International Journal of Hybrid Information Technology
Vol. 9, No. 5 (2016), pp. 131-144
http://dx.doi.org/10.14257/ijhit.2016.9.5.11

Improved Particle Swarm Optimization with Dynamic Fractional Order Velocity and Wavelet Mutation

Lingyun Zhou[1, 2] and Lixin Ding[1, *]

[1]*State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China*
[2]*College of Computer Science, South-Central University for Nationalities, Wuhan, Hubei 430074, China*
zhouly@mail.scuec.edu.cn[1], lxding@whu.edu.cn[*]

Abstract

Particle Swarm Optimization (PSO) is one of the most powerful algorithms for optimization. Traditional PSO algorithm tends to suffer from slow convergence and trapping into local optimum. In this paper, an improved PSO algorithm is proposed by combining dynamic fractional order technology and the wavelet mutation strategy. In the proposed method, a dynamic fractional order velocity update equation is designed to control the convergence rate. Furthermore, the wavelet mutation mechanism is employed to improve the swarm diversity and escape from the local optimums. The experimental results show that the proposed algorithm can provide fast convergence speed and high convergence precision based on the ten classic test functions.

Keywords*: Dynamic fractional order; Wavelet mutation; Particle swarm optimization; Convergence rate*

1. Introduction

Particle Swarm Optimization (PSO) which was developed by Eberhart and Kennedy is a population-based stochastic optimization algorithm [1]. It mimics the group behavior as fish schooling and birds flocking. The algorithm is initialized with a population of particles, where each particle betokens a plausible solution in a D-dimensional space. Then each particle updates its new flying velocity according to its own best experience and entire swarm's best experience. Subsequently, the particle is updated with its new position.

Assume a *D*-dimensional optimization problem *f* and a swarm consisting of n particles. The current position of the *i*th particle is a vector x_i, and the velocity of this particle is also a vector v_i. The best position encountered by the *i*th particle is denoted as P_i and P_g is the best position found by the entire population group. The swarm is manipulated in some form resembling the following equations:

$$\begin{aligned} v_i(t) = v_i(t-1) + c_1 r_1 (P_i(t-1) - x_i(t-1)) \\ + c_2 r_2 (P_g(t-1) - x_i(t-1)) \end{aligned} \quad (1)$$

$$x_i(t) = x_i(t-1) + v_i(t) \ , \quad (2)$$

where $i = 1, 2, \dots, n$ is the particle's index and *t* indicates the iteration number. c_1 and c_2 are positive constants, which are referred to as cognitive and social parameters, respectively. And r_1 and r_2 is random numbers that is uniformly distributed within the interval [0, 1].

PSO has become one of the most popular optimization techniques and has been successfully applied to the many areas such as the social model, digital image processing, and pattern recognition and so on. However, like other population-based optimization techniques, the original PSO has the shortcomings as slow convergence and getting into a local optimum easily. Therefore, how to accelerate the convergence speed and how to avoid the local optimal solution are two important issues in the PSO research [2].

In order to overcome the problems of PSO, a number of PSO variants have been developed. Some schemes suggested adjusting parameters such as the constriction factor, acceleration coefficient and inertia weight [3-5]. The inertia weight which is the most widely employed in the PSO variants is introduced to influence convergence speed. And as explored in [5], the inertia weight and the constriction factor are mathematically equivalent in some cases. The work indicates that a relatively large inertia weight is better for global exploration, while a small inertia weight enhances the ability of local exploitation. Subsequently, in order to better maintain the balance between local search and global search, linearly decreasing methods [6] and fuzzy methods [7] have also been developed to adjust the inertia weight over the course of search process. Generally, adjusting parameters can improve the performance of PSO, but the particles are still easily to trap into local optima, the problem of premature convergence remains.

Topological structures have also been studied widely such as ring, pyramid, ortho-cyclic [8], and so on. Kennedy [9] indicated that a small neighborhood might perform better on complex or multimodal problems, while a large neighborhood is more conducive to simple or unimodal optimization problems. Further, in order to improve the robustness of the PSO, dynamically changing neighborhood structures have been proposed [10]. In addition, Mendes *et al.* [11] introduced a fully informed particle swarm which used five topologies, where the particles make use of the information from all the neighbors around it. This method is proven to be very effective in solving single objective global optimization problems. However, it is not suitable for multimodal optimization problems because of the topology-based neighborhood selection method. B. Y. Qu *et al.* [12] proposed a distance-based locally informed PSO for multimodal optimization, which makes use of the neighborhood information to realize niching behavior. In this method, the neighborhoods are estimated in terms of Euclidean distance so as to form different stable niches that can converge to different global peaks. Usually, the PSO variants with different neighborhood structures have an enhanced ability to prevent premature convergence. However, but the disadvantage is that it is difficult to select the suitable topological structures and the neighborhood size in practice. How to choose the neighbors determines how diverse the influence will be and how efficient the algorithm will be.

Another area of focus is to explore the learning strategies for particles. In CLPSO [13], the proposed comprehensive learning strategy encourages each particle to learn from different particles on different dimensions. This learning strategy can keep the diversity of the swarm and eliminate premature convergence. In OLPSO [14], an orthogonal learning strategy is constructed via orthogonal experimental design to avoid the oscillation phenomenon in PSO and guides the particles to fly in better directions. Wang *et al.* [15] investigated integrating a generalized opposition-based learning (GOBL) strategy, where GOBL is employed to escape from local optimum. Inspired by the scheme, multiple good examples can guide a crowd towards making progress. ELPSO uses an example set of multiple global best particles to update the positions of the particles in order to maintain better diversity. These learning strategies improve the search performance, but it may increase the number of the object function evaluation.

Integration of other evolutionary algorithms and optimization techniques with PSO is also a hot topic. Angeline [16] has first introduced into PSO a selection operator that used in a genetic algorithm. Apart from selection, crossover, and mutation operations that have been adopted from genetic algorithm, more other operations have also been employed to overcome the drawback of trapping in the local optima. Ahmed *et al.* [17] proposed a hybrid PSO (HPSOM), in which a constant mutating space is used in mutations. However, the mutating space is kept unchanged all the time throughout the search. In search of a better model of mutation operations using the PSO algorithm, the HPSOWM which was formulated by [18], adopts a mutation with a dynamic mutating space by incorporating a wavelet function. The PSO's mutating space is dynamically varying along the search based on the properties of the wavelet function. The HPSOWM perform more efficiently than the PSO with constriction and inertia weight factors and other hybrid PSOs due to the wavelet mutation strategy [18].

Recently, Pires *et al*. [19] presented a novel method (FPSO) for controlling the convergence rate of PSO by using fractional calculus concepts. This approach was tested for several well-known functions and the results showed that it contributes to improve the convergence rate of PSO. The schemes [20-22] successfully applied the method to image segmentation, and compared it with other modified PSO algorithm, the experiment results showed that the method converged very well. However, the algorithm converged fast at the beginning but converged slow in the late, and the accuracy of the search solution is not high.

Aiming to maintain the fast converge rate of the fractional order PSO while obtain higher search quality, this paper proposes a dynamic fractional-order PSO combining with the wavelet mutation based on the work of the paper [19, 23]. The remainder of this paper is organized as follows. Section II after briefly introduces the PSO with the fractional order velocity, the proposed improved PSO algorithm with dynamic fractional order velocity (IFWPSO) is discussed. Section III presents in detail the results of numerical experiments on some well-known test functions. Finally, Section IV concludes with some discussions.

2. Proposed Algorithm

2.1. PSO with Fractional Order Velocity

The fractional calculus is a generalization of the conventional integration and differentiation to include non-integer values in the powers of the derivatives or integrals. Many natural phenomena can be more accurately modeled by fractional differ-integrals. It has attracted the attention of many researchers, being applied in various scientific fields such as engineering, computational mathematics, fluid mechanics. The discrete time implementation of the Grűnwald-Letnikov definition of fractional derivative with fractional coefficient of is defined as [24]:

$$D^{\alpha}[x(t)] = \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^{k}\Gamma(\alpha+1)\,x(t-kT)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \tag{3}$$

where $\alpha \in [0,1]$, $\Gamma(\cdot)$ is the gamma function. T is the sampling period and r is the truncation order.

An important property revealed by the equation (3) is that an integer-order derivative just implies a finite series while the fractional-order derivative needs an infinite number of terms. Therefore, integer derivatives are local operators, while fractional derivatives have a memory of all the past events. However, the influence of past events decreases over time. These characteristics make fractional calculus well suited to describe the dynamic phenomena of particle's trajectory fully.

E.J. Solteiro Pires [19] first applied the Grűnwald-Letnikov definition of fractional derivative to the PSO, considering only the first r=4 terms of differential derivative given by (3), and the velocity update equation with fractional derivative was given by the equation:

$$v(t) = \alpha v(t-1) + \frac{1}{2}\alpha(1-\alpha)v(t-2) + \frac{1}{6}\alpha(1-\alpha)(2-\alpha)v(t-3)$$
$$+ \frac{1}{24}\alpha(1-\alpha)(2-\alpha)(3-\alpha)v(t-4) \tag{4}$$
$$+ c_1 r_1 (P_i(t-1) - x_i(t-1)) + c_2 r_2 (P_g(t-1) - x_i(t-1))$$

2.2. Dynamic Fractional Order

In this section, the dynamic fractional order strategy is introduced to obtain better convergence rate. In PSO with fractional order velocity update rule (4), the value of α has a vital role in controlling the convergence rate of the PSO algorithm. The results of [23] showed that, when the value of α fixed, the algorithm converged fast at the beginning but converged slow in the late or even fell into stagnation. When α varied linearly, the performance of the algorithm was similar to the algorithm using inertia weight.

To examine the impact of the parameter α on the anytime behavior of PSO with fractional velocity, several tests were done. Those with smaller values of α are initially better, but easily lead to search stagnation. The smaller values more probability result in worse final solution quality. However, a larger value of α are initially worse, but produces much better results towards the end of the run. So, if we set the value of a smaller value in the early running period and then varied the value of α to a larger value, the performance of the algorithm would be improved. To consider this possibility, a function is developed to compute the value of based on the sigmoid function, which is defined by:

$$f(x) = 1/(1+e^{-x}) \tag{5}$$

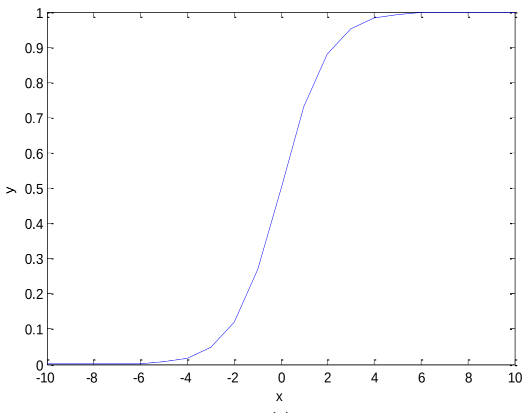
This function is an *S*-type function that can be seen in Figure 1 (a). It will be improved to meet the needs. The value of α is assigned as follows:

$$\alpha = \lambda / (1 + e^{-\beta \times iter/\max\_iter+\theta}), \tag{6}$$

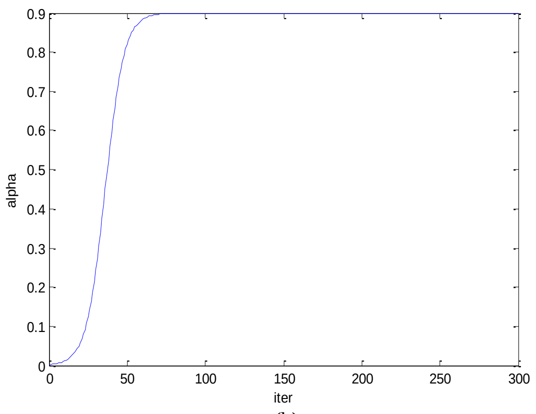
where *iter* represents the current iteration number, *max_iter* is the maximum number of iterations. λ, β and θ are parameters. λ is the vertical scaling factor. The original sigmoid function can be vertical scaling transformation, when $0 < \lambda < 1$, it represents the original sigmoid function compressed in the vertical direction. β is the horizontal scaling factor. The original sigmoid function can be horizontal scaling transformation. The greater the value of β takes, the smaller the original sigmoid function can be horizontally stretched. θ is the translation factor. The value of these parameters can be set according to specific cases. We ensure the value of α in the range (0, 0.9]. For example, when λ=0.9, β=35, θ=5, *max_iter*=300, the value of α varies between 0.0062 and 0.9. The graphics of α can be seen in Figure 1 (b).

2.3. Wavelet Mutation

The PSO algorithm with fractional order velocity can converge faster compared with the PSO, but it still easily falls into local optimal value. To address this problem, the mutation operation can be imposed to increasing the probability of escape from local optima. The wavelet mutation strategy that employed by the PSO was first proposed in [18]. With respect to the Gaussian and Cauchy mutation which were often used by PSO, the wavelet mutation is easier to carry out effective search space search and can effectively balance the global search and local search, since the Morlet wavelet generate positive and negative numbers are the same, and most of the



(a)



(b)

Figure 1. The Sigmoid Function (A) and the Function (B)

energy is concentrated in a smaller range [18]. It is defined by

$$\sigma = \frac{1}{\sqrt{a}} e^{-\left(\frac{\phi}{s}\right)^2 / 2} \cos\left(5\left(\frac{\phi}{s}\right)\right), \qquad (7)$$

where $\phi \in [-2.5s, 2.5s]$, it can be concluded that $\sigma \in [-1, 1]$. s is a parameter and can be defined by

$$s = \exp(-\ln(g) \times (1 - \frac{iter}{\max\_iter})^{\xi} + \ln(g)), \quad (8)$$

where ξ is the shape parameter of the monotonic increasing function, and g is the upper limit of the parameter s. *iter* represents the current iteration number, *max_iter* is the maximum number of iterations. According to equation (8), it can be concluded that $s \in [1, g]$. Assume that the mutation probability of particles is $p \in [0, 1]$, x_{maxi} is the upper limit of the search space and x_{mini} is the lower limit of the search space, the equation can be define as

$$m(x_i^d(t)) = \begin{cases} P_g^d(t) + \sigma \times (x_{\mathrm{maxi}} - P_g^d(t)) & \text{if } \sigma > 0 \\ x_i^d(t) + \sigma \times (P_g^d(t) - x_{\mathrm{mini}}) & \text{if } \sigma \le 0 \end{cases} \qquad (9)$$

As can be seen from (9), after performing mutation, when σ closes to 1, the *d*-dimensional velocity of the *i*th particle will be close to the maximum value of *d*-dimensional velocity. When σ closes to -1, the *d*-dimensional velocity of the *i*th particle will be close to the minimum *d*-dimensional velocity. The mutation probability p is a random number between [0, 1]. When the dimension of the problem is relatively low, the value of p is usually taken between the values [0.8 0.5]. When the dimension of the problem is relatively high, the value of p is usually taken [0.2 0.1]. If the value of p is too large, the number of mutation would increase, which would destroy the current swarm search information and result in poor results and also consume a large amount of running time [18].

2.4. Algorithm Procedure and Analysis

Figure 2 shows the flowchart of the proposed algorithm IFWPSO and the steps involved are given as follows.

Step 1 Initialization: The initial positions of all particles are generated randomly within the D-dimensional search space with velocities initialized to 0.

Step 2 Terminal Condition Check: If the terminal condition is needed, the algorithm terminates. Otherwise, go to Step 3 for a new round of iteration. The terminal condition can be the predefined maximum evaluation number of iterations or reaches the specified precision.

Step 3 velocity Updating: If the current iteration number is less than 5, every particle follows the velocity update rule (1) and save the history velocity. If the current iteration number is greater than or equal to 5, every particle follows the velocity update rule (4) and updates the history velocity.

Step 4 Position Updating: Every particle follows the position update rule (2) to adjust its position. The position needs to be clamped by the search space.

Step 5 Wavelet Mutation Operation: The mutation operation is used to mutate the elements of particles. The details of the operation are as follows. Generates a random number $z \in [0, 1]$. If z is less than a predefined parameter $p \in [0, 1]$, then the wavelet mutation operation will be performed according to the rule (7)-(9). For each particle, a random number k will be generated between $[0, D]$, the kth-dimension of the particle is selected for the mutation.

Step 6 Evaluating: Evaluate the new population. Update the personal best position and the group best position. Go to Step 2.

The proposed IFWPSO is actually using the dynamic fractional order velocity equation to update the particle velocity and combined with wavelet mutation in the basic framework of original PSO algorithm. IFWPSO contains three main parts, including initialization, particle velocity and position update and wavelet mutation operation. The time complexity of the former two is the same as the PSO, that is O $(n \bullet D)$, and the last one is O (D). The dynamic fractional order strategy and the wavelet mutation strategy will not increase the time complexity of the PSO algorithm. So, the time complexity of the IFWPSO is O $(n \bullet D)$, which is a similar computational complexity with the PSO. However, the memory complexity of the IFWPSO is larger than the PSO since the dynamic fractional order strategy needs to track the last four steps of each particle’s velocity.

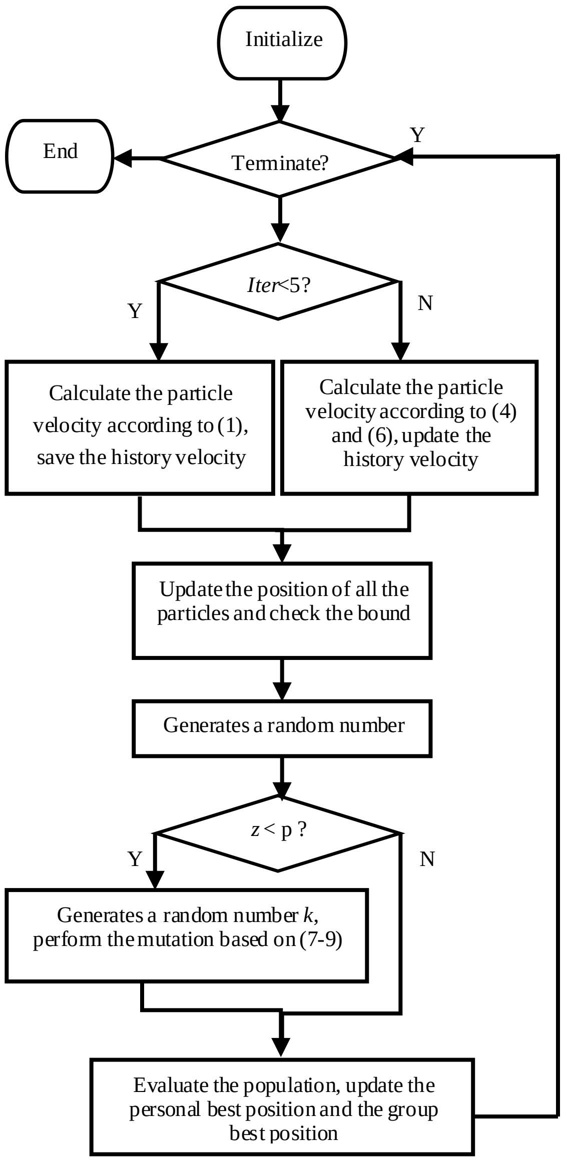

Initialize
End
Terminate?
Y
Iter<5?
Y
N
Calculate the particle velocity according to (1), save the history velocity
Calculate the particle velocity according to (4) and (6), update the history velocity
Update the position of all the particles and check the bound
Generates a random number
z < p ?
Y
N
Generates a random number k, perform the mutation based on (7-9)
Evaluate the population, update the personal best position and the group best position

Figure 2. Flowchart of IFWPSO

3. Experimental Results

To test the performance of the IFWPSO, computational experiments are designed and implemented. The computational procedures described above have been implemented in MATLAB environment on the hardware environment is an Intel Core 3.20 GHz CPU, 4GB RAM computer. This section presents and duly discusses the experimental results of our comparative study. The proposed algorithm are tested on ten test functions, including the five functions that directly taken from literature [19] and another five widely used test functions. They are listed in Table 1.

Table 1. Test Functions

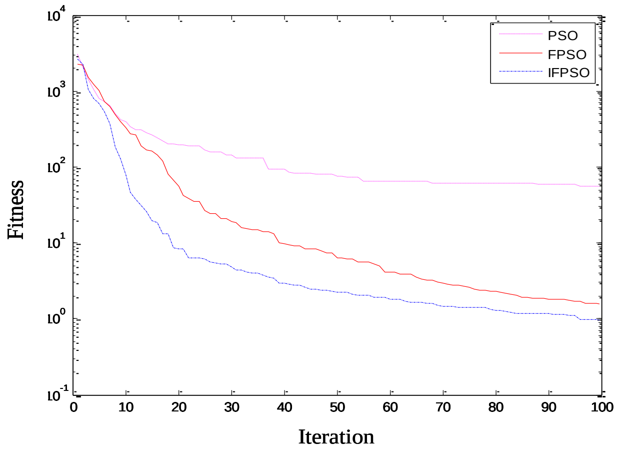
	Function name	Test function	Dimension	Domain range	Minimum
f1	Bohachevsky	$x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	[-50, 50]	0.0
f2	Colville	$100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	4	[-10, 10]	0.0
f3	Drop wave	$-\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}$	2	[-10, 10]	-1.0
f4	Easom	$-\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	2	[-100, 100]	-1.0
f5	Rastrigin	$\sum_{i=1}^{30} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12, 5.12]	0.0
f6	Sphere	$\sum_{i=1}^{30} x_i^2$	30	[-100, 100]	0.0
f7	Rosenbrock	$\sum_{i=1}^{30} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-2, 2]	0.0
f8	Ackley	$-20\exp(-0.2 \times \sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}) - \exp(\frac{1}{30}\sum_{i=1}^{30}\cos(2\pi x_i)) + 20 + e$	30	[-32, 32]	0.0
f9	Griewank	$\frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600, 600]	0.0
f10	Schwefel	$-\sum_{i=1}^{10} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	10	[-500, 500]	-4189.8

Table 2. Success Rates For Test Functions F1- F10

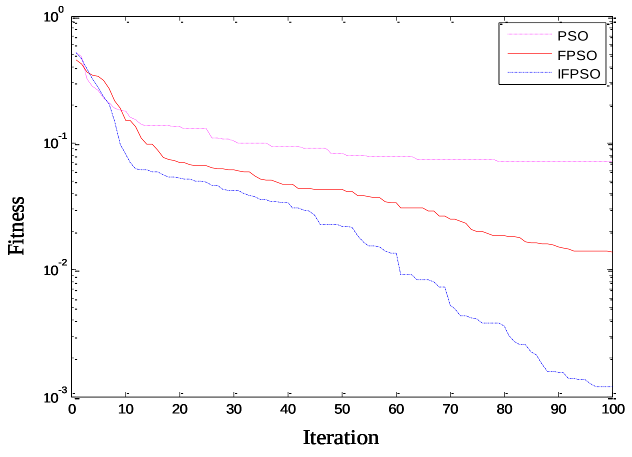
	PSO		**FPSO**		**IFPSO**		**IFWPSO**	
	Iteration	***Success rate***	***Iteration***	***Success rate***	***Iteration***	***Success rate***	***Iteration***	***Success rate***
f1	10000	0	106	100	123	100	**16**	100
f2	10000	0	**306**	100	1536	100	793	100
f3	7724	35	73	100	95	100	**63**	100
f4	4284	75	**662**	100	676	100	1320	100
f5	10000	0	7168	30	360	100	**48**	100
f6	10000	0	3848	65	2234	100	**669**	100
f7	10000	0	3708	65	251	100	**168**	100
f8	10000	0	8146	20	801	100	**647**	100
f9	10000	0	1978	85	2665	100	**1224**	95
f10	10000	0	10000	0	9524	5	**1906**	85

3.1. Impact of Dynamic Fractional Order Velocity on the Convergence of the PSO

With the purpose of studying the impact of dynamic fractional order velocity on the convergence of the PSO, some experiments have been conducted. Here using two benchmark functions: Bohachevsky function and Drop wave function. These two functions are shown in Table as f_1 and f_3, respectively. For each comparison algorithm, the number of the population sets to 30 and the maximum velocity of the particle sets to be half of the search space. In PSO, parameters c_1 and c_2 both can be set to 1.193, inertia weight ω can be set to ω=0.721. In FPSO, the value of parameter α is the same with the literature [12]. In the PSO with the dynamic fractional order velocity (IFPSO), λ=0.9, β=35, θ=5. The maximum number of iterations can be set to 100 for all the algorithms. The results concerning the search behavior of the algorithms on these two functions are illustrated Figure 3. The Figure is plotted based on the mean values of 20 independent runs of each algorithm that can represent the search behavior of the algorithms in most cases. It can be seen that the FPSO converges faster than the PSO and the IFPSO converges fastest.



(a)

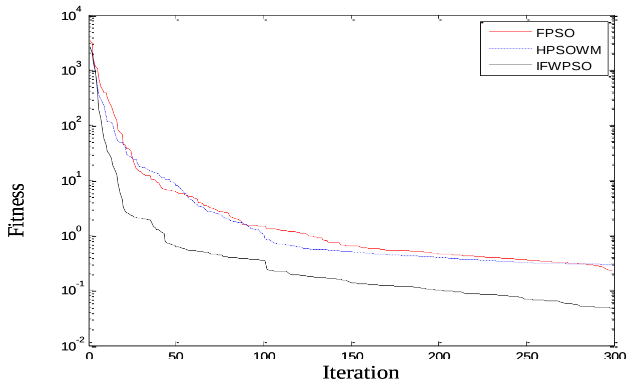


(b)

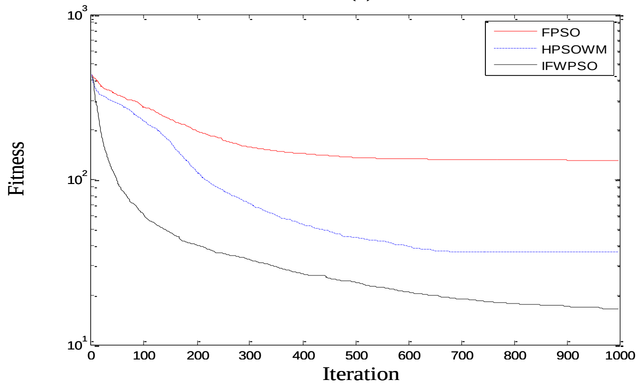
Figure 3. Convergence Curve of Three PSO Algorithms

3.2. Comparison and Analysis

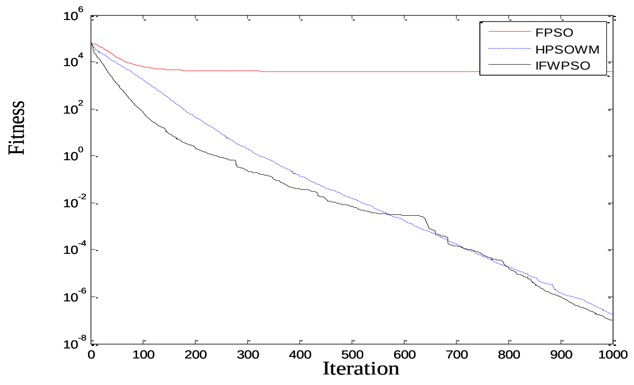
To test the performance of the proposed algorithm (IFWPSO), it is compared with PSO, FPSO [19] and IFPSO on 10 benchmark functions. The parameter settings in this section are as follows. The number of the population sets to 30 and the maximum velocity of the particle sets to be half of the search space. In our simulations, the values of the main parameters are c_1=c_2=1.193, ω=0.721, λ=0.9, β=35, θ=5, ξ=5.0, and g=10000



(a)



(b)



(c)

Figure 4. Convergence Traces of the Algorithms for Three Representative Test Functions. (A) F3. (B) F5. (C) F6

The test functions can be divided into two categories. The first one is the category of the low dimensional function, *f1*- *f4* belong to this type. The second one is the category of high dimensional functions, *f5* - *f10* belong to this type. Parameter p has been set to 0.5 on the first category test functions and *p*=0.1 on the second category test functions. Furthermore, all the algorithms use the same maximum number of function evaluations 10000 in each run for each test function and each algorithm is tested 20 times independently for every function and the mean results are used in the comparison.

The number of iterations needed to successfully reach the stopping criterion and the success rates for all the algorithms on test functions *f*1- *f*10 are recorded and presented in Table 2. The best results are marked in boldface.

Table 3. Solutions Accuracy Comparisons

	FPSO			**HPSOWM**			**IFWPSO**		
	Mean	***Best***	***Deviation***	***Mean***	***Best***	***Deviation***	***Mean***	***Best***	***Deviation***
f1	1.67e-10	3.33e-16	4.04e-10	6.27e-13	**0**	2.43e-12	**0**	**0**	0
f2	2.18e-01	6.99e-04	2.92e-01	2.93e-01	**2.79e-04**	7.50e-01	**4.58e-02**	1.50e-03	4.46e-02
f3	1.05e-09	7.70e-14	3.07e-09	**6.96e-14**	**0**	1.97e-13	5.00e-05	**0**	1.90e-04
f4	7.50e-01	5.45e-13	4.44e-01	**1.63e-13**	**0**	5.19e-13	5.50e-01	**0**	5.10e-01
f5	1.30e+02	5.67e+01	3.64e+01	3.63e+01	2.09e+01	9.82e+00	**1.65e+01**	**7.97e+00**	5.64e+00
f6	4.00e+03	2.06e-07	5.03e+03	1.75e-07	2.79e-08	1.80e-07	**9.89e-08**	**1.04e-09**	1.76e-07
f7	2.33e+02	3.63e+00	2.96e+02	3.02e+01	2.06e+01	1.67e+01	**2.82e+01**	**6.67e+00**	2.16e+01
f8	1.08e+01	9.34e-01	8.60e+00	**8.77e-05**	4.69e-05	3.47e-05	1.96e+00	**2.77e-05**	6.03e+00
f9	4.06e+01	2.90e-07	4.60e+01	**2.32e-02**	1.59e-07	2.75e-02	2.65e-02	**6.15e-09**	2.46e-02
f10	7.75e+01	4.59e+01	1.78e+01	7.12e+01	5.74e+01	1.37e+01	**5.11e+01**	**1.15e+01**	2.09e+01

The results show that IFWPSO converges very fast on most of the functions and it achieves a much higher success rate than other PSO algorithms on most of these 10 test functions. The better performance is due to the better fine search generated by the dynamic fractional order particle velocity and wavelet mutation. Especially for high dimensional function, the number of iterations that IFWPSO needed is much less than that of FPSO. It indicates that FPSO easily falls into local optimal solution in solving complex high dimensional optimization functions. However, IFWPSO can effectively avoid getting into local optimums because of applying wavelet mutation. The addition of wavelet mutation strategy in dynamic fractional order particle swarm algorithm can effectively escape from local optima. Figure 4 gives the convergence traces of the three algorithms over the iterations, from top to bottom are the curves for three representative functions *f2, f5* and *f6.*

In order to compare the solution quality, the solutions obtained by IFWPSO are compared with the ones obtained by FPSO [19] and HPSOWM [18]. According to the complexity of the test functions, the maximum number of iterations is set to 300 for the first category test functions and the maximum number of iterations is set to 1000 for the second one. Table 3 compares the mean values, the best values and the standard deviations of the solutions found. The best results are marked in boldface.

It can be observed that IFWPSO achieve the best solutions on most of the test functions. Experimental results and comparisons verify that the dynamic fractional order velocity combined with wavelet mutation indeed helps the IFWPSO perform better than the traditional PSO and some recent PSO variants on most of the test functions in terms of solution accuracy and convergence speed.

4. Conclusions

In this paper, a dynamic fractional PSO algorithm combining wavelet mutation (IFWPSO) is presented. It uses a dynamic fractional particle velocity update equations to update the particle velocity which can enhance the convergence rate of the algorithm. And it employs the wavelet mutation strategy which can effectively prevent the algorithm from falling into a local optimum. It can be seen from the experimental results that the proposed algorithm can perform better in convergence speed and solution quality, especially for high dimensional optimization functions.

Acknowledgment

The authors thank the financial support for the work from Chinese National Natural Science Foundation (61103248, 61379059), the Fundamental Research Funds for the Central Universities (CZY14011).

References

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", Proceedings of IEEE International Conference on Neural Networks, **(1995)**.
[2] A. Banks, J. Vincent and C. Anyakoha, Natural Computing, vol. 7, **(2008)**.
[3] M. Clerc and J. Kennedy, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space", Evolutionary Computation, IEEE Transactions, vol. 6, **(2002)**, pp. 58-73.
[4] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer", Proceedings of IEEE International Conference on Evolutionay Computation, **(1998)**, pp. 69-73.
[5] R. C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", Proceedings of the 2000 Congress, IEEE, **(2000)**.
[6] Y. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", Proceedings of the 1999 Congress on, IEEE, **(1999)**.
[7] Y. Shi and R. C. Eberhart, "Fuzzy Adaptive Particle Swarm Optimization", Proceedings of the 2001 Congress, IEEE, **(2001)**.
[8] K. Ganapathy, V. Vaidehi, B. Kannan and H. Murugan, "Expert Systems with Applications", vol. 41, **(2014)**.
[9] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", Proceedings of the 1999 Congress, IEEE, **(1999)**.
[10] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator", Proceedings of the 1999 Congress on. IEEE, **(1999)**.
[11] R. Mendes, J. Kennedy and J. Neves, "Evolutionary Computation", IEEE Transactions, vol. 8, **(2004)**.
[12] B. Y. Qu, P. N. Suganthan and S. Das, "Evolutionary Computation", IEEE Transactions, vol. 17, **(2013)**.
[13] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Evolutionary Computation", IEEE Transactions, vol. 10, **(2006)**.
[14] Z. H. Zhan, J. Zhang, Y. Li and Y. H. Shi, "Evolutionary Computation", IEEE Transactions, vol. 15, **(2011)**.
[15] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu and M. Ventresca, "Information Sciences", vol. 181, **(2011)**.
[16] P. J. Angeline, "Using Selection to Improve Particle Swarm Optimization", Proceedings of IEEE International Conference on Evolutionary Computation, **(1998)**.
[17] A. A. A. Esmin, G. Lambert-Torres G and A. C. Z. de Souza, "Power Systems", IEEE Transactions, vol. 20, **(2005)**.
[18] S. H. Ling, H. H. C. Iu, K. Y. Chan, H. K. Lam, B. C. Yeung and F. H. Leung, "Systems, Man, and Cybernetics", Part B: Cybernetics, IEEE Transactions, vol. 38, **(2008)**.
[19] E. J. S. Pires, J. A. T. Machado, P. B. de Moura Oliveira, J. B. Cunha and L. Mendes, "Nonlinear Dynamics, vol. 61, **(2010)**.
[20] M. S. Couceiro, R. P. Rocha, N. M. F. Ferreira and J. T. Machado, "Signal, Image and Video Processing", vol. 6, **(2012)**.
[21] P. Ghamisi, M. S. Couceiro and F. M. L. Martins, "Atli Benediktsson", Journal, vol. 52, **(2014)**.
[22] P. Ghamisi, M. S. Couceiro, J. A. Benediktsson and N. M. Ferreira, "Expert Systems with Applications", vol. 39, **(2012)**.
[23] L. Y. Zhou, S. B. Zhou and A. S. Muhammad, "Nonlinear Dynamic", vol. 77, **(2014)**.
[24] Y. S. Mishura, "Stochastic Calculus for Fractional Brownian Motion and Related Processes", Berlin: Springer, **(2008)**.

Authors

Lingyun Zhou received the B.S. and M.S. degrees in College of Computer Science from South-Central University for Nationalities, China, in 2001 and 2004, respectively. She is a PhD candidate in State Key Laboratory of Software Engineering, Computer School at Wuhan University in China. And she is currently a lecturer in College of Computer Science of South-Central University for Nationalities.

Lixin Ding received his B. Sc. and M. Sc. degrees from the Department of Applied Mathematics, Hunan University, Changsha, China, in 1989 and 1992, respectively, and

Ph. D. degree from the State Key Laboratory of Software Engineering (SKLSE), Wuhan University, Wuhan, China, in 1998. From 1998 to 2000, he was a Postdoctoral researcher at the Department of Armament Science and Technology, Naval University of Engineering, Wuhan, China. He is currently a Professor at the SKLSE, Wuhan University. He is also the Director of Zhuhai R&D Center of SKLSE. He has published more than 80 research articles in domestic and foreign academic journals, such as IEEE Transactions on Communications, IEEE Transactions on Engineering Management, Evolutionary Computation, Computers & Mathematics with Applications, Neurocomputing, Transactions of the Institute of Measurement and Control, Dynamics of Continuous, Discrete and Impulsive Systems, Journal of Network and Computer Applications, Applied Mathematics & Information Sciences, Information, Sensor Letters, KSII Transactions on Internet and Information Systems, Neural, Parallel & Scientific Computation, ICIC Express Letters, Journal of Computational Information Systems, Science China: Information Sciences, Science Bulletin, Mathematica Numerica Sinica *etc*. His research interests include intelligence computation, intelligent information processing, machine learning and cloud computing.