Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9676279, 8 pages
https://doi.org/10.1155/2020/9676279

Research Article

Fruit Fly Optimization Algorithm Based on Single-Gene Mutation for High-Dimensional Unconstrained Optimization Problems

Xiao-dong Guo,[1] **Xue-liang Zhang**,[1] **and Li-fang Wang**[2]

[1]*School of Mechanical Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China*
[2]*School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China*

Correspondence should be addressed to Li-fang Wang; wanglifang@tyust.edu.cn

Received 6 August 2020; Revised 19 October 2020; Accepted 1 November 2020; Published 17 November 2020

Academic Editor: S. A. Edalatpanah

The fruit fly optimization (FFO) algorithm is a new swarm intelligence optimization algorithm. In this study, an adaptive FFO algorithm based on single-gene mutation, named AFFOSM, is designed to aim at inefficiency under all-gene mutation mode when solving the high-dimensional optimization problems. The use of a few adaptive strategies is core to the AFFOSM algorithm, including any given population size, mutation modes chosen by a predefined probability, and variation extents changed with the optimization progress. At first, an offspring individual is reproduced from historical best fruit fly individual, namely, elite reproduction mechanism. And then either uniform mutation or Gauss mutation happens by a predefined probability in a randomly selected gene. Variation extent is dynamically changed with the optimization progress. The simulation results show that AFFOSM algorithm has a better accuracy of convergence and capability of global search than the ESSMER algorithm and several improved versions of the FFO algorithm.

1. Introduction

In the recent years, swarm intelligence has become a research focus of optimization design field because of its special advantages, such as simple to operate, quick convergence rate, and powerful ability of global search. New swarm intelligence technologies, such as genetic algorithms (GAs) [1], ant colony optimization (ACO) [2], particle swarm optimization (PSO) [3], artificial fish-swarm optimization (AFSO) [4], artificial bee colony algorithm (ABC) [5], firefly algorithm (FA) [6], FFO [7, 8], biogeography-based optimization (BBO) [9], whale optimization algorithm (WOA) [10], butterfly optimization algorithm (BOA) [11], gaining sharing knowledge-based algorithm (GSK) [12], and their respective improved versions are emerging in endlessly and have been widely applied to the fields of optimization design [13–20].

Fruit fly is an insect which eats plants that are decaying, especially fruits. Fruit flies acquire chemical information in their environment through smell and taste receptors on the surface of their bodies and then regulate behaviors, such as foraging, aggregation, mating, and spawning. In these processes, olfactory plays an important role over long distances and shorter ranges.

FFO algorithm [7, 8] is proposed by Pan in 2011. FFO simulates the foraging behavior of fruit flies and is a fast, structure-simple, and easily realized algorithm. Therefore, it already attracted broad attention in recent years [21–44] and has been successfully applied to a wide range of practical problems [45–52].

Fruit flies have olfactory and visual abilities superior to other species. When foraging, fruit flies first use their own olfactory organs to smell the odor from food source and exchange odor information to the surrounding fruit flies, a process known as the olfactory foraging phase. Then, the flies used their visual organs to find and fly to the locations of the flies that had gathered the best odor information, a process named the vision foraging phase.

A series of studies show that some unreasonable algorithmic design makes FFO algorithm ill-equipped to jump out of local extremum and to handle complex, high-dimensional, and nonlinear problems. So with this as the starting point of the paper, a small population, adaptive and improved version of the FFO algorithm, named AFFOSM, is developed based on the single-gene mutation mode, in which the only one gene of an offspring is different from the elite individual.

On the contrary to a single-gene mutation mode, all-gene mutation mode is adopted by most of optimization algorithms, such as FFO, PSO, and ABC algorithm, in which each gene of an offspring is different from the elite or parent individual.

The efficiency of the AFFOSM algorithm presented in this research is evaluated by solving 6 test problems. Optimization results demonstrate that AFFOSM is very competitive compared to the state-of-the-art single-gene optimization methods.

The rest of the paper is structured as follows. Section 2 describes the related research work about the further analysis and modification to FFO Algorithm. Section 3 describes the developments of FFO algorithm, from all-gene mutation to single-gene mutation. AFFOSM algorithm developed in this study will be covered in detail in Section 4. Test problems and optimization results are presented and discussed in Sections 5. Section 6 summarizes the main findings of this study and suggests directions for future.

2. Related Work

It is worth noting that, in the FFO algorithm, fruit fly individual is represented in its coordinates in a 2D plane, and the corresponding variable value is calculated as the reciprocal of the Euclidean distance between individual and ordinate origin, as illustrated in Figure 1.

Another noteworthy thing about the FFO algorithm is elitist reproduction strategy. Once a historical best solution is found, fruit fly individuals will fly to it and look for the food resources around it before a newly best solution is found.

There are also some disadvantages as follows:

(1) The ability to solve the problem whose theoretical optimal solution is negative is not available.

(2) It is difficult and time-consuming for population initialization when definition domain is far away from ordinate origin.

(3) Obviously, it is not good choice that search range is fixed, compared to dynamic one.

(4) Most of searching behavior happens around ordinate origin due to nonuniform search in definition domain. FFO algorithm is workable to deal with a class of problems, such as quite a lot of test function whose theoretical optimal solution is very close to zero and is poor for most of the optimization problems in practical projects.

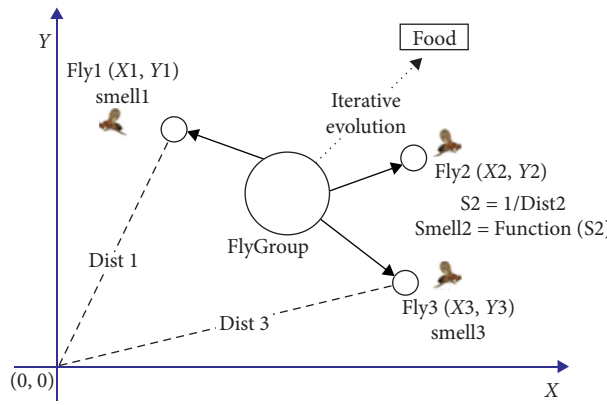

Figure 1: Foraging process of fruit fly.

(5) Elitist reproduction strategy could make fruit fly swarm easy to fall into local extremum and not capable to solve complex, high-dimensional and nonlinear problems.

Given the abovementioned facts, many improvements have been made in recent years.

Fu-qiang Xu and Tao [53] presented the G-FFO algorithm with sign processing in a random manner. Inspired by probability estimation for code words in adaptive arithmetic coding, a FFO algorithm with adaptive sign processing (FFOASP) is proposed [54].

Wu Lei et al. propose SEDI-FFO algorithm in which more fruit flies would fly in the search direction that was best for finding the optimal solution or at least in a direction close to the optimal direction [34].

Based on hybrid location information exchange mechanism, HFFO algorithm is proposed that enables flies to communicate with each other and conduct local search in a swarm-based approach [36].

Fan et al. propose WFFO algorithm in which an effective whale-inspired hunting strategy is introduced to replace the random search plan of the original FFO [38].

Niu et al. propose an improved FFO algorithm based on differential evolution (DFFO) by modifying the expression of the smell concentration judgment value and by introducing a differential vector to replace the stochastic search [52].

CEFFO algorithm is proposed in which trend search is applied to enhance the local searching capability of fruit fly swarm, and coevolution mechanism is employed to avoid the premature convergence and improve the ability of global searching [40].

SCA_FFO algorithm [41] is developed by introducing the logic of the sine-cosine algorithm. The fruit fly individual adopts the way to fly outward or inward to find the global optimum.

3. FFO: From All-Gene to Single-Gene Mutation

In general, the nonconstrained optimization problem can be formulated as an n-dimensional minimization problem as follows:

$$\begin{aligned} &\min f(x), \\ &\text{s.t.} \quad x = (x_1, x_2, \ldots, x_n), \\ &\qquad x_j \in \left(l_j,\ u_j\right), \quad j = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where f is a boundary objective function, $x = (x_1, x_2, \ldots, x_n)$ is the set of decision variables, n is the dimensionality, and l_j and u_j are the lower and upper bounds of the decision variable x_j, respectively.

3.1. FFO Algorithm. In FFO, fruit fly individual is represented in its coordinates in a plane and generated in uniform mutation around historically best solution, also called current population location:

$$\begin{aligned} X_i &= X\_\text{axis} \pm \sigma \times \text{rand}(), \\ Y_i &= Y\_\text{axis} \pm \sigma \times \text{rand}(), \end{aligned} \tag{2}$$

where (X_i, Y_i), $(X_\text{axis}, Y_\text{axis})$is the coordinate pair of current individual x_i and current population δ,σ is the amplitude of uniform mutation, and rand is the uniformly distributed random numbers between 0 and 1. The value of x_i is the reciprocal of the Euclidean distance between fruit fly individual and ordinate origin:

$$x_i = \frac{1}{\sqrt{X_i^2 + Y_i^2}}. \tag{3}$$

It can be found that the mechanism of individual representing limits the performance of FFO.

3.2. LGMS-FFO Algorithm. Shan et al. [55] use one-dimensional coordinate of fruit fly to denote the individual location, and then, let it be equal to the value of x_i, which can be formulated as follows:

$$\begin{aligned} x_i &= X_i, \\ \delta &= X\_\text{axis}. \end{aligned} \tag{4}$$

Based on a new linear generation mechanism of candidate solution, LGMS-FFO algorithm is proposed:

$$\begin{aligned} \omega &= \omega_0 \times \alpha^{\text{Iter}}, \\ x_i &= \delta \pm \omega \times [l + (u - l) \times \text{rand}()], \end{aligned} \tag{5}$$

where ω is a weight factor to tune variation extent, ω_0 is the initial weight, α is the weight coefficient, and Iter is the current generation.

Obviously, all-gene mutation mode is used to generate offspring individuals in FFO and LGMS-FFO algorithms. However, the higher dimensionality the functions to be optimized are, the lower the probability of excellent individuals is. This in turn has caused a low convergence rate for solving high-dimensional functions.

3.3. IFFO Algorithm. Different from FFO and LGMS-FFO, single-gene mutation mode is introduced that only one gene is selected randomly to mutate in the IFFO algorithm. It is demonstrated that the single-gene mutation mode is a better choice in performance than the all-gene mutation mode for solving high-dimensional functions.

IFFO algorithm is presented in which a control parameter σ is used to tune self-adaptively the search scope in a random direction of current swarm location, and offspring individuals are generated in the single-gene uniform mutation mode [28]:

$$\begin{aligned} &\sigma \longleftarrow \sigma_{\max} \times \exp\left(\log\left(\frac{\sigma_{\min}}{\sigma_{\max}}\right) \times \frac{\text{Iter}}{\text{Iter}_{\max}}\right), \\ &\begin{cases} x_{i,d} \longleftarrow \delta_d \pm \sigma \times \text{rand}(), \\ x_{i,j} \longleftarrow \delta_j, \quad i = 1, 2, \ldots, k,\ j = 1, 2, \ldots, n,\ j \neq d, \end{cases} \end{aligned} \tag{6}$$

where $\sigma_{\max}$ is the maximum radius, $\sigma_{\min}$ is the minimum radius, and d is a random integer between 1 and n.

4. AFFOSM Algorithm

ESSMER algorithm, an improved evolutionary strategy with single-gene mutation and elite reproduction, is presented for solving high-dimensional function [56]. In the ESSMER algorithm, the best father individual is chosen to generate $\lambda + k$ offsprings, λ by Gauss mutation, and k by uniform mutation:

$$\begin{aligned} &\begin{cases} x_{i,d} = \delta_d + \sigma_t N(0, 1), \\ x_{i,j} = \delta_j, \quad i = 1, 2, \ldots, \lambda,\ j = 1, 2, \ldots, n,\ j \neq d, \end{cases} \\ &\begin{cases} x_{i,d} = \delta_d + l_d + (u_d - l_d)\text{rand}(), \\ x_{i,j} = \delta_j, \quad i = 1, 2, \ldots, k,\ j = 1, 2, \ldots, n,\ j \neq d, \end{cases} \end{aligned} \tag{7}$$

where d is a random integer between 1 and n. σ_t is a variation extent related to the optimization process, and its initial value σ_0 is equal to 2. σ_t is reduced by a quarter once the stagnant iterations (recorded as flag) reach a default iteration.

Inspired by ESSMER, AFFOSM algorithm, an adaptive FFO algorithm based on single-gene mutation, is designed in this article. At first, an offspring individual is reproduced from historical best father individual. And a randomly selected gene is modified by either uniform mutation or Gauss mutation occurring by a predefined probability. Compared with ESSMER, the initial variation extents are entirely dependent on the problems to be optimized. The initial amplitude of uniform variation is equal to the width of the definition domain, a very broad range. The initial amplitude of Gauss mutation is equal to a tenth of the definition domain, a relatively little range.

In the process of optimization, the amplitude σ_f of uniform variation is cyclically adjusted. At the beginning of each iteration, σ_f is reduced by a quarter. σ_f will be reverted to the initial value until a better solution is achieved or a predefined accuracy is reached.

The amplitude σ_s of Gauss mutation is dynamically changed with the optimization progress. Those continuous

Table 1: Test function.

Function	Expression	Definition domain	$f(x^*)$
$f1$	$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	$[-30, 30]$	0
$f2$	$f(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0
$f3$	$f(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp((1/n)\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$[-32, 32]$	0
$f4$	$f(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	$[-10, 10]$	0
$f5$	$f(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$	$[-600, 600]$	0
$f6$	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]$	0

Functions $f1$ and $f2$ are unimodal. Function $f1$ is the Rosenbrock function, also referred to as the valley or banana function. Its global minimum lies in a narrow, parabolic valley. Function $f2$ is the sphere function, also referred to as the harmonic function with the only global minimum. Functions $f3$–$f6$ are, respectively, Ackley, Alpine, Griewank, and Rastrigin function. They are multimodal, and each of them has a large number of local minima and is difficult to be optimized.

stagnant iterations are recorded by a variable namedflag. The amplitude σ_s remains unchanged as long as flag is less than a predefined value, such as 30. Otherwise, the amplitude σ_s is reduced successively by a quarter unless flag is not less than 30. Meanwhile, a variable, written as σ_{temp}, is used to save and restore the new amplitude σ_s once a better solution is found.

AFFOSM algorithm mainly includes 3 steps as follows:

Step 1. Initialization.

Set population size N and maximum iteration $\text{iter}_{\max}$. Set $\sigma_{\max} \longleftarrow u - l$, $\sigma_{\min} \longleftarrow 10^{-7}$, $\sigma_f \longleftarrow \sigma_{\max}$, and $\sigma_s \longleftarrow \sigma_{\max}/10$.

Step 2. Osphresis foraging phase.

Update the mutation amplitudes.

Before a new iteration, the mutation amplitudes are updated as follows:

$\sigma_f \longleftarrow \sigma_f \times 0.75$

if $\sigma_f < \sigma_{\min}$, then $\sigma_f \longleftarrow \sigma_{\max}$

if flag = 30, then $\sigma_s \longleftarrow \sigma_s \times 0.75$, $\sigma_{\text{temp}} \longleftarrow \sigma_s$, flag $\longleftarrow 0$

Where the variable σ_{temp} is used to guarantee the continuity of the Gauss mutation amplitude.

Generate new solutions

For each individual, the uniform mutation is executed with probability 0.2:

$$\begin{cases} x_{i,d} = \delta_d \pm \sigma_f \text{rand}(), \\ x_{i,j} = \delta_j, \quad j = 1, 2, \ldots, n,\ j \neq d. \end{cases} \tag{8}$$

The Gauss mutation is executed with probability 0.8:

$$\begin{cases} x_{i,d} = \delta_d + \sigma_s N(0, 1), \\ x_{i,j} = \delta_j, \quad j = 1, 2, \ldots, n,\ j \neq d, \end{cases} \tag{9}$$

where d is a random integer between 1 and n.

Step 3. Vision foraging phase.

Evaluate each new solution. If a better solution is discovered, then update δ and set flag, σ_f, and σ_s to be 0, $\sigma_{\max}$, and σ_{temp}. Otherwise, let flag equal to flag + 1 and return Step 2.

Table 2: The performance on test functions with $n = 100$ and 200.

Function		ESSMER	IFFO	SFFO	AFFOSM
f1	$n = 100$	1.49E + 02 /1.04E + 02	1.91E + 02 /4.08E + 02	1.68E + 02 /6.20E + 02	1.25E + 02 /5.02E + 01
	$n = 200$	4.31E + 02 /4.02E + 02	4.46E + 02 /4.38E + 02	4.59E + 02 /7.01E + 02	3.31E + 02 /3.16E + 02
f2	$n = 100$	3.98E − 14 /4.63E − 14	3.97E − 16 /9.66E − 17	3.66E − 16 /1.12E − 16	7.43E − 43 /6.64E − 42
	$n = 200$	1.34E − 07 /1.69E − 07	3.25E − 15 /6.09E − 16	3.01E − 15 /4.12E − 16	1.15E − 22 /1.61E − 22
f3	$n = 100$	5.27E − 08 /5.15E − 08	7.51E − 09 /7.90E − 10	7.74E − 09 /9.77E − 10	5.37E − 13 /1.91E − 13
	$n = 200$	6.13E − 05 /2.56E − 05	1.47E − 08 /1.17E − 09	1.53E − 08 /1.21E − 09	1.34E − 12 /1.08E − 12
f4	$n = 100$	1.32E − 07 /1.37E − 07	1.16E − 07 /5.39E − 08	1.37E − 07 /4.56E − 08	4.10E − 14 /8.40E − 15
	$n = 200$	4.53E − 04 /1.71E − 04	6.63E − 07 /1.51E − 07	6.73E − 07 /1.23E − 07	3.84E − 12 /2.27E − 12
f5	$n = 100$	2.14E − 01 /1.45E − 01	9.90E − 03 /2.41E − 02	1.23E − 02 /2.21E − 02	7.40E − 03 /2.51E − 02
	$n = 200$	2.95E − 01 /1.76E − 01	9.90E − 03 /2.72E − 02	1.23E − 02 /1.81E − 02	9.90E − 03 /1.22E − 02
f6	$n = 100$	1.26E − 12 /1.28E − 02	9.95E − 01 /7.88E − 01	9.95E − 01 /7.86E − 01	2.01E − 07 /4.96E − 01
	$n = 200$	3.76E − 06 /7.57E − 01	1.69E + 01 /5.38E + 00	1.89E + 01 /3.77E + 00	2.00E + 00 /1.49E + 00
R	$n = 100$	3.17	2.667	2.833	1.333
	$n = 200$	3.264	2.556	2.819	1.361

The computational complexity of AFFOSM is related to the size of populations N and the number of iterations $\text{iter}_{\max}$. The computational complexity of one fruit fly at each iteration is one, and then, the computational complexity of AFFOSM can be summarized as $O(N \times \text{iter}_{\max})$, which is the same as original FFO, IFFO, and ESSMER.

5. Test Functions and Results Analysis

To verify the proposed AFFOSM algorithm, a total of 6 benchmark problems with different characteristics are listed in Table 1 where n denotes the dimensionality of the functions and $f(x^*)$ is the global optimal.

As another improved version of FFO algorithm, SFFO [57] adjusts adaptively its search along an appropriate decision variable from its previous experience in generating promising solutions.

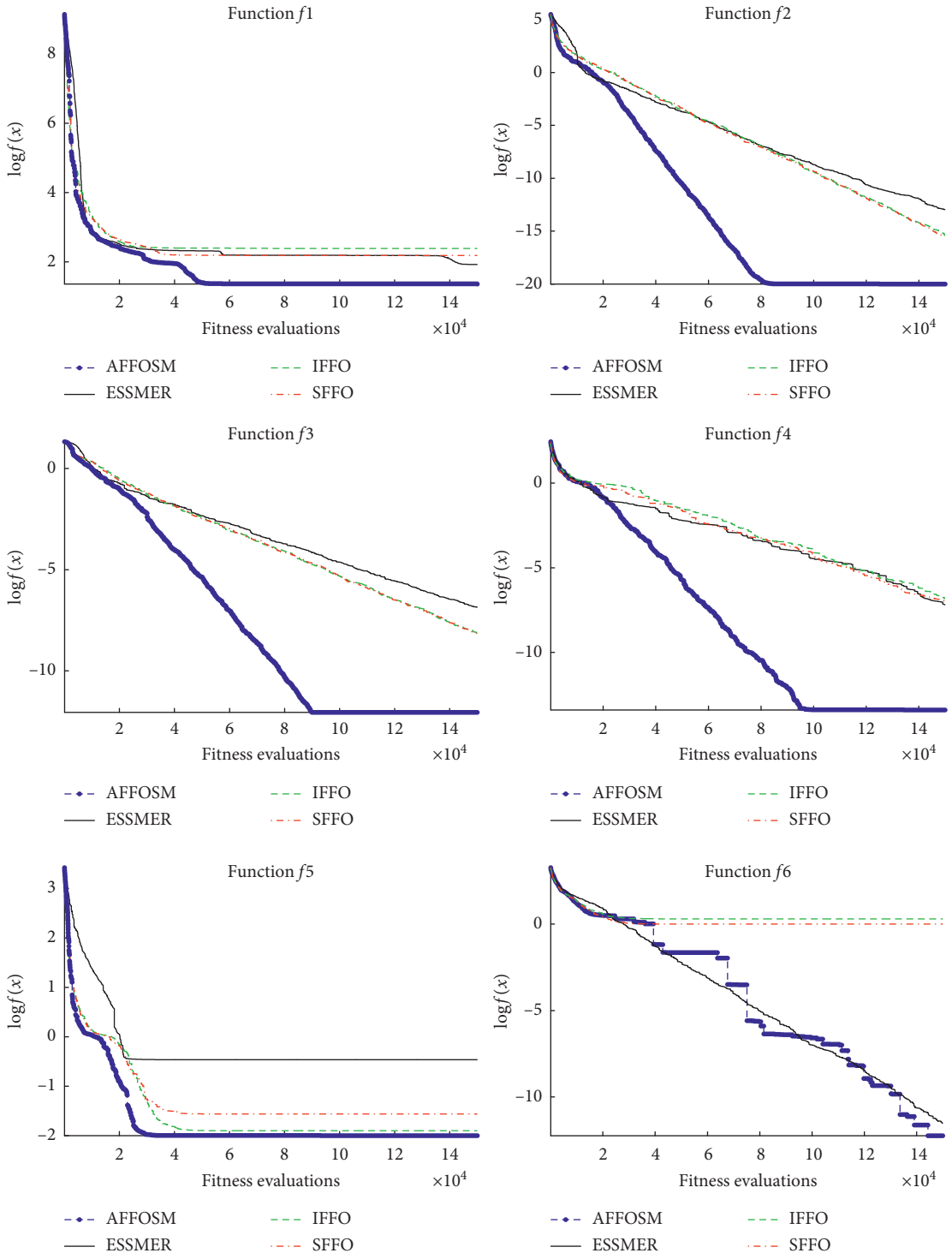

Figure 2: Performance comparison on test functions on $n = 100$.

In the ESSMER algorithm, population size $N = 10, \lambda = 8, k = 2$, and $\sigma_0 = 2$. For IFFO and SFFO algorithms, $N = 3$ and $\sigma_{\max} = (u - l)/2$. For the AFFOSM algorithm, $N = 3$, $\sigma_{\max} = u - l$, and $\sigma_{\min} = 10^{-7}$ for all algorithms.

These algorithms are coded on Matlab 7.1 and run on Windows 10 operating system and Intel(R) Core(TM) i7-6600U CPU @ 2.60 GHz 2.81 GHz with 8G RAM.

Each problem is run 25 times independently. The median/standard deviations over these 25 runs and the average rank of four algorithms on Friedman's test [58] are reported in Table 2. Comparisons of convergence curves on $n = 100$ are also presented in Figure 2.

Comparing with the ESSMER algorithm, the AFFOSM algorithm performs better for functions $f1$–$f5$ with both

$n = 100$ and $n = 200$. AFFOSM algorithm produces a smaller median; the 0, 29, 5, 7, and 2 orders of magnitude reduced with $n = 100$ and the 0, 15, 7, 8, and 2 orders with $n = 200$, respectively.

Comparing with the IFFO algorithm on $n = 100$, the AFFOSM algorithm performs better for all test functions. The smaller median is gained with the 0, 27, 4, 7, 0, and 6 orders of magnitude reduced. Comparing with the IFFO algorithm on $n = 200$, the AFFOSM algorithm performs better for functions $f1$, $f2$, $f3$, and $f4$. The smaller median is gained with the 0, 7, 4, and 5 orders of magnitude reduced. Meanwhile, the same median value is gained for function $f5$.

Comparing with the SFFO algorithm on $n = 100$, the AFFOSM algorithm performs better for all test functions. The smaller median is gained with the 0, 27, 4, 7, 1, and 6 orders of magnitude reduced.

Comparing with the IFFO algorithm on $n = 200$, the AFFOSM algorithm performs better for functions $f1$, $f2$, $f3$, $f4$, and $f5$. The smaller median is gained with the 0, 7, 4, 5, and 1 orders of magnitude reduced.

For test functions $f1$, $f2$, $f3$, $f4$, and $f5$, AFFOSM performs best on both $n = 100$ and $n = 200$. ESSMER algorithm, meanwhile, performs the best only for function f6 on both $n = 100$ and $n = 200$. The IFFO algorithm performs the best only in the case of function $f5$ on $n = 200$. Unfortunately, the SFFO algorithm is the best one in no case.

Similarly, the average rank R on Friedman's test shows the proposed AFFOSM scheme is superior to other ones based on single-gene mutation.

6. Conclusion and Future Work

For high-dimensional problems, the algorithms based on single-gene mutation have better performance on convergency and accuracy than those based on all-gene mutation. To overcome the shortage of the FFO to solve the high-dimensional optimization problems, single-gene mutation and adaptive mutation range control technique are introduced into the AFFOSM algorithm proposed in this article. In the AFFOSM algorithm, the main function of Gauss mutation is to search locally around the historic best solution. Uniform mutation not only plays a role to search globally in the whole space but also improves the efficiency of local search when uniform mutation range is significantly smaller than the Gauss mutation range. Unrelated to iterations, mutation range is adjusted by the reference of the optimization progression. Simulations show that the AFFOSM algorithm is better than those based on single-gene mutation, such as ESSMER, IFFO, and SFFO.

An interesting topic for future research would be investigating the real impact of some parameters values in AFFOSM to analyze their contributions to the algorithm performance.

Another interesting topic for future research would be applying the AFFOSM algorithm to solve constraint or multiobjective optimization problems and practical engineering problems such as benchmark structural optimization problems.

Data Availability

The data and code used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61003053).

References

[1] J. J. Grefenstette, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2, pp. 95–99, 1988.

[2] E. Bonabeau and H. M. Botee, "Evolving ant colony optimization," *Advances in Complex Systems*, vol. 1998, no. 1, pp. 149–159, 1998.

[3] Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation*, IEEE, Seoul, South Korea, August 2002.

[4] X. L. Li, "An optimizing method based on autonomous animats: fish-swarm algorithm," *Systems Engineering-Theory & Practice*, vol. 22, no. 11, pp. 32–38, 2002.

[5] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[6] X. S. Yang, "Firefly algorithm: levy flights and global optimization," 2010, https://arxiv.org/abs/1003.1464.

[7] W. T. Pan, *Fruit Fly Optimization Algorithm*, Tsang Hai publishing, Taibei, China, 2011.

[8] W. T. Pan, "A new evolutionary computation approach: fruit fly optimization algorithm," in *Proceedings of the Conference of Digital Technology and Innovation Management*, pp. 382–391, Taipei, Taiwan, November 2011.

[9] H. Garg and Harish, "An efficient biogeography based optimization algorithm for solving reliability optimization problems," *Swarm and Evolutionary Computation*, vol. 24, pp. 1–10, 2015.

[10] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[11] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing*, vol. 23, pp. 715–734, 2018.

[12] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 7, pp. 1501–1529, 2020.

[13] X. J. Bi and Y. J. Wang, "A modified artificial bee colony algorithm and its application," *Journal of Harbin Engineering University*, vol. 33, no. 1, pp. 117–123, 2012.

[14] H. Garg and S. P. Sharma, "Multi-objective reliability-redundancy allocation problem using particle swarm optimization," *Computers & Industrial Engineering*, vol. 64, no. 1, pp. 247–255, 2013.

[15] G. G. Wang, L. Guo, A. H. Gandomi, G. S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.
[16] H. Garg and Harish, "A hybrid PSO-GA algorithm for constrained optimization problems," *Applied Mathematics and Computation*, vol. 274, pp. 292–305, 2016.
[17] G. G. Wang and Y. Tan, "Improving metaheuristic algorithms with information feedback models," *IEEE Transactions on Cybernetics*, vol. 49, no. 2, pp. 542–555, 2019.
[18] R. S. Patwal, N. Narang, and H. Garg, "A novel TVAC-PSO based mutation strategies algorithm for generation scheduling of pumped storage hydrothermal system incorporating solar units," *Energy*, vol. 142, no. 1, pp. 822–837, 2018.
[19] A. W. Mohamed, A. A. Hadi, and K. M. Jambi, "Novel mutation strategy for enhancing shade and lshade algorithms for global numerical optimization," *Swarm and Evolutionary Computation*, vol. 50, Article ID 100455, 2019.
[20] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," in *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 145–152, San Sebastian, Spain, June 2017.
[21] C. Li, S. Xu, W. Li, and L. Hu, "A novel modified fly optimization algorithm for designing the self-tuning proportional integral derivative controller," *Journal of Convergence Information Technology*, vol. 7, no. 16, pp. 69–77, 2012.
[22] L. Wang, X. L. Zhang, and S. Y. Wang, "A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem," *Knowledge-Based Systems*, vol. 48, pp. 17–23, 2013.
[23] J. Y. Han, C. Z. Liu, and L. G. Wang, "Dynamic double subgroups cooperative fruit fly optimization algorithm," *Pattern Recognition & Artificial Intelligence*, vol. 26, no. 11, pp. 1057–1067, 2013.
[24] J. Y. Han and C. Z. Liu, "Efficient fruit fly optimization algorithm with reverse cognition," *Computer Engineering*, vol. 11, pp. 223–225, 2013.
[25] J. Y. Han and C. Z. Liu, "Fruit fly optimization algorithm based on history cognition," *Journal of Frontiers of Computer Science & Technology*, vol. 8, no. 3, pp. 368–375, 2014.
[26] X. Yuan, X. Dai, J. Zhao, and Q. He, "On a novel multi-swarm fruit fly optimization algorithm and its application," *Applied Mathematics and Computation*, vol. 233, no. 3, pp. 260–271, 2014.
[27] X. L. Zheng, L. Wang, and S. Y. Wang, "A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem," *Knowledge-Based Systems*, vol. 57, pp. 95–103, 2014.
[28] Q. K. Pan, H. Y. Sang, J. H. Duan, and L. Gao, "An improved fruit fly optimization algorithm for continuous function optimization problems," *Knowledge-Based Systems*, vol. 62, pp. 69–83, 2014.
[29] C. Z. Liu and J. Y. Han, "Adaptive fruit fly optimization algorithm based on bacterial migration," *Computer Engineering &Science*, vol. 36, no. 4, pp. 690–696, 2014.
[30] J. Ning, B. Wang, H. Li, and B. Xu, "Research on and application of diminishing step fruit fly optimization algorithm," *Journal of Shenzhen University Science and Engineering*, vol. 31, no. 4, pp. 367–373, 2014.
[31] Z. X. Liu, Y. F. Wang, and Y. Zhang, "Multiple population fruit fly optimization algorithm for automatic warehouse order picking operation scheduling problem," *Journal of Wuhan University of Technology*, vol. 36, no. 3, pp. 71–75, 2014.
[32] X. D. Guo, L. F. Wang, and X. L. Zhang, "Fruit fly optimization algorithm based on adaptive step size," *Journal of North University of China*, vol. 9, no. 1, 2016.
[33] W. Wang and X. Liu, "Melt index prediction by least squares support vector machines with an adaptive mutation fruit fly optimization algorithm," *Chemometrics and Intelligent Laboratory Systems*, vol. 141, pp. 79–87, 2015.
[34] L. Wu, W. Xiao, and L. Zhang, "An improved fruit fly optimization algorithm based on selecting evolutionary direction intelligently," *International Journal of Computational Intelligence Systems*, vol. 9, no. 1, pp. 80–90, 2016.
[35] Y. Wang and L. Feng, "Novel double subgroups and partition sampling based fruit fly optimization algorithm," *Journal of ZheJiang University (Engineering Science)*, vol. 51, pp. 2292–2298, 2017.
[36] S. X. Lv, Y. R. Zeng, and L. Wang, "An effective fruit fly optimization algorithm with hybrid information exchange and its applications," *International Journal of Machine Learning & Cybernetics*, vol. 9, no. 10, pp. 1623–1648, 2017.
[37] T. S. Du, X. T. Ke, J. G. Liao, and Y. J. Shen, "DSLC-FOA: improved fruit fly optimization algorithm for application to structural engineering design optimization problems," *Applied Mathematical Modelling*, vol. 55, pp. 314–339, 2018.
[38] Y. Fan, P. Wang, A. A. Heidari et al., "Boosted hunting-based fruit fly optimization and advances in real-world problems," *Expert Systems with Applications*, vol. 159, Article ID 113502, 2020.
[39] W. Zhong, J. Niu, Y. Liang, X. Kong, and F. Qian, "Multi-strategy fruit fly optimization algorithm and its application," *Ciesc Journal*, vol. 66, no. 12, pp. 4888–4894, 2015.
[40] X. M. Han, Q. M. Liu, H. Z. Wang, and L. M. Wang, "Novel fruit fly optimization algorithm with trend search and co-evolution," *Knowledge Based Systems*, vol. 141, pp. 1–17, 2018.
[41] Y. Fan, P. Wang, A. A. Heidari, M. Wang, and C. Li, "Rationalized fruit fly optimization with sine cosine algorithm: a comprehensive analysis," *Expert Systems with Applications*, vol. 157, Article ID 113486, 2020.
[42] L. Wang, S. X. Lv, and Y. R. Zeng, "Literature survey of fruit fly optimization algorithm," *Kongzhi Yu Juece/Control and Decision*, vol. 32, no. 7, pp. 1153–1162, 2017.
[43] H. Han, "Analysis on fruit fly optimization algorithm," *Computer Systems & Applications*, vol. 22, pp. 783–791, 2017.
[44] L. Wang and X. L. Zheng, "Advances in fruit fly optimization algorithms: control theory & applications," 2017.
[45] W. T. Pan, "Using fruit fly optimization algorithm optimized general regression neural network to construct the operating performance of enterprises model," *Journal of Taiyuan University of Technology (Social ences Edition)*, vol. 29, no. 4, pp. 1–5, 2011.
[46] W. T. Pan, "A new Fruit Fly Optimization Algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69–74, 2012.
[47] H. Li, S. Guo, H. Zhao, C. Su, and B. Wang, "Annual electric load forecasting by a least squares support vector machine with a fruit fly optimization algorithm," *Energies*, vol. 5, no. 12, pp. 4430–4445, 2012.
[48] H. Z. Li, S. Guo, and C. J. Li, "A hybrid forecasting model based on fruit fly optimization algorithm and least squares support vector machine:-the case of logistics demand forecasting of China," *Journal of Quantitative Economics*, vol. 10, no. 1, p. 378, 2012.

[49] D. Y. Shi, J. Lu, and L. J. Lu, "A judge model of the impact of lane closure incident on individual vehicles on freeways based on RFID technology and FOA-GRNN method," *Journal of Wuhan University of Technology*, vol. 34, pp. 63–68, 2012.

[50] X. Wang, K. Du, B. Qin, and H. J. Xu, "Drying rate modeling based on FOALSSVR," *Control Engineering of China*, vol. 19, no. 7, pp. 630–638, 2012.

[51] H. Z. Li, S. Guo, C. J. Li, and J. Q. Sun, "A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm," *Knowledge-Based Systems*, vol. 37, pp. 378–387, 2013.

[52] J. Niu, W. Zhong, Y. Liang, N. Luo, and F. Qian, "Fruit fly optimization algorithm based on differential evolution and its application on gasification process operation optimization," *Knowledge-Based Systems*, vol. 88, pp. 253–263, 2015.

[53] F. Xu and Y. Tao, "The improvement of fruit fly optimization algorithm-using bivariable function as example," in *Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012)*, pp. 1516–1520, Atlantis Press, Paris, France, May 2014.

[54] X. Guo, X. Zhang, and L. Wang, "Fruit fly optimisation algorithm with adaptive sign processing," *International Journal of Computing Science and Mathematics*, vol. 6, no. 6, pp. 538–545, 2015.

[55] D. Shan, G. Cao, and H. Dong, "LGMS-FOA an improved fruit fly optimization algorithm for solving optimization problems," *Mathematical Problems in Engineering*, vol. 2013, Article ID 108768, 9 pages, 2013.

[56] X. Wang and S. Yu, "Improved evolution strategies for high-dimensional optimization," *Control Theory & Applications of China*, vol. 23, no. 01, pp. 148–151, 2006.

[57] H. Y. Sang, Q. K. Pan, and P. Y. Duan, "Self-adaptive fruit fly optimizer for global optimization," *Natural Computing*, vol. 18, no. 4, pp. 785–813, 2017.

[58] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm & Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.