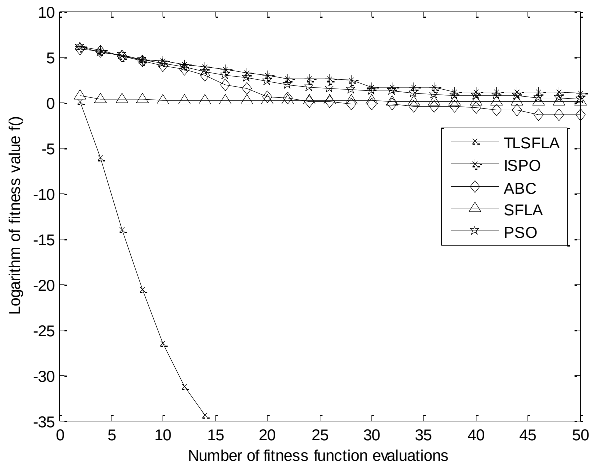
(a) f_1 Function



(b) f_9 Function



(c) f_{10} Function



(d) f_{11} Function

Figure 1. The Evolution Trend of f_1, f_9, f_{10} and f_{11}

From Figure 1, it can be seen that TLSFLA has not only strong Optimization capability, but also converges faster. Each algorithm can get better result after small iteration times, in Figure 1 (b) and (d), theoretical optimal solution 0 can be found after evolving to some value, but 0 cannot be in log, so the trend of TLSFLA is imperfect, all of other 4 algorithms fall into local optimal.

4.3. Comparison of TLSFLA with Improved SFLA Variants

As to verify high calculation precision of TLSFLA, Experiments are conducted to compare 8 SFLA variants on Sphere, Griewank, Ackley and Rastrgin functions in the Table 4.

Table 4. Comparison Results for 8 SFLA Variants

Function	Mean+ Std.Dev			
	Sphere	**Griewank**	**Ackley**	**Rastrgin**
[21]	7.5100e-018±7.5300e-[illegible]	1.1900e-002±1.29e-002	1.2700e-008±7.7500e-[illegible]	9.4521e+000±3.4243e+000
[22]	7.5000e-003±1.3000e-003	1.1100e-002±1.5200e-002	7.3800e-002±1.6400e-003	6.6463e+000±3.4642e+000
[23]	7.4950e-017±1.4838e-16	1.2107e-007±2.8321e-006	2.0263e-004±3.2010e-004	7.709e-002±1.1230e-002
[24]	7.5600e-011±2.2670e-010	7.4100e-004±1.4500e-002	2.8680e-008±4.9700e-003	3.1743e+001±6.3130e+000
[25]	2.9490e-003±7.1700e-004	8.5587e-002±8.4717e-002	4.0741e-002±1.0185e-003	9.7575 +000±3.3458 +000
[26]	9.0000e-006±1.7000e-005	4.3305e-002±3.2415 e-003	2.606e-003±4.265e-003	9.3870e+000±3.1769e+000
[27]	5.1272e-024±8.8207e-024	**0.0000e+000±0.0000e+000**	5.1337e-014±6.0403e-014	**0.0000e+000±0.0000e+000**
Our approach	**9.7187e-031±1.3359e-029**	**0.0000e+000±0.0000e+000**	**5.8872e-016±0.0000e+000**	**0.0000e+000±0.0000e+000**

Table 5. Results of 8 SFLA Variants Using Friedman Test

algorithm	**ranking**
Our approach	**1.25**
[27]	1.75
[23]	3.75
[21]	4.50
[24]	5.25
[26]	6.00
[22]	6.25
[25]	7.25

We can see from table 4 that TLSFLA surpasses other 7 SFLA variants on mean value and standard deviation. [27] and TLSFLA achieve the same theoretical optimum on multimodal functions Griewank and Rastrgin, however, [27] is obviously inferior to TLSFLA on Sphere and Ackley, and standard deviation on Ackley is 0 and the result is stable.

5. Conclusions

This paper presents two-phases learning shuffled frog leaping algorithm based on standard SFLA. TLSFLA employs elitist Gaussian Learning in global information exchange and improve the rule in local research. Experimental verifications on a set of constrained benchmark functions show that TLSFLA attains better performance than standard swarm intelligent algorithms and SFLA variants; it can promote the convergence velocity and global optimum.

However, from the above analyses, TLSFLA is not suitable for all kinds of function comparing with ABC and ISPO, for example, f_5, f_6, f_8 and f_{13}. It suggests that the presented approach might not be enough to prevent the search falling into local optimum. So our future study will concentrate on how to improve the efficiency of the proposed method [28, 29].

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant (No. 61261039, 61263029 and 61202464), Science Foundation of Jiangxi Province under Grant (No.20132BAB211031), Science and Technology Support Program of Jiangxi Province under Grant(No.20142BBG70034), Jiangxi Province Department of Education Science and Technology Project under Grant(No. GJJ13762 and GJJ13763)and Science and Technology Support Program of Nanchang City under Grant(No.2013HZCG011 and 2013HZCG006).

References

[1] M. M. Eusuff and K. E. Lansey, “Optimization of water distribution network design using the shuffled frog leaping algorithm”, Water Resources Planning and Management, vol. 3, **(2003)**, pp.210.
[2] P. Moscato, “On evolution, search, optimization, genetic algorithms and martial arts: Towards mimetic algorithms”, Caltech Concurrent Computation Program Report 826, California Institute of Technology **(1989)**, pp.16-20.
[3] J. Kennedy and R. C. Eberhart, “Particle swarm optimization”, Proceedings of IEEE International Conference on Neural Networks, **(1995)** December, pp. 1942-1948; Perth, Australia.
[4] L. Xuehui, Y. Ye and L. Xia, “Modified shuffled frog-leaping algorithm to solve traveling salesman problem”, Journal on Communications, vol.7, **(2009)**, pp.130.
[5] T. Niknam and E. A. Farsani, “A hybrid evolutionary algorithm for distribution feeder reconfiguration”, SCIENCE CHINA Technological Sciences, vol.4, **(2010)**, pp.950.
[6] A. Rahimi-Vahed and A. Hossein Mirzaei, “A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem”, Computers & Industrial Engineering, vol.53, **(2007)**, pp.642.
[7] X. Li, J. Luo, M. Chen and N. Wang, “An improved shuffled frog-leaping algorithm with extremely optimization for continuous optimization”, Information Sciences, vol.192, **(2012)**, pp.143.
[8] S. Hui and Z. Jia, “Application of Particle Sharing Based Particle Swarm Frog Leaping Hybrid Optimization Algorithm in Wireless Sensor Network Coverage Optimization”, Journal of Information and Computational Science, vol.14, **(2011)**, pp.3181.
[9] C. Gonggui, L. Hihuan and C. Jinfu, “SFL Algorithm Based Dynamic Optimal Power Flow in Wind Power Integrated System”, Automation of Electric Power Systems, vol.4, **(2009)**, pp.25.
[10] Z. Guangyu, “Meme triangular probability distribution shuffled frog-leaping algorithm”, Computer Integrated Manufacturing Systems, vol.10, **(2009)**, pp.1979.
[11] T. Niknam, B. B. Firouzi and H. D. Mojarrad, “A new evolutionary algorithm for non-linear economic dispatch”, Expert Systems with Applications, vol.38, **(2011)**, pp.13301.
[12] Z. Xiaodan, H. Feng, Z. Li and Z. Cairong, “Improved Shuffled Frog Leaping Algorithm Based on Molecular Dynamics Simulations”, Journal of Data Acquisition & Processing, vol.3, **(2012)**, pp. 327.
[13] L. Jian-ping and L. Xia, “Improved shuffled frog leaping algorithm for solving TSP”, Journal of Shenzhen University Science and Engineering, vol.2, **(2010)**, pp.173.
[14] J. Jianguo, L. Jin, L. Xiu-ping, S. Jielin and T. Mi, “A Shuffled frog leaping algorithm using niche technology”, Chinese Journal of Computational Mechanic, vol.6, **(2012)**, pp.960.
[15] C. Wen-hua, L. Xiaobing, W. Wei and W. Jiesheng, “Survey on shuffled frog leaping algorithm”, Control and Decision, vol.4, **(2012)**, pp.481.
[16] C. Yuhong, S. Fuchun, W. Weijun and Y. Chun, “An Improved Particle Swarm Optimization Algorithm with Search Space Zoomed Factor and Attractor”, Chinese Journal of Computers, vol.1, **(2011)**, pp.115.
[17] Y. Xin, L. Yong and L. Guangming, “Evolutionary programming made faster”, IEEE Trans Evolutionary Computation,vol.2, **(1999)**, pp.82.
[18] Y. Shi and R. C. Eberhar, “A modified particle swarm optimization”, Proceedings of the Computational Intelligence, **(1998)** May 69-73, Anchorage, USA.
[19] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm”, Journal of Global Optimization, vol. 3, **(2007)**, pp.459.

[20] J. Zhen, Z. Jiarui, L. Huilian and W. Qinghua, “A novel Intelligent Single Particle Optimizer”, Chinese Journal of Computers, vol. 3, **(2010)**, pp.556.
[21] Z. Ziyang, W. Daobo and L. Yuanyuan, “Improved Shuffled Frog Leaping Algorithm for Continuous Optimization Problem”, Proceedings of IEEE Congress on Evolutionary Computation, **(2009)** May, pp. 2992-2995; Trondheim, Norway.
[22] Z. Pengjun and L. Sanyan, “Shuffled frog leaping algorithm for solving complex functions”, Application Research of Computers, vol. 7, **(2009)**, pp.2435.
[23] S. Chittineni, A. N. S. Pradeep, G. Dinesh, S. C. Satapathy and P. V. G. D, and P. Reddy, “A Parallel Hybridization of Clonally Selection with Shuffled Frog Leaping Algorithm for Solving Global Optimization Problems (P-AISFLA)”, Lecture Notes in Computer Science, vol. 7077, **(2011)**, pp.211.
[24] H. Yichao, Q. Wenlong and X. Jiwei, “Improved shuffled frog leaping algorithm and its convergent analysis”, Computer Engineering and Applications, vol. 22, **(2011)**, pp. 37.
[25] L. Wenzhi, L. Shuai, X. Yang and T. Xiongyan, “Virtual Network Embedding based on Shuffled Frog Leaping Algorithm in TUNIE”, International Journal of Advancements in Computing Technology, vol. 11, **(2012)**, pp. 402.
[26] T. Zhong, N. Sheng and L. Zhi-sheng, “Research on the Shuffled Frog Leaping Algorithm Based on Local Orthogonal Crossover Operator”, International Journal of Advancements in Computing Technology, vol. 23, **(2012)**, pp. 800.
[27] Z. Xiaodan, Z. Li and Z. Cairong, “An improved shuffled frog leaping algorithm for solving constrained optimization problems”, Journal of Shandong University (Engineering Science), vol.1, **(2013)**, pp.1.
[28] L. Xu, X. Li and S. X. Yang, “Intelligent Information Processing and System Optimization”, Intelligent Automation and Soft Computing, vol. 7, **(2011)**, pp.829.
[29] X. Li-zhong, F. Xiao-liang and W. Cheng-lin, “Sequential Fusion Filtering for Networked Multi-Sensor Systems Based on Noise Estimation”, ACTA ELECTRONICA SINIC, vol. 1, **(2014)**, pp.160.

Authors

Jia Zhao, He received the M.A. degree in technology of computer application form Nanchang Hongkong University, Nanchang, China. He has been an assistant professor in the School of Information Engineering, Nanchang Institute of Technology, since 2011. His research interests include evolutionary computation, swarm intelligence, wireless sensor networks, images processing.

Li Lv, She received the M.A. degree in Computer Architecture form Jiangxi Normal University, Nanchang, China. She has been a assistant professor in the School of Information Engineering, Nanchang Institute of Technology, since 2013. Her research interests include evolutionary computation, wireless sensor networks, embedded system.