

MARKOV CHAIN AND ADAPTIVE PARAMETER SELECTION ON PARTICLE SWARM OPTIMIZER

Chao-Wei Chou, Jiann-Horng Lin* and Rong Jeng

Department of Information Management
I-Shou University, Kaohsiung 840, Taiwan
{choucw, jhlin, rjeng}@isu.edu.tw

*Corresponding author , E-mail address: jhlin@isu.edu.tw

ABSTRACT

Particle Swarm Optimizer (PSO) is such a complex stochastic process so that analysis on the stochastic behavior of the PSO is not easy. The choosing of parameters plays an important role since it is critical in the performance of PSO. As far as our investigation is concerned, most of the relevant researches are based on computer simulations and few of them are based on theoretical approach. In this paper, theoretical approach is used to investigate the behavior of PSO. Firstly, a state of PSO is defined in this paper, which contains all the information needed for the future evolution. Then the memory-less property of the state defined in this paper is investigated and proved. Secondly, by using the concept of the state and suitably dividing the whole process of PSO into countable number of stages (levels), a stationary Markov chain is established. Finally, according to the property of a stationary Markov chain, an adaptive method for parameter selection is proposed.

KEYWORDS

Markov chain, Memory-less property, Order Statistics, Particle Swarm Optimizer, Percentile, Stationary Markov chain

1. INTRODUCTION

The Particle Swarm Optimizer (PSO), introduced by Kennedy and Eberhart, is a stochastic optimization technique likened to the behavior of a flock of birds or the sociological behavior of a group of people [10], [11]. PSO is a population based optimization technique. The population is called a swarm. PSO is motivated from the simulation of social behavior of the group of individuals and therefore different from other evolutionary computational methods. It generates new values for all particles (individuals) in the swarm (population). PSO memorizes the previous individual and social (swarm) experience and uses them to help generate new particles. This idea for generation of new individuals differentiates PSO from other population-based algorithms.

PSO is a high performance optimizer that possesses several desirable attributes, including the fact that the basic algorithm is very easy to understand and implement. It is similar in some ways to genetic algorithm (GA) and evolutionary algorithms. But it requires very less computational bookkeeping and fewer lines of code. Hence, it gains increasing popularity with its wide applicability and effectiveness in performing difficult optimization tasks. Among other applications, it has been applied to many problems, including multi-objective problems, mini-max problems, integer programming problems, errors-in-variables problems, and numerous

engineering applications. It also shares the ability of the genetic algorithm to handle arbitrary nonlinear cost functions, but with a much simpler implementation. It clearly demonstrates good possibilities for widespread use in electromagnetic optimization [3]. Though PSO has the advantage of being easy, it suffers from the “curse of dimensionality” just like other stochastic optimization algorithms including GA. This implies that the performance deteriorates as the dimensions of the search space increase. It does not possess the ability to improve upon the quality of the solutions as the number of generations was increased. We give a simple explanation in the following. Consider a basic stochastic global search algorithm that generates a sample from a uniform distribution on the entire search space. The algorithm stops when it generates a solution that falls in the optimality region, a small volume of the search space surrounding the global optimum such that the fitness is under a given threshold. The probability of generating a sample inside the optimality region is simply the volume of the optimality region divided by the volume of the search space. This probability will decrease exponentially as the dimensionality of the search space increases. Given this explanation, it is clear that it is typically significantly harder to find the global optimum of a high-dimensional problem, compared with a low-dimensional problem with similar topology. Beside the “curse of dimensionality,” another drawback of PSO is associated with the lack of a mechanism responsible for the control of the magnitude of the velocities, which may incur the danger of swarm explosion and divergence. To address the explosion problem a threshold on the absolute value of the velocity that can be assumed by any particle was incorporated. Also, PSO suffers from “being trapped in local minima” or “having slower convergence rate.”

To overcome the above difficulties, many researches are devoted to improving the performance of the original PSO. For example, the original PSO does not have an inertia weight, i.e., $\omega(r) = 0$ in (6), while Shi and Eberhart [23] introduced the improvement by adding an inertia weight, which results in faster convergence and increases the overall performance of PSO. Angeline [1] introduced a form of selection such that the properties making some solutions superior are transferred directly to some less effective particles. Angeline used a tournament selection process based on the particles' current fitness, copying the current positions and velocities of the better half of the population onto the worse half. This technique improved the performance of PSO in three of the four test functions. Kennedy [17] proposed the LBEST method by dividing the swarm into multiple “neighborhoods,” with each neighborhood maintaining its own local best solution. This approach is less prone to becoming trapped in local minima, but typically has slower convergence.

Since PSO and Genetic Algorithm (GA) both work with a population of solutions, Bergh and Engelbrecht [2] combined the searching abilities of both methods. Based on the compensatory property of GA and PSO, they propose a new algorithm, a variation on the traditional PSO algorithm, called the cooperative particle swarm optimizer (CPSO) that combines the evolution ideas of both. On the other side Parsopoulos and Vrahatis [22] propose a technique aiming to compute all global optimizers, while avoiding local optimizers, through PSO. Because the existing methods are all simulation-based, in this paper the theoretical part of PSO will be the main concern.

A simple explanation of the PSO operation is as follows. Each particle in the whole swarm represents a possible solution to the optimization task at hand. During iterations each particle accelerates in the direction of its own personal (individual) best solution found so far, as well as in the direction of the global best position discovered so far by all the particles in the swarm. This means that if a particle discovers a promising new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process.

Instead of using evolutionary operators to manipulate the individuals, like in evolutionary computational algorithms, each particle in PSO flies in the search space with a velocity that is dynamically adjusted according to its own flying experience and its companions' flying experience. Through this paper, we assume a D-dimensional search space S with

$$S = (a_1, b_1) \times (a_2, b_2) \times \cdots \times (a_D, b_D), \quad (1)$$

where a_1, a_2, \dots, a_D and b_1, b_2, \dots, b_D are the lower bounds and upper bounds of respective dimensions. We have a swarm consisting of I particles on S. Let i be the index of particle and r be the index of iteration. The position of the i th particle after its r th iteration is a D-dimensional random vector

$$\tilde{X}_i(r) = (X_{i,1}(r), X_{i,2}(r), \dots, X_{i,D}(r)) \in S, i = 1, 2, \dots, I; r = 1, 2, \dots, R. \quad (2)$$

The velocity acting on the i th particle at its r th iteration is also a D-dimensional random vector

$$\tilde{V}_i(r) = (V_{i,1}(r), V_{i,2}(r), \dots, V_{i,D}(r)) \in S, i = 1, 2, \dots, I; r = 1, 2, \dots, R. \quad (3)$$

The personal best position of the i th particle after its r th iteration is recorded and represented as

$$\tilde{X}_i^P(r) = (X_{i,1}^P(r), X_{i,2}^P(r), \dots, X_{i,D}^P(r)) \in S, i = 1, 2, \dots, I; r = 1, 2, \dots, R. \quad (4)$$

The global best position after the r th iteration of the i th particle is recorded and represented as

$$\tilde{X}_i^G(r) = (X_{i,1}^G(r), X_{i,2}^G(r), \dots, X_{i,D}^G(r)) \in S, i = 1, 2, \dots, I; r = 1, 2, \dots, R. \quad (5)$$

The conditions of the whole swarm before the first iteration are given as follows. $\tilde{X}_i(0) \in S$ is given at random, i.e., $X_{i,1}(0), X_{i,2}(0), \dots, X_{i,D}(0), i = 1, 2, \dots, I$ are mutually independent and uniformly distributed on $(a_1, b_1), (a_2, b_2), \dots, (a_D, b_D)$, respectively. The initial velocities are set to be zero vectors, i.e. $\tilde{V}_i(0) = \mathbf{0}$. The initial personal and global best positions are set to be that

$$\tilde{X}_i^P(0) \equiv X_i(0)$$

and

$$\tilde{X}_i^G(0) \equiv \text{the best of } X_1^P(0), X_2^P(0), \dots, X_D^P(0).$$

Here a continuous, non-constant fitness function f is given and the positions \tilde{X} is said to be "better" than \tilde{Y} means that \tilde{X} has lower fitness function value, i.e. $f(\tilde{X}) < f(\tilde{Y})$; \tilde{X} is said to be "best" if $f(\tilde{X})$ is the minimum. Now the initial conditions are determined, the iterations can keep going on. Before each position renewal, the previous personal best and global best positions are used to compute the new velocities and then the new position is determined according to ((6) and (7)). After position renewal, the new personal best and global best positions are computed and possibly updated immediately according to (8) and (9). The iteration is by the following order:

first iteration of first particle ($r = 1, i = 1$), first iteration of second particle ($r = 1, i = 2$), ..., first iteration of I th particle ($r = 1, i = I$), second iteration of first particle ($r = 2, i = 1$), ..., In (6)-(9), the action of the r th iteration, $r = 1, 2, \dots, R$, on particle i , $i = 1, 2, \dots, I$, is given as follows :

$$\tilde{V}_i(r) = \omega(r)\tilde{V}_i(r-1) + (\tilde{U}^{(0,c_1)}(r) \otimes (\tilde{X}_i^P(r-1) - \tilde{X}_i(r-1))) + (\tilde{U}^{(0,c_2)}(r) \otimes (\tilde{X}_{i-1}^G(r-1) - \tilde{X}_i(r-1))), \quad (6)$$

where $\omega(r)$ is a parameter called the inertia weight and is a function of r ; c_1, c_2 are positive constants, referred to as cognitive and social parameters, respectively;

$$\tilde{U}^{(0,c_k)}(r) = U_1^{(0,c_k)}(r), U_2^{(0,c_k)}(r), \dots, U_D^{(0,c_k)}(r), k = 1, 2,$$

where the components

$U_d^{(0,c_k)}(r), d = 1, 2, \dots, D$, are i.i.d. uniform random variables in the range $(0, c_k)$, $k = 1, 2$.

The vector product \otimes is defined as

$$(a_1, a_2, \dots, a_D) \otimes (b_1, b_2, \dots, b_D) \equiv (a_1 b_1, a_2 b_2, \dots, a_D b_D)$$

After the computation of the new velocity, the new position of the particle i after its r th iteration is computed by

$$\tilde{X}_i(r+1) = \tilde{X}_i(r) + \tilde{V}_i(r+1) \quad (7)$$

With the new position in (7) and its new fitness function value $f(\tilde{X}_i(r+1))$, the personal best position and global best position are possibly renewed using (8) and (9).

$$\tilde{X}_i^P(r) \equiv \text{the better of } \tilde{X}_i^P(r-1) \text{ and } \tilde{X}_i(r), \quad (8)$$

$$\tilde{X}_i^G(r) \equiv \text{the better of } \tilde{X}_{i-1}^G(r) \text{ and } \tilde{X}_i^P(r), \quad (9)$$

Here $\tilde{X}_{i-1}^G(r)$ denotes the global best after the r th iteration of the particle ($i-1$). Note that, to be consistent in (9). As for $i = 1$, we define

$$\tilde{X}_0^G(1) \equiv \tilde{X}_I^G(0), \tilde{X}_0^G(2) \equiv \tilde{X}_I^G(1), \dots, \tilde{X}_0^G(R) \equiv \tilde{X}_I^G(R-1).$$

In the iteration process, $\omega(r)$ in (6) is employed to manipulate the impact of the previous velocity on the current velocity. It can resolve the tradeoff between the global (wide ranging) and local (nearby) exploration ability of the swarm. A larger $\omega(r)$ encourages global exploration (moving to previously not encountered areas of the search space), while a smaller one promotes local exploration, i.e., fine-tuning the current search area. A suitable value for *inertia weight* provides the desired balance between the global and local exploration ability of the swarm and, consequently, improves the effectiveness of the algorithm. Experimental results suggest that it is preferable to initialize the inertia weight to a large value, giving priority to global exploration of the search space, and gradually decreasing so as to obtain refined solutions. But the condition depends on the scenarios, or more precisely, on the fitness function.

2. PROBLEM STATEMENT AND PRELIMINARIES

Up to present most of the modified methods are still faced with a dilemma of compromise between the cases of being trapped in local minima or having slower convergence rate. Among them many papers deal with the selection of parameters to obtain the best performance.

For the selection of parameter ω , most of them use decreasing values during the iteration process, i.e. $\omega = \omega(r)$ is a linearly decreasing function of r from around 1.0 to 0.4 [6, 12, 14, 18, 19, 24], or a exponentially decreasing function of r [15], while some treat as constant throughout the iteration process between 0.5 and 1 [7, 13, 21]. Among others, in [8] ω is a linearly increasing function of r .

For the selection of parameters c_1, c_2 , most of them use the same constant value ($c_1 = c_2 = 2$ in [14, 16, 18, 20, 21] and $c_1 = c_2 = 1$ in [19]), or fixed constant pair ($(c_1, c_2) = (2.0, 1.0)$ in [4], $(c_1, c_2) = (3.0, 5.0)$ in [12], $(c_1, c_2) = (1.8, 1.0)$ in [13]). On a different side, [7] increases c_1 from 0.35 to 2.4 and decreases c_2 from 2.4 to 0.35 throughout the iteration process.

As for the selection of parameter V_{\max} , almost all of them use a constant value except that [9] decreases it throughout the iteration process.

By considering the Markov property which will be introduced and proved in the following, it seems to have a better way of parameter selection since the fate of PSO depends on the present achievement (fitness value) instead of iteration r . This details will be explained in the following. As far as our investigation is concerned, most of the relevant researches are based on computer simulations and seldom of them are based on theoretical approach. Hence, theoretical approach about the behaviour of PSO will be the main concern in this paper. Some theoretical results as well as heuristic adaptive methods on parameter selection will be proposed.

Before going into the details of the theoretical results, some preliminary results will be given first. Theorem 2.1 and Lemma 2.1 are from Chou etc. [5]. Theorem 2.1 gives the result that even the non-cooperative independent and identically distributed (i.i.d.) searching algorithm will converge and find the solution eventually under suitable conditions, even though no personal experience and group experience are incorporated in the searching process. The theorem is stated in the following.

Theorem 2.1. (Converge in probability of i.i.d. search) Let f be a D-dimensional continuous, non-constant function on domain S in (1), and $f(\tilde{X}_0)$ be the minimum value of f on S , i.e.

$$f(\tilde{X}_0) \leq f(\tilde{X}), \forall \tilde{X} \in S.$$

Furthermore, let $\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_n$ be i.i.d. uniform random vector on S in (1) and

$f(\tilde{X}_{1n}) \leq f(\tilde{X}_{2n}) \leq \dots \leq f(\tilde{X}_{nn})$ be the order statistics of $f(\tilde{X}_1), f(\tilde{X}_2), \dots, f(\tilde{X}_n)$. Then we have

$$\lim_{n \rightarrow \infty} f(\tilde{X}_{1n}) = f(\tilde{X}_0) \text{ in probability.}$$

Theorem 2.1 proposes a convergence behavior of a non-cooperative i.i.d. searching algorithm if the fitness function is continuous.

Furthermore, the real distribution can be approximated by empirical distribution for large data set and the estimation of probability percentile is of great importance in practical application. In Lemma 2.1, the minimum of n realized function values can be used to estimate the $1/n$ -percentile of the distribution of the function value and, with the help of Theorem 2.1, the estimate will converge to the true minimum. This can help to estimate the percentiles for the distribution and can be used in our adaptive parameter selection.

Lemma 2.1. (Estimation of Percentile) As in Theorem 1, let $\tilde{Y}, \tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_n$ be i.i.d. uniform random vector and the random functions $f(\tilde{X}_{1n}) \leq f(\tilde{X}_{2n}) \leq \dots \leq f(\tilde{X}_{nn})$ be the order statistics of $f(\tilde{X}_1), f(\tilde{X}_2), \dots, f(\tilde{X}_n)$ and $f(\tilde{x}_{1n}) \leq f(\tilde{x}_{2n}) \leq \dots \leq f(\tilde{x}_{nn})$ be their realized function values. Then

$$\Pr(f(\tilde{Y}) \leq f(\tilde{x}_{1n})) \sim \frac{1}{n}, \text{ i.e. } \lim_{n \rightarrow \infty} \frac{\Pr(f(\tilde{Y}) \leq f(\tilde{x}_{1n}))}{\frac{1}{n}} = 1.$$

From lemma 2.1, we know that, for large n , the realized value $f(\tilde{x}_{1n})$ can be treated as a good estimate for $(1/n)$ -percentile of $f(\tilde{Y})$. The result of Lemma 1 gives a theoretical base for estimation of percentiles. To be more precise, when $\alpha = 1/n$, then $f(\tilde{x}_{1n})$ can be a good estimate of α -percentile for f values. The percentiles will help to build our Markov chain as well as index of comparison in the following.

The rest of this paper is organized as follows. In Section 3 we define the state of a PSO and accordingly investigate the memory-less property of the state. This sets the foundation for the Markov chain in later sections. In Section 4 a Markov chain is established and investigated. Adaptive method of parameter selection in PSO process is then presented. In Section 5, some conclusion and discussion is given for the future research.

3. MEMORY-LESS PROPERTY OF PSO STATES

In this section, the state of PSO is defined and its memory-less property is investigated. Also, the definition and the concept of α -Dominating is given for the measure of achievement in a probability sense.

3.1. A definition of the state of a PSO

Before we investigate the Markov (memory-less) property of the states of PSO, we must well define the states of PSO and investigate them. We hope that the state defined in this paper can be a good indicator throughout the whole process. Thus the states must contain as much information as possible that is concerned in the process.

In the following we use the index of time t to represent every position change and function evaluation, i the index of particle. $t = 1$ is the time of the first position change and function evaluation of the first particle ($i = 1$) and $t = 2$ is the time of the first position change and function

evaluation of the second particle ($i = 2$), and so on. For every t only one particle moves and re-compute its fitness function value. The index i of the moving particle and the time t satisfy the following relation:

$$i = \begin{cases} t \pmod{I}, & \text{if } t \pmod{I} \neq 0 \\ i = I, & \text{if } t \pmod{I} = 0 \end{cases} \quad (10)$$

$$\tilde{V}_i(t) = \begin{cases} \omega(t)\tilde{V}_i(t-1) + (\tilde{U}^{(0,c_1)}(t) \otimes (\tilde{X}_i^P(t-1) - \tilde{X}_i(t-1))) \\ \quad + (\tilde{U}^{(0,c_2)}(t) \otimes (\tilde{X}^G(t-1) - \tilde{X}_i(t-1))), & \text{for } i \text{ satisfies (10),} \\ 0, & \text{other } i. \end{cases} \quad (11)$$

$$\tilde{X}_i(t) = \begin{cases} \tilde{X}_i(t-1) + \tilde{V}_i(t) & \text{for } i \text{ satisfies (10),} \\ \tilde{X}_i(t-1), & \text{other } i. \end{cases} \quad (12)$$

$$\tilde{X}_i^P(t) = \begin{cases} \text{the better one of } (\tilde{X}_i^P(t-1), \tilde{X}_i(t)), & \text{for } i \text{ satisfies (10),} \\ \tilde{X}_i^P(t-1) & \text{other } i. \end{cases} \quad (13)$$

$$\tilde{X}^G(t) = \text{the better one of } (\tilde{X}^G(t-1), \tilde{X}_i^P(t)), \quad \text{for } i \text{ satisfies (10)}. \quad (14)$$

Here $\tilde{X}_i(t), i = 1, 2, \dots, I$, denote the position (vector) at time t ; $\tilde{X}_i^P(t), i = 1, 2, \dots, I$, the personal best position (vector) at time t ; $\tilde{V}_i(t)$, for i satisfies (10), the moving particle's velocity at time t and $\tilde{X}^G(t)$ denotes the global best position at time t . We use t instead of r in (11) to (14) because the state changes with t increases. Every time only one particle moves according to (11) and (12). Then function evaluation is made to renew the position of the moving particle, the fitness function value is re-computed and the personal best position (of moving particle) and global best position (of whole swarm) are possibly changed ((13) and (14)). Note that every time only one particle is moved and the other particles inherit their previous values of positions and personal best positions ((12) and (13)). Also note that $f(\tilde{X}_i^P(t)), i = 1, 2, \dots, I$, and $f(\tilde{X}^G(t))$ are all non-increasing functions of t .

Combine the formulas (11)-(14) and define the state at time t , as $\tilde{W}(t)$,

$$\tilde{W}(t) \equiv (\tilde{X}_1(t), \tilde{X}_2(t), \dots, \tilde{X}_I(t), \tilde{X}_1^P(t), \tilde{X}_2^P(t), \dots, \tilde{X}_I^P(t), \tilde{V}_1(t), \tilde{V}_2(t), \dots, \tilde{V}_I(t), \tilde{X}^G(t)). \quad (15)$$

Note that in (15) the information needed for the future revolution is contained in the present state. Thus we have the following memory-less property of the state.

3.2. The memory-less property of the PSO state

As mentioned at the end of previous paragraph, a theorem on the memory-less property of the PSO state is given.

Theorem 3.1. The states of the PSO, as defined in (15), are memory-less, i.e. the conditional distribution function of the next state $\tilde{W}(t+1)$, given the present and historical states, $\tilde{W}(t), \tilde{W}(t-1), \dots, \tilde{W}(1)$, is the same as the conditional distribution function of $\tilde{W}(t+1)$, given only the present state $\tilde{W}(t)$, i.e.

$$DF(\tilde{W}(t+1) | \tilde{W}(t), \tilde{W}(t-1), \dots, \tilde{W}(1)) = DF(\tilde{W}(t+1) | \tilde{W}(t)). \quad (16)$$

Proof. Without loss of generality we assume that at present time t only particle i moves and re-compute its fitness function, while at future time $(t+1)$ only particle $(i+1)$ moves and re-compute its fitness function. We discuss the change of the components from that in $\tilde{W}(t)$ to that in $\tilde{W}(t+1)$ in the following. Firstly, the future velocity of particle will be set to be zero, i.e. $\tilde{V}_i(t+1) = 0$ since particle will not move at time $(t+1)$. The other components of $\tilde{W}(t+1)$ inherit their previous values in $\tilde{W}(t)$ except only four components: $\tilde{V}_{i+1}(t+1) = 0, \tilde{X}_{i+1}(t+1), \tilde{X}_{i+1}^P(t+1)$, and $\tilde{X}^G(t+1)$. Among them $\tilde{V}_{i+1}(t+1) = 0$ depends only on the present state $\tilde{W}(t)$ by (11). The other three also depend only on the present state $\tilde{W}(t)$ by (12), (13) and (14) respectively. Thus $\tilde{W}(t+1)$ depends only on $\tilde{W}(t)$. This completes the proof.

3.3. The Concept of α _Dominating

With the memory-less property, the fate of PSOs depends only on the present state while not on the past history. Therefore what is the most important is “what the state is now” instead of “what the state used to be.” Therefore we can investigate the influence of the present state on the future state regardless of the historical state. Also, the future states of the PSOs do not depend directly on the time t . This paper will give the result that the state of PSOs constitutes a stationary Markov chain and therefore depend on t at most through the present state. We explain this phenomenon in details in the following. As t increases, the state may gain considerably better achievement, but it is not definite. If for a specific sample process, as t increases while the state may gain no significant achievement, then the future state distribution will not be favored even t is very large. Hence it seems better to consider different parameters portfolio during different state stage instead of time stage as the previous research did.

To evaluate the achievement (goodness) of a state, we define an index for the state to represent the present achievement of PSO. This index is based on the concept of probability. Since the achievement is position-dependent, firstly we define an index relevant to the positions.

Definition 3.1. (α _Dominating of positions) A position (vector) $Y_0 \in S$ is said to be α _Dominating, $0 \leq \alpha \leq 1$, with respect to f and its domain S , if the probability that a uniform distributed random vector is better than Y_0 is α , i.e. $\Pr(f(Y_1) < f(Y_0)) = \alpha$.

We discuss two extreme cases in the following:

- (i) When $\alpha = 0$, Y_0 is a best position.
- (ii) When $\alpha = 1$, Y_0 is a worst position.

With α smaller, the position (fitness function) is better achieved, but it is not to be assured that the position is geographically nearer the global optimum. Maybe it is only nearer to one local optimum. In the following, we discuss a concept similar to that in Definition 3.1 with respect to the state space.

Definition 3.2. (α _Dominating of states) A state \tilde{W} is said to be α _Dominating, or with α value (in dominating), with respect to f and its domain S , if its global best position is α _Dominating.

According to Definition 3.2, the immediate result in Lemma 3.1 follows.

Lemma 3.1. The α value (in dominating) of a state depends only on its global best position.

Furthermore, we define one relation between two states in Definition 3.3.

Definition 3.3. (Equal_Dominating relation) Two state \tilde{W}_1 and \tilde{W}_2 are said to be Equal_Dominating to each other, if they are both α _Dominating for the same α .

Note that a state is better if its α value is smaller and two states are of little difference in view of achievement if they are Equal_Dominating. Lemma 3.2 gives the result that it's an equivalent relation.

Lemma 3.2. The relation of Equal_Dominating is an equivalent relation.

Proof. It's rather intuitive that the Equal_Dominating relation is reflexive, symmetric and transitive. We omit the proof.

According to Lemma 3.2, we can divide the whole state space of PSO into uncountable number of equivalent classes. Each class corresponds to one value. Within one class every pair of the states, \tilde{W}_1 and \tilde{W}_2 , are Equal_Dominating to each other. Whereas for any pair of states that are not in the same class, they are not Equal_Dominating. But the total number of α values, or equivalent classes, is uncountable and thus very difficult for analysis. In the following, the equivalent classes will be merged into countable ones and the analysis tasks will thus be simplified.

4. MARKOV CHAIN ON PSO LEVELS

In this section we merge the equivalent classes into countable ones, as mentioned in the end of Section 3, to reduce the number of equivalent classes. We call the merged classes "levels". The classification of levels will be according to (17) and Definition 4.1 in the following.

$$\tilde{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n, \dots), 1 = \alpha_0 > \alpha_1 > \alpha_2 > \dots > \alpha_n > \dots > 0 \quad (17)$$

Definition 4.1. Given a countable sequence $\{\alpha_n, n=0,1,\dots\}$ as in (17), a state \tilde{W} is said to be Level_ n _Dominating, , (or simply "in Level n "), denote by $L(\tilde{W}) = n$, with respect to f, S , and $\tilde{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n, \dots)$, if the state \tilde{W} is α _Dominating with $\alpha_n \geq \alpha > \alpha_{n+1}$.

With Definition 4.1 the states are merged into countable number of classes, called levels, by taking into account the countable intervals of α -value instead of uncountable single α -value. Now a similar relation can be defined as that in Definition 4.2.

Definition 4.2. (Equal_Level relation) Two states are said to be Equal_Level to each other, if $L(\tilde{W}_1) = L(\tilde{W}_2) = n$ for some $n \in \{0, 1, 2, \dots\}$.

Just as in Lemma 3.2, a similar result of equivalent equation for Equal_Level is stated in the following without proof.

Lemma 4.1. The relation of Equal_Level is an equivalent relation.

Note that if two states are Equal_Dominating, then they must be Equal_Level, but not vice versa. To help the proof in the following theorem, define $\Omega_n, n = 0, 1, 2, \dots$, to be the classes of state space for \tilde{W} such that $\Omega_n = \{\tilde{W} \mid L(\tilde{W}) = n\}$.

Give an example by setting $\alpha_n = 10^{-n}$, i.e. $\tilde{\alpha} = (10^0, 10^{-1}, 10^{-2}, \dots)$, then we have the sequence $\{\Omega_n, n = 0, 1, 2, \dots\}$ be a partition of the whole state space. In this example a state is in Level- n means that its global best position attains the 10^{-n} -percentile but not yet the $10^{-(n+1)}$ -percentile. With the aid of Definition 4.2, a Markov chain can be established.

Theorem 4.1. In PSO, the stochastic process of the levels of the state $\{L(\tilde{W}(t)), t = 1, 2, \dots\}$, or simply denote by $\{L(t), t = 1, 2, \dots\}$ as defined in Definition 4.2, constitutes a Markov chain.

Proof. Given the history of the previous levels

$$L(t) = m, L(t-1) = m_{t-1}, L(t-2) = m_{t-2}, \dots, L(1) = m_1, m, m_{t-1}, m_{t-2}, \dots, m_1 \in \{0, 1, \dots\},$$

the conditional probability of $L(\tilde{W}(t+1)) = n \in \{0, 1, 2, \dots\}$ is

$$\begin{aligned} & \Pr[L(\tilde{W}(t+1)) = n \mid L(\tilde{W}(t)) = m, L(\tilde{W}(t-1)) = m_{t-1}, L(\tilde{W}(t-2)) = m_{t-2}, \dots, L(\tilde{W}(1)) = m_1] \\ &= \iiint_{\tilde{W}(t+1) \in \Omega_n} DF(\tilde{W}(t+1) \mid L(\tilde{W}(t)) = m, L(\tilde{W}(t-1)) = m_{t-1}, L(\tilde{W}(t-2)) = m_{t-2}, \dots, L(\tilde{W}(1)) = m_1) d\mu \\ &= \iiint_{\tilde{W}(t+1) \in \Omega_n} DF(\tilde{W}(t+1) \mid L(\tilde{W}(t)) = m) d\mu \\ &= \Pr[L(\tilde{W}(t+1)) = n \mid L(\tilde{W}(t)) = m], \forall n, m, m_{t-1}, m_{t-2}, \dots, m_1 \end{aligned}$$

Note that the second equality holds by the result in Theorem 3.1 (formula (16)).

The theorem is proved.

By the result of Theorem 4.1 the state (level) transition probability can be defined as

$$\Pr[L(t+1) = n \mid L(t) = m] \equiv p_{m,n}(t), \forall m, n, t. \quad (18)$$

Observe the last equality in the proof of Theorem 4.1. Since the conditional probability in the formula is independent of t , i.e. the resulting conditional probability for different in (18) are all the same. Hence, we have the following further result.

Theorem 4.2. The Markov chain defined in Theorem 4.1 is a stationary (time homogeneous) one, i.e. the formula in (18) is reduced to

$$\Pr[L(t+1) = n \mid L(t) = m] \equiv p_{m,n}, \forall m, n, t \quad (19)$$

In (19), the state (level) transition probability $p_{m,n}$ is independent of t . This justifies our argument that parameter selection should be a function of state (achievement) instead of a function of t (or iteration). The transition probability matrix can be denoted by

$$P \equiv [p_{m,n}] \equiv \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n} & \cdots \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ p_{m,0} & p_{m,1} & \cdots & p_{m,n} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}. \quad (20)$$

Note that $p_{m,n} = 0, \forall m > n$, i.e. P is an upper triangular matrix since the levels will at least stay the same comparing with previous state, if not increase. In (20) $p_{m,n}$ is one step transition probability and P is one step transition probability matrix. Besides the one step probability, l -step probability can also be established for general positive integer l . Define the l -step transition probability and l -step transition probability matrix as

$$\Pr[L(t+l) = n \mid L(t) = m] \equiv p_{m,n}^l, \forall l \geq 1, m, n \quad (21)$$

and

$$P^l \equiv [p_{m,n}^l] \equiv \begin{bmatrix} p_{0,0}^l & p_{0,1}^l & \cdots & p_{0,n}^l & \cdots \\ p_{1,0}^l & p_{1,1}^l & \cdots & p_{1,n}^l & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ p_{m,0}^l & p_{m,1}^l & \cdots & p_{m,n}^l & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}. \quad (22)$$

For the same reason as in (20), $p_{m,n}^l = 0, \forall m > n$, i.e. P^l is an upper triangular matrix. By the theory of Markov chain, P^l can be easily computed by the matrix product of matrix P , i.e. $P^l = \underbrace{PP \cdots P}_l$.

Take a closer look at Theorem 4.2. The PSO faces the common fate of a stationary Markov chain: the iteration number r (or equivalently, t) may be very large, while the present achievement is poor (for example, just like a new beginner), then the efforts done before will be almost in vain. To be more precise, the future fate of the PSO depends only on the present achievement (state) but not directly on the iteration number or t . The theoretical results convince us that: different parameters may render different transition probability in different levels of achievements rather than in different number of iterations. As a result, unlike most of the methods in other papers, we propose a heuristic idea. When dynamically choosing (tuning) parameters, the present achievement (level) will be taken into consideration instead of the number of iterations as traditional papers did. For example, we can compare the transition probability between different parameter set (transition probability matrix is parameter dependent). During different level of

achievements, choose the parameter set which is most favorable and help to promote the levels to higher ones.

The further application can be the investigation in the relationship between parameter set and transition probability for different optimizers. One reliable method is by using of computer simulation. For example, given a fixed fitness function, we can estimate the transition probability $p_{n,n+1}$ in different parameter set and choose the optimum parameter set in each level to maximize the value of $p_{n,n+1}(\omega, V_{\max}, c_k)$ (a function of ω, V_{\max}, c_k) or minimize the value of $p_{n,n}(\omega, V_{\max}, c_k)$. Here for practical purpose we assume that the probability that PSO jumps more than or equals to two levels at one time is very small with respect to jumps one level, i.e.

$$\frac{P_{n,n+l}}{P_{n,n+1}} \rightarrow 0, \forall l \geq 2.$$

The method to find the percentile of the function value of each fitness function can be done by simulation according to the result of Lemma 1. For the example given before, $\tilde{\alpha}=(10^0, 10^{-1}, 10^{-2}, \dots)$, the percentile of the fitness function value $f_n, n=0, 1, 2, \dots$, satisfying the relation that $\Pr[f(X) < f_n] = 10^{-n}, n=0, 1, 2, \dots$, can be estimated by use of the minimum of 10^n i.i.d. random function evaluations.

5. CONCLUSION

In this section we firstly propose some indexes to measure and compare the performance of different PSO algorithms, and then some steps of adaptive parameters selection are given based on the theoretical results in this work.

5.1. Index for comparison

Some fair measures to compare the different algorithms are given here. And they are divided into three groups. Among them Groups A and B are on effectiveness and Group C are on efficiency.

A1. Error_Hit Ratio: the portion of number of simulations the algorithm success in reducing the function value below a specified threshold (error) using fewer than the maximum allocated number of function evaluations, it measures the “success” probability in the searching process.

A2. α _Hit Ratio: the portion of number of simulations the algorithm successes in achieving the α _Dominating state using fewer than the maximum allocated number of function evaluations.

A3. Level_ n _Hit Ratio: the portion of number of simulations the algorithm successes in achieving the Level_ n state using fewer than the maximum allocated number of function evaluations.

B1. Best Value of Fitness Function: the smallest function value the algorithm can attain by using the same fixed number of function evaluations, it measure the extent of closeness to best value the algorithm can attain.

B2. Best α _Dominating: the smallest α value of the state the algorithm can attain by using the same fixed number of function evaluations.

B3. Best Level_ n _Dominating: the largest n of the state the algorithm can attain by using the same fixed number of function evaluations.

C1. Improving Ratio based on errors: to achieve the same accuracy (error), the ratio between the numbers of function evaluations needed by the non-cooperative i.i.d. random selection and that by PSOs, or between two PSOs, it measures the ratio of the times of function evaluations needed by two methods.

C2. Improving Ratio based on α value: to achieve the same α value, the ratio between the numbers of function evaluations needed by the non-cooperative i.i.d. random selection and that by PSOs, or between two PSOs.

C3. Improving Ratio based on level_ n : to achieve the same level, the ratio between the numbers of function evaluations needed by the non-cooperative i.i.d. random selection and that by PSOs, or between two PSOs.

5.2. Adaptive parameters selection

At the end of this work, we give some steps for practical adaptive parameters selection, which can be implemented by computer simulations.

1. Set the values $\tilde{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n)$, here for practical purpose we use finite dimensions.
2. Find by simulation the function values f_1, f_2, \dots, f_n such that $\Pr[f(X) \leq f_i] = \alpha_i, i = 1, 2, \dots, n$, the respective function values are the threshold of each level.
3. Find by simulation the best portfolio of parameters (such that $p_{m,m+1}$ is maximum) at different levels (dynamically parameters selection).
4. The best portfolio of parameters at each level found in step 3 constitute the best level-dependent parameter sets, $\{PS(m), m=1, 2, \dots, n\}$.

REFERENCES

- [1] P. Angeline, "Using selection to improve particle swarm optimization," in Proc. IJCNN'99, Washington, DC, July 1999, pp. 84–89.
- [2] F. V. D. Bergh and A. P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization," IEEE Trans. Evol. Comput., vol. 8, pp. 225–239, June 2004.
- [3] D. W. Boeringer and D. H. Werner, "Particle Swarm Optimization Versus Genetic Algorithms for Phased Array Synthesis", IEEE Trans .Ant. Prop., vol. 52, no. 3, Mar. 2004.
- [4] X. Cai , Z. Cui , J. Zeng , and Y. Tan, "Dispersed particle swarm optimization", Information Processing Letters, vol.105, no.6, pp.294-301, MAR. 16 2008.
- [5] C.W. Chou, H.H. Lin and J.H. Lin, "A note on Particle Swarm Optimization: An integrated and theoretical approach," in 4th IEEE International Conference on Computational Cybernetics, Tallinn, Estonia, AUG. 2006.
- [6] L. D. S. Coelho and B. M. Herrera, "Fuzzy Identification Based on a Chaotic Particle Swarm Optimization Approach Applied to a Nonlinear Yo-yo Motion System", IEEE Transactions on Industrial Electronics, vol.54, no.6, pp. 3234-3245, DEC. 2007.

- [7] S. Das, A. Abraham and A. Konar, "Automatic Clustering Using an Improved Differential Evolution Algorithm", *IEEE Transactions on Systems, Man and Cybernetics Part A-Systems and Humans*, vol.38, no.1, pp.218-237, JAN. 2008.
- [8] S. Das and A. Konar, "A swarm intelligence approach to the synthesis of two-dimensional IIR filters", *Engineering Applications of Artificial Intelligence*, vol.20, no.8, pp.1086-1096, DEC. 2007.
- [9] G. G. Dimopoulos and C. A. Frangopoulos, "Optimization of energy systems based on Evolutionary and Social metaphors", *Energy*, vol.33, no.2, pp.171-179, FEB. 2008.
- [10] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.
- [11] R. C. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools: Academic*, 1996, ch. 6, pp. 212–226.
- [12] M. El-Telbany and F. El-Karimi, "Short-term forecasting of Jordanian electricity demand using particle swarm optimization", *Electric Power System Research*, vol.78, no.3, pp.425-433, MAR. 2008.
- [13] H.W. Ge, L.Sun, Y.C. Liang, and F. Qian, "An Effective PSO and AIS-Based Hybrid Intelligent Algorithm for Job-Shop Scheduling", *IEEE Transactions on Systems, Man and Cybernetics Part A-Systems and Humans*, vol.38, no.2, pp.358-368, FEB. 2008.
- [14] F.A. Guerra and L.D.S. Coelho, "Multi-step ahead nonlinear identification of Lorenz's chaotic system using radial basis neural network with learning by clustering and particle swarm optimization", *Chaos Solitons & Fractals*, vol.35, no.5, pp.967-979, MAR. 2008.
- [15] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm", *Chaos Solitons & Fractals*, vol.37, no.3, pp.698-705, AUG. 2008.
- [16] Y. T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions", *Applied Soft Computing*, vol.8, no.2, pp.849-857, MAR. 2008.
- [17] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in *Proc. 2000 Congr. Evolutionary Computing*, 2000, pp. 1507–1512.
- [18] M. Khodier, N. Dib, and J. Ababneh, "Design of multi-band multi-section transmission line transformer using particle swarm optimization", *Electrical Engineering*, vol.90, no.4, pp.293-300, APR. 2008.
- [19] Z. J. Lee, "An integrated algorithm for gene selection and classification applied to microarray data of ovarian cancer", *Artificial Intelligence in Medicine*, vol.42, no.1, pp.81-93, JAN. 2008.
- [20] Y. Liu and X. Gu, "Skeleton-Network Reconfiguration Based on Topological Characteristics of Scale-Free Networks and Discrete Particle Swarm Optimization", *IEEE Transactions on Power Systems*, vol.22, no.3, pp.1267-1274, AUG. 2007.
- [21] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance", *Neural Networks*, vol.21, no.2-3, pp.427-436, MAR.-APR. 2008.
- [22] K. E. Parsopoulos and M. N. Vrahatis, "On the Computation of All Global Minimizers Through Particle Swarm Optimization," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 211–224, June 2004.
- [23] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evolutionary Computation*, Anchorage, AK, May 1998.
- [24] H. S. Yazdi and S. Effati, "Eigenvalue spread criteria in the particle swarm optimization algorithm for solving of constraint parametric problems", *Applied Mathematics and Computation*, vol.192, no.1, pp.40-50, SEP. 1 2007.

Authors

Chao-Wei Chou received his B. S. degree in Electrical Engineering from National Sun-Yat Sen University, Taiwan in 1988. He then received his M.S. degree in Business Administration from National Taiwan University, Taiwan in 1994. He received his Ph. D. in Statistics from National Sun-Yat Sen University, Taiwan in 2004. He has subsequently joined the Department of Information Management of I-Shou University as an assistant Professor. His research interests include artificial intelligence, computer network, information security and information economics.



Jiann-Horng Lin received his B. S. and M. S. both in Computer Science and Information Engineering from Feng-Chia University, Taiwan in 1987 and 1989, respectively. He then received his Ph.D. in Electrical Engineering and Computer Science from Syracuse University, New York in 1999. He is currently an assistant professor at the Department of Information Management at I-Shou University, Taiwan. He is also the department chair from 2004 to 2007. His research interests include artificial intelligence, data mining, chaos and information theory. He is also interested in the area of evolutionary computation and bioinformatics. Dr. Lin is a member of the IEEE, the Association for Computing Machinery and the Institute of Information and Computing Machinery.



Rong Jeng received his B. S. degree in Control Engineering from National Chiao-Tung University, Taiwan in 1981. He then received his M.S. degree in Electrical Engineering from Memphis State University, Memphis, TN in 1986. He received his Ph. D. in Electrical and Computer Engineering from University of Cincinnati, Cincinnati OH in 1991. He is currently an associate professor at the Department of Information Management at I-Shou University, Taiwan. He was also the department chair from 1991 to 1994. His research interests include Operations Research, Simulation, information theory.

