IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 29, 2019, accepted April 29, 2019, date of publication May 2, 2019, date of current version May 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2914534

An Improved Firefly Algorithm With Specific Probability and Its Engineering Application

CHUNFENG WANG AND XINYUE CHU
College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, China
Corresponding author: Xinyue Chu (2316607219@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 11671122, in part by the Key Project of Henan Educational Committee under Grant 19A110021, and in part by the School Research Project under Grant 20180562.

ABSTRACT Firefly algorithm (FA) belongs to the swarm intelligence algorithm, which is famous for its strong exploration, a small number of parameter settings and effortless operation. However, there are some drawbacks in the searching process for FA, as the poor accuracy of the solution, high-computational time complexity, and doughty oscillation. These phenomenons are attributed to two factors: 1) in classical FA, the firefly, which is gloomier than others can be attracted by any one of them and 2) FA cannot fully utilize the information of objective function and its fitness. In this paper, to overcome these shortcomings, based on specific probability p_{fit}, a new modified firefly algorithm (pFA) is proposed. In this algorithm, for speeding up the convergence, the specific probability p_{fit} determined by the value of fitness of the firefly is used to choose a neighbor among the better fireflies compared with the predefined firefly, which helps the predefined firefly to move toward a better direction. If there is no neighbor, the opposite learning strategy is employed to lead the firefly to move. The performance of pFA is tested on some well-known benchmark functions. The findings of the test show that pFA is outperformed to FA and some other state-of-the-art algorithms. Finally, we apply pFA to solve four engineering applications.

INDEX TERMS Firefly algorithm, specific probability, opposite learning, engineering application.

I. INTRODUCTION

The swarm intelligence algorithm is computational intelligence algorithm by simulating collective intelligence of biological groups. It came into being and developed rapidly, since the traditional optimization technics, namely deterministic algorithms, like branch and bound algorithms, is incapable of solving complex practical problems, such as data mining [1], [2], 0-1 knapsack problems [3], [4], vehicle routing problem [5], [6]. For a deterministic algorithm, it can not solve complex practical problems mainly because it often requires these problems have some good properties, such as continuous, smooth, convex, monotonic, etc. However, many of these problems do not have such properties. Different from the deterministic methods, a stochastic algorithm has less requirements on these problems, so it has good performance to solve these problems with little information of objective function, higher dimensional, multiobjective optimization and even no concrete form of objective function. What's more, it mainly simulates collective behavior of biological groups to solve optimization problems. Up to now, many swarm intelligence algorithms have been proposed, e.g. Artificial Bee Colony (ABC) [7]–[9], Ant Colony Optimization (ACO) [10], Particle Swarm Optimization (PSO) [11], Simulated Annealing (SA) [12], Cuckoo Search (CS) [13], Tabu Search (TS) [14], Harmony Search (HS) [15], Firefly Algorithm (FA) [16]–[18] and so on.

As one of swarm intelligence algorithms, firefly algorithm (FA) has captured much attention of many scholars since it was proposed by Yang in 2008 [19]. And it was applied to solve many problems, including network reconfiguration of unbalanced distribution networks [20], visual tracking [21], vision-based railway overhead inspection system [22], job shop scheduling problem [23] and so forth.

Since FA was presented, it has many variants, which can be divided into three aspects: (1) the fixed step α easily makes the algorithm getting into the local optima, in order to overcome such drawback, variable strategies for step setting were utilized to modify it [24]–[27]. (2) FA is good at exploiting, while PSO prefers to explore. It is a good idea

The associate editor coordinating the review of this manuscript and approving it for publication was Khalid Aamir.

that make up for FA with PSO in terms of performance. Thus, Sivaranjani and Kumar [28] proposed hybrid particle swarm optimization-firefly algorithm (HPSOFA) to solve combinatorial optimization of non-slicing VLSI floorplanning. In FA, except the darker firefly, the brighter firefly almost does not move, only the brightest one move at random. Xia et al. [29] employed three novel operators in a hybrid optimizer based on the two algorithms. In [30], pattern search algorithm was utilized as a local optimization method to improve firefly algorithm. In order to enhance the search of the brighter one, Wang et al. [31] hybridize firefly algorithm with differential evolution (DE). To solve the capacitated facility location problem, Rahmani and Mirhassani [32] proposed a hybrid firefly-Genetic Algorithm. In order to solve vehicle routing problems, Goel and Maini [33] combined ant colony and firefly algorithm to improve the performance of firefly algorithm. (3) in order to further improve the performance of FA, some authors combine FA with classical optimization techniques. For example, Gandomi *et al.* [34] combined chaos with FA to improve its exploration and the robustness of solution. Kotteeswaran and Sivakumar [35] introduced Lévy -flight into FA to enhance the performance of FA.

Furthermore, in classical FA, for any darker firefly, it can be allured by all the fireflies that are brighter than it. Thus, it can make the computation complexity of time higher and be easy to cause oscillations in the search process. In order to alleviating its negative effects, in 2016, Wang et al. [36] proposed a new FA: firefly algorithm with random attraction (RaFA). In the paper, each firefly x_i is only attracted by the firefly x_j, which is randomly selected from others except x_i, rather than all fireflies. Therefore, to a large extent, the time complexity is reduced. However, it can't guarantee that there is a better direction to direct x_i, it may reduce the accuracy and convergence speed of the algorithm. In order to overcome such disadvantage, Wang *et al.* [37] introduced an improved FA: firefly algorithm with neighborhood attraction (NaFA). x_i identified k-neighbor around it to compose a circle topology, and then if x_j, which belongs to the k-neighbor, is brighter than x_i, x_i will move toward x_j.

Although the aforementioned FA variants have a better performance than the classical FA, there is still room for improvement. Based on the above considerations, we propose an improved FA algorithm(pFA). The contributions of this paper are given as follows:

(1) We put these fireflies, whose values of fitness are bigger than $x_i's$, into a set K, and then we randomly select a firefly in the K as its neighbor with p_{fit}, which can reduce computational time complexity, speed up the convergence, and avoid oscillation in the iteration.

(2) To deal with the situation that there is no neighbor, we take the opposite learning strategy into account to help the firefly jump out of a local position and increase the diversity of the population.

The rest of the paper is composed as follows. FA algorithm is described in Section 2. And then, the proposed algorithm pFA is depicted in Section 3. Section 4-Section 5 show the corresponding experimental results. In Section 6, we apply pFA to solve four practical problems. Finally, in Section 8, there are some conclusions.

II. CLASSICAL FIREFLY ALGORITHM

In FA, it mainly imitates the flashing behavior among fireflies. Each firefly is considered as a potential solution in search space, and the move behavior among fireflies stands for solutions's upgrading to find a better solution. Moreover, in order to guarantee the algorithm on the right road, it is idealized by 3 hypotheses which are depicted as follows:

(1) Fireflies are asexual; namely, they can be attracted with each other regardless of gender.

(2) The size of the attraction is proportional to the brightness of the flashing. For two random fireflies x_i and x_j, if x_i is gloomier than x_j, then x_i is attracted by x_j. If x_i is the best one among all fireflies, it will move randomly in search space.

(3) The brightness of fireflies is relative with the value of objective function.

A. BASIC PARAMETERS

In FA, both brightness (I) and attractiveness (β) play an important role. In general, for maximum problem, the brightness of a firefly is positively relative with the value of objective function, and for minimum problem, they are negatively relative with each other.

Prior to defining the attractiveness(namely β), firstly, the Euclidean distance of x_i and x_j should be calculated as follows:

$$r_{ij} = \sqrt{\sum_{t=1}^{D}(x_{it} - x_{jt})^2}, \tag{1}$$

where D is the dimension of the problem.

Then, attractiveness is described as follows:

$$\beta = \beta_0 \times e^{-\gamma \times r_{ij}^2}, \tag{2}$$

where β_0 is the maximum attractiveness, and γ is the absorption coefficient.

B. THE MOVEMENT OF THE FIREFLY

Supposed that x_i is attracted by x_j, and then, x_i will move as following formula:

$$x_i^{t+1} = x_i^t + \beta \times (x_j^t - x_i^t) + \alpha \times (rand - 0.5), \tag{3}$$

where x_i^t represents the position of x_i at the t-th iteration. $rand \in [0, 1]$, α is the random step, and $\alpha \in [0, 1]$.

C. THE DETAILED STEP OF THE FA

Step 1: Initialize population NP, and set parameters, such as β_0 and α.

Step 2: Calculate r_{ij} and β with formula (1) and (2), respectively.

Step 3: Upgrade solution with formula (3).

Step 4: Generate a better solution by comparing the previous solution with the current solution.

Step 5: For the new solution, calculate its value of objective function.

Step 6: Judge whether the algorithm satisfies the terminate condition(up to the max number of iterations($ItMax$)). If satisfied, the algorithm end, and output optimal solution; else, return to Step 2.

III. THE IMPROVED FA: PFA

In FA, each firefly can be attracted many times by the brighter fireflies, which may cause severe oscillation. Thus, in order to alleviate this disadvantage, we proposed an improved algorithm abbreviated as pFA. In pFA, the attraction among fireflies is reduced greatly, and it can also reduce computational time complexity and increase the accuracy of solution. The detailed description is as follows.

In pFA, firstly, we calculate the value of fitness(fit) of each firefly x_i with formula (4):

$$fit = \begin{cases} \dfrac{1}{1+f(x_i)}, & \text{if } f(x_i) \geq 0, \\ 1+|f(x_i)|, & \text{else}, \end{cases} \tag{4}$$

where $f(x_i)$ is the objective function of firefly x_i. And then, those values of fitness are bigger than $x_i's$(namely $i-th$ firefly's) are put into a set K. Finally, we utilize the roulette wheel selection to choose its neighbor, the formula is shown as the following:

$$p_{fit} = \frac{fit(x_k)}{\sum_{x_t \in K} fit(x_t)}. \tag{5}$$

For x_i, we randomly select a firefly x_k as its neighbor with p_{fit} in set K, not in all fireflies. By this way, the worse fireflies are sifted out, and the remainders are better than x_i. Furthermore, from (5), we can see that, the better the firefly is, the more chances to be chosen it has. Thus, no matter which one is selected, the chosen one will always give a good direction to lead x_i rightly. Obviously, this strategy can not only speed up the convergence and guarantee the diversity of population, but also reduce selection pressure for fireflies.

However, there exists such case that no firefly is better than x_i, that is to say x_i is the best one in the whole population. Thus, it falls into local optimum easily. To deal with this case, the opposite learning skill is adopted by utilizing the information of x_i.

Based on the above discussion, we can rewrite formula (3) as follows:

$$x_i^{t+1} = \begin{cases} x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_k^t - x_i^t) \\ \quad +\alpha(rand - 0.5), & \text{if } K \neq \emptyset, \\ l + u - x_i^t, & \text{else}, \end{cases} \tag{6}$$

where l and u are the lower bound and upper bound of variables, respectivey; x_k is chosen from set K with p_{fit}.

As far as the step α, this paper employs the following formula to update:

$$\alpha_t = \alpha_0 \alpha_{t-1}, \tag{7}$$

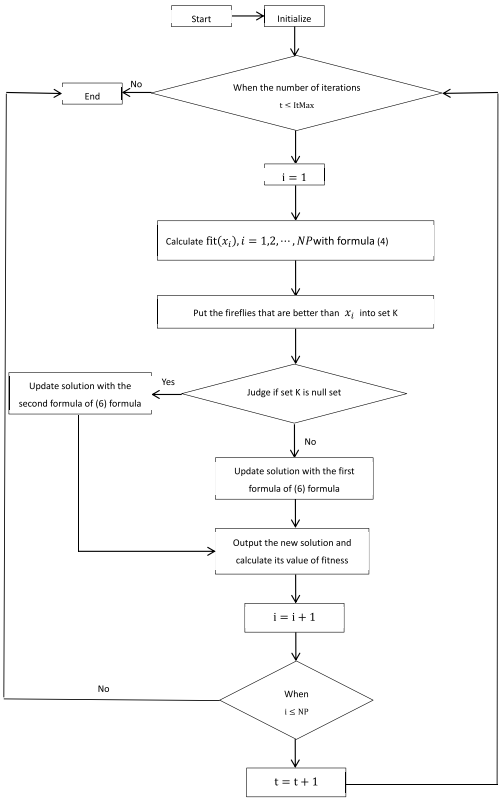

FIGURE 1. The flow chart of pFA.

where $\alpha_0 = 0.7$. And the α_0's scope should be between 0 and 1, $\alpha_1 = 0.25$.

The detailed description of pFA is given in algorithm 1.

Algorithm 1 Pseudo-Code of pFA

01:Initialize the population size NP, $\{x_i | i = 1, 2, \cdots, NP\}$, the maximum number of iterations $ItMax$.
02: **while** $t <= ItMax$ **do**
03: for $i = 1$ to NP **do**
04: Calculate the value of fitness of x_i.
05: better fireflies are chosen to form the set K.
06: Choose a firefly x_k from set K with probability p_{fit}
07: Move x_i according to (6).
08: Compute the fitness value of the new x_i.
08: end;
09: Rank the fireflies and find the current best;
10: $t = t + 1$;
11: end

The flow chart of pFA is shown in Fig. 1

A. TIME COMPLEXITY

For the optimization problem (f), suppose that $O(f)$ is computational time complexity of evaluating its function value. Thus, the computational time complexity of FA is $O(ItMax * NP^2 * f)$, while the computational time complexity of RaFA and NaFA are $O(ItMax * NP * f)$ and $O(ItMax * k * NP * f)$, respectively, k is the number of neighbor in NaFA.

TABLE 1. Benchmark functions.

Functions	Range	Optimal value
$f_1 = \sum_{i=1}^{D} x_i^2$	[-100,100]	0
$f_2 = \sum_{i=1}^{D} \mid x_i \mid + \prod_{i=1}^{D} \mid x_i \mid$	[-10,10]	0
$f_3 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
$f_4 = \max_i \{\mid x_i \mid, 1 \le i \le D\}$	[-100,100]	0
$f_5 = \sum_{i=1}^{D} i x_i^2$	[-10,10]	0
$f_6 = \sum_{i=1}^{D} i x_i^4$	[-1.28,1.28]	0
$f_7 = \sum_{i=1}^{D} \mid x_i \mid^{(i+1)}$	[-1,1]	0
$f_8 = \sum_{i=1}^{D} 10^6 \frac{i-1}{D-1}$	[-100,100]	0
$f_9 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	[-100,100]	0
$f_{10} = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[-1.28,1.28]	0
$f_{11} = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12,5.12]	0
$f_{12} = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{D} x_i^2/D}) - \exp(\sum_{i=1}^{D} \cos(2\pi x_i/D) + 20 + e$	[-32,32]	0
$f_{13} = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0
$f_{14} = 0.5 + \frac{sin(\sqrt{\sum_{i=1}^{D} x_i^2})^2 - 0.5}{(1+0.001\sum_{i=1}^{D} x_i^2)^2}$	[-100,100]	0
$f_{15} = \frac{\sum_{i=1}^{D} (x_i^4 - 16x_i^2 + 5x_i)}{D}$	[-5,5]	0
$f_{16} = \sum_{i=1}^{D} \mid x_i sin(x_i) + 0.1x_i \mid$	[-10,10]	0
$f_{17} = \begin{cases} \sum_{i=1}^{D} (x_i^2 - 10\cos(2\Pi x_i) + 10), & \mid x_i \mid < 0.5 \\ \sum_{i=1}^{D} ((\frac{random(2x_i)}{2})^2 - 10\cos(\Pi random(2x_i)) + 10), & \mid x_i \mid \ge 0.5 \end{cases}$	[-5.12,5.12]	0
$f_{18} = \frac{\Pi}{D} 10\sin^2(\Pi y_1) + \frac{\Pi}{D} \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10sin^2(\Pi y_{i+1})]$ $+ \frac{\Pi}{D} (y_D - 1)^2 + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -1 \end{cases}$	[-50,50]	0
$f_{19} = \sum_{i=1}^{D} (20 \frac{i-1}{D-1} * z_i)^2, z = x * M$	[-100,100]	0
$f_{20} = (1000x_1)^2 + \sum_{i=2}^{D} z_i^2, z = x * M$	[-100,100]	0
$f_{21} = \sum_{i=1}^{D} z_i^2, z = x - o$	[-100,100]	0

In this paper, for pFA, its computational time complexity is $O(ItMax * NP * f)$. Apparently, when it comes to computational time complexity, NaFA is equal to pFA, however, considering the accuracy and convergence speed of the algorithm, pFA is still better than NaFA.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, the performance of pFA is tested on 21 common benchmark functions derived from CEC 2005 and compared with FA, RaFA, and NaFA. Here, $f_1 - f_9$ are unimodal functions, f_{10} is noise function, $f_{11} - f_{18}$ are multimodal functions, f_{19} and f_{20} are orthogonal functions and f_{21} is shifted sphere function. The specific descriptions of these functions are given in Table 1.

A. PARAMETER SETTING

To make it fair, for all algorithms, the population size NP is set to 40, and the maximum iterations($ItMax$) is set to 2500. Both the initial β_0 and γ are set to 1, and except for pFA, $\alpha = 0.25$. The α in pFA is discussed in Experiment 1. Moreover, for NaFA, K is set to 3(the number of neighbors). Furthermore, to deeply analyze the performance of pFA statistically, all algorithms run 30 times on 30, 50 and 100 dimension, respectively, and we take minimum, mean and deviation as the evaluation criterion to analyze the statistical results.

B. EXPERIMENT 1: DETERMINE THE VALUE OF THE α_0 AND α_1 IN FORMULA (7)

In our algorithm, the values of α_0 and α_1 in formula (7) play a vital role. In the early stages of the iteration, the larger value of α can enhance the global search capability of the algorithm. As the iteration progresses, the smaller value of α can give the algorithm a strong local search ability. Thus, the α should be regressive with iterating, To determine which value is better for pFA, we set α_0 to 0.1, 0.3, 0.5, 0.7, 0.9, and α_1 to 0.15, 0.35, 0.55, 0.75, 0.95, respectively. And compare them on 21 benchmark functions with 30 dimension. The results are given in Table 2 and Table 3.

According to the data given in Table 2, for most functions, the best value of α_0 is chosen in [0.5, 0.7]. For f_{18}, $\alpha_0 = 0.9$ is

TABLE 2. Results obtained by pFA with different values of α_0 and α_1 on 30-dimension.

Function	$\alpha_0 = 0.1, \alpha_1 = 0.25$	$\alpha_0 = 0.3, \alpha_1 = 0.25$	$\alpha_0 = 0.5, \alpha_1 = 0.25$	$\alpha_0 = 0.7, \alpha_1 = 0.25$	$\alpha_0 = 0.9, \alpha_1 = 0.25$
f_1	0(0)	0(0)	0(0)	0(0)	1.03e-224(0)
f_2	1.59e-280(0)	6.39e-281(0)	1.80e-281(0)	3.08e-279(0)	4.37e-113(4.58e-113)
f_3	1.78e-119(9.76e-119)	1.64e-158(9.03e-158)	6.35e-155(3.48e-154)	1.65e-133(9.06e-133)	3.50e-140(1.40e-140)
f_4	2.66e-242(0)	2.64e-240(0)	9.00e-243(0)	1.19e-241(0)	4.46e-113(5.38e-114)
f_5	0(0)	0(0)	0(0)	0(0)	1.55e-225(0)
f_6	0(0)	0(0)	0(0)	0(0)	(0)
f_7	0(0)	0(0)	0(0)	0(0)	1.36e-234(0)
f_8	0(0)	0(0)	0(0)	0(0)	3.28e-220(0)
f_9	0(0)	0(0)	0(0)	0(0)	2.2396e-63(1.1914e-62)
f_{10}	1.06e-05(1.02e-05)	1.08e-05(9.68e-05)	1.39e-05(1.88e-05)	1.30e-05(1.35e-05)	8.94e-06(9.51e-06)
f_{11}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{12}	3.49e-15(1.52e-15)	3.13e-15(1.22e-15)	3.37e-15(1.44e-15)	3.49e-15(1.52e-15)	8.94e-15(3.45e-15)
f_{13}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{14}	0.0097(6.23e-11)	0.0097(3.41e-07)	0.0097(3.3167e-10)	0.0097(2.16e-07)	0.0097(1.01e-10)
f_{15}	-38.4619(2.0583)	-39.0839(2.3885)	-38.9468(2.2689)	-39.52(2.4386)	-46.8126(4.8447)
f_{16}	1.78e-278(0)	1.97e-280(0)	2.85e-282(0)	1.21e-282(0)	0.0034(0.0064)
f_{17}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{18}	0.7996(0.2319)	0.8171(0.2756)	0.7627(0.2608)	0.8048(0.3094)	0.0468(0.0267)
f_{19}	1.86e-12(1.02e-11)	1.09e-13(6.00e-13)	5.46e-12(2.52e-11)	9.41e-18(5.14e-17)	4.06e-15(2.18e-14)
f_{20}	0(0)	0(0)	0(0)	0(0)	2.28e-221(0)
f_{21}	0.0094(1.00e-11)	0.0110(1.21e-10)	0.0064(2.83e-18)	0.0108(9.11e-19)	0.0115(2.73e-18)

TABLE 3. Results obtained by pFA with different values of α_0, and α_1 on 30-dimension.

Function	$\alpha_0 = 0.7, \alpha_1 = 0.15$	$\alpha_0 = 0.7, \alpha_1 = 0.35$	$\alpha_0 = 0.7, \alpha_1 = 0.55$	$\alpha_0 = 0.7, \alpha_1 = 0.75$	$\alpha_0 = 0.7, \alpha_1 = 0.95$
f_1	0(0)	0(0)	0(0)	0(0)	0(0)
f_2	7.80e-280(0)	9.41e-282(0)	8.86e-279(0)	1.12e-280(0)	4.37e-280(0)
f_3	9.27e-127(5.08e-126)	8.94e-133(4.89e-132)	1.16e-155(6.36e-155)	9.98e-148(5.41e-147)	3.30e-97(1.80e-96)
f_4	1.40e-239(0)	7.76e-242(0)	1.36e-238(0)	1.41e-241(0)	4.87e-242(0)
f_5	0(0)	0(0)	0(0)	0(0)	(0)
f_6	0(0)	0(0)	0(0)	0(0)	(0)
f_7	0(0)	0(0)	0(0)	0(0)	(0)
f_8	0(0)	0(0)	0(0)	0(0)	(0)
f_9	0(0)	0(0)	0(0)	0(0)	0(0)
f_{10}	1.13e-05(1.01e-05)	9.88e-06(1.06e-05)	1.18e-05(9.54e-05)	1.04e-05(9.87e-05)	1.03e-05(1.27e-05)
f_{11}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{12}	3.37e-15(1.44e-15)	3.84e-15(1.70e-15)	3.25e-15(1.34e-15)	3.49e-15(1.52e-15)	3.49e-15(1.52e-15)
f_{13}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{14}	0.0097(3.16e-09)	0.0097(5.65e-09)	0.0097(1.65e-08)	0.0097(8.64e-11)	0.0097(2.38e-11)
f_{15}	-39.0518(3.2473)	-39.4378(2.0107)	-38.7370(2.5830)	-39.6825(2.8559)	-39.1156(2.1769)
f_{16}	4.01e-282(0)	1.27e-281(0)	2.17e-282(0)	1.38e-282(0)	9.86e-282(0)
f_{17}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{18}	0.7197(0.2452)	0.7063(0.2046)	0.7107(0.2567)	0.7918(0.2931)	0.7459(0.2802)
f_{19}	6.43e-24(3.52e-23)	1.42e-22(7.82e-22)	1.02e-17(5.62e-17)	9.87e-12(5.39e-11)	1.95e-20(1.06e-19)
f_{20}	0(0)	0(0)	1.13e-27(1.98e-27)	2.06e-17(6.46e-17)	4.80e-09(9.40e-09)
f_{21}	0.0126(5.86e-18)	.0.0122(4.89e-18)	0.0065(3.47e-17)	0.0146(5.06e-18)	0.0059(1.51e-18)

best, and for $f_{19}, f_{21}, \alpha_0 = 0.7$ is better than others. In Table 3, for f_3 and f_{14}, $\alpha_1 = 0.55/0.75$ is best, except them, its better that the value of α_1 is chosen in [0.15, 0.35] based on overall consideration. Thus, it is recommended that this combination, $\alpha_0 = 0.7, \alpha_1 = 0.25$, should be selected.

C. EXPERIMENT 2: COMPARISON WITH OTHER FAS ON MIN, WORST, MEAN AND STANDARD DEVIATION

In this part, to verify the superiority of pFA, we compare it with FA, RaFA and NaFA on 21 benchmark functions, which are described in Table 1.

When the dimension of the problem is 30, the computational results of FA, RaFA and NaFA and pFA are given in Table 4. From the table, it can be seen that, for the mean, minimum and standard deviation, pFA is better than others on $f_2 - f_{21}$. Among them, for $f_5 - f_9, f_{11}, f_{13}, f_{17}$ and f_{20}, pFA can find the minimum value of 0, and even the mean and standard deviation are 0, which demonstrates its high accuracy and better robustness; For $f_3, f_4, f_{10}, f_{12}, f_{16}$ and f_{19}, the minimum value found by pFA is close to 0. Moreover, both of FA and pFA can find the minimum value of 0 on f_1 and f_6, while RaFA and NaFA fail to find it. Finally, for f_{14}, f_{15}, f_{18} and f_{21}, pFA is nearly equal to others.

The computational results on 50 and 100 dimension are given in Tables 5 and 6. Comparing with the results in Table 4, we can see, with dimension increasing, the mean, minimum and standard deviation of FA, RaFA and NaFA are all getting worse, while the results of pFA is almost invariable. It shows that pFA is still effective in solving high dimensional problems.

In order to further analyze the performance of pFA, we give some convergent curve graphs of $f_3, f_7, f_{11}, f_{15}, f_{19}$ and f_{21} on 30, 50 and 100 dimension, respectively. The results are shown in Figs. 2-4. In Fig. 2, for f_3, pFA almost converge to 0 when the iteration ends, and all of FA,RaFA and NaFA present slow increasing trend throughout the iteration process. For f_7, f_{11}

TABLE 4. Computational results of FA, RaFA, NaFA, and pFA for each function on 30 dimension.

Function	Algorithm	Time	Min	Mean	Std
f_1	FA	249	**0**	**0**	**0**
	RaFA	4	874.6161	2.11e+03	647.9445
	NaFA	6	450.5938	1.17e+03	508.6607
	pFA	62	**0**	**0**	**0**
f_2	FA	159	**9.73e-322**	0.8723	1.5287
	RaFA	4	13.2200	18.6540	3.3840
	NaFA	6	9.3045	15.1840	3.4537
	pFA	69	1.26e-292	**3.08e-279**	**0**
f_3	FA	25	551.8415	1.90e+03	874.7558
	RaFA	4	1.17e+03	3.18e+03	1.40e+03
	NaFA	8	1.59e+03	2.87e+03	799.3186
	pFA	93	**2.84e-273**	**1.65e-133**	**9.06e-133**
f_4	FA	25	0.0062	2.9432	2.5806
	RaFA	2	15.4290	20.0772	3.4026
	NaFA	5	10.3884	16.2469	2.7713
	pFA	76	**1.57e-254**	**1.19e-241**	**0**
f_5	FA	22	1.76e-04	2.2007	4.2620
	RaFA	4	174.1170	325.1527	77.6503
	NaFA	6	62.8256	183.4959	69.9538
	pFA	55	**0**	**0**	**0**
f_6	FA	164	**0**	**0**	**0**
	RaFA	5	0.0233	0.1905	0.2088
	NaFA	8	0.0025	0.0998	0.0747
	pFA	83	**0**	**0**	**0**
f_7	FA	33	7.59e-08	1.13e-06	6.33e-07
	RaFA	5	5.39e-08	5.39e-05	1.32e-04
	NaFA	8	1.01e-08	2.19e-06	2.82e-06
	pFA	83	**0**	**0**	**0**
f_8	FA	31	1.38e+06	4.91e+06	2.53e+06
	RaFA	5	5.80e+06	3.26e+07	1.79e+07
	NaFA	8	2.54e+06	9.51e+06	6.17e+06
	pFA	107	**0**	**0**	**0**
f_9	FA	1	1	2.5330	1.3322
	RaFA	0.5	1	2.4333	1.1351
	NaFA	0.7	**0**	1.9000	1.1552
	pFA	44	**0**	**0**	**0**
f_{10}	FA	1088	0.0193	0.0552	0.0250
	RaFA	31	0.0742	0.3144	0.2032
	NaFA	181	0.0705	0.2063	0.0900
	pFA	125	**3.30e-08**	**1.30e-05**	**1.35e-05**
f_{11}	FA	63	31.8386	53.5950	12.5248
	RaFA	4	101.9936	146.5316	21.3669
	NaFA	10	65.0032	99.0897	16.2696
	pFA	27	**0**	**0**	**0**
f_{12}	FA	22	6.21e-15	1.67e-14	5.31e-15
	RaFA	4	7.3980	9.5945	0.9476
	NaFA	6	5.5759	8.2415	1.1312
	pFA	30	**2.66e-15**	**3.49e-15**	**1.52e-15**
f_{13}	FA	27	5.02e-04	0.8384	0.3028
	RaFA	5	0.9594	0.9955	0.0076
	NaFA	46	0.7904	0.9476	0.0485
	pFA	35	**0**	**0**	**0**
f_{14}	FA	21	0.1270	0.2915	0.0843
	RaFA	3	0.4419	0.4735	0.0107
	NaFA	59	0.4297	0.4706	0.0127
	pFA	77	**0.0097**	**0.0097**	**2.16e-07**
f_{15}	FA	113	-70.7927	-67.3345	**2.2962**
	RaFA	5	-47.4986	-41.3536	2.7055
	NaFA	9	-53.9663	-48.5161	3.4484
	pFA	123	**-45.8318**	**-39.5200**	2.4386
f_{16}	FA	29	0.0302	0.3067	0.4919
	RaFA	4	7.1728	12.1989	2.5347
	NaFA	6	2.1486	7.6929	1.8520
	pFA	70	**1.58e-289**	**1.21e-282**	**0**
f_{17}	FA	17	30	56.1333	17.7234
	RaFA	5	66.8561	121.6540	23.2301
	NaFA	17	40.4044	69.3322	13.0052
	pFA	38	**0**	**0**	**0**
f_{18}	FA	47	**1.58e-27**	**0.1424**	0.3472
	RaFA	7	5.3744	530.7299	2.30e+03
	NaFA	13	5.9968	15.1325	7.0716
	pFA	182	0.3828	0.8048	**0.3094**
f_{19}	FA	35	2.18e+03	1.03e+04	5.44e+03
	RaFA	6	3.07e+04	5.82e+04	2.04e+04
	NaFA	11	9.83e+03	3.75e+04	1.45e+04
	pFA	135	**0**	**9.41e-18**	**5.14e-17**
f_{20}	FA	27	1.58e+04	2.63e+04	7.21e+03
	RaFA	4	2.27e+03	6.05e+03	2.27e+03
	NaFA	8	3.44e+03	6.76e+03	2.40e+03
	pFA	78	**0**	**0**	**0**
f_{21}	FA	30	0.0173	0.0173	3.34e-18
	RaFA	12	0.0131	0.0131	8.38e-18
	NaFA	36	0.0137	0.0137	6.54e-18
	pFA	78	**0.0108**	**0.0108**	**9.11e-19**

TABLE 5. Computational results of FA, RaFA, NaFA, and pFA for each function on 50 dimension.

Function	Algorithm	Time	Min	Mean	Std
f_1	FA	49	3.03e-104	1.60e-04	6.17e-04
	RaFA	5	3.26e+03	5.15e+03	1.83e+03
	NaFA	7	2.67e+03	4.49e+03	1.08e+03
	pFA	68	**0**	**0**	**0**
f_2	FA	23	1.0787	13.9119	8.7174
	RaFA	4	29.6556	39.2844	4.1969
	NaFA	6	27.7283	38.6810	6.0088
	pFA	74	**3.31e-277**	**1.47e-266**	**0**
f_3	FA	34	5.41e+03	9.71e+03	3.07e+03
	RaFA	6	4.09e+03	1.06e+04	4.58e+03
	NaFA	12	5.60e+03	9.23e+03	2.80e+03
	pFA	129	**5.39e-179**	**5.12e-62**	**2.80e-61**
f_4	FA	26	7.2956	14.0661	2.8475
	RaFA	3	18.3396	23.7321	2.9446
	NaFA	6	18.2441	22.5221	2.6046
	pFA	83	**4.44e-242**	**2.41e-234**	**0**
f_5	FA	22	3.8608	42.1408	29.4447
	RaFA	4	726.9976	1.47e+03	260.9366
	NaFA	6	585.9786	1.06e+03	316.2513
	pFA	65	**0**	**0**	**0**
f_6	FA	41	2.51e-09	1.27e-04	1.63e-04
	RaFA	6	0.2298	0.9258	0.4317
	NaFA	10	0.2569	0.7123	0.3796
	pFA	118	**0**	**0**	**0**
f_7	FA	41	2.17e-07	1.34e-06	8.20e-07
	RaFA	6	6.67e-08	8.38e-05	2.55e-04
	NaFA	9	1.70e-09	3.98e-06	4.54e-06
	pFA	109	**0**	**0**	**0**
f_8	FA	38	6.40e+06	1.81e+07	8.36e+06
	RaFA	6	3.92e+07	1.06e+08	5.49e+07
	NaFA	10	1.12e+07	4.31e+07	1.94e+07
	pFA	150	**0**	**0**	**0**
	RaFA	0.6	4	7.5667	1.6333
	NaFA	0.9	2	6.5000	1.9608
	pFA	52	**0**	**0.8000**	**0.9965**
f_{10}	FA	1422	0.0918	0.1879	0.0638
	RaFA	38	0.3719	1.0542	0.5028
	NaFA	230	0.4187	1.0951	0.5019
	pFA	168	**3.44e-07**	**8.09e-06**	**7.92e-06**
f_{11}	FA	44	60.6924	100.9517	21.1218
	RaFA	4	238.4711	300.2502	31.4020
	NaFA	8	180.3972	246.0055	24.5482
	pFA	31	**0**	**0**	**0**
f_{12}	FA	60	3.10e-14	1.1351	0.7038
	RaFA	4	9.2711	10.8784	0.8054
	NaFA	8	8.8794	10.6574	0.8643
	pFA	34	**2.66e-15**	**3.84e-15**	**1.70e-15**
f_{13}	FA	20	1.0000	1.0000	7.26e-10
	RaFA	4	1.0000	1.0000	1.48e-06
	NaFA	39	0.9937	0.9994	0.0014
	pFA	44	**0**	**0**	**0**
f_{14}	FA	73	0.4147	0.4798	0.0177
	RaFA	3	0.4850	0.4930	0.0031
	NaFA	64	0.4888	0.4932	0.0023
	pFA	82	**0.0097**	**0.0097**	**6.87e-08**
f_{15}	FA	38	-70.1781	-66.0138	1.9884
	RaFA	7	**-40.9722**	-37.7798	**1.8800**
	NaFA	13	-48.9140	-43.6976	2.6474
	pFA	167	-41.4030	**-35.9725**	2.5076
f_{16}	FA	28	0.7043	3.0358	1.6669
	RaFA	4	18.9912	25.1443	2.7943
	NaFA	8	16.6327	20.4971	2.5943
	pFA	76	**2.20e-277**	**1.43e-266**	**0**
f_{17}	FA	18	68	111.9750	28.0144
	RaFA	5	193.7580	258.1913	29.0084
	NaFA	23	150.2852	189.0966	23.2016
	pFA	43	**0**	**0**	**0**
f_{18}	FA	54	1.0212	4.6927	2.1361
	RaFA	9	9.7041	6.15e+03	2.20e+04
	NaFA	15	15.5344	1.14e+03	4.19e+03
	pFA	241	**0.5145**	**0.8954**	**0.2381**
f_{19}	FA	45	1.26e+04	4.59e+04	2.28e+04
	RaFA	8	8.21e+04	1.73e+05	6.05e+04
	NaFA	16	6.17e+04	1.23e+05	3.69e+04
	pFA	195	**0**	**1.95e-66**	**1.07e-65**
f_{20}	FA	30	2.98e+04	4.84e+04	1.29e+04
	RaFA	5	6.51e+03	1.05e+04	2.43e+03
	NaFA	9	7.90e+03	1.36e+04	3.53e+03
	pFA	92	**0**	**0**	**0**
f_{21}	FA	74	0.0091	0.0091	3.42e-18
	RaFA	19	0.0073	0.0073	4.39e-18
	NaFA	93	0.0062	0.0062	**2.24e-18**
	pFA	113	**0.0059**	**0.0059**	2.95e-18

TABLE 6. Computational results of FA, RaFA, NaFA, and pFA for each function on 100 dimension.

Function	Algorithm	Time	Min	Mean	Std
f_1	FA	28	1.25e+03	3.28e+03	1.61e+03
	RaFA	5	1.05e+04	1.39e+04	2.04e+03
	NaFA	8	1.24e+04	1.73e+04	2.75e+03
	pFA	79	**0**	**0**	**0**
f_2	FA	25	45.7986	93.6254	19.3097
	RaFA	4	75.6588	91.2161	8.1710
	NaFA	7	86.5001	105.3091	9.7488
	pFA	83	**1.51e-265**	**7.46e-265**	**0**
f_3	FA	33	6.76e+03	9.85e+03	2.40e+03
	RaFA	6	5.97e+03	9.63e+03	3.22e+03
	NaFA	13	3.91e+03	9.63e+03	3.22e+03
	pFA	129	**8.96e-199**	**1.32e-56**	**6.58e-59**
f_4	FA	29	19.7073	29.5103	3.3598
	RaFA	3	22.9635	29.3519	2.7539
	NaFA	6	25.5820	30.6770	2.2555
	pFA	91	**6.78e-237**	**2.67e-227**	**0**
f_5	FA	25	724.7314	2.04e+03	716.0177
	RaFA	4	4.13e+03	6.54e+03	1.19e+03
	NaFA	7	5.13e+03	7.82e+03	1.22e+03
	pFA	74	**0**	**0**	**0**
f_6	FA	56	0.2542	1.1921	0.6860
	RaFA	9	3.0133	6.3736	2.0038
	NaFA	16	4.0194	9.5038	2.7436
	pFA	197	**0**	**0**	**0**
f_7	FA	60	4.56e-07	1.41e-06	7.20e-07
	RaFA	8	1.53e-06	2.14e-04	8.78e-04
	NaFA	14	2.94e-08	3.41e-06	5.64e-06
	pFA	177	**0**	**0**	**0**
f_8	FA	55	3.44e+07	8.77e+07	2.27e+07
	RaFA	9	1.70e+08	3.71e+08	1.13e+07
	NaFA	17	9.24e+07	2.51e+08	9.16e+07
	pFA	260	**0**	**0**	**0**
f_9	FA	2	9	17.3333	4.1133
	RaFA	0.8	19	24.5667	3.5006
	NaFA	1.3	15	21.2000	2.4691
	pFA	63	**4**	**8.2000**	**2.3839**
f_{10}	FA	2086	1.1660	3.0710	1.2766
	RaFA	56	3.9583	7.2508	3.0172
	NaFA	349	4.7223	11.4290	4.7380
	pFA	274	**1.044e-07**	**1.40e-05**	**1.43e-05**
f_{11}	FA	29	182.9112	283.1775	32.5341
	RaFA	5	666.9288	743.7117	43.0929
	NaFA	10	573.5190	643.1476	30.7975
	pFA	39	**0**	**0**	**0**
f_{12}	FA	28	6.2005	8.4004	1.1137
	RaFA	5	11.2087	12.1025	0.4671
	NaFA	11	11.0521	12.9293	0.6050
	pFA	38	**2.66e-15**	**4.08e-15**	**1.77e-15**
f_{13}	FA	27	1.0000	1.0000	0
	RaFA	1	1.0000	1.0000	0
	NaFA	5	1.0000	1.0000	0
	pFA	61	**0**	**0**	**0**
f_{14}	FA	115	0.4979	0.4991	4.12e-04
	RaFA	4	0.4980	0.4988	3.49e-04
	NaFA	71	0.4976	0.4987	3.83e-04
	pFA	93	**0.0097**	**0.0097**	**1.78e-08**
f_{15}	FA	58	-62.4677	-59.1327	**1.7.38**
	RaFA	12	-38.9661	-34.5243	1.7015
	NaFA	23	-43.6207	-37.4587	2.0631
	pFA	270	**-35.8802**	**-33.1229**	1.4444
f_{16}	FA	28	12.8347	22.8740	4.6885
	RaFA	5	51.0554	61.2341	4.7078
	NaFA	9	52.8338	60.5799	4.9688
	pFA	90	**1.79e-265**	**2.50e-257**	**0**
f_{17}	FA	27	234.5053	322.2796	46.2017
	RaFA	5	575.2118	672.3144	46.8110
	NaFA	20	477.7268	546.7777	36.9873
	pFA	55	**0**	**0**	**0**
f_{18}	FA	90	13.6919	42.0941	12.5882
	RaFA	15	237.7223	1.77e+05	2.70e+05
	NaFA	24	3.03e+04	1.82e+05	2.21e+05
	pFA	399	**0.7731**	**1.0352**	**0.0986**
f_{19}	FA	78	1.29e+05	2.36e+05	5.19e+04
	RaFA	14	3.45e+05	5.29e+05	1.11e+05
	NaFA	26	3.82e+05	5.66e+05	1.22e+05
	pFA	362	**0**	**1.95e-66**	**1.07e-65**
f_{20}	FA	45	6.43e+04	9.95e+04	2.72e+04
	RaFA	7	1.75e+04	2.45e+04	5.03e+03
	NaFA	15	2.16e+04	2.91e+04	4.63e+03
	pFA	152	**0**	**0**	**0**
f_{21}	FA	165	0.0038	0.0038	1.62e-18
	RaFA	29	0.0037	0.0037	1.93e-18
	NaFA	217	0.0033	0.0033	**1.15e-18**
	pFA	210	**0.0031**	**0.0031**	2.02e-18

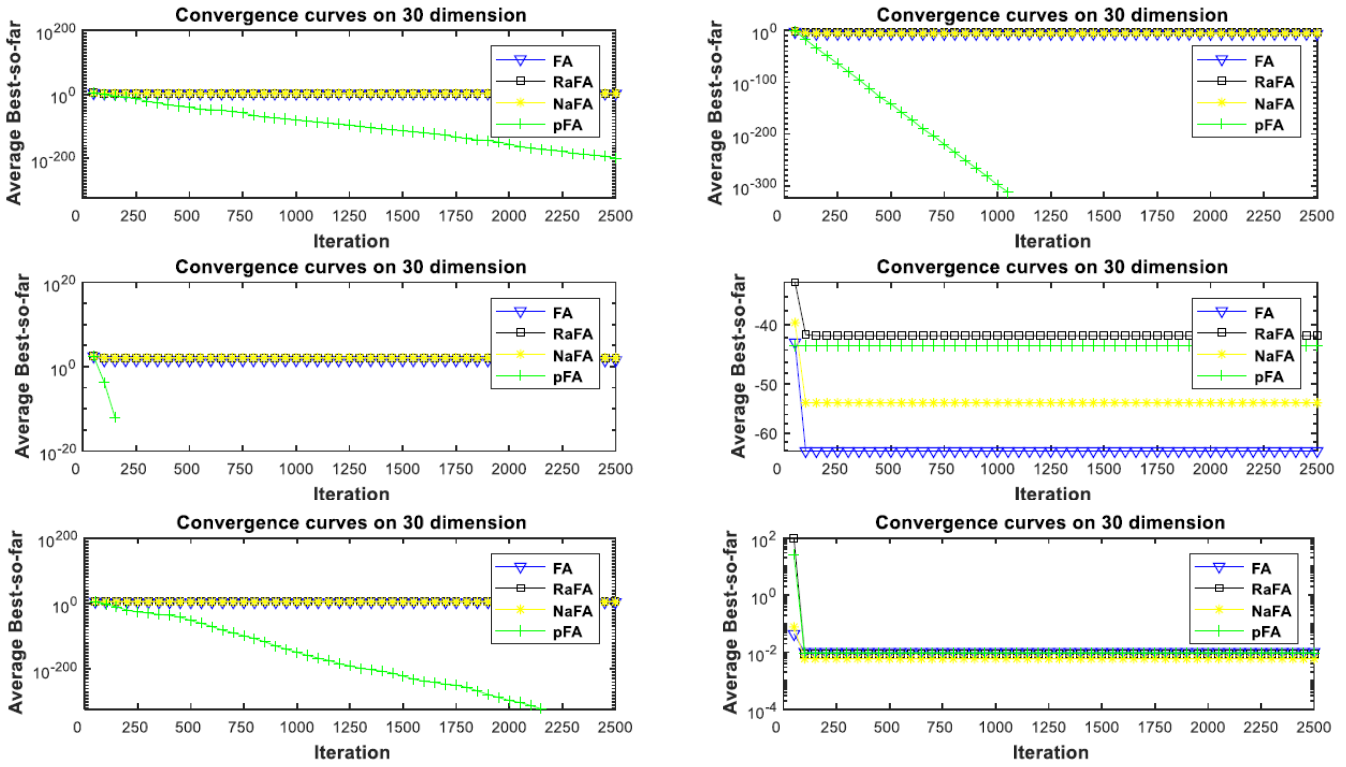

FIGURE 2. Convergence curves of f_3, f_7, f_{11}, f_{15}, f_{19}, and f_{21} on 30 dimension.

and f_{19}, pFA is able to converge to 0 within the iteration process, while others change little. For f_{15}, RaFA is better slightly than others, and for f_{21}, pFA is worse than others at the beginning, with the iteration goes, it quickly caught up with others. As shown in Figs. 3 and 4, as the dimension increase, the convergence of pFA becomes slower in f_3. For f_7, f_{11} and f_{21}, all of them remain unchanged, and for f_{15} and f_{19}, pFA becomes better, while others change little.

In summary, the performance of pFA is better than FA, RaFA and NaFA on most functions, and for majority functions, pFA can find the minimum value of 0. And it also effectively for solving higher dimension problem.

V. EXPERIMENT 3: COMPARISON WITH OTHER FAS BASED ON A PROBABILISTIC METRIC

In [48], Gomes et al. proposed a new way of comparison among metaheuristic optimization algorithms. In this way, it can be clearly seen which algorithm is better. In this paper, as shown in [48], we regard P_{better} as the probability that represents pFA is better than FA, RaFA and NaFA. If pFA is worse than FA, RaFA and NaFA, we regard P_{worse} as the probability, and if pFA is equal to FA, RaFA and NaFA, we regard P_{equal} as the probability. Here, the population size, dimension and the maximum iterations are set to 40, 50 and 2500, respectively. Moreover, we select $f_3, f_7, f_{11}, f_{15}, f_{19}$ and f_{21} as benchmark functions, and for each algorithm, it was run 30 times. The results are shown in Table 7.

As shown in Table 7, for f_3, f_7, f_{11} and f_{19}, pFA is the best, and it is 100% likely to find the optimal value in a single run. For f_{15}, it can be seen that pFA is worse than FA, RaFA and NaFA with the probability of 100%, 72% and 99%, respectively. And for f_{21}, pFA is better than FA, RaFA and NaFA with the probability of 100%.

VI. EXPERIMENT 4: COMPARISON WITH DE AND DES

In this part, the validity of the improved algorithm is further tested by comparing with DE, chDE [49], jDE [50], aDE [51] and IMMSADE [52] on $f_1 - f_9, f_{11}, f_{13}, f_{15}, f_{17}, f_{19}$ and f_{21}. These benchmark functions are consistent with literature [52]. For the sake of fairness, all parameter settings are derived from literature [52]. For each algorithm, it runs 30 times independently. The dimension D and the population size NP are set to 30, 100, respectively, and the maximum iterations($ItMax$) is set to 3000. We utilize the statistics of average value(mean) and standard deviation(std) to evaluate the performance of algorithms after 30 times running. The statistical results are displayed in Table 8. And except for the statistics of pFA, statistics of others are derived from their original papers. For example, the parameter settings of IMMSADE algorithm are the same as literature [52]: all of λ_i^0, F_i^0 and CR_i^0 are randomly chosen from [0.7, 1.0], [0.1, 0.8] and [0.3, 1.0], respectively. Besides, the parameter settings of DE originate from literature [49], in Which CR is set to 0.9 and F is set to 0.9, 0.5. For pFA, β_0, and γ are set to 1.

TABLE 7. Computational results of FA, RaFA, NaFA, and pFA based on a probabilistic metric for f_3, fa_7, f_{11}, f_{15}, f_{19}, and f_{21}.

Function	Algorithm	P_{better}	P_{equal}	P_{worse}
f_3	FA	1.00	0.00	0.00
	RaFA	1.00	0.00	0.00
	NaFA	1.00	0.00	0.00
f_7	FA	1.00	0.00	0.00
	RaFA	1.00	0.00	0.00
	NaFA	1.00	0.00	0.00
f_{11}	FA	1.00	0.00	0.00
	RaFA	1.00	0.00	0.00
	NaFA	1.00	0.00	0.00
f_{15}	FA	0.00	0.00	1.00
	RaFA	0.28	0.00	0.72
	NaFA	0.01	0.00	0.99
f_{19}	FA	1.00	0.00	0.00
	RaFA	1.00	0.00	0.00
	NaFA	1.00	0.00	0.00
f_{21}	FA	1.00	0.00	0.00
	RaFA	1.00	0.00	0.00
	NaFA	1.00	0.00	0.00

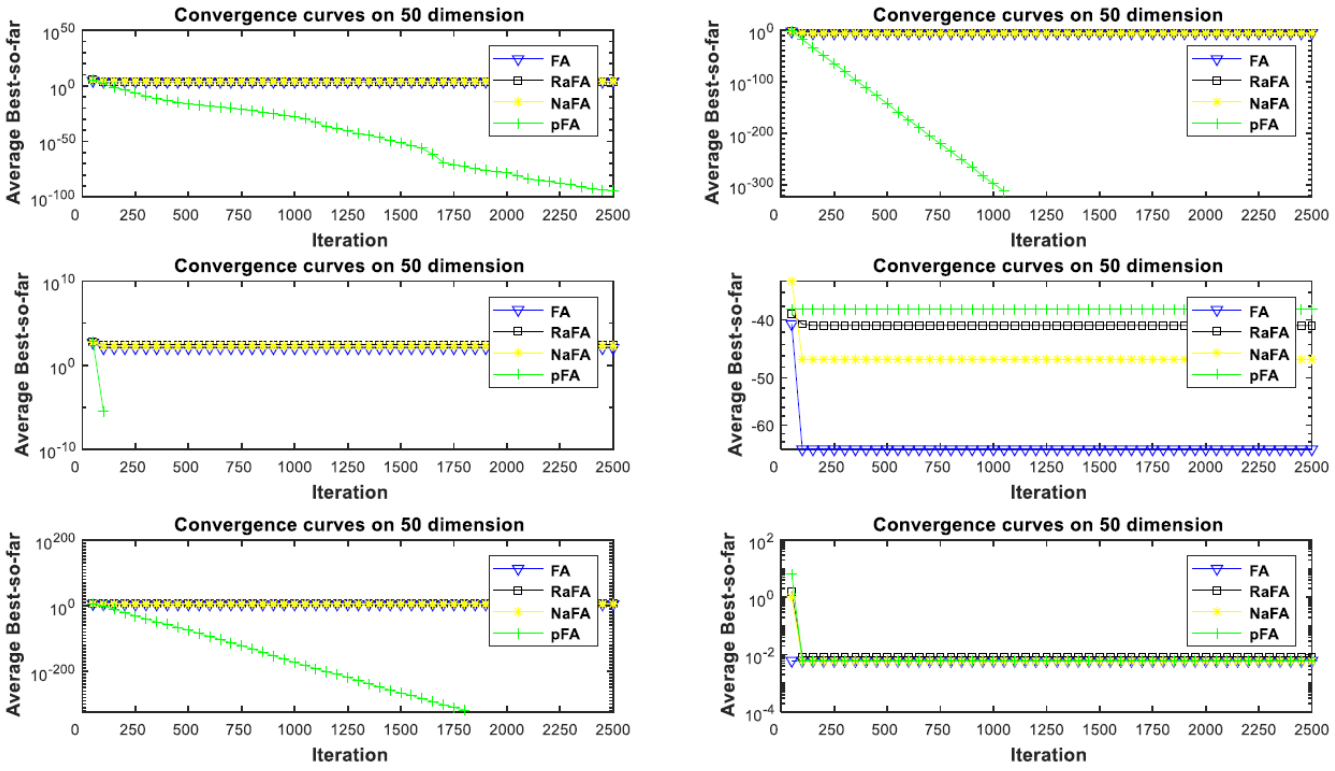

FIGURE 3. Convergence curves of f_3, f_7, f_{11}, f_{15}, f_{19}, and f_{21} on 50 dimension.

As shown in Table 8, for f_1, $f_3 - f_7$ and f_9, both mean and std of pFA are superior to others'. In particular, both of them are 0 on f_1, f_3, f_6 and f_7. For f_8 and f_{15}, the values of mean and std of all algorithms are 0. For f_{13}, pFA is as good as aDE and IMMSADE, of which mean and std are 0. For f_2 and f_{17}, the performance of pFA is worse than others'. Furthermore, mean and std of pFA is slightly lower than others' in f_{21}. In conclusion, The results show that pFA is reliable and effective.

VII. APPLICATION OF PFA

In the engineering field, many practical problems can be boiled down to the optimization problem under a certain mathematical model. Many practices prove: the intelligent algorithm is introduced into the field of engineering optimization to solve all kinds of complex problems, which will obtain lots of economic and social benefits. In this part, in order to further verify the superiority of pFA, we use pFA to solve the four major engineering problems: pressure vessel design, the structure design of tension-pressure spring, Three-bar truss design and I-beam vertical deflection. Besides, the results are compared with those results obtained by some other algorithms.

Moreover, first of all, we need to solve the inequality constraints of the following problems. In this paper, we adopt Deb's rules [53] to solve constraint conditions. The detailed description of Deb's rules is given as follows:

TABLE 8. Computational results of DE and DEs.

Function	Algorithm	Mean	Std
f_1	DE	6.45e-37	9.42e-37
	chDE	1.02e-56	2.80e-56
	jDE	3.24e-39	3.48e-39
	aDE	2.63e-56	2.23e-56
	IMMSADE	5.68e-165	**0**
	pFA	**0**	**0**
f_2	DE	1.27e-05	1.41e-05
	chDE	1.65e-04	1.90e-04
	jDE	2.5800	1.2800
	aDE	1.23e-01	7.11e-02
	IMMSADE	**1.71e-137**	**9.20e-137**
	pFA	4.86e-55	2.66e-54
f_3	DE	1.08e-33	1.55e-33
	chDE	7.16e-53	2.08e-52
	jDE	2.11e-36	2.58e-36
	aDE	2.93e-53	2.68e-53
	IMMSADE	5.44e-161	2.93e-160
	pFA	**0**	**0**
f_4	DE	4.16e-18	3.83e-18
	chDE	3.06e-34	4.40e-34
	jDE	3.40e-23	2.49e-23
	aDE	6.75e-33	4.42e-33
	IMMSADE	4.64e-48	2.50e-47
	pFA	**1.10e-321**	**0**
f_5	DE	1.99e-01	3.39e-01
	chDE	1.02e+01	5.0600
	jDE	1.91e-05	1.12e-05
	aDE	2.77e-04	1.10e-03
	IMMSADE	3.14e-88	1.69e-87
	pFA	**4.82e-283**	**0**
f_6	DE	8.13e-31	8.51e-31
	chDE	9.73e-51	1.97e-50
	jDE	1.57e-33	1.69e-33
	aDE	1.71e-50	1.74e-50
	IMMSADE	3.99e-165	**0**
	pFA	**0**	**0**
f_7	DE	1.49e-36	1.93e-36
	chDE	7.19e-56	3.33e-55
	jDE	5.34e-39	4.91e-39
	aDE	5.15e-56	5.00e-56
	IMMSADE	9.09e-125	4.89e-124
	pFA	**0**	**0**
f_8	DE	**0**	**0**
	chDE	**0**	**0**
	jDE	**0**	**0**
	aDE	**0**	**0**
	IMMSADE	**0**	**0**
	pFA	**0**	**0**
f_9	DE	4.32e-03	1.30e-03
	chDE	3.87e-03	1.02e-03
	jDE	4.88e-03	1.42e-03
	aDE	4.85e-03	1.41e-03
	IMMSADE	1.33e-04	7.24e-05
	pFA	**4.32e-06**	**4.04e-06**
f_{11}	DE	**0**	**0**
	chDE	2.46e-04	1.33e-03
	jDE	**0**	**0**
	aDE	**0**	**0**
	IMMSADE	**0**	**0**
	pFA	0.0776	0.2426
f_{13}	DE	1.44e+02	2.20e+01
	chDE	2.5200	2.1300
	jDE	4.5900	3.5800
	aDE	**0**	**0**
	IMMSADE	**0**	**0**
	pFA	**0**	**0**
f_{15}	DE	**0**	**0**
	chDE	**0**	**0**
	jDE	**0**	**0**
	aDE	**0**	**0**
	IMMSADE	**0**	**0**
	pFA	**0**	**0**
f_{17}	DE	**1.35e-19**	**2.40e-35**
	chDE	**1.35e-19**	3.13e-29
	jDE	**1.35e-19**	2.41e-35
	aDE	**1.35e-19**	2.41e-35
	IMMSADE	**1.35e-19**	2.95e-32
	pFA	0.5411	0.2116
f_{21}	DE	3.41e-01	4.50e-02
	chDE	1.60e-01	4.55e-02
	jDE	3.09e-01	4.01e-02
	aDE	3.18e-01	4.11e-02
	IMMSADE	**6.10e-02**	**1.72e-02**
	pFA	0.49e-01	0.98e-01

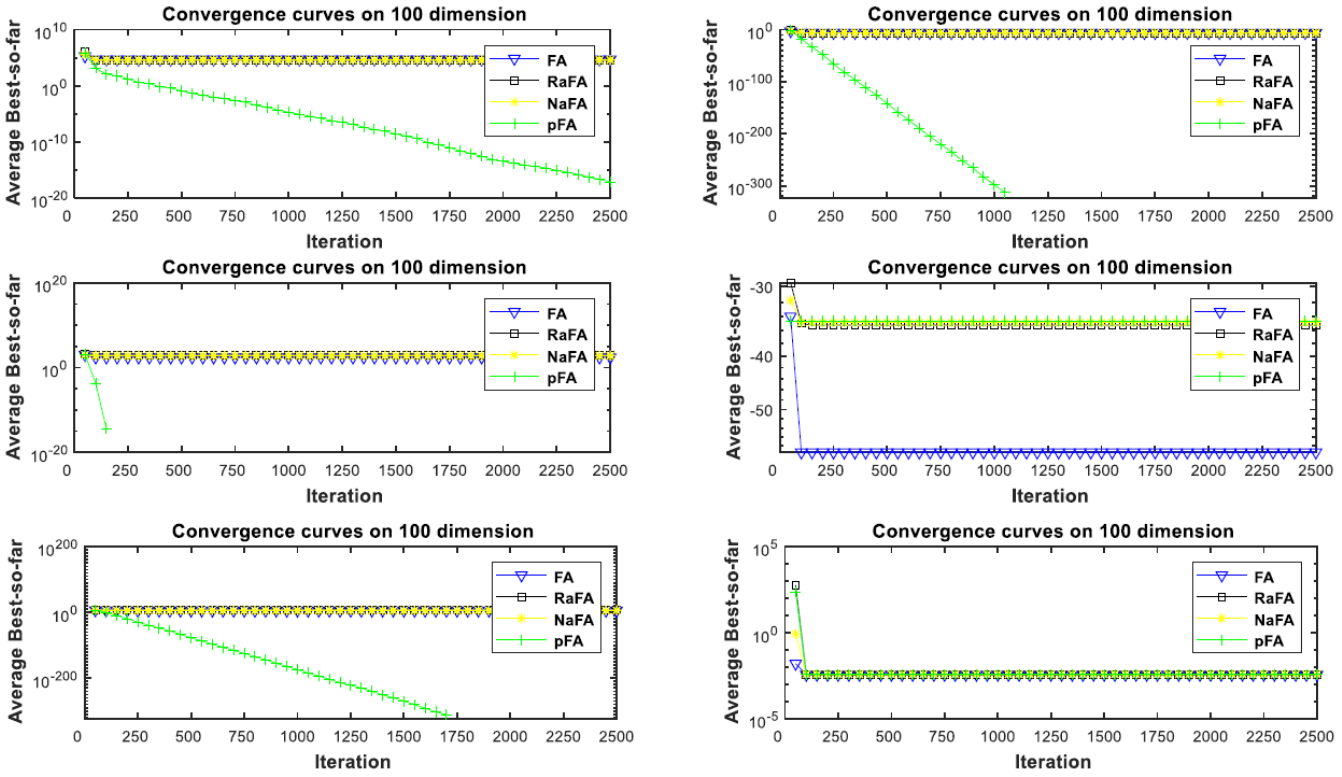

FIGURE 4. Convergence curves of f_3, f_7, f_{11}, f_{15}, f_{19}, and f_{21} on 100 dimension.

(1) Between a feasible solution and an infeasible solution, the feasible solution is preferred.

(2) The infeasible solution is regarded as a feasible solution, when the infeasible solution violates the constraints very rarely.

(3) For two feasible solutions, the solution with better objective function value is better.

(4) For two infeasible solutions, the solution violating constraints very little is better.

A. PRESSURE VESSEL DESIGN

The pressure vessel design is from literature [38] and belongs to the widely used structural design benchmark problem. It minimize mainly the total cost$f(x)$, which includes the cost of materials, forming and welding. In this problem, there are some parameters: the thickness of shell (T_s), the thickness of the head (T_h), the inner radius (R) and the head (L). The detailed description is shown in Fig. 5 [38]. In this paper, we let x_1, x_2, x_3 and x_4 represents T_s, T_h, R and L, respectively. Thus, the variable vector can be written as $x = [x_1, x_2, x_3, x_4]$.

The optimization problem can be written as follows:

$$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

subject to

$$g_1(x) = -x_1 + 0.0193x_3 \le 0,$$

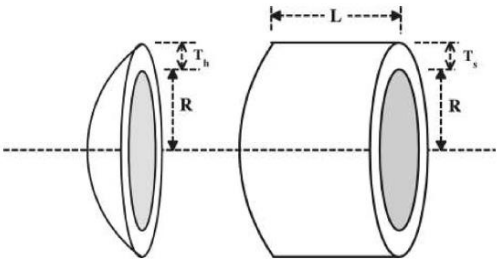

FIGURE 5. Pressure vessel design problem.

$$g_2(x) = -x_3 + 0.00954x_3 \le 0,$$
$$g_3(x) = -\pi x_3^2 x_4 - 4/3(\pi x_3^3) + 1296000 \le 0,$$
$$g_4(x) = x_4 - 240 \le 0,$$

where

$$0 \le x_1 \le 99,$$
$$0 \le x_2 \le 99,$$
$$10 \le x_3 \le 200,$$
$$10 \le x_4 \le 200,$$

For this optimization problem, there are many scholars utilize different algorithms to solve it, such as memory based Hybrid Dragonfly algorithm(MHDA) [38], Cuckoo Search algorithm (CS) [39], improved Particle Swarm Optimizer (IPSO) [40], Artificial Bee Colony algorithm (ABC) [41] and

TABLE 9. Comparison the best solution obtained by different algorithms for pressure vessel design.

Optimum variables	MHDA	CS	IPSO	ABC	PGABC	pFA
x_1	0.778169	0.812500	0.812500	0.812500	0.7781686	**0.957100**
x_2	0.384649	0.437500	0.437500	0.437500	0.3846491	**0.005900**
x_3	40.3196	42.0984456	42.098445	42.098446	40.3210545	**49.554600**
x_4	200	176.6363595	176.6365950	176.636596	199.9802367	**101.976400**
$f(x)$	5885.3353	6059.7143348	6059.7143	6059.714339	5885.40322828	**2727.320000**

TABLE 10. The statistical results obtained by different algorithms for pressure vessel design.

Algorithm	Min	Worst	Mean	Std
MHDA	5885.3353	5885.3353	5885.3353	**0**
CS	6059.714	6495.3470	6447.7360	502.693
IPSO	6059.7143	NA	6289.92881	305.78
ABC	6059.714339	NA	6245.308144	205
PGABC	5885.403285	5895.126804	5887.557024	2.745290
pFA	**2727.320000**	**4511.1000**	**3517.000**	562.9108

TABLE 11. Comparison the best solution obtained by different algorithms for structure design of tension-pressure spring.

Optimum variables	CPSO	MPM	IOD	SPA	TGA	pFA
x_1	0.051728	0.050000	0.053396	0.051480	0.051989	**0.051825**
x_2	0.357644	0.315900	0.399180	0.351661	0.363965	**0.359999**
x_3	11.244543	14.250000	9.185400	11.632201	10.890522	**11.099404**
$f(x)$	0.0126747	0.0128334	0.0127303	0.0127048	0.0126810	**0.0126652**

penalty guided ABC (PGABC) [42]. To be fair, the maximum iteration is set to 1500, and the population size is set to 50. For each algorithm, program runs 30 times independently. The best solutions obtained by different algorithms are shown in Table 9. In order to verify the performance of pFA, we compare different algorithms by min, worst, mean and std. The results are given in Table 10. And except for the statistics of pFA, statistics of others are derived from literature [38].

Frm Table 11, we can see that, for pressure vessel design problem, pFA can obtain the best result. In Table 10, considering min, worst and mean, we can know that the results obtained by pFA are superior apparently to others, and for standard deviation, pFA is worse than other algorithms. Based on above analysis, we can draw a conclusion: the performance of pFA is superior to others, and it is more applicable to solve the pressure vessel design problem.

B. STRUCTURE DESIGN OF TENSION-PRESSURE SPRING

The structure design of tension-pressure spring problems (see Fig. 6) is chosen from literature [43], it aimed mainly at minimizing its quality under satisfying the constraint conditions. The constraint conditions include minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. There are three design variables: coil diameter of spring ($d(x_1)$), average diameter of spring coil ($D(x_2)$) and the effective number of circles ($P(x_3)$).

The optimization problem can be written as follows:

$$\min f(x) = (x_3 + 2)x_2x_1^2$$

$$\text{subject to } g_1(x) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0,$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0,$$

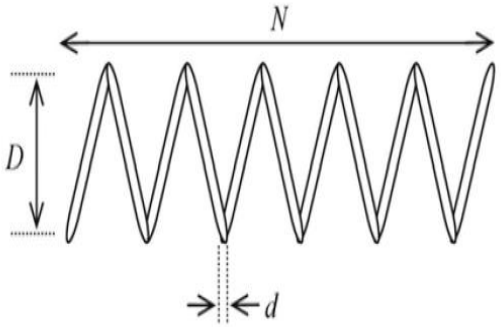

FIGURE 6. Structure design of tension-pressure spring.

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0,$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0,$$

where

$$0.05 \leq x_1 \leq 2,$$
$$0.25 \leq x_2 \leq 1.3,$$
$$2 \leq x_3 \leq 15,$$

For this optimization problem, it was solved by different algorithms, such as an effective co-evolutionary particle swarm optimization (CPSO) [43], a study of mathematical programming methods (MPM) [44], introduction to Optimum Design (IOD) [45], a self-adaptive penalty approach (SPA) [46] and genetic algorithms through the use of dominance-based tournament selection (TGA) [47]. For each algorithm, program runs 30 times independently.

TABLE 12. The statistical results obtained by different algorithms for structure design of tension-pressure spring.

Algorithm	Min	Worst	Mean	Std
CPSO	0.0126747	0.012730	0.012924	5.198500e-005
MPM	0.0128334	NA	NA	NA
IOD	0.0127303	NA	NA	NA
SPA	0.0127048	0.012769	0.012822	3.939000e-005
TGA	0.0126810	0.0127420	0.012973	5.900000e-005
pFA	**0.0126652**	**0.0126735**	**0.0126676**	**2.311605e-006**

TABLE 13. Comparison the best solution obtained by different algorithms for three-bar truss design.

Optimum variables	BWOA	MBA	MFO	DEDS	CS	pFA
x_1	0.788666327	0.7885650	0.788244770931922	0.78867513	0.0.78867	**0.788676772**
x_2	0.408273202	0.4085597	0.40946695784741	0.40824828	0.40902	**0.408243657**
$f(x)$	263.8958435	263.8958522	263.895979682	263.8958434	263.9716	**263.8958433**

TABLE 14. Comparison the best solution obtained by different algorithms for I-beam vertical deflection.

Optimum variables	ARSM	Improved ARSM	CS	SOS	pFA
x_1	37.05	48.42	50.000000	50.00000	**50.00000**
x_2	80.00	79.99	80.000000	80.00000	**80.00000**
x_3	1.71	0.90	0.900000	0.90000	**1.36985**
x_4	2.31	2.40	2.3216715	0.32179	**5.00000**
$f(x)$	0.0157	0.0131	0.0130747	0.0130741	**0.007100**

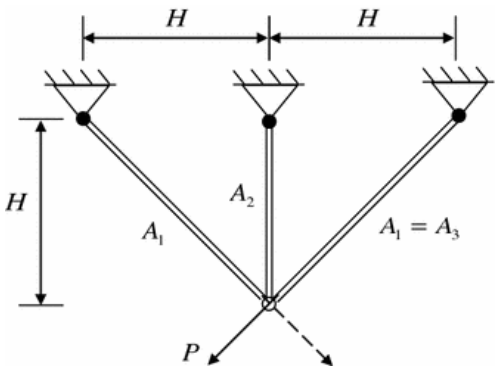

FIGURE 7. Three-bar truss design.

The best solutions obtained by different algorithms are shown in Table 11. In order to verify the performance of pFA, we compare different algorithms by min, worst, mean and std. The results are given in Table 12. And except for the statistics of pFA, statistics of others are derived from literature [43].

As shown in Tables 11 and 12, for the structure design of tension-pressure spring, compared with other algorithms, pFA can find the best solution and obtain the best objective function value. For pFA, its worst solution is better than the best solution obtained by other algorithms, and its mean and standard deviation are also better than others. That is to say, the pFA is better than others in solving this problem.

C. THREE-BAR TRUSS DESIGN

For this problem, to minimize the weight subject to stress, deflection, and buckling constraints, the two parameters $A_1(x_1)$ and $A_2(x_2)$ should be optimized(see Fig. 7). The problem of three-bar truss design has been studied by many scholars. In order to solve this case, Chen [54] proposed the balanced variant of WOA, that is BWOA. Sadollah [55] utilized Mine blast algorithm (MBA) to solve this problem. Moreover, there are other algorithms used to optimize it. Such as, Mirjalili [56], Zhang [57] and Gandomi [58] adopted MFO, DEDS and CS, respectively.

The optimization problem can be written as follows:

$$\begin{aligned}
&\min f(x) = (2\sqrt{2}x_1 + x_2) \times l \\
&\text{subject to } g_1(x) = P(\sqrt{2}x_1 + x_2)/(\sqrt{2}x_1^2 + 2x_1x_2) - \sigma \le 0, \\
&\qquad g_2(x) = Px_2/(\sqrt{2}x_1^2 + 2x_1x_2) - \sigma \le 0, \\
&\qquad g_3(x) = P/(\sqrt{2}x_2 + x_1) - \sigma \le 0,
\end{aligned}$$

where

$$\begin{aligned}
&0 \le x_1 \le 1, \\
&0 \le x_2 \le 1, \\
&l = 100cm, \quad P = 2kN/cm^2, \ \sigma = 2kN/cm^2.
\end{aligned}$$

For each algorithm, it runs 30 times independently. NP is set 20. The results of the above-mentioned algorithms are given in Table 13. Observing the Table 13, we can see that the best weight of the problem of three-bar truss design is 263.8958433 when x_1 and x_2 are set as 0.788676772, 0.408243657, respectively, which is obtained by pFA. For BWOA, MBA, MFO, DEDS and CS, the results obtained by them are worse than that of pFA's.

D. I-BEAM VERTICAL DEFLECTION

For I-beam vertical deflection problem(see Fig. 8), its aim is to minimize the vertical deflection of an I-beam. Moreover, The cross-sectional area and stress constraints should be satisfied at the same time. For this problem, there are

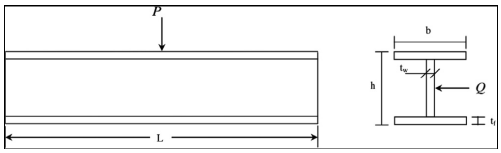

FIGURE 8. I-beam vertical deflection.

4 variables: length(b), height(h), and two thick-nesses of this problem(t_w,t_f). For convenience, we set this 4 variables as x_1, x_2, x_3 and x_4, respectively.

The optimization problem can be written as follows:

$$\begin{aligned}
\min\ & f(x) = 5000/(x_3(x_2 - 2x_4)/12 + x_1x_4^3/6 \\
& \quad + 2x_1x_4((x_2 - x_4)/2)^2) \\
\text{subject to}\ & g_1(x) = 2x_1x_3 + x_3(x_2 - 2x_4 - 300 \le 0, \\
& \quad g_2(x) = 18x_2 \times 10^4/(x_3(x_2 - 2x_4)^3 + 2x_1x_3(4x_4^2 \\
& \quad + 3x_2(x_2 - 2x_4))) + 15x_1 \times 10^3/((x_2 - 2x_4)x_3^3 \\
& \quad + 2x_3x_1^3) - 56 \le 0,
\end{aligned}$$

where

$$\begin{aligned}
10 \le x_1 \le 50, \\
10 \le x_2 \le 80, \\
0.9 \le x_3 \le 5, \\
0.9 \le x_4 \le 5,
\end{aligned}$$

For each algorithm, it runs 30 times independently; NP is set to 20. The results obtained by ARSM[59], improved ARSM[59], CS[58], SOS[60] and pFA are given in Table 14. From the statistical data in Table 14, it can be seen that the optimal value of this problem is 0.007100, which is obtained by pFA. Meanwhile, for ARSM, improved ARSM, CS and SOS, the best result is 0.0130741, which is much worse than that of pFA's.

VIII. CONCLUSION

In this paper, in order to reduce computational time complexity, speed up the convergence, avoid oscillation in the iteration and overstep the local optimum, a novel firefly algorithm (pFA) was proposed. First, it utilizes the probability p_{fit} to choose a firefly with high quality, and then the opposite learning skill is employed to generate a new firefly in this situation, where all fireflies are worse than x_i.

In pFA, we use the probability selection to determine the firefly that x_i move towards to. In this way, the attraction among fireflies can be reduced, thus, it can avoid oscillation, Improve the precision of solution and speed up convergence. And the usage of opposite learning strategy makes the diversity of population increased and enhance exploration. The comparison results show that the performance of pFA is superior to others. Furthermore, it also obtains satisfying results when pFA is applied to solve four engineering optimization problems.

In the future, we will do more experiments to test the performance of pFA, for example, we can use it to deal with multi-objective optimization problems and cluster.

REFERENCES

[1] D. Martens, B. Baesens, and T. Fawcett, "Editorial survey: Swarm intelligence for data mining," *Mach. Learn.*, vol. 82, no. 1, pp. 1–42, Jan. 2011.
[2] K. Vadim, "Overview of different approaches to solving problems of data mining," *Procedia Comput. Sci.*, vol. 123, pp. 234–239, Feb. 2018.
[3] Y. Liu, "The fully polynomial approximation algorithm for the 0-1 knapsack problem," *Theory Comput. Syst.*, vol. 35, no. 5, pp. 559–564, Oct. 2002.
[4] S. Sahni, "Approximate algorithms for the 0/1 knapsack problem," *J. ACM*, vol. 22, no. 1, pp. 115–124, Jan. 1975.
[5] Z. Fu, R. Eglese, and L. Y. O. Li, "Erratum: A new tabu search heuristic for the open vehicle routing problem," *J. Oper. Res. Soc.*, vol. 57, no. 8, p. 1018, Aug. 2006.
[6] W. Y. Szeto, Y. Z. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *Eur. J. Oper. Res.*, vol. 215, no. 1, pp. 126–135, Nov. 2011.
[7] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in *Foundations of Fuzzy Logic and Soft Computing*, vol. 4529. Berlin, Germany: Springer, 2007, pp. 789–798.
[8] D. Karaboga, B. Akay, and C. Ozturk, "Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks," in *Modeling Decisions for Artificial Intelligence*, vol. 4617. Berlin, Germany: Springer, 2007, pp. 318–329.
[9] K. Singh and S. Sundar, "Artifical bee colony algorithm using problem-specific neighborhood strategies for the tree t-spanner problem," *Appl. Soft Comput.*, vol. 62, pp. 110–118, Jan. 2018.
[10] M. Dorigo and T. Stutzle, *Ant Colony Optimization* (A Bradford Book). London, U.K.: MIT Press, 2004.
[11] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, pp. 687–697, Jan. 2008.
[12] S. Alizadeh and M. Ghazanfari, "Learning FCM by chaotic simulated annealing," *Chaos, Solitons Fractals*, vol. 41, no. 3, pp. 1182–1190, Aug. 2009.
[13] X. S. Yang and S. Deb, "Engineering optimization by cuckoo search," *Int. J. Math. Model. Numer. Optim.*, vol. 1, no. 4, pp. 330–343, Dec. 2010.
[14] Y. Suzuki and J. D. Cortes, "A tabu search with gradual evolution process," *Comput. Ind. Eng.*, vol. 100, pp. 52–57, Oct. 2016.
[15] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
[16] X. S. Yang, "Firefly algorithm,"in *Engineering Optimization*. Hoboken, NJ, USA: Wiley, 2010.
[17] X.-S. Yang and X. He, "Firefly algorithm: Recent advances and application," *Int. J. Swarm Intell.*, vol. 1, no. 1, pp. 36–55, Aug. 2013.
[18] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792. Berlin, Germany: Springer, 2009, pp. 169–178.
[19] X. S. Yang, "Firefly algorithm," in *Nature-Inspired Metaheuristic Algorithms*. London, U.K.: Luniver Press, 2008.
[20] M. Kaur and S. Ghosh, "Network reconfiguration of unbalanced distribution networks using fuzzy-firefly algorithm," *Appl. Soft Comput.*, vol. 49, pp. 868–886, Dec. 2016.
[21] M. L. Gao, L.-L. Li, X.-M. Sun, L.-J. Yin, H.-T. Li, and D.-S. Luo, "Firefly algorithm (FA) based particle filter method for visual tracking," *Optik-Int. J. Light Electron Opt.*, vol. 126, no. 18, pp. 1705–1711, 2015.
[22] L. Aydin, "A new approach based on firefly algorithm for vision-based railway overhead inspection system," *Measurement*, vol. 74, pp. 43–55, Oct. 2015.
[23] K. C. Udaiyakumar and M. Chandrasekaran, "Application of firefly algorithm in job shop scheduling problem for minimization of makespan," *Procedia Eng.*, vol. 97, pp. 1798–1807, Dec. 2014.
[24] S. Yu *et al.*, "A variable step size firefly algorithm for numerical optimization," *Appl. Mathematics Comput.*, vol. 263, pp. 214–220, Jul. 2015.
[25] S. Yu, S. Su, Q. Lu, and L. Huang, "A novel wise step strategy for firefly algorithm," *Int. J. Comput. Math.*, vol. 91, no. 12, pp. 2507–2513, May 2014.

[26] I. Fister *et al.*, "Memetic firefly algorithm for combinatorial optimization," in *Proc. 5th Int. Conf. Bioinspired Optim. Methods Appl.*, May 2012, pp. 75–86.
[27] H. Wang *et al.*, "Firefly algorithm with adaptive control parameters," *Soft Comput.*, vol. 21, no. 17, pp. 5091–5102, Sep. 2017.
[28] P. Sivaranjani and A. S. Kumar, "Hybrid particle swarm optimization-firefly algorithm (HPSOFF) for combinatorial optimization of non-slicing VLSI floorplanning," *J. Intell. Fuzzy Syst.*, vol. 32, no. 1, pp. 661–669, Jan. 2017.
[29] X. W. Xia *et al.*, "A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm," *J. Comput. Sci.*, vol. 26, pp. 488–500, May 2018.
[30] A. M. Beigi and A. Maroosi, "Parameter identification for solar cells and module using a hybrid firefly and pattern search algorithms," *Sol. Energy*, vol. 171, no. 1, pp. 435–446, Sep. 2018.
[31] W. Wang *et al.*, "A hybrid firefly algorithm for continuous optimization problems," in *Proc. Int. Conf. Cloud Comput. Secur.*, vol. 10040, Nov. 2016, pp. 522–531.
[32] A. Rahmani and S. A. Mirhassani, "A hybrid firefly-genetic algorithm for the capacitated facility location problem," *Inf. Sci.*, vol. 283, pp. 70–78, Nov. 2014.
[33] R. Goel and R. Maini, "A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems," *J. Comput. Sci.*, vol. 25, pp. 28–37, Mar. 2018.
[34] A. H. Gandomi, X-S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 1, pp. 89–98, Jan. 2013.
[35] R. Kotteeswaran and L. Sivakumar, "Levy guided firefly algorithm based tuning of decentralised PI controller of nonlinear multivariable system—Coal gasifier," *IFAC Proc. Volumes*, vol. 47, no. 1, pp. 127–134, Mar. 2014.
[36] H. Wang, W. Wang, H. Sun, and S. Rahnamayan, "Firefly algorithm with random attraction," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 1, pp. 33–41, Feb. 2016.
[37] H. Wang *et al.*, "Firefly algorithm with neighborhood attraction," *Inf. Sci.*, vol. 382, pp. 374–387, Mar. 2017.
[38] S. K. S. Ranjini and S. Murugan, "Memory based hybrid dragonfly algorithm for numerical optimization problems," *Expert Syst. Appl.*, vol. 83, pp. 63–78, Oct. 2017.
[39] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, Jan. 2013.
[40] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," *Eng. Optim.*, vol. 36, no. 5, pp. 585–605, Oct. 2004.
[41] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1001–1014, 2012.
[42] H. Garg, "Solving structural engineering design optimization problems using an artificial bee colony algorithm," *J. Ind. Manage. Optim.*, vol. 10, no. 3, pp. 777–794, Nov. 2014.
[43] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Eng. Appl. Artif. Intell.*, vol. 20, pp. 89–99, Feb. 2007.
[44] A. D. Belegundu, "A study of mathematical programming methods for structural optimization," Dept. Civil Environ., Univ. Iowa, Iowa City, Iowa, 1982.
[45] J. S. Arora, *Introduction to Optimum Design*. New York, NY, USA: McGraw-Hill, 1989.
[46] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Comput. Ind.*, vol. 41, no. 2, pp. 113–127, Mar. 2000.
[47] C. A. C. Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Adv. Eng. Inform.*, vol. 16, pp. 193–203, Jul. 2002.
[48] W. J. S. Gomes, A. T. Beck, R. H. Lopez, and L. F. F. Miguel, "A probabilistic metric for comparing metaheuristic optimization algorithms," *Struct. Saf.*, vol. 70, pp. 59–70, Jan. 2018.
[49] G. Zhenyu, C. Bo, Y. Min, and C. Binggang, "Self-adaptive chaos differential evolution," in *Proc. Int. Conf. Natural Comput.*, vol. 4221, 2006, pp. 972–975.
[50] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.
[51] N. Noman, D. Bollegala, and H. Iba, "An adaptive differential evolution algorithm," *IEEE Congr. Evol. Comput.*, Jul. 2011, pp. 2229–2236.
[52] S. Wang, Y. Li, and H. Yang, "Self-adaptive differential evolution algorithm with improved mutation mode," *Appl. Intell.*, vol. 47, no. 3, pp. 644–658, Oct. 2017.
[53] E. Mezura-Montes and C. A. C. Coello, "An empirical study about the usefulness of evolution strategies to solve constrained optimization problems," *Int. J. Gen. Syst.*, vol. 37, no. 4, pp. 443–473, 2008.
[54] H. Chen, Y. Xu, M. Wang, and X. Zhao, "A balanced whale optimization algorithm for constrained engineering design problems," *Appl. Math. Model.*, vol. 71, pp. 45–59, Jul. 2019.
[55] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2592–2612, May 2013.
[56] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.
[57] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Inf. Sci.*, vol. 178, no. 15, pp. 3043–3074, Aug. 2008.
[58] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–25, Jan. 2013.
[59] G. Wang, "Adaptive response surface method using inherited latin hypercube design points," *J. Mech. Des.*, vol. 125, no. 2, pp. 210–220, Jun. 2003.
[60] M.-Y. Cheng and D. Prayogo, "Symbiotic Organisms Search: A new metaheuristic optimization algorithm," *Comput. Struct.*, vol. 139, pp. 98–112, Jul. 2014.

CHUNFENG WANG was born in Kaifeng, Henan, China, in 1978. He received the M.S. degree from Henan Normal University, in 2006, and the Ph.D. degree from Xidian University, in 2012. He has published over 40 papers in domestic and foreign academic journals. He has applied for one national invention patents. He hosted and participated in four national natural science foundation and two provincial projects. His research interests include swarm intelligent algorithms, optimization theory and its applications, and Bayesian network learning.

XINYUE CHU was born in Zhoukou, Henan, China, in 1992. She is currently pursuing the M.S. degree with Henan Normal University. She has published one paper in foreign academic journal.