Journal of Information Hiding and Multimedia Signal Processing ©2018 ISSN 2073-4212
Ubiquitous International Volume **9**, Number **6**, November **2018**

EFPSO: An Effective Fuzzy Particle Swarm Optimization and Its Applications

Dongping Tian

Institute of Computational Information Science
Baoji University of Arts and Sciences
No.1 Hi-Tech Avenue, Hi-Tech District, Baoji, Shaanxi 721013, P.R. China

Institute of Computer Software
Baoji University of Arts and Sciences
No.44 Baoguang Road, Weibin District, Baoji, Shaanxi 721016, P.R. China
tiandp@ics.ict.ac.cn, tdp211@163.com

Received January, 2018; Revised May, 2018

Abstract. *Particle swarm optimization (PSO) is a population-based stochastic optimization technique that has been widely applied to solve a number of complex optimization problems. However, PSO is easily trapped into the local optima and appears premature convergence during the search process. To address these issues, we present an effective fuzzy particle swarm optimization (abbreviated as EFPSO) algorithm by introducing a two-input and two-output fuzzy logic controller (FLC) with nine inference rules into the canonical PSO in this paper. To be specific, the increment of global optimum and the maximal focus distance of particles are employed as the two-input variables, while the inertia weight and the constraint factor are adaptively adjusted according to the control information translated from the FLC in the search process of particles. Conducted experiments in the task of benchmark function optimization and standard image segmentation demonstrate that the EFPSO proposed in this paper significantly outperforms several other existing PSO variants in the literature.*
Keywords: PSO, FLC, Inertia weight, SA, Premature convergence, GA

1. **Introduction.** Particle swarm optimization is a swarm intelligence and swarm search algorithm originating from artificial life and evolutionary computation [1]. Due to the convenience of realization and promising optimization ability, PSO has been successfully applied in solving various function optimization problems, or the problems that can be transformed to the function optimization problems since its advent in 1995. Currently, PSO has become one of the most preferred choices for optimization problems due to its lesser memory requirements and better performance for providing solutions closer to optimum on different benchmark and engineering problems such as computer vision [2], engineering optimization [3], economic dispatch [4], and other related research areas [5], etc. It should be noted that the performance of PSO can be changed by varying the values of its parameters, such as the inertia weight, cognitive and social acceleration coefficients, and the learning coefficient. Although PSO is a powerful optimization tool with very few parameters to be adjusted, its operation itself is complicated and difficult to understand. On the other hand, similar to other evolutionary algorithms [6,7], due to the lack of knowledge of the dynamic search process, it is hard, if not impossible, to design

a mathematical model to adaptively adjust these parameters conveniently, especially for the inertia weight.

Over the last few decades, some understanding of the PSO search process has been accumulated, and the linguistic description of the search process is also available. These understanding and linguistic descriptions make fuzzy inference system a good candidate for dynamically tuning these parameters of particle swarm optimization. Based on this recognition, a two-input and two-output FLC is introduced into the canonical PSO to improve its optimization performance. Here, the two-input variables utilized here are as follows: one is the increment of global optimum (IGO) at successive generations, which reflects how particles will move in the subsequent search process according to their current situation. Exactly speaking, the sign of IGO indicates where particles will move towards (positive sign (+): move towards the objective gradually, negative sign (-): move far away from the objective gradually, and zero: stay at the original position), and the absolute value of IGO denotes the moving distance of particles. In addition, it should be noted that all of the situations discussed here are based on the premise of minimal optimization problems. The other input variable is the maximal focus distance (MFD) of particles that is able to reflect the current distribution of particles in the whole swarm, which is utilized to check the premature convergence of the PSO algorithm. Obviously, a larger MFD implies a more divergent swarm whereas a smaller one implies a more convergent swarm. The two-output variables include the inertia weight and constraint factor. The basic idea behind this scheme is to get a better balance between the exploitation and exploration during the search process of the particle swarm.

The rest of this paper is organized as follows. Section 2 reviews some related work on PSO, especially some classical fuzzy particle swarm optimization methods obtained from the literature. Section 3 introduces the canonical PSO algorithm. In section 4, the proposed fuzzy particle swarm optimization is elaborated from three aspects including the fuzzy logic controller, input-output variables of FLC and the fuzzy PSO system design, respectively. Section 5 reports the experimental results and analysis. Finally, this paper is ended with some important conclusions and future work in section 6.

2. **Related Work.** From the literature, it can be clearly observed that most of the current existing particle swarm optimization variants can be roughly divided into four categories.

(1) **Swarm initialization**. Like other swarm based stochastic optimization algorithms, PSO is first initialized with a population of random solutions (here refers to the positions of each particle) in the search space, and then begins to enter a loop in order to continue to search for optimal solutions by updating the particle's velocities and positions until some termination conditions are satisfied. In our previous work [8], two kinds of chaotic maps are firstly exploited to improve the quality of the initial population for PSO with promising results. Subsequent work [9] employs a similar chaotic opposition-based population initialization instead of the purely random one to improve the PSO. In particular, our recent work [10] introduces chaotic map based initialization and Gaussian mutation mechanism as well as a local re-initialization strategy into the standard PSO. Extensive experiments on several well-known benchmark functions demonstrate its effectiveness and efficiency. In the context of swarm initialization, there has been very little work in this research direction. However, it is reported that PSO tends to the characteristics of low stability owing to its non-uniformly distributed initial particles by [11]. Moreover, it is quite clear that the initial population can definitely affect the convergence speed of swarm and consequently the optimization performance of PSO.

(2) **Parameter selection**. Note that the proper selection of control parameters such

as inertia weight and acceleration coefficients can markedly influence the convergence of PSO. As the representative work, Tanweer et al.[12] present a self regulating particle swarm optimization (SRPSO) which incorporates the self-regulating inertia weight determined by the best particle for better exploration and self-perception on global search direction determined by the rest particles for exploitation in the search space. In [13], a simplified PSO is developed based on the stochastic inertia weight. It is clearly to be seen that this variant removes the velocity parameter and obtains inertia weight by means of random distribution to enhance the local and global search ability of PSO. In addition, Clerc and Kennedy [14] introduce a constriction factor into the standard particle swarm optimization that is a function of c_1 and c_2 to insure the convergence of PSO. In the work of Ratnaweera et al.[15], a self-organizing hierarchical particle swarm optimizer is put forward with time-varying acceleration coefficients to control the local search and convergence to the global optimum solution. Besides, many other acceleration coefficients [16-19] have also been formulated for PSO to improve its performance, and more details of them can be gleaned from the corresponding literature.

(3) **Topology structure**. To increase the diversity of the swarm, different topologies based particle swarm optimization has been developed in recent years. The salient neighborhood structures applied to PSO include the ring topology [20], the von Neumann topology [21] and the small world topology [22]. In [23], a PSO with expanding neighborhood topology is developed by combining particle swarm optimization and the variable neighborhood search to solve the well-known constrained shortest path problem. Recently, the work by Majercik [24] applies fluid neural networks to create dynamic neighborhood topologies and introduces fluid neural network particle swarm optimization (FNN-PSO) with a dynamic neighborhood mechanism. In the meanwhile, Wang et al.[25] develop a hybrid topology scale-free Gaussian-dynamic PSO for real power loss minimization problem involving fully connected topology and ring topology simultaneously. In more recent work [26], a dynamic tournament topology strategy is exploited to improve particle swarm optimization. To summarize, a suitable topological structure can effectively enhance the performance of PSO.

(4) **Hybrid versions**. To hybridize PSO with other auxiliary search techniques has been an active topic of research in swarm and evolutionary computation for decades. The work of Davoodi et al.[27] combines an improved quantum-behavior PSO with a simplex algorithm to solve the load flow problem. In [28], a hybrid PSO with artificial bee colony is presented for high-dimensional optimization problems. In addition, it should be noted that the combination of PSO with other evolutionary computation techniques like selection [29], crossover [30] and mutation [31,32] of genetic algorithm has now become a popular technique for enhancing the performance of PSO. As can be seen from the literature aforementioned, hybridization is a desirable strategy to keep the balance between exploration and exploitation for PSO as well as to prevent the stagnation of the swarm by leveraging the strengths of each of the components of the algorithm.

It seems not wise to give detailed descriptions for various PSO variants existed in the literature. Instead, this paper attempts to look into them through a unified view that may help grasp the essentials of these particle swarm optimization algorithms for other related researchers. So in this section, we will focus on several representative fuzzy PSO methods in the literature as follows, especially the formulation and the construction of their corresponding fuzzy logic controllers. Besides, note that a set of same symbols appeared many times in this work are summarized in the following table so as to improve the readability.

TABLE 1. The symbols and their corresponding meanings

Notation	Meaning
ω	inertia weight
c_1, c_2	acceleration coefficients
r_1, r_2	random numbers under a uniform distribution in [0,1]
α	constraint factor
t	current iteration
f_i	fitness value of the i-th particle
f_{avg}	average fitness value of the swarm for the current generation
N	maximum number of particles

2.1. **FPSO proposed by Shi & Eberhart [33].** In literature [33], a two-input and one-output FLC has been designed to improve the performance of canonical particle swarm optimization. Specifically, the two input variables mentioned here are the current best performance evaluation ($CBPE$) and the current inertia weight. The only one output variable is the change of the inertia weight. In order to make $CBPE$ be applicable to a wide range of optimization problems, it is normalized as follows:

$$NCBPE = \frac{CBPE - CBPE_{\min}}{CBPE_{\max} - CBPE_{\min}} \tag{1}$$

where $CBPE_{min}$ denotes the real minimum, $CBPE_{max}$ is the non-optimal $CBPE$. Note that the non-optimal $CBPE$ here represents that any solution with $CBPE$ greater or equal to $CBPE_{max}$ is not an acceptable solution to the minimization problem (assume minimization problems). All three fuzzy variables are defined to have three fuzzy sets and nine rules in the fuzzy system correspondingly. Simulation results show that PSO with a fuzzy system tuning its inertia weight can improve its performance to a large extent.

2.2. **FATPSO proposed by Liu & Abraham [34].** In literature [34], a two-input and two-output FLC has been devised. One of the input variables is $NCBPE$ that is the same as defined in [33]. The other is the current velocity (CV) of the particle. In addition, one of the output variables is ρ, the scaling factor to control the domain of the particle's oscillation. Another is V_{ck}, which controls the change of the velocity threshold according to the following formula:

$$V_c = e - [10(1 + V_{ck})] \tag{2}$$

A new velocity update strategy is designed as follows:

$$V_{ij}(t+1) = \omega \hat{v} + c_1 r_1 (x_{ij}^{\#}(t) - x_{ij}(t)) + c_2 r_2 (x_j^*(t) - x_{ij}(t)) \tag{3}$$

$$\hat{V} = \begin{cases} v_{ij}, & if \quad |v_{ij}| \geqslant v_c \\ u(-1,1)v_{\max}/\rho, & if \quad |v_{ij}| < v_c \end{cases} \tag{4}$$

where u(-1,1) is the random number, uniformly distributed in the interval [-1,1], and ρ is the scaling factor to control the domain of the particle's oscillation according to v_{max}. v_c is the minimum velocity threshold, a tunable threshold parameter to limit the minimum of the particle's velocity. Note that there are two-input and two-output based on six rules in the fuzzy adaptive turbulent particle swarm optimization (FATPSO) system. Through numerical experiment, it validates that the performance degrades little as the optimization problem's dimension increases.

2.3. **FPSO proposed by Yadmellat & Salehizadeh & Menhaj [16].** In literature [16], a new fuzzy tuned inertia weight particle swarm optimization (FIPSO) is presented based on the linguistic fuzzy control structure. More specifically, FIPSO is a two-input (t and $\Delta v(t)$) and one-output (inertia weight ω) FLC based particle swarm optimization algorithm. $\Delta v(t)$ can be defined as below.

$$\Delta v_{av}(t) = |v_{av}(t) - v_{av}(t-1)| \tag{5}$$

$$v_{av}(t) = \frac{1}{m \cdot D} \sum_{m} \sum_{D} v_{id} \tag{6}$$

where $\Delta v_{av}(t)$ denotes the average relative velocity. Note that nine dynamic fuzzy control rules are established for ω, which can meet the different requirement of PSO for inertia weight during the different search stages. Simulation results have shown that FIPSO has a better convergence compared to the other versions of PSO variants. Furthermore, the performance of it does not degrade significantly as the problem's dimension scales up.

2.4. **FPSO proposed by Tian & Zhao [17].** In literature [17], Tian et al. formulate a novel fuzzy particle swarm optimization with two-input and two-output, in which the fitness variance ($Delt$) and mean extremal deviation ($Total$) are considered as the input parameters of FLC so as to measure the discreteness of swarm in the search space and the population diversity respectively. Through this way, the inertia weight and learning factor of the extended term of PSO can be adaptively adjusted during the search process.

$$Delt = \frac{1}{N} \sum\nolimits_{i=1}^{N} (f_i - f_{ave})^2 \tag{7}$$

$$Total = \frac{1}{POP} \sum\nolimits_{i=1}^{POP} (pBest - gBest)^2 \tag{8}$$

Note that in Eq.(8), POP, $pBest$ and $gBest$ denote the swarm size, individual and global extremums, respectively.

2.5. **Other FPSOs in the literature [18,19,35-40].** Apart from the FPSO algorithms aforementioned, Kang et al.[18] propose a fuzzy based approach for tuning acceleration parameters of the PSO algorithm. Bajpai and Singh [32] present a fuzzy adaptive particle swarm optimization for bidding strategy in uniform price spot market. A hybrid meta-heuristic fuzzy scheme [36] has been designed based on discrete particle swarm optimization variable neighborhood search to solve quadratic assignment problem. In particular, the representations of the position and velocity of the particles in PSO is extended from the real vectors to fuzzy matrices in this hybrid fuzzy scheme. In [37], FCPSO-H is built to overcome the weaknesses of local optimum and curse of dimensionality, in which fuzzy logic is employed to control the acceleration coefficients in velocity equation for each particle. Recently, Juang et al.[38] come up with an adaptive fuzzy PSO based on the standard particle swarm optimization algorithm. This PSO utilizes fuzzy set theory to adjust PSO acceleration coefficients adaptively, and is thereby able to improve accuracy and efficiency of searches. Besides, Robati et al.[39] put forward a balanced fuzzy particle swarm optimization (BF-PSO) to solve the fundamental optimization problem entitled traveling salesman problem. Khan et al.[40] develop a fuzzy logic based multi-objective particle swarm optimization algorithm to efficiently solve the distributed local area networks topology design problem. In more recent work [19], to make up for the drawbacks of trapping into local optima and premature convergence, Neshat has developed FAIPSO. Its acceleration coefficients c_1 and c_2 are adaptively adjusted for each particle based on a fuzzy inference system, which comprises six inputs, two outputs and ten rules. In the meanwhile, a parabolic model is used to reduce its inertia weight. Alternatively, a range

of vision is defined for each of the particles and every one of the particles searches within this range. Especially another two-input, one-output and 14-rule based fuzzy inference system is formulated to adaptively control the range of the vision. In sum, all of the FPSO methods mentioned above possess respective advantages (information sharing, fast convergence and robustness) and disadvantages (premature convergence, local optimum and curse of dimensionality, etc).

3. **Particle Swarm Optimization.** Particle swarm optimization is a population-based technique for optimization, which simulates the social behavior of the fish schooling or bird flocking. In PSO system, each candidate solution is called a particle, each particle moves in the search space with a velocity that is dynamically adjusted according to the corresponding particle's experience and the particle's companions experience. Mathematically, the particles are manipulated according to the following equations:

$$v_{id}(t+1) = \omega \times v_{id}(t) + c_1 \times r_1 \times [p_{id}(t) - x_{id}(t)] + c_2 \times r_2 \times [p_{gd}(t) - x_{id}(t)] \quad (9)$$

$$x_{id}(t+1) = x_{id}(t) + \alpha \times v_{id}(t+1) \quad (10)$$

Note that ω is the inertia weight, it has characteristics that are reminiscent of the temperature parameter in the simulated annealing (SA). A large inertia weight facilitates a global exploration while a small inertia weight facilitates a local exploitation. α is generally used to control the weight of the velocity. The i-th particle is represented as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$. The best previous position (the position giving the best fitness value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol g. The rate of the position change (velocity) for particle i is represented as $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. During the update, the maximum velocity of each dimension of a particle is restricted to v_{max}, whose coordination of every dimension is also restricted to the permission scope. D represents the dimension of the search space. Note that in Eq.(9), the first part is the previous velocity of the particle, while the second is the "cognitive" part, representing the exploiting of its own experience, where c_1 is individual factor. The third is the "social" part, denoting the shared information and mutual cooperation among the particles, and c_2 is societal factor.

4. **Fuzzy Particle Swarm Optimization.** In this section, the proposed EFPSO will be elaborated from three aspects of the fuzzy logic controller, input-output variables of FLC and the fuzzy PSO system design, respectively.

4.1. **Fuzzy logic controller.** An FLC is composed of a knowledge base, that includes the information given by the expert in the form of linguistic control fuzzy rules, a fuzzification interface, which has the effect of transforming crisp data into fuzzy sets, an inference system, that uses them together with the knowledge base to make inference by means of a reasoning method, and a defuzzification interface, that translates the fuzzy control action thus obtained to a real control action using a defuzzification method.

4.2. **FLC input-output variables.** In this paper, two variables are selected as inputs for FLC. One is the increment of global optimum (IGO) in successive generations defined as follows:

$$IGO = p_g(t-1) - p_g(t), t \geqslant 2 \quad (11)$$

where $p_g(t-1)$ and $p_g(t)$ denote the global optimum of the swarm at generation $t-1$ and t, respectively. t is constricted to be greater or equal to 2. The initial value of IGO, that is, when t equals to 1, is predetermined according to different optimization problems (here refers to minimal optimization). From Eq.(11), it is easy to see that the value of

IGO can be positive, zero or negative. To be specific, if it is positive, which indicates particles move towards the objective gradually. On the contrary, if it is negative, which implies particles move far away from the objective. In addition, if IGO is equal to zero, then particles still stay at their original positions. Another input variable is the maximal focus distance (MFD) of particles formulated by [10], which can be defined as follows:

$$MFD = \max_{i=1\cdots m} \left(\sqrt{\sum_{d=1}^{D} (p_{ld} - x_{id})^2} \right) \tag{12}$$

where m is the number of neighborhood particles, p_{ld} is the previous best position, and x_{id} represents the sub-vector of the d-th dimension of the i-th particle in the search space. Similar to the commonly used measure variance of particles, MFD is utilized to check whether the PSO algorithm plunges into the local optima or not. In other words, the primary purpose of MFD is to evaluate the discreteness and diversity of the particles in the swarm. Based on the above discussions, a corresponding control strategy is adopted to tune the distribution of particles. Fig. 1 illustrates the scheme of the proposed EFPSO algorithm.

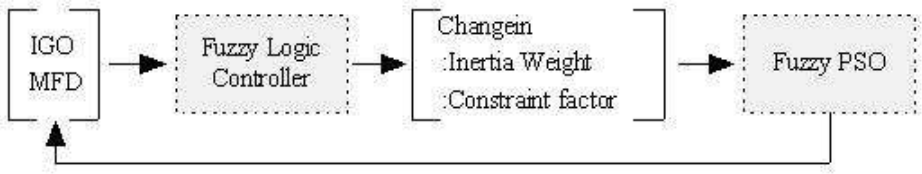

FIGURE 1. Scheme of the proposed fuzzy PSO

4.3. **Fuzzy PSO system design.** In our proposed fuzzy PSO system, note that the two-input variables IGO and MFD adopt the Gaussian membership function for fuzzy logic controller.

$$f(x) = \exp\left(-\frac{1}{2}y^2\right), y = \frac{8(x - x_1)}{x_2 - x_1} - 4 \tag{13}$$

The two-output variables inertia weight (IW) and constraint factor (CF) adopt the Trimple membership function as below,

$$f(x) = \begin{cases} 0, & x < x_1 \\ \frac{2(x-x_1)}{x_2-x_1}, & x_1 \leqslant x \leqslant \frac{x_1+x_2}{2} \\ \frac{2(x_2-x)}{x_2-x_1}, & \frac{x_1+x_2}{2} \leqslant x \leqslant x_2 \\ 0, & x > x_2 \end{cases} \tag{14}$$

Fig. 2 illustrates the framework of the proposed fuzzy particle swarm optimization. Note that the control rules have the following forms: Rule(i, j), if IGO is IGO_i and (or) MFD is MFD_j then IW is IW_k and CF is CF_h, for i=1,2,3; j=1,2,3; k=1,2,3; h=1,2,3, where IGO_i and MFD_j belong to the linguistic set $\{Low, Medium, High\}$. The output variables IW and CF are in the interval [0,1].

Table 2 lists the nine control rules of the proposed fuzzy particle swarm optimization algorithm. It should be noted that the first two columns denote the two-input, 3 and 4 columns are the two-output variables, the numbers in parentheses indicate the weight corresponding to the inference rule. The last column represents the fuzzy operators AND

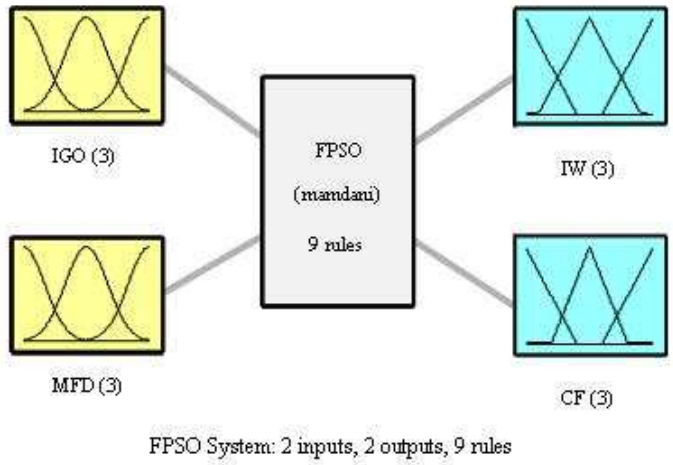

FIGURE 2. Framework of the fuzzy PSO

(1) and OR (2) for antecedents of the fuzzy rules. In addition, notice that the fuzzification and defuzzification processes leverage mamdani and centroid methods, respectively. Meanwhile, Fig. 3 displays the surface viewers based on the two-input variables IGO and MFD as well as the two-output variables IW and CF respectively, which will be helpful to understand the fuzzy inference system constructed in this paper.

TABLE 2. Nine control rules of the fuzzy PSO

[Rules]				
1	1	1	3(1):	2
1	2	1	2 (1):	2
1	3	1	1 (1):	1
2	1	3	3 (1):	1
2	2	3	2 (1):	1
2	3	3	1 (1):	1
3	1	1	3 (1):	1
3	2	1	2 (1):	1
3	3	1	1 (1):	1

5. **Experimental Results and Analysis.** To validate the effectiveness of the EFPSO proposed in this paper, six well-known benchmark functions obtained from the literature are leveraged to evaluate its performance, all of which have the same minimum value (viz. zero) except for f_6 with 0.000381827. Their expressions, initialization ranges, X_{max} and V_{max} are described in Table 3. For the sake of fair comparison, the parameters involved are set as follows: the acceleration coefficients $c_1 = c_2 = 2.0$, the swarm size is set to 80, the test function's dimensions are 30 and the maximal iteration number is set to 5000 respectively. Without loss of generality, the mean fitness value (Mean) and standard deviation (Std.) are utilized to measure the performance of each PSO algorithm based on the above parameter configurations over 30 independent runs. To obtain an unbiased comparison, the EFPSO algorithm is implemented on the platform of Intel Core Duo CPU

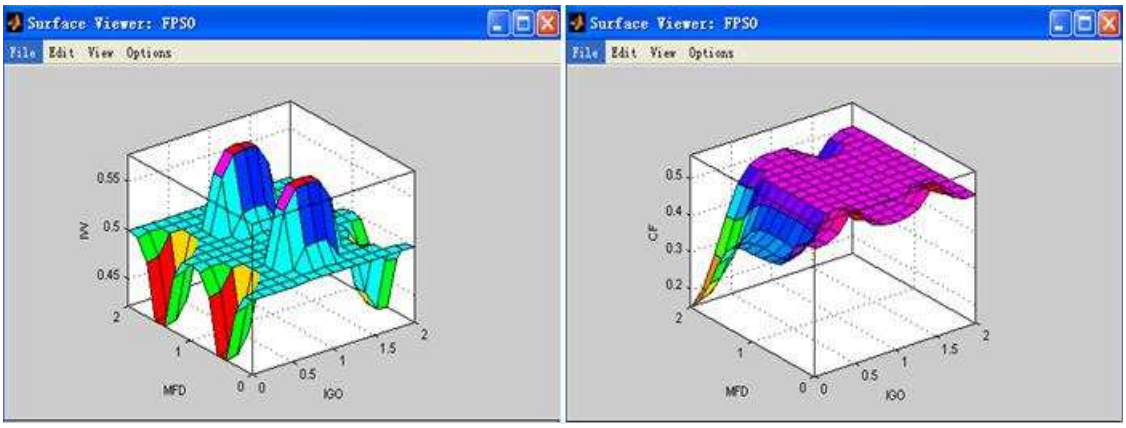

FIGURE 3. Surface viewers of the constructed fuzzy PSO

2.0 GHz PC running the Windows XP professional edition in Matlab 7.0 is also provided to demonstrate its performance. Besides, it should be noted that the experimental results of CMA-ES, JADE and OLPSO-G methods are directly acquired from literature [41] for comparison except for FPSO [17].

TABLE 3. The benchmark functions employed in our experiment

Func.	Expression	Initial range	X_{max}	V_{max}
f_1	$\sum_{i=1}^{d} {x_i}^2$	[-100,100]	100	100
f_2	$\sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-5,10]	100	100
f_3	$\sum_{i=1}^{d} i \times {x_i}^4 + random[0,1)$	[-1.28,1.28]	10	10
f_4	$\sum_{i=1}^{d} ({x_i}^2 - 10\cos(2\pi x_i) + 10)$	[-2.56,2.56]	10	10
f_5	$\frac{1}{4000}\sum_{i=1}^{d} (x_i)^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{x_i}}\right) + 1$	[-600,600]	600	600
f_6	$418.9829 \times n - \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i \rvert})$	[-500,500]	500	500

From the results listed in Table 4, it can be easily observed that the EFPSO proposed in this paper significantly outperforms the other PSO algorithms for functions f_1, f_2 and f_5 respectively. As for f_3, even though the *Mean* value of EFPSO is slightly worse than that of JADE, our method achieves the better standard deviation compared to that of JADE algorithm. With regard to f_4, EFPSO is obviously superior to CMA-ES, OLPSO-G and FPSO apart from obtaining the same best optimization result as JADE. Concerning the test function f_6, our approach can still achieve better performance in comparison with the other three PSO variants except for JADE. On the other hand, it should be noted that the standard deviations of the proposed fuzzy PSO are consistently smaller than that of other PSO algorithms, which further demonstrate the robustness of EFPSO for numerical optimization. All in all, the fuzzy schemes developed in this paper are able to get a better balance between exploitation and exploration during the search process, which is largely owing to the premature convergence of swarms can be effectively avoided by adopting the

fuzzy strategies based on the maximal focus distance among particles and the increment of global optimum at successive generations.

TABLE 4. Performance comparison among different PSO variants

Func.	Metric	CMA-ES	JADE	OLPSO-G	FPSO	EFPSO
f_1	Mean	4.56e-16	1.30e-54	4.10e-54	6.18e-17	7.28e-56
	Std.	1.13e-16	9.20e-54	6.32e-54	8.06e-17	9.15e-55
f_2	Mean	2.33e-15	3.20e-01	2.15e+01	9.46e-01	7.16e-16
	Std.	7.70e-16	1.10e-00	2.99e+01	8.83e-01	8.01e-16
f_3	Mean	5.92e-02	6.80e-04	1.16e-02	6.97e-01	9.51e-04
	Std.	1.73e-02	2.50e-04	4.10e-03	8.32e-01	3.13e-05
f_4	Mean	1.76e+02	0	1.07e-00	3.72e-17	0
	Std.	1.39e+01	0	9.90e-01	6.65e-19	0
f_5	Mean	9.59e-16	2.00e-04	4.80e-03	4.26e-12	8.03e-19
	Std.	3.50e-16	1.40e-03	8.63e-03	5.16e-12	6.64e-20
f_6	Mean	3.15e+03	7.10e-00	3.84e+02	5.43e+02	3.69e+01
	Std.	5.79e+02	2.80e+01	2.17e+02	7.06e+02	2.23e+01

To present a total comparison on the optimization performance between EFPSO and other PSO variants, Table 5 shows the detailed results from the non-parametric Wilcoxon rank sum tests [42] at 5% significance level. Note that the number of benchmark functions (out of the 6 tested functions) that the EFPSO is significantly better than (*Better*), almost the same as (*Same*), and significantly worse than (*Worse*) the compared PSO algorithm are reported here. The total score (*Total*) is calculated by subtracting *Worse* from *Better*. Obviously, the *Total* values shown in Table 5 apparently demonstrate the significance of EFPSO over the other selected particle swarm optimization algorithms.

TABLE 5. Statistical analysis of wilcoxon-tests between EFPSO and its competitors

Item	CMA-ES	JADE	OLPSO-G	FPSO
Better	5	3	6	6
Same	1	1	0	0
Worse	0	2	0	0
Total	5	1	6	6

In addition, to further illustrate the effect of the proposed EFPSO algorithm, we also apply it in the task of standard image segmentation, whose goal is to partition an image into non-overlapping objects of interest based on the intrinsic features, such as color, texture, intensity and contrast, etc. Note that the threshold segmentation is a basic method in the field of image segmentation, and the most commonly used threshold method is the Otsu algorithm [43] whose core idea can be described as follows: let the pixels of a given image be represented in l gray levels $\{0, 1, \cdots, l-1\}$, suppose that the pixels are dichotomized into two classes: object and background, denoted by C_0 with gray levels $\{0, 1, \cdots, t\}$ and C_1 with gray levels $\{t+1, t+2, \cdots, l-1\}$ respectively by a threshold at level t. $p_i = n_i/N$, where n_i represents the number of pixels at level i while N denotes the total number of pixels appeared in the image. As a result, the probabilities of class occurrence and the class mean levels for C_0 can be defined as below:

$$\omega_o(t) = \sum_{i=0}^{t} p_i, \mu_0(t) = \sum_{i=0}^{t} ip_i/\omega_0 \tag{15}$$

FIGURE 4. The original and the segmented images of Lena

Similarly, the probabilities of class occurrence and the class mean levels for the background can be described as:

$$\omega_1(t) = \sum_{i=t+1}^{l-1} p_i, \mu_1(t) = \sum_{i=t+1}^{l-1} ip_i/\omega_1 \tag{16}$$

The variance formula between these two groups is $d(t) = \omega_0(t)\omega_1(t)(\mu_0(t) - \mu_1(t))^2$. And the corresponding gray level value t^* is the best threshold when the variance function achieving the maximum value, i.e., $t^* = Argmax\{d(t)\}$. So it can be seen that how to determine the threshold value of Otsu method is the key to image segmentation. Here, we exploit the EFPSO algorithm proposed in this paper to solve the segmentation threshold. Note that due to the limited space, the standard images Lena and Peppers with 512×512 pixels are employed here to validate the performance of the proposed fuzzy PSO algorithm. At the same time, we compare them with the standard PSO (SPSO) and genetic algorithm (GA), respectively. The main parameter settings of GA are described as follows: elite selection strategy, crossover rate is 0.7, mutation rate is 0.4, migration fraction is 0.2, population size is 50 and the maximum generation is 100 served as the stopping criteria. Figures 4-5 illustrate the original images and the segmented results

FIGURE 5. The original and the segmented images of Peppers

yielded by GA, SPSO and the proposed EFPSO respectively, which further verifies the superiority of our approach over other swarm intelligence based methods in the task of image segmentation.

6. **Conclusions and Future Work.** As one of the most important swarm intelligence based algorithms, PSO attracts increasing attention as a new optimization technique for solving complex optimization problems. In this paper, we have proposed a two-input and two-output fuzzy logic controller based particle swarm optimization algorithm. The increment of global optimum and maximal focus distance of particles are used as the two-input variables, while the inertia weight and constraint factor as the two-output variables that are adaptively adjusted according to the control information translated from the FLC during the search process. Through extensive experiments, we demonstrate that the fuzzy PSO proposed in this paper has not only the powerful ability to search the global optimum, but also effectively prevent the premature convergence of the particle swarm optimization algorithm. In particular, the EFPSO has already been successfully applied in the multimedia retrieval systems to train the feedforward neural network, which further validates its effectiveness and efficiency.

As for future work, we plan to introduce this approach into the other real-world research fields, such as integrated circuit design, multimedia semantic understanding and engineering optimization scheduling, etc. Lastly, and arguably most importantly, the qualitative relationship between the particle's distribution and the convergence of PSO will be elaborated comprehensively from the perspective of mathematics.

Acknowledgment. The author would like to sincerely thank the anonymous reviewers for their valuable comments and insightful suggestions that have helped to improve the paper. Besides, the author thanks Prof. Zhongzhi Shi for stimulating discussions and helpful hints. This work is partially supported by the National Program on Key Basic Research Project (No.2013CB329502), the Key R&D Program of the Shaanxi Province of China (No.2018GY-037) and the Special Research Project of the Educational Department of Shaanxi Province of China (No.18JK0051).

REFERENCES

[1] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of the Int'l Conf. on Neural Networks*, pp. 1942–1948, 1995.

[2] E. Vellasques, R. Sabourin and E. Granger, Fast intelligent watermarking of heterogeneous image streams through mixture modeling of PSO populations, *Applied Soft Computing*, vol. 13, no. 6, pp. 3130–3148, 2013.

[3] C. Zhang, Y. Liu and Y. Zhao, Application of dynamic neighborhood small population particle swarm optimization for reconfiguration of shipboard power system, *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1255–1262, 2013.

[4] T. Niknam, H. Mojarrad and M. Nayeripour, A new fuzzy adaptive particle swarm optimization for non-smooth economic dispatch, *Energy*, vol. 35, no. 4, pp. 1764–1778, 2010.

[5] J. Koupaei, S. Hosseini and F. Ghaini, A new optimization algorithm based on chaotic maps and golden section search method, *Engineering Applications of Artificial Intelligence*, vol. 50, no. 4, pp. 201–214, 2016.

[6] Z. Meng, J.-S. Pan and H. Xu, QUasi-affine transformation evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization, *Knowledge-Based Systems*, vol. 109, pp. 104–121, 2016.

[7] Z. Meng and J.-S. Pan, Monkey king evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization, *Knowledge-Based Systems*, vol. 97, pp. 144–157, 2016.

[8] D. Tian and T. Zhao, Particle swarm optimization based on tent map and logistic map, *Journal of Shaanxi University of Science and Technology*, vol. 28, no. 2, pp. 17–23, 2010.

[9] W. Gao, S. Liu and L. Huang, Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique, *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 11, pp. 4316–4327, 2012.

[10] D. Tian, Particle swarm optimization with chaos-based initialization for numerical optimization, *Intelligent Automation and Soft Computing*, vol. 24, no. 2, pp. 331–342, 2018.

[11] R. He, Y. Wang, Q. Wang, et al., An improved particle swarm optimization based on self-adaptive escape velocity, *Journal of Software*, vol. 16, no. 12, pp. 2036–2044, 2005.

[12] M. Tanweer, S. Suresh and N. Sundararajan, Self regulating particle swarm optimization algorithm, *Information Sciences*, vol. 294, pp. 182–202, 2015.

[13] Z. Zhao, S. Huang and W. Wang, Simplified particle swarm optimization algorithm based on stochastic inertia weight, *Application Research of Computers*, vol. 31, no. 2, pp. 361–364, 2014.

[14] M. Clerc and J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[15] A. Ratnaweera, S. Halgamuge and H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[16] P. Yadmellat, S. Salehizadeh and M. Menhaj, A new fuzzy inertia weight particle swarm optimization, *Proc. of the Int'l Conf. on Computational Intelligence and Natural Computing*, pp. 507–510, 2009.

[17] D. Tian and T. Zhao, Particle swarm optimization algorithm based on fuzzy controller, *Computer Engineering and Design*, vol. 31, no. 24, pp. 5335–5338, 2010.
[18] Q. Kang, L. Wang and Q. Wu, Research on fuzzy adaptive optimization strategy of particle swarm algorithm, *International Journal of Information Technology*, vol. 12, no. 3, pp. 66–76, 2006.
[19] M. Neshat, FAIPSO: fuzzy adaptive informed particle swarm optimization, *Neural Computing and Applications*, vol. 23, no. 1, pp. 95–116, 2013.
[20] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, *Proc. of the 6th Int'l Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.
[21] J. Kennedy and R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 4, pp. 515–519, 2006.
[22] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 1931–1938, 1999.
[23] Y. Marinakis, A. Migdalas and A. Sifaleras, A hybrid particle swarm optimization C variable neighborhood search algorithm for constrained shortest path problems, *European Journal of Operational Research*, vol. 261, no. 3, pp. 819–834, 2017.
[24] S. Majercik, Using fluid neural networks to create dynamic neighborhood topologies in particle swarm optimization, *Proc. of the 9th Int'l Conf. on Swarm Intelligence*, pp. 270–277, 2014.
[25] C. Wang, Y. Liu, Y. Zhao, et al., A hybrid topology scale-free Gaussian-dynamic particle swarm optimization algorithm applied to real power loss minimization, *Engineering Applications of Artificial Intelligence*, vol. 32, no. 6, pp. 63–75, 2014.
[26] L. Wang, B. Yang and J. Orchard, Particle swarm optimization using dynamic tournament topology, *Applied Soft Computing*, vol. 48, no. 11, pp. 584–596, 2016.
[27] E. Davoodi, M. Hagh and S. Zadeh, A hybrid improved quantum-behaved particle swarm optimization-simplex method (IQPSOS) to solve power system load flow problems, *Applied Soft Computing*, vol. 21, pp. 171–179, 2014.
[28] Z. Li, W. Wang, Y. Yan, et al., PS-ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems, *Expert Systems with Applications*, vol. 42, no. 22, pp. 8881–8895, 2015.
[29] P. Agarwalla and S. Mukhopadhyay, Efficient player selection strategy based diversified particle swarm optimization algorithm for global optimization, *Information Sciences*, vol. 397-398, pp. 69–90, 2017.
[30] A. Meng, Z. Li, H. Yin, et al., Accelerating particle swarm optimization using crisscross search, *Information Sciences*, vol. 329, pp. 52–72, 2016.
[31] Z. Meng, J.-S. Pan and L. Kong, Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution, *Knowledge-Based Systems*, vol. 141, pp. 92–112, 2018.
[32] H. Wang, W. Wang and Z. Wu, Particle swarm optimization with adaptive mutation for multimodal optimization, *Applied Mathematics and Computation*, vol. 221, pp. 296–305, 2013.
[33] Y. Shi and R. Eberhart, Fuzzy adaptive particle swarm optimization, *Proc. of the IEEE Int'l Conf. on Evolutionary Computation*, pp. 101–106, 2001.
[34] H. Liu and A. Abraham, Fuzzy adaptive turbulent particle swarm optimization, *Proc. of the 5th Int'l Conf. on Hybrid Intelligent Systems*, pp. 445–450, 2005.
[35] P. Bajpai and S. Singh, Fuzzy adaptive particle swarm optimization for bidding strategy in uniform price spot market, *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2152–2160, 2007.
[36] H. Liu and A. Abraham, An hybrid fuzzy variable neighborhood particle swarm optimization algorithm for solving quadratic assignment problems, *Journal of Universal Computer Science*, vol. 13, no. 9, pp. 1309–1331, 2007.
[37] Z. Afsahi and M. Meybodi, Improving cooperative PSO using fuzzy logic, *Research and Development in Intelligent Systems XXVI*, pp. 219–232, 2009.
[38] Y. Juang, S. Tung and H. Chiu, Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions, *Information Sciences*, vol. 181, no. 20, pp. 4539–4549, 2011.
[39] A. Robati, G. Barani, H. Pour, et al., Balanced fuzzy particle swarm optimization, *Applied Mathematical Modelling*, vol. 36, no. 5, pp. 2169–2177, 2012.
[40] S. Khan and A. Engelbrecht, A fuzzy particle swarm optimization algorithm for computer communication network topology design, *Applied Intelligence*, vol. 36, no. 1, pp. 161–177, 2012.
[41] Z. Zhan, J. Zhang, Y. Li, et al., Orthogonal learning particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[42] J. Gibbons and S. Chakraborti, Nonparametric Statistical Inference (Edited by M. Lovric), *International Encyclopedia of Statistical Science*, Springer, Berlin, Heidelberg, pp. 977–979, 2011.

[43] N. Otsu, A threshold selection method from gray-level histograms, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.