Turbulent Particle Swarm Optimization Using Fuzzy Parameter Tuning

Ajith Abraham and Hongbo Liu

Abstract. Particle Swarm Optimization (PSO) algorithm is a stochastic search technique, which has exhibited good performance across a wide range of applications. However, very often for multi-modal problems involving high dimensions the algorithm tends to suffer from premature convergence. Premature convergence could make the PSO algorithm very difficult to arrive at the global optimum or even a local optimum. Analysis of the behavior of the particle swarm model reveals that such premature convergence is mainly due to the decrease of velocity of particles in the search space that leads to a total implosion and ultimately fitness stagnation of the swarm. This paper introduces Turbulence in the Particle Swarm Optimization (TPSO) algorithm to overcome the problem of stagnation. The algorithm uses a minimum velocity threshold to control the velocity of particles. TPSO mechanism is similar to a turbulence pump, which supplies some power to the swarm system to explore new neighborhoods for better solutions. The algorithm also avoids clustering of particles and at the same time attempts to maintain diversity of population. We attempt to theoretically analyze that the algorithm converges with a probability of 1 towards the global optimal. The parameter, the minimum velocity threshold of the particles is tuned adaptively by a fuzzy logic controller embedded in the TPSO algorithm, which is further called as Fuzzy Adaptive

Ajith Abraham
Centre for Quantifiable Quality of Service in Communication Systems,
Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
e-mail: ajith.abraham@ieee.org
http://www.softcomputing.net

Ajith Abraham and Hongbo Liu
School of Computer Science and Engineering, Dalian Maritime University,
116026 Dalian, China

Hongbo Liu
Department of Computer, Dalian University of Technology, 116023 Dalian, China
e-mail: lhb@dlut.edu.cn

A. Abraham et al. (Eds.): Foundations of Comput. Intel. Vol. 3, SCI 203, pp. 291–312.
springerlink.com

TPSO (FATPSO). We evaluated the performance of FATPSO and compared it with the Standard PSO (SPSO), Genetic Algorithm (GA) and Simulated Annealing (SA). The comparison was performed on a suite of 20 widely used benchmark problems. Empirical results illustrate that the FATPSO could prevent premature convergence very effectively. It clearly outperforms the considered methods, especially for high dimension multi-modal optimization problems.

1 Introduction

Particle Swarm Optimization (PSO) algorithm is mainly inspired by social behaviour patterns of organisms that live and interact within large groups. In particular, PSO incorporates swarming behaviours observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the idea of swarm intelligence is emerged ([14]). It could be applied to solve various function optimization problems, or the problems that can be transformed to function optimization problems. PSO has exhibited good performance across a wide range of applications ([19, 15, 25, 26, 1, 21, 5, 22]). However, its performance deteriorates as the dimensionality of the search space increases, especially for multi-modal optimization problems ([13, 20]). PSO algorithm often demonstrates faster convergence speed in the first phase of the search, and then slows down or even stops as the number of generations is increased. Once the algorithm slows down, it is difficult to achieve better fitness values. This state is called as stagnation or premature convergence. The trajectory of particles was given a lot of importance rather than their velocities. In this paper, we attempt to discuss the relation between the algorithm convergence and the velocities of the particles. It is found that the stagnation state is mainly due to a decrease of velocity of particles in the search space which leads to a total implosion and ultimately fitness stagnation of the swarm. We introduce Turbulent Particle Swarm Optimization (TPSO) algorithm to improve the optimization performance and overcome the premature convergence problem. The basic idea is to drive those lazy particles and get them to explore new search spaces. TPSO uses a minimum velocity threshold to control the velocity of particles and also avoids clustering of particles and maintains diversity of population in the search space. The minimum velocity threshold of the particles is tuned adaptively by using a fuzzy logic controller in the algorithm, which is further called as Fuzzy Adaptive TPSO (FATPSO).

The Chapter is organized as follows. Particle swarm optimization is reviewed briefly and the effects on the change of the velocities of particles are analyzed in Section 2. In Section 3, we describe the TPSO model and the fuzzy adaptive processing method. Experiment settings, results and discussions are given in Section 4 followed by some conclusions in the last Section.

2 Particle Swarm Optimization

Particle swarm optimization refers to a relatively new family of algorithms that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. Some researchers have done much work on its study and development during the recent years ([29, 20, 16, 12]). We review briefly the standard particle swarm model, and then analyze the various effects in the change in the velocities of particles.

2.1 Standard Particle Swarm Model

The particle swarm model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the fitness f can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector $\mathbf{p}_i$ (i is the index of the particle), and a velocity represented by a velocity-vector $\mathbf{v}_i$. Each particle remembers its own best position so far in a vector $\mathbf{p}_i^{\#}$, and its j-th dimensional value is $p_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\mathbf{p}^*$, and its j-th dimensional value is p_j^*. During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined by Eq.(1). The new position is then determined by the sum of the previous position and the new velocity by Eq.(2).

$$\begin{aligned} v_{ij}(t) = w v_{ij}(t-1) + c_1 r_1 (p_{ij}^{\#}(t-1) - p_{ij}(t-1)) \\ + c_2 r_2 (p_j^*(t-1) - p_{ij}(t-1)) \end{aligned} \tag{1}$$

$$p_{ij}(t) = p_{ij}(t-1) + v_{ij}(t) \tag{2}$$

where r_1 and r_2 are the random numbers, uniformly distributed within the interval [0,1] for the j-th dimension of i-th particle. c_1 is a positive constant, called as coefficient of the self-recognition component, c_2 is a positive constant, called as coefficient of the social component. The variable w is called as the inertia factor, which value is typically setup to vary linearly from 1 to near 0 during the iterated processing. From Eq.(1), a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm.

In the particle swarm model, the particle searches the solutions in the problem space within a range $[-s, s]$ (If the range is not symmetrical, it can be translated to the corresponding symmetrical range). In order to guide the particles effectively in the search space, the maximum moving distance during one iteration is clamped in between the maximum velocity $[-v_{max}, v_{max}]$ given in Eq.(3), and similarly for its moving range given in Eq.(4):

$$v_{i,j} = sign(v_{i,j})min(|v_{i,j}|, v_{max}) \tag{3}$$

$$p_{i,j} = sign(p_{i,j})min(|p_{i,j}|, p_{max}) \tag{4}$$

The value of v_{max} is $\alpha \times s$, with $0.1 \leq \alpha \leq 1.0$ and is usually chosen to be s, i.e. $\alpha = 1$.

2.2 Velocities Analysis in Particle Swarm

Some previous studies have discussed the trajectory of particles and the convergence of the algorithm ([3, 29, 27]). It has been shown that the trajectories of the particles oscillate as different sinusoidal waves and converge quickly, sometimes prematurely. We analyze the effects of the change in the velocities of particles.

The gradual change of the particle's velocity can be explained geometrically. During each iteration, the particle is attracted towards the location of the best fitness achieved so far by the particle itself and by the location of the best fitness achieved so far across the whole swarm. From Eq.(1), $v_{i,j}$ can attain a smaller value, but if the second term and the third term in RHS of Eq.(1) are both small, it cannot resume a larger value and could eventually loose the exploration capabilities in the future iterations. Such situations could occur even in the early stages of the search. When the second term and the third term in RHS of Eq.(1) are zero, $v_{i,j}$ will be damped quickly with the ratio of w. In other words, if a particle's current position coincides with the global best position/particle, the particle will only move away from this point if its previous velocity and w are non-zero. If their previous velocities are very close to zero, then all the particles will stop moving once they catch up with the global best particle, which many lead to premature convergence. In fact, this does not even guarantee that the algorithm has converged to a local minimum and it merely means that all the particles have converged to the best position discovered so far by the swarm. This state owes to the second term and the third term in the RHS of Eq.(1), the cognitive components of the PSO. But if the cognitive components of the PSO algorithm are invalidated, all particles always search the solutions using the initial velocities. Then the algorithm is merely a degenerative stochastic search without the characteristics of PSO.

3 Turbulent Swarm Optimization

We introduce a new velocity update approach for the particles in PSO, and analyze its effect on the particle's behavior. We also illustrate a Fuzzy Logic Controller (FLC) scheme to adaptively control the parameters ([11, 30, 17]).

3.1 Velocity Update of the Particles

As discussed in the previous Section, one of the main reason for premature convergence of PSO is due to the stagnation of the particles exploration of a new search space. We introduce a strategy to drive those lazy particles and let them explore better solutions. If a particle's velocity decreases to a threshold v_c, a new velocity is assigned using Eq.(6). Thus, we present the turbulent particle swarm optimization using new velocity update equations:

$$\begin{aligned} v_{ij}(t) = w\hat{v} + c_1 r_1(x_{ij}^{\#}(t-1) - x_{ij}(t-1)) \\ + c_2 r_2(x_j^*(t-1) - x_{ij}(t-1)) \end{aligned} \tag{5}$$

$$\hat{v} = \begin{cases} v_{ij} & \text{if } |v_{ij}| \geq v_c \\ u(-1,1)v_{max}/\rho & \text{if } |v_{ij}| < v_c \end{cases} \tag{6}$$

where $u(-1,1)$ is the random number, uniformly distributed with 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 particles' velocity. Fig. 1 illustrates the trajectory of a single particle in standard particle swarm optimization (SPSO) and turbulent particle swarm optimization (TPSO) respectively.

The change of the particle's situation is directly correlated to two parameter values, v_c and ρ. A large v_c shortens the oscillation period, and it provides a great probability for the particles to leap over local minima using the same number of iterations. But a large v_c compels particles in the quick "flying" state, which leads them not to search the solution and forcing them not to refine the search. In other words, a large v_c facilitates a global search while a

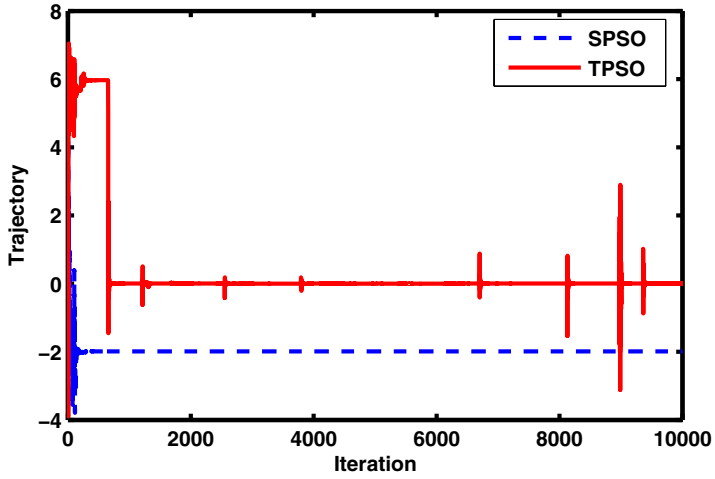

Fig. 1 Trajectory of a single particle

smaller value facilitates a local search. By changing it dynamically, the search ability is dynamically adjusted. The value of ρ changes directly the particle oscillation domain. It is possible for particles not to jump over the local minima if there would be a large local minimum available in the objective search space. But the particle trajectory would more prone to oscillate because of a smaller value of ρ. For the desired exploration-exploitation trade-off, we divide the particle search into three stages. In the first stage the values for v_c and ρ are set at large and small values respectively. In the second stage, v_c and ρ are set at medium values and in the last stage, v_c is set at a small value and ρ is set at a large value. This enable the particles to take very large steps to explore solutions in the early stages, by scanning the whole solution space for good local minima and then in the final stages particles perform a fine grain search. The use of fuzzy logic would be suitable for dynamically tuning the velocity threshold, since it starts a run with an initial value which is changed during the run. By using the fuzzy control approach, the parameters can be adaptively regulated according to the problem environment.

3.2 Fuzzy Parameter Control

A Fuzzy Logic Controller (FLC) is composed of a knowledge base, that includes the information given by the expert in the form of linguistic control 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]. The generic structure of an FLC is shown in Figure 2.

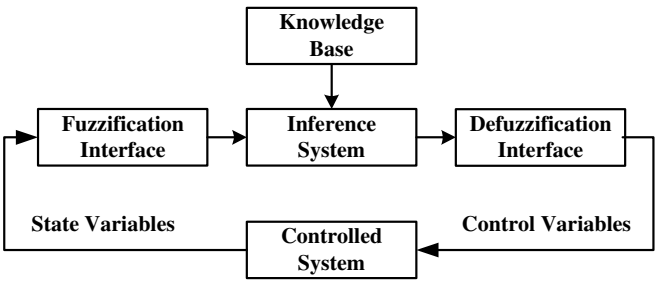

Fig. 2 Generic structure of an FLC

In the proposed algorithm, two variables are selected as inputs to the fuzzy system: the Current Best Performance Evaluation ($CBPE$) ([24]) and the Current Velocity (CV) of the particle. For adapting to a wide range of optimization problems, $CBPE$ is normalized as Eq.(7):

$$NCBPE = \frac{CBPE - CBPE_{min}}{CBPE_{max} - CBPE_{min}} \tag{7}$$

where $CBPE_{min}$ is the estimated (or real) minimum, $CBPE_{max}$ is the worst solution to the minimization problem, which usually is the $CBPE$ at half the number of iterations. If we do not have any prior information about the objective function and if it is difficult to estimate $CBPE_{min}$ and $CBPE_{max}$, we can do some preliminary experiments by decreasing linearly from 1 to 0 during the run. One of the output variables is ρ, the scaling factor to control the domain of the particle's oscillation. Another is Vck, which controls the change of the velocity threshold according to Eq.(8):

$$v_c = e - [10(1 + Vck)] \tag{8}$$

The fuzzy inference system is listed briefly as follows:
[System]
Name='$FATPSO$'

[Input1] Name='$NCBPE$'
Range=[0 1]
NumMFs=3
MF1='Low':'gaussmf', [0.005 0]
MF2='Medium':'gaussmf', [0.03 0.1]
MF3='High':'gaussmf', [0.25 1]

[Input2]
Name='CV'
Range=[0 1e-006]
NumMFs=2
MF1='Low':'trapmf', [0 0 1e-030 1e-020]
MF2='High':'trapmf', [1e-010 1e-008 1e-006 1e-006]

[Output1]
Name='Vck'
Range=[-1 2.2]
NumMFs=3
MF1='Low':'trimf', [-1 -0.8 -0.5]
MF2='Medium':'trimf', [-0.6 0 0.2]
MF3='High':'trimf', [0.1 1.1 2.2]

[Output2]
Name='ρ'
Range=[1 120]
NumMFs=3
MF1='Small':'trimf', [1 1 4]
MF2='Medium':'trimf', [2.214 10.71 59.29]
MF3='Large':'trimf', [47.15 120 120]

[Rules]
1 1, 3 0 (1) : 1
2 0, 2 0 (1) : 1
3 2, 1 0 (1) : 1
1 1, 0 3 (1) : 2
2 0, 0 2 (1) : 2
3 2, 0 1 (1) : 2

In the above mentioned list, there are three parts: the first part is the configuration of the fuzzy system, the second one is the definition of the membership functions, and the third one is the rule base. There are two inputs and two outputs based on six rules. In the rule base, the first two columns correspond to the input variables, the second two columns correspond to the output variables, the fifth column displays the weight applied to each rule, and the sixth column is short form that indicates whether this is an AND (1) rule or an OR (2) rule. The numbers in the first four columns refer to the index number of the membership function, in which the number 1 encodes fuzzy set 'Low', 2 encodes 'Medium', and 3 encodes 'High'. For example, the first rule is "If ($NCBPE$ is Low) and (CV is Low) then (Vck is High) with the weight 1". The general structure of the FATPSO is illustrated in Algorithm 1.

Algorithm 1. FATPSO

01. Initialize parameters and the particles
02. While (the end criterion is not met) do
03. $t = t + 1$
04. Calculate the fitness value of each particle
05. $x^* = argmin_{i=1}^{n}(f(x^*(t-1)), f(x_1(t)),$
06. $f(x_2(t)), \cdots, f(x_i(t)), \cdots, f(x_n(t)))$
07. For i= 1 to n
08. $x_i^{\#}(t) = argmin_{i=1}^{n}(f(x_i^{\#}(t-1)), f(x_i(t))$
09. For j = 1 to d
10. If $abs(v_{ij}) < 1e-6$
11. Obtain the velocity threshold
12. {
13. fismat = readfis('$FATPSO.fis$')
14. FO = evalfis([$NCBPE$ CV], fismat)
15. }
16. Endif
17. Update the j-th dimension value of $\mathbf{x}_i$
18. and $\mathbf{v}_i$ according to Eqs. (1), (2) and (3)
19. Next j
20. Next i
21. End While

4 Convergence Analysis of TPSO

For analyzing the convergence of the proposed algorithm, we first introduce the definitions and lemmas [8, 9, 10], and then theoretically prove that the proposed variable neighborhood particle swarm algorithm converges with a probability 1 or strongly towards the global optimal.

Consider the problem (P) as

$$(P) = \begin{cases} minf(\mathbf{x}) \\ \mathbf{x} \in \Omega = [-s, s]^n \end{cases} \tag{9}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$. $\mathbf{x}^*$ is the global optimal solution to the problem (P), let $f^* = f(\mathbf{x}^*)$. Let

$$\begin{aligned} D_0 &= \{\mathbf{x} \in \Omega | f(\mathbf{x}) - f^* < \varepsilon\} \\ D_1 &= \Omega \setminus D_0 \end{aligned} \tag{10}$$

for every $\varepsilon > 0$.

Assume that the i-th dimensional value of the particle's velocity decreases to a threshold v_c, then the shaking strategy is activated, and a turbulent velocity is generated by Eq.(6). In $u(-1,1)v_{max}/\rho$, $u(-1,1)$ is a normal distributed random number within the interval [-1,1], and the scaling factor ρ is a positive constant to control the domain of the particle's oscillation according to v_{max}. Therefore the turbulent velocity $\hat{v}$ belongs to the normal distribution. If $v_{max} = s$, then $\hat{v} \sim [-\frac{s}{\rho}, \frac{s}{\rho}]$. During the iterated procedure from the time t to $t+1$, let q_{ij} denote that $\mathbf{x}(t) \in D_i$ and $\mathbf{x}(t+1) \in D_j$. Accordingly the particles' positions in the swarm could be classified into four states: q_{00}, q_{01}, q_{10} and q_{01}. Obviously $q_{00} + q_{01} = 1$, $q_{10} + q_{11} = 1$.

Definition 1 (Convergence in terms of probability). *Let ξ_n a sequence of random variables, and ξ a random variable, and all of them are defined on the same probability space. The sequence ξ_n converges with a probability of ξ if*

$$\lim_{n \to \infty} P(|\xi_n - \xi| < \varepsilon) = 1 \tag{11}$$

for every $\varepsilon > 0$.

Definition 2 (Convergence with a probability of 1). *Let ξ_n a sequence of random variables, and ξ a random variable, and all of them are defined on the same probability space. The sequence ξ_n converges almost surely or almost everywhere or with probability of 1 or strongly towards ξ if*

$$P\left(\lim_{n \to \infty} \xi_n = \xi\right) = 1; \tag{12}$$

or

$$P\left(\bigcap_{n=1}^{\infty}\bigcup_{k\geq n}[|\xi_n-\xi|\geq\varepsilon]\right)=0 \tag{13}$$

for every $\varepsilon>0$.

Lemma 1 (Borel-Cantelli Lemma). *Let* $\{A_k\}_{k=1}^{\infty}$ *be a sequence of events occurring with a certain probability distribution, and let* A *be the event consisting of the occurrences of a finite number of events* A_k *for* $k=1,2,\cdots$. *Then*

$$P\left(\bigcap_{n=1}^{\infty}\bigcup_{k\geq n}A_k\right)=0 \tag{14}$$

if

$$\sum_{n=1}^{\infty}P(A_n)<\infty; \tag{15}$$

$$P\left(\bigcap_{n=1}^{\infty}\bigcup_{k\geq n}A_k\right)=1 \tag{16}$$

if the events are totally independent and

$$\sum_{n=1}^{\infty}P(A_n)=\infty. \tag{17}$$

Lemma 2 (Particle State Transference). $q_{01}=0$*;* $q_{00}=1$*;* $q_{11}\leq c\in(0,1)$ *and* $q_{10}\geq 1-c\in(0,1)$.

Proof. In the proposed algorithm, the best solution is updated and saved during the whole iterated procedure. So $q_{01}=0$ and $q_{00}=1$.

Let $\hat{\mathbf{x}}$ is the position with the best fitness among the swarm so far as the time t, i.e. $\hat{\mathbf{x}}=\mathbf{p}^*$. As the definition in Eq. (10), $\exists r>0$, when $\|\mathbf{x}-\hat{\mathbf{x}}\|_\infty\leq r$, we have $|f(\mathbf{x})-f^*|<\varepsilon$. Denote $Q_{\hat{\mathbf{x}},r}=\{x\in\Omega|\|\mathbf{x}-\hat{\mathbf{x}}\|_\infty\leq r\}$. Accordingly

$$Q_{\hat{\mathbf{x}},r}\subset D_0 \tag{18}$$

Then,

$$\begin{aligned}P\{(\mathbf{x}+\Delta\mathbf{x})\in Q_{\hat{\mathbf{x}},r}\}&=\prod_{i=1}^{n}P\{|x_i+\Delta x_i-\hat{x}_i|\leq r\}\\&=\prod_{i=1}^{n}P\{\hat{x}_i-x_i-r\leq\Delta x_i\leq\hat{x}_i-x_i+r\}\end{aligned} \tag{19}$$

where x_i, Δx_i and $\hat{x}_i$ are the i-th dimensional values of $\mathbf{x}$, $\Delta\mathbf{x}$ and $\hat{\mathbf{x}}$, respectively. Moreover, $\hat{v}\sim[-\frac{s}{\rho},\frac{s}{\rho}]$, so that

$$P((\mathbf{x}+\Delta\mathbf{x})\in Q_{\hat{\mathbf{x}},r})=\prod_{i=1}^{n}\int_{\hat{x}_i-x_i-r}^{\hat{x}_i-x_i+r}\frac{\rho}{2\sqrt{2\pi}s}e^{-\frac{\rho^2y^2}{2s^2}}dy \tag{20}$$

Denote $P_1(\mathbf{x})=P\{(\mathbf{x}+\Delta\mathbf{x})\in Q_{\hat{\mathbf{x}},r}\}$ and $\mathbb{C}$ is the convex closure of level set for the initial particle swarm. According to Eq. (20), $0<P_1(\mathbf{x})<1$ ($\mathbf{x}\in\mathbb{C}$). Again, since $\mathbb{C}$ is a bounded closed set, so $\exists\hat{\mathbf{y}}\in\mathbb{C}$,

$$P_1(\hat{\mathbf{y}})=\min_{\mathbf{x}\in\mathbb{C}}P_1(\mathbf{x}),\quad 0<P_1(\hat{\mathbf{y}})<1. \tag{21}$$

Considering synthetically Eqs. (18) and (21), so that

$$q_{10}\geq P_1(\mathbf{x})\geq P_1(\hat{\mathbf{y}}) \tag{22}$$

Let $c=1-P_1(\hat{\mathbf{y}})$, thus,

$$q_{11}=1-q_{10}\leq 1-P_1(\hat{\mathbf{y}})=c\quad(0<c<1) \tag{23}$$

and

$$q_{10}\geq 1-c\in(0,1) \tag{24}$$

□

Theorem 1. *Assume that the TPSO algorithm provides position series* $\mathbf{p}_i(t)(i=1,2,\cdots,n)$ *at time* t *by the iterated procedure.* $\mathbf{p}^*$ *is the best position among the swarm explored so far, i.e.*

$$\mathbf{p}^*(t)=arg\min_{1\leq i\leq n}(f(\mathbf{p}^*(t-1)),f(\mathbf{p}_i(t))) \tag{25}$$

Then,

$$P\left(\lim_{t\to\infty}f(\mathbf{p}^*(t))=f^*\right)=1 \tag{26}$$

Proof. For $\forall\varepsilon>0$, let $p_k=P\{|f(\mathbf{p}^*(k))-f^*|\geq\varepsilon\}$, then

$$p_k=\begin{cases}0 & \text{if } \exists T\in\{1,2,\cdots,k\},\ \mathbf{p}^*(T)\in D_0\\ \bar{p}_k & \text{if } \mathbf{p}^*(t)\notin D_0,\ t=1,2,\cdots,k\end{cases} \tag{27}$$

According to Lemma 2,

$$\bar{p}_k=P\{\mathbf{p}^*(t)\notin D_0,t=1,2,\cdots,k\}=q_{11}^k\leq c^k. \tag{28}$$

Hence,

$$\sum_{k=1}^{\infty}p_k\leq\sum_{k=1}^{\infty}c^k=\frac{c}{1-c}<\infty. \tag{29}$$

According to Lemma 1,

$$P\left(\bigcap_{t=1}^{\infty}\bigcup_{k\geq t}|f(\mathbf{p}^*(k))-f^*|\geq\varepsilon\right)=0 \tag{30}$$

As defined in Definition 2, the sequence $f(\mathbf{p}^*(t))$ converges almost surely or almost everywhere or with probability 1 or strongly towards f^*. The theorem is proven. □

5 Experiments and Discussions

In our experiments the algorithms used for comparison were mainly SPSO (standard PSO) ([6]), FATPSO (fuzzy adaptive turbulent PSO), Genetic Algorithm(GA) ([2]) and Simulated Annealing (SA) ([18, 28]). The four algorithms share many similarities. GA and SA are powerful stochastic global search and optimization methods, which are also inspired from the nature like the PSO.

Genetic algorithms mimic an evolutionary natural selection process. Generations of solutions are evaluated according to a fitness value and only those candidates with high fitness values are used to create further solutions via crossover and mutation procedures.

Simulated annealing is based on the manner in which liquids freeze or metals re-crystalize in the process of annealing. In an annealing process, a

Table 1 Parameter settings for the algorithms

SPSO		
	Swarm size	20
	Self-recognition coefficient c_1	1.49
	Social coefficient c_2	1.49
	Inertia weight w	$0.9 \rightarrow 0.1$
FATPSO		
	Swarm size	20
	Self-recognition coefficient c_1	1.49
	Social coefficient c_2	1.49
	Inertia weight w	0.7
GA		
	Size of the population	20
	Probability of crossover	0.8
	Probability of mutation	0.02
SA		
	Number operations before temperature adjustment	20
	Number of cycles	10
	Temperature reduction factor	0.85
	Vector for control step of length adjustment	2

Table 2 Numerical benchmark functions

Rosenbrock (f_1):	$f_1 = \sum_{i=1}^{n}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$; $\mathbf{x} \in [-2.048, 2.048]^n$, $min(f_1(\mathbf{x}^*)) = f_1(\mathbf{1}) = 0.$
Quadric (f_2):	$f_2 = \sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$; $\mathbf{x} \in [-100, 100]^n$, $min(f_2(\mathbf{x}^*)) = f_2(\mathbf{0}) = 0.$
Schwefel 2.22 (f_3):	$f_3 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$; $\mathbf{x} \in [-10, 10]^n$, $min(f_3(\mathbf{x}^*)) = f_3(\mathbf{0}) = 0.$
Schwefel 2.26 (f_4):	$f_4 = 418.9829n - \sum_{i=1}^{n}(x_i sin(\sqrt{\lvert x_i \rvert}))$; $\mathbf{x} \in [-500, 500]^n$, $min(f_4(\mathbf{x}^*)) = f_4(\mathbf{420.9687}) \approx 0.$
Levy (f_5):	$f_5(\mathbf{x}) = \frac{\pi}{n}\Big(ksin^2(\pi y_1) + \sum_{i=1}^{n-1}((y_i - a)^2 (1 + ksin^2(\pi y_{i+1}))) + (y_n - a)^2\Big)$, $y_i = 1 + \frac{1}{4}(x_i - 1), \quad k = 10, \quad a = 1;$ $\mathbf{x} \in [-10, 10]^n$, $min(f_5(\mathbf{x}^*)) = f_5(\mathbf{1}) = 0.$
Generalized Shubert (f_6):	$f_6 = \prod_{i=1}^{n}\sum_{j=1}^{5}(jcos((j+1)x_i + j))$; $\mathbf{x} \in [-10, 10]^n$, $min(f_6(\mathbf{x}^*))$ is unknown.
Rastrigin (f_7):	$f_7 = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10)$ $\mathbf{x} \in [-5.12, 5.12]^n$, $min(f_7(\mathbf{x}^*)) = f_7(\mathbf{0}) = 0.$
Griewank (f_8):	$f_8 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$; $\mathbf{x} \in [-300, 300]^n$, $min(f_8(\mathbf{x}^*)) = f_8(\mathbf{0}) = 0.$
Ackley (f_9):	$f_9 = -20exp(-0.2\sqrt{\frac{1}{n}\sum_i^n x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$; $\mathbf{x} \in [-32, 32]^n$, $min(f_9(\mathbf{x}^*)) = f_9(\mathbf{0}) = 0.$
Zakharov (f_{10}):	$f_{10} = \sum_i^n x_i^2 + (\sum_i^n \frac{1}{2} i x_i)^2 + (\sum_i^n \frac{1}{2} i x_i)^4$; $\mathbf{x} \in [-10, 10]^n$, $min(f_{10}(\mathbf{x}^*)) = f_{10}(\mathbf{0}) = 0.$

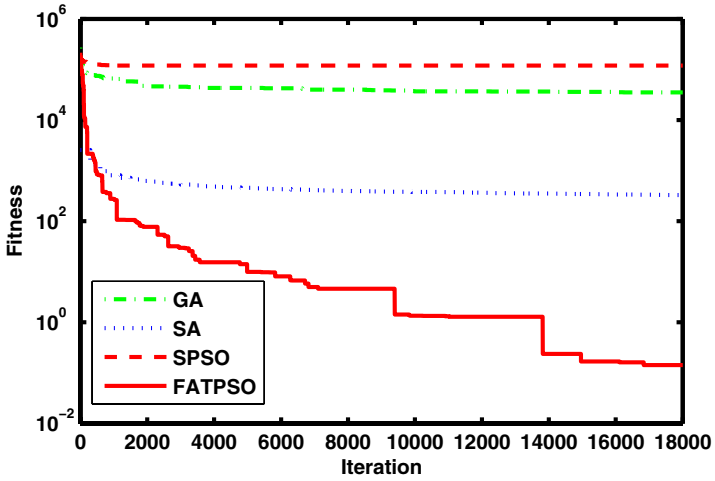

Fig. 3 30-D Quadric (f_2) function performance

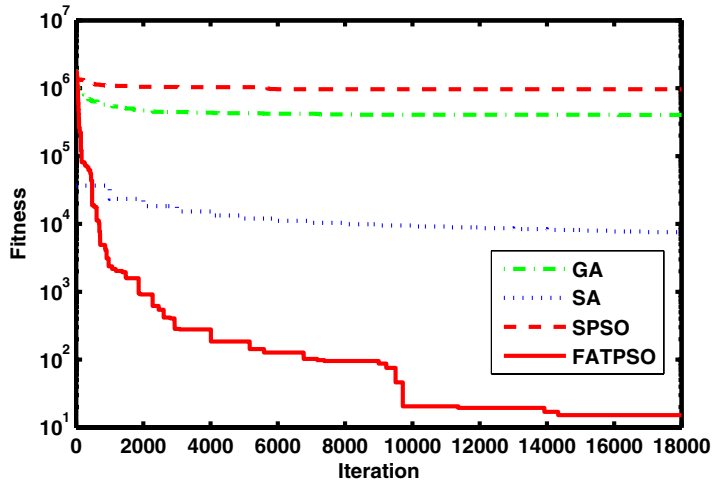

Fig. 4 100-D Quadric (f_2) function performance

melt, initially at high temperature and disordered, is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. In terms of computational simulation, a global minimum would correspond to such a "frozen" (steady) ground state at the temperature $T = 0$.

Both methods are valid and efficient methods in numeric programming and have been employed in various fields due to their strong convergence properties. In the experiments, the specific parameter settings for each of the considered algorithms are described in Table 1. Each algorithm was tested with all the numerical functions shown in Table 2. The first two functions,

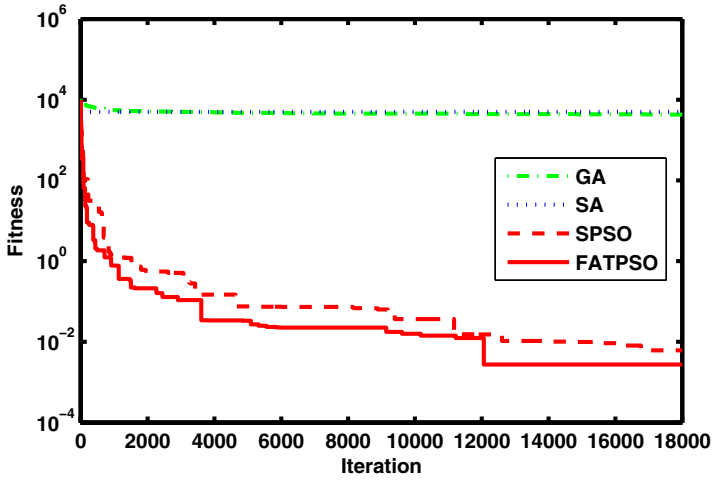

Fig. 5 30-D Schwefel 2.26 (f_4) function performance

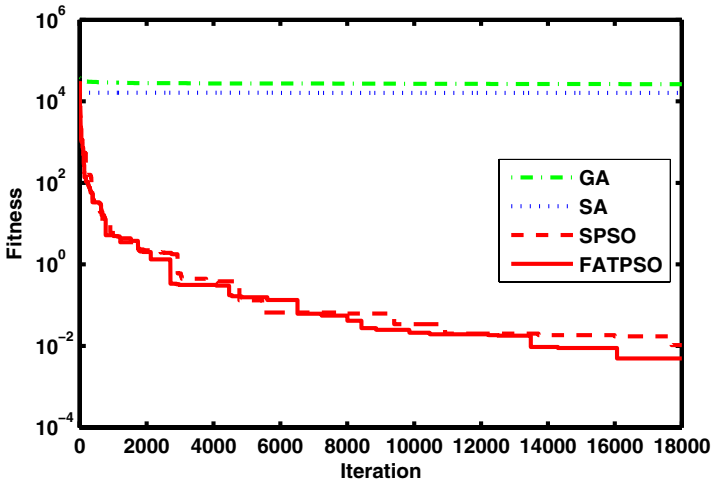

Fig. 6 100-D Schwefel 2.26 (f_4) function performance

namely Rosenbrock's and Quadric function, have a single minimum, while the other functions are highly multimodal with multiple local minima. A new function, Generalized Shubert was constructed temporarily for which global minimum function is unknown for us. It is also useful for us to validate the algorithms without knowing the optimal value. Some of the functions have the sum of their variables, some of them have the product (multiplying), some of them have dimensional effect (ix_i). We tested the algorithms on the different functions in 30 and 100 dimensions, yielding a total of 20 numerical benchmarks. For each of these functions, the goal was to find the global

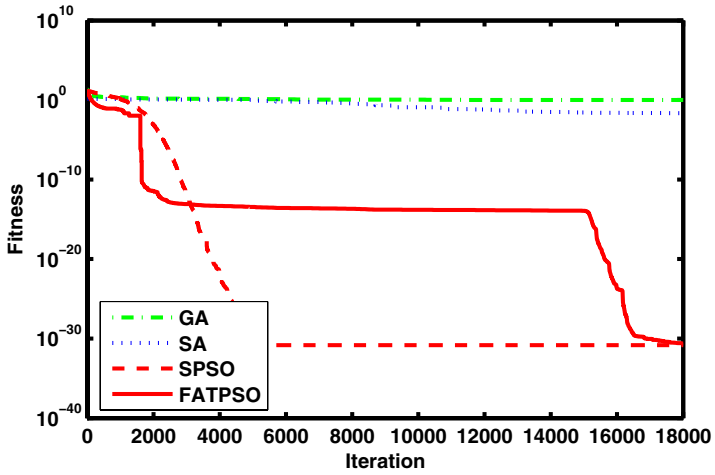

Fig. 7 30-D Levy (f_5) function performance

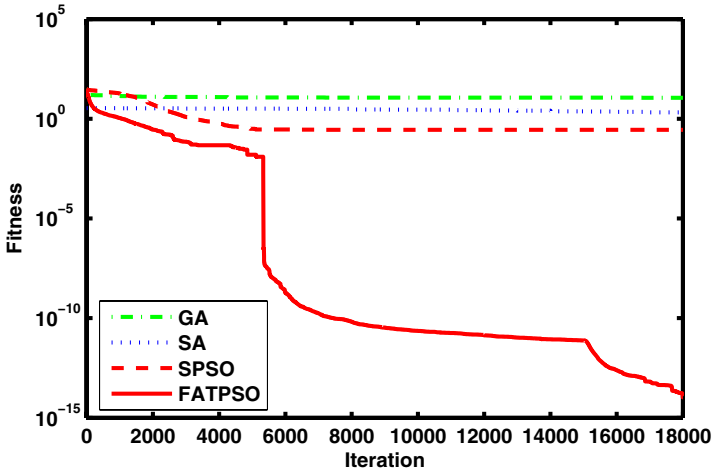

Fig. 8 100-D Levy (f_5) function performance

minima. Each algorithm (for each benchmark) was repeated 10 times with different random seeds. Each trial used a fixed number of 18,000 iterations. The objective functions were evaluated 360,000 times in each trial. Since the swarm size in all PSOs was 20, the size of the population in GA was 20 and the number operations before temperature adjustment (SA) were 20. The average fitness values of the best solutions throughout the optimization run were recorded and the averages and the standard deviations were calculated from the 10 different trials. The standard deviation indicates the differences in the results during the 10 different trials.

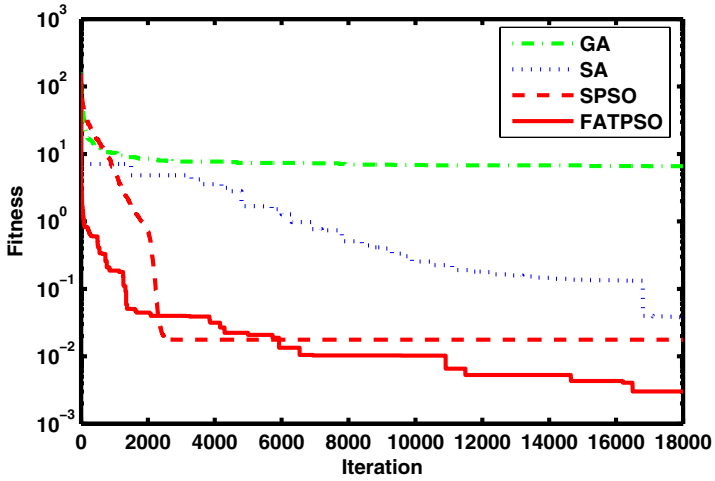

Fig. 9 30-D Griewank (f_8) function performance

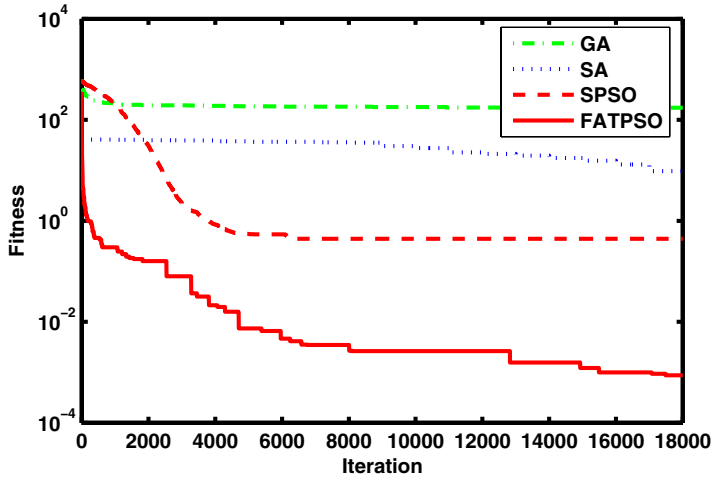

Fig. 10 100-D Griewank (f_8) function performance

Figures 3 to 12 illustrate the mean best function values for the ten functions with two different dimensions (i.e. 30-D and 100-D) using the four algorithms. Each algorithm for different dimensions of the same objective function has similar performance. But in general, the higher the dimension is, the higher the fitness values are. It is observed that for almost all algorithms, the solutions get trapped in a local minimum within the first 2000 iterations except for FATPSO. For the low dimensional problems, SA is usually a cost-efficient choice. For example, SA for 30-D f_8 has a good performance than that in other situations. It is interesting that even if other algorithms are very close

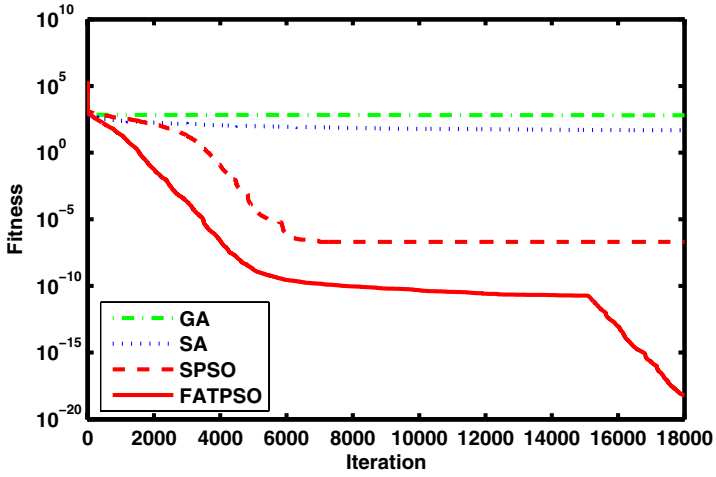

Fig. 11 30-D Zakharov (f_{10}) function performance

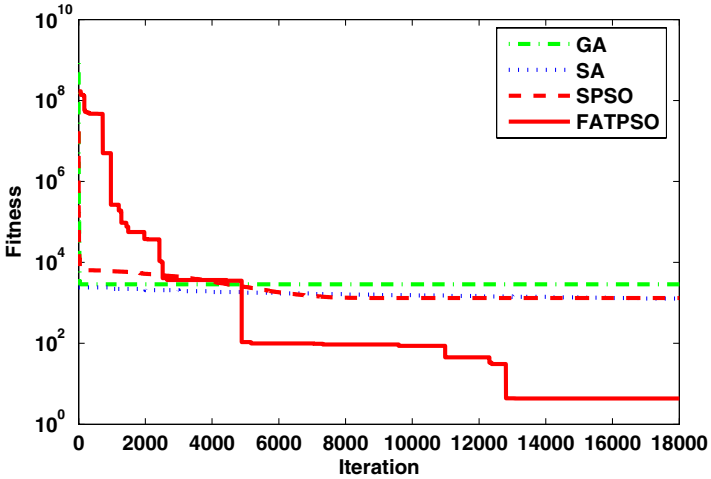

Fig. 12 100-D Zakharov (f_{10}) function performance

to or better than FATPSO in 30-D benchmarks, but a very large difference emerges in the case of 100-D benchmark problems. FATPSO becomes much better than other algorithms in general besides for f_4. The averages and the standard deviations for 10 trials are showed in Table 3. The larger the averages are, wider the standard deviations are usually. There is not too large difference of the standard deviations between the different algorithms for the same benchmark functions. Referring to the empirical results depicted in Table 3, for most of considered functions, FATPSO demonstrated a consistent performance pattern among all the considered algorithms. FATPSO

Table 3 Performance comparison for the function optimization problems

f	D	SPSO	FAPSO	GA	SA
f_1	30	25.4594	1.1048e-004	222.9510	29.0552
		±16.5424	±0.0017	±26.4874	±4.8291
	100	228.6963	6.9026e-004	7.2730e+003	138.3233
		±675.7348	±0.0080	±459.1044	±38.1029
f_2	30	1.1927e+005	2.9699	3.7843e+004	382.7578
		±41.3785	±24.9744	±4.4308e+003	±103.9384
	100	9.6398e+005	54.0376	4.0615e+005	9.5252e+003
		±3.7652e+004	±482.4480	±2.2613e+004	±4.8500+003
f_3	30	2.3732e-008	5.9520e-006	20.2291	0.4991
		±0.3763	±1.3009e-005	±1.4324	±1.8212
	100	55.5606	9.2702e-004	1.2391e+013	23.4349
		±2.3719e-007	±2.6465	±1.2269e+017	±5.0520
f_4	30	0.0501	0.0279	4.5094e+003	4.9754e+003
		±0.2215	±0.1086	±294.7204	±4.2394
	100	0.0481	0.0220	2.7101e+004	1.6131e+004
		±0.7209	±0.6902	±528.3332	±51.7519
f_5	30	1.4685e-031	1.5535e-030	1.0734	0.1617
		±0.0021	±2.6040e-012	±0.1996	±0.4583
	100	0.2806	2.6011e-011	11.4534	2.8817
		±2.1761	±0.1219	±0.4760	±0.4526
f_6	30	-7.4305e+033	-4.0465e+034	-5.1931e+020	-1.5457e+032
		±2.3497e+033	±1.2176e+034	±6.9217e+020	±1.2010e+016
	100	-2.9776e+096	-3.2111e+114	-1.5347e+055	-3.0040e+104
		±1.2330e+096	±2.4430e+114	±9.4580e+054	±4.2442e+101
f_7	30	33.7291	8.4007e-010	204.0560	32.7997
		±17.7719	±9.3676	±6.8450	±6.9936
	100	391.0421	19.9035	1.2070e+003	177.8810
		±176.3618	±115.9034	±23.8156	±37.7808
f_8	30	0.0177	0.0102	6.8463	0.3193
		±0.3157	±0.0149	±0.6060	±1.7880
	100	0.4400	0.0720	179.5966	31.4270
		±14.4633	±0.6945	±7.3908	±11.4656
f_9	30	0.6206	5.4819e-004	1.7437	0.6606
		±0.2996	±0.0086	±0.0427	±0.0657
	100	1.0666	0.0011	2.3570	1.0167
		±0.3921	±0.0059	±0.0079	±0.0532
f_{10}	30	2.0098e-007	5.911e-011	659.0997	62.2253
		±52.8218	±0.0626	±12.0276	±46.5389
	100	1.3223e+003	90.1373	2.8632e+003	1.5625e+003
		±1.4259e+003	±1.7697e+004	±4.7935e-013	±294.7468

performed extremely well with the exception of 30-D f_4, 100-D f_4, 30-D f_5, 30-D f_{10}, in which the results have little difference between the considered algorithms. It is to be noted that FATPSO could be an ideal choice for

solving complex problems (example f_2) when all other algorithms failed to give a better solution.

6 Conclusions

We introduced the Turbulent Particle Swarm Optimization (TPSO) as an alternative method to overcome the problem of premature convergence in the conventional PSO algorithm. TPSO uses a minimum velocity threshold to control the velocity of particles. TPSO mechanism is similar to a turbulence pump, which supply some power to the swarm system. The basic idea is to control the velocity the particles to get out of possible local optima and continue exploring optimal search spaces. The minimum velocity threshold can make the particle continue moving and maintain the diversity of the population until the algorithm converges. We proposed a fuzzy logic based system to tune adaptively the velocity threshold, which is further called as Fuzzy adaptive TPSO (FATPSO). We evaluated and compared the performance of SPSO, FATPSO, GA and SA algorithms on a suite of 20 widely used benchmark problems. The results from our research demonstrated that FATPSO generally outperforms most of the other considered algorithms, especially for high dimensional, multimodal functions.

Acknowledgements. The authors would like to thank Prof. Xiukun Wang, Drs. Ran He and Bo Li for their scientific collaboration in this research work. This work was partially supported by NSFC (60873054).

References

1. Boeringer, D.W., Werner, D.H.: Particle swarm optimization versus genetic algorithms for phased array synthesis. IEEE Transactions on Antennas and Propagation 52(3), 771–779 (2004)
2. Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Dordrecht (2000)
3. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
4. Cordón, O., Herrera, F., Peregrin, A.: Applicability of the fuzzy operators in the design of fuzzy logic controllers. Fuzzy Sets and Systems 86, 15–41 (1997)
5. Du, F., Shi, W.K., Chen, L.Z., Deng, Y., Zhu, Z.F.: Infrared image segmentation with 2-D maximum entropy method based on particle swarm optimization. Pattern Recognition Letters 26, 597–603 (2005)
6. Eberhart, R.C., Shi, Y.H.: Comparison between genetic algorithms and particle swarm optimization. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 611–616 (1998)
7. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transations on Evolutionary Computation 3(2), 124–141 (1999)

8. Feller, W.: An Introduction to Probability Theory and Its Application, 3rd edn. John Wiley & Sons, Chichester (1968)
9. Guo, C., Tang, H.: Global convergence properties of evolution stragtegies. Mathematica Numerica Sinica 23(1), 105–110 (2001)
10. He, R., Wang, Y., Wang, Q., Zhou, J., Hu, C.: An improved particle swarm optimization based on self-adaptive escape velocity. Journal of Software 16(12), 2036–2044 (2005)
11. Herrera, F., Lozano, M.: Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions. Soft Computing 7, 545–562 (2003)
12. Jiang, C.W., Etorre, B.: A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation. Mathematics and Computers in Simulation 68, 57–65 (2005)
13. Kennedy, J., Spears, W. M.: Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 78–83 (1998)
14. Kennedy, J., Eberhart, R.: Swarm intelligence. Morgan Kaufmann Publishers, Inc., San Francisco (2001)
15. Lu, W.Z., Fan, H.Y., Lo, S.M.: Application of evolutionary neural network method in predicting pollutant levels in downtown area of Hong Kong. Neurocomputing 51, 387–400 (2003)
16. Mahfouf, M., Chen, M.Y., Linkens, D.A.: Adaptive weighted swarm optimization for multiobjective optimal design of alloy steels. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 762–771. Springer, Heidelberg (2004)
17. Mark, L., Shay, E.: A fuzzy-based lifetime extension of genetic algorithms. Fuzzy Sets and Systems 149, 131–147 (2005)
18. Orosz, J.E., Jacobson, S.H.: Analysis of static simulated annealing algorithms. Journal of Optimzation theory and Applications 115(1), 165–182 (2002)
19. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Natural Computing 1, 235–306 (2002)
20. Parsopoulos, K.E., Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization. IEEE Transactions on Evolutionary Computation 8(3), 211–224 (2004)
21. Phan, H.V., Lech, M., Nguyen, T.D.: Registration of 3D range images using particle swarm optimization. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 223–235. Springer, Heidelberg (2004)
22. Schute, J.F., Groenwold, A.A.: A study of global optimization using particle swarms. Journal of Global Optimization 31, 93–108 (2005)
23. Shi, Y.H., Eberhart, R.C., Chen, Y.: Implementation of evolutionary fuzzy systems. IEEE Transactions on Fuzzy System 7(2), 109–119 (1999)
24. Shi, Y.H., Eberhart, R.C.: Fuzzy adaptive particle swarm optimization. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 101–106 (2001)
25. Sousa, T., Silva, A., Neves, A.: Particle swarm based data mining algorithms for classification tasks. Parallel Computing 30, 767–783 (2004)
26. Ting, T., Rao, M., Loo, C.K., Ngu, S.S.: Solving unit commitment problem using hybrid particle swarm optimization. Journal of Heuristics 9, 507–520 (2003)

27. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317–325 (2003)
28. Triki, E., Collette, Y., Siarry, P.: A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. European Journal of Operational Research 166, 77–92 (2005)
29. van den Bergh, F.: An analysis of particle swarm optimizers, PhD thesis, University of Pretoria, South Africa (2002)
30. Yun, Y.S., Gen, M.: Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics. Fuzzy Optimization and Decision Making 2, 161–175 (2003)