2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Autonomous Learning Adaptation for Particle Swarm Optimization

Wenyong Dong, Jiangsen Tian, Xu Tang, Kang Sheng and Jin Liu

***Abstract*—In order to improve the performance of PSO, this paper presents an Autonomous Learning Adaptation method for Particle Swarm Optimization (ALA-PSO) to automatically tune the control parameters of each particle. Although PSO is an ideal optimizer, one of its drawbacks focuses on its performance dependency on its parameters, which differ from one problem to another. In ALA-PSO, each particle is viewed as an intelligent agent and aims at improving itself performance, and can autonomously learn how to tune its parameters from its own experiment of successes and failures. For each particle, it means successful movement if the value of objective function in current position is improved than previous position, otherwise means failure. In case of successful movement, the parameters that are positive correlation with the direction of forward movement should be increased otherwise should be decreased. Meanwhile, in case of unsuccessful movement, inverse operation should be performed. The proposed parameter adaptive method is compared with several existing adaptive strategies, and the results show that ALA-PSO is not only effective, but also robust in different categories benchmarks.**

I. INTRODUCTION

Population-based optimization technique has become popular since 1990s, and Particle Swam Optimization (PSO) plays an important role in the family of Nature-inspired Computing (NC) [Kennedy and Eberhart, 1995]. Inspired by biological methods in birds flocking and fish schooling, the particles in PSO and its variants use a simple kinematic equation to collaboratively complete the optimization task. Each particle learns not just from personal experience, but also from each other, especially the leader. For most NC, at least two operators (or factors) should be considered to balance contradiction between exploration and exploitation, PSO is not an exception. It employs three key parameters, inertial weight, cognitive learning factor and social learning factor, to integrate the local search with global search together.

Wenyong Dong is with the Computer School, Wuhan University, Wuhan city, China (dwy@whu.edu.cn). He is presently with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA.

Jiangsen Tian is with the Computer School, Wuhan University, Wuhan city, China (e-mail:447204295@qq.com).

Kang Sheng is with the Computer School, Wuhan University, Wuhan city, China (e-mail:360148942@qq.com).

Xu Tang is with the School of Resource and Environmental Science, Wuhan University, Wuhan city, China (Corresponding author e-mail: 441541168 @qq.com).

Jin Liu is with the Computer School, Wuhan University, Wuhan city, China (e-mail: typon@gmail.com).

This work was in part supported by the NSF of China under Grant Nos. 61170305, 60873114, 61070013, U1135005, and NSF of USA under Grant CMMI-1162482.

Although PSO and its variants have been successfully applied to complex optimization problems, such as, continue functions optimization, combinatorial optimization, multi-objective optimization, and dynamic optimization, there are still rooms for improvement [Banks *et al*., 2007]. The improved methods can be divided into five categories: 1) Adaptive methods to tune PSO parameters; 2) Different population topology to maintain PSO diversity; 3) Multi-population or swarms to cooperatively capture multiple optima; 4) Mixture PSO with other search methods to speed up PSO convergence; and 5) New designed kinematic equation to balance the contradiction between exploitation and exploration [Hu *et al*., 2012; Kang *et al*, 2013; Liang *et al*., 2006; Reyes-Sierra and Coello, 2006].

In this paper, we aim to address issue of adaptive tuning for PSO parameters from a viewpoint of autonomous learning during the course of search process. We regard each particle as an agent and give them more thoughts, let them can think and make decision based on their performance metric. As discussed above, a new Autonomous Learning Adaptation method for Particle Swarm Optimization (ALA-PSO) is proposed. In ALA-PSO, each particle aims at improve itself performance, and can autonomously learn how to tune its parameters from its own experiment of successes and failures. For each particle, it means success movement if the value of objective function in current position is improved than previous position, otherwise means failure. In case of successful movement, the parameters that are positive correlation with the direction of forward movement should be increased otherwise should be decreased. Meanwhile, in case of unsuccessful movement, inverse operation should be performed.

The paper is organized as follows. Section II reviews some advance on the PSO parameter adaptation. Section III presents the details of the proposed ALA-PSO, including the tune rules for each particle. Section Ⅳ gives and analyzes experimental results. Section V gives the conclusion and future directions.

II. REVIEW OF THE PARAMETER ADAPTATION FOR PSO

There are two approaches to study how to choose the PSO parameters: 1) some guidelines are obtained from theories on stability and convergence [Ozcan and Mohan, 1999; Clerc and Kennedy, 2002; Kadirkamanathan *et al*., 2006; Jiang *et al*., 2007; Martinez and Gonzalo, 2008; Zhang *et al*., 2009]; and 2) some practical strategies for PSO parameters adaptation are drawn from deep experiment analysis [Jiao *et al*., 2008;

Chatterjee and Siarry, 2006; Shi and Eberhart, 2001; Ratnaweera *et al*., 2004; Zhan *et al*., 2009; Juang *et al*., 2011; Yamaguchi and Yasuda, 2006]. Because PSO and its variants are very abundant since dozen years, so we focus on the PSO with inertial weight, and in which there are three parameters need to set, which are inertial weight ω, cognitive learning factor c_1 and social learning factor c_2.

A. Theoretical Guidelines for Parameter Setting

In theory, the three PSO parameters take different roles in search process. ω is used to balance the global and local search ability, and it is similar to the temperature parameter in the simulated annealing [Eberhart and Shi, 1998]. A large ω facilitates a global search while a small one facilitates a local search. c_1 often has the role to increase diversity, while c_2 speeds up convergence of PSO. Meanwhile, c_1 collaborates with c_2 to constitute attractor of each particle. In the following, we focus on some theoretical results on the PSO with inertial weight, and its update rules with inertial weight for particle i at iteration k+1 is

$$\begin{cases} \mathbf{v}_i^{k+1} = \omega \mathbf{v}_i^k + c_1 \cdot r_1 \cdot (\mathbf{l}_i^k - \mathbf{x}_i^k) + c_2 \cdot r_2 \cdot (\mathbf{g}^k - \mathbf{x}_i^k) \\ \mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \end{cases} \tag{1}$$

where i stands for the ith paricle, and k stands for the kth iteration; $\mathbf{v}_i^k$ is the movement velocity; ω stands for inertial weight; c_1 is the cognitive learning factor; r_1 and r_2 are random numbers whose distribution is uniform over [0, 1]; $\mathbf{l}_i^k$ is the ith particle's best position found so far; $\mathbf{x}_i^k$ is the position of the ith particle at the kth iteration; c_2 is the social learning factor; and $\mathbf{g}^k$ is the best position found so far by the entire population.

There are two categories have been proposed to study the issue of PSO parameter selection: 1) deterministic models, and 2) stochastic models. The former can be found in [Ozcan and Mohan, 1999; Bergh, 2002; Clerc and Kennedy, 2002; and Yasuda et al., 2006]. The latter can be found in [Jiang et al., 2007; Martginez and Gonzalo, 2008]]. All those theoretical results provide insights into how particle swarm system works, and how select proper parameters for different optimization scenarios. Although the two models deal with PSO from different views, the similar theoretical results can be found.

The movement trajectories analysis by [Clerc and Kennedy, 1999], and show that when proper parameters are given, all the particles in swarm eventually approach their corresponding stationary points as

$$x_i^s = \frac{c_1 \cdot \mathbf{l}_i^s + c_2 \cdot \mathbf{g}^s}{c_1 + c_2} \tag{2}$$

where x_i^s is the stationary points of the ith particle; $\mathbf{l}_i^s$ is the ith particle best position found so far when PSO is trapped into stagnation; and $\mathbf{g}^s$ is the best position found so far when PSO stagnates.

From the view of deterministic dynamics, a sufficient stability condition on the parameter selection is established by [Kadirkamanathan *et al*, 2006]]

$$\left\{ (\omega, \phi) : |\omega| < 1,\ \omega \neq 0,\ \phi < \frac{1 - 2|\omega| + \omega^2}{1 + \omega} \right\} \tag{3}$$

where $\phi = {(c_1 + c_2)}/{2}$.

The other typical theoretical results are obtained by [Bergh, 2002] and [Jiang *et al*., 2007] as follows:

$$\{ (\omega, \phi) : 0 \le \omega < 1,\ \omega > \phi - 1 \} \tag{4}$$

$$\{ (\omega, \phi) : 0 \le \omega < 1,\ 0 < \phi / 2 < \omega + 1 \} \tag{5}$$

Those theoretical results only give instructive range or dependency relationship of parameter setting, as for concrete values of those parameters for different optimization problem, they are helpless.

B. Strategies for Adaptive PSO Parameter Selection

Because of the drawback existing in theoretical analysis mentioned above, other researchers try to propose some practical strategies for PSO parameter selection and adaptive tuning during the optimization process. The existing methods for adaptive tuning parameters can be classified into two categories: 1) entire-based parameter adaptation which means all the particles in swam adopt the same parameters, and 2) individual-based parameter adaptation which means each particle in swarm has its own parameters need to adaptive tuning.

At the earliest some researches emphasize on the adaptation of only one PSO parameter, i.e., ω. The adaptation strategy based on linear function is proposed by [Shi and Eberhart, 2001], in which only the ω is set to linearly decrease with the iteration count:

$$\omega(i) = \omega_{\max} - \frac{i \cdot (\omega_{\max} - \omega_{\min})}{I} \tag{6}$$

where $\omega(i)$ is the inertia weight at the ith iteration; $\omega_{\max}$ is the maximum inertial weight, and usually set to 0.9; $\omega_{\min}$ is the minimum inertial weight, and usually set to 0.4; and I is the maximum iteration count.

In order to improve the performance of adaptive method based on linear function, some nonlinear versions are proposed by [Jiao *et al*., 2008; Chatterjee and Siarry, 2006]] as follows:

$$\omega(i) = \omega_0 u^{-i} \tag{7}$$

where ω_0=0.9, and $u \in$ [1.001, 1.005].

$$\omega(i) = \omega_{\max} + \frac{(I - i)^n \cdot (\omega_{\max} - \omega_{\min})}{I^n} \tag{8}$$

where n is the nonlinear parameters ranging from 0.1 to 2.

The method based on fuzzy rules to adjust ω is proposed by [Shi and Eberhart, 2001]. All fuzzy rules are designed to tune on the basis of performance of the best particle.

In the individual-based parameter adaptation strategy, each particle has its own ω, c_1 and c_2, and tunes them independently. One typical strategy is proposed by [Yamaguchi and Yasuda, 2006] as follows:

$$\begin{cases} c_{1,i}^{k+1} = c_{1,i}^{k} + \alpha_i^k (c_{1,best}^k - c_{1,i}^k) \\ c_{2,i}^{k+1} = c_{2,i}^{k} + \alpha_i^k (c_{2,best}^k - c_{2,i}^k) \end{cases} \quad (9)$$

where $c_{1,i}^k$ is the cognitive learning factor of the ith particle at the kth generation; α_i^k is selected from 0 or 2/I depending on whether the best particle improve or not; $c_{1,best}^k$ is the cognitive learning factor of the best particle at the kth generation; $c_{2,i}^k$ is the social learning factor of the ith particle at the kth generation; and $c_{2,best}^k$ is the social learning factor of the best particle at the kth generation.

Another typical version belong to individual-based parameter adaptation strategy is proposed by [Hu *et al*., 2013]. In this method, depending on the relative position between each particle and the global best particle, the strategy to tune ω, c_1 and c_2 of each particle as follows:

$$\begin{cases} \omega_i^{k+1} = \omega_i^k + \sigma_i^k \times \left[2(\mathbf{g}^k - \mathbf{x}_i^{k+1})^T \times \mathbf{v}_i^k \right] \\ c_{1,i}^{k+1} = c_{1,i}^k + \sigma_i^k \cdot r_{1,i}^k \times \left[2(\mathbf{g}^k - \mathbf{x}_i^{k+1})^T \times (\mathbf{l}_i^k - \mathbf{x}_i^k) \right] \\ c_{2,i}^{k+1} = c_{2,i}^k + \sigma_i^k \cdot r_{2,i}^k \times \left[2(\mathbf{g}^k - \mathbf{x}_i^{k+1})^T \times (\mathbf{g}^k - \mathbf{x}_i^k) \right] \end{cases} \quad (10)$$

where ω_i^k is the inertia weight of the ith particle at the kth generation; σ_i^k is called step size, also known as Polyak's step size; $r_{1,i}^k$ and $r_{2,i}^k$ are random numbers whose distribution are uniform over [0, 1].

III. AUTONOMOUS LEARNING ADAPTATION FOR PSO

To address issue of PSO parameter adaptive tuning, this work proposes autonomous learning adaptation (ALA) strategies to enhance PSO performance by giving autonomous thinking ability for each particle to select its proper parameters. The term of "autonomous" means each particle can act on one's own, and make decide for itself, and it is different from "automatically". In the following, we firstly introduce our method of autonomous learning adaptation, and then, the procedure of ALA-PSO will be depicted.

A. Autonomous Learning Adaptation

To reduce the sensitivity of PSO to its parameter settings, many strategies are proposed to adaptively tune them. But none of them are based on individual-level autonomous decision. In this paper, except parameter adaptation for each particle, we do not change the basic kinematic equation of PSO.

The main ideas are depicted in Fig. 1. Each particle can be viewed as an intelligent agent, who is an autonomous entity which can observe through sensors to receive the global best position from environments and acts upon an environment using actuators (i.e. it can update the global best position if its optimization value of current position is better than the one of global best position) and directs its activity towards achieving goals (i.e. it is rational to update its local best position and parameters). Particle as an intelligent agent may also learn or use knowledge to achieve their goals. It uses its rules to update its parameters, as well as its local best position found so far.

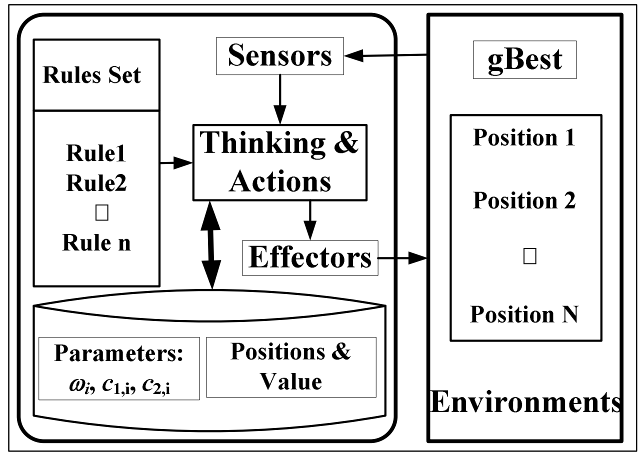


Fig. 1. Each particle is viewed as an intelligent agent, and it is composed of the following parts: Sensors, Effectors, Rules set, Knowledge base, and Thinking & Action.

A system is autonomous to the extent that its behavior is determined by its own experience and decision rules, and each particle should do whatever action (i.e., move its position) is expected to improve its performance measure (i.e., optimization value), on the basis of the evidence provided by the percept sequence (i.e., the global best position) and whatever built-in knowledge (i.e., its current parameters, position, and so on) the particle has. In order to achieve its target, we define the rules as

Rule 1 (kinematic equation for each particle):

if *Not Stop*, then

$$\begin{cases} \mathbf{v}_i^{k+1} = \omega_i \mathbf{v}_i^k + c_{1,i} \cdot r_1 \cdot (\mathbf{l}_i^k - \mathbf{x}_i^k) + c_{2,i} \cdot r_2 \cdot (\mathbf{g}^k - \mathbf{x}_i^k) \\ \mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \end{cases} \quad (11)$$

where $c_{1,i}$ and $c_{2,i}$ are cognitive learning factor and social learning factor of ith particle. This rule is the movement equation, and it is carried out if the stop criterion does not meet.

Rule 2 (parameter update methods if new position leads to performance improved):

if $f(\mathbf{x}_i^{k+1})$ is better than $f(\mathbf{x}_i^k)$, then

$$\begin{cases} \omega_i^{k+1} = H(\omega_{\min}, \omega_i^k e^{\sigma_0 \cos<\mathbf{v}_i^k, \mathbf{v}_i^{k+1}>}, \omega_{\max}) \\ c_{1,i}^{k+1} = H(c_{1,\min}, c_{1,i}^k e^{\sigma_1 \cos<(\mathbf{l}_i^k - \mathbf{x}_i^k), \mathbf{v}_i^{k+1}>}, c_{1,\max}) \\ c_{2,i}^{k+1} = H(c_{2,\min}, c_{2,i}^k e^{\sigma_2 \cos<(\mathbf{g}^k - \mathbf{x}_i^k), \mathbf{v}_i^{k+1}>}, c_{2,\max}) \end{cases} \quad (12)$$

where

$\omega_{\min}$ and $\omega_{\max}$ are the minimum and maximum value of inertial weight;

σ_0 is a coefficient for inertial weight;

$c_{1,\min}$ and $c_{1,\max}$ the minimum and maximum value of cognitive learning factor;

σ_1 is a coefficient for cognitive learning factor;

$c_{2,\min}$ and $c_{2,\max}$ the minimum and maximum value of global learning factor; and

σ_2 is a coefficient for global learning factor.

H is a projection operator on R and defined as follows:

$$H(x_{\min}, x, x_{\max}) = \begin{cases} x_{\min}, & x < x_{\min} \\ x, & x_{\max} \le x \le x_{\min} \\ x_{\max}, & x > x_{\max} \end{cases} \quad (13)$$

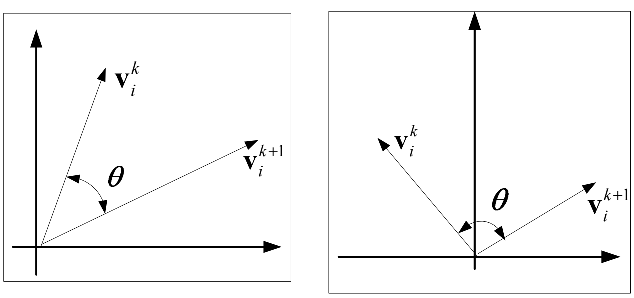
The rationale behind rule 2 is demonstrated in Fig. 2. In case the particle performance improves, ω will increase if the angle between velocity and its previous one is litter than $\pi/2$, otherwise, decrease. The same actions should be performed in the other coefficients, i.e., cognitive and global learning factor.

Rule 3 (parameter update methods if new position leads to performance degenerated):

if $f(\mathbf{x}_i^{k+1})$ is worse than $f(\mathbf{x}_i^k)$, then

$$\begin{cases} \omega_i^{k+1} = H(\omega_{\min}, \omega_i^k e^{-\sigma_0 \cos<\mathbf{v}_i^k, \mathbf{v}_i^{k+1}>}, \omega_{\max}) \\ c_{1,i}^{k+1} = H(c_{1,\min}, c_{1,i}^k e^{-\sigma_1 \cos<(\mathbf{l}_i^k - \mathbf{x}_i^k), \mathbf{v}_i^{k+1}>}, c_{1,\max}) \\ c_{2,i}^{k+1} = H(c_{2,\min}, c_{2,i}^k e^{-\sigma_2 \cos<(\mathbf{g}^k - \mathbf{x}_i^k), \mathbf{v}_i^{k+1}>}, c_{2,\max}) \end{cases} \quad (14)$$

The rationale behind rule 3 is similar with rule 2, that is, if particle performance degenerates, the parameter should decrease if its direction of the corresponding vector is same with the ongoing direction, otherwise, increase.



(a) (b)

Fig. 2. (a) In case velocity and its previous one is litter than $\pi/2$, (b) In case velocity and its previous one is large than $\pi/2$

Based on rules 2 and 3, each particle adaptively tunes its parameters according to its search ability. Experiments have shown the effectiveness of the proposed method.

B. Procedure of ALA-PSO

As discussed above, each particle in swarm is regarded as intelligent agent. The implement modes can be divided into two categories: synchronous and synchronous methods. Because we focus on our parameter adaptive strategies, so we only study the synchronous mode, i.e., in each horizon (iteration), all particles will update their states, and then, the next horizon begins. Thus, we can organize the procedure of ALA-PSO as Algorithm 1.

Algorithm 1: Autonomous Learning Adaptation method for Particle Swarm Optimization (ALA-PSO)

Step 1) (**Setting the algorithm parameters**) set the number of particles, and their parameters ranges.

Step 2) (**Initialization**) randomly generated the swarm, and restrict their parameters in allowable ranges. And then, update the global best position and local best positions;

Step 3) (**Movement**) For all particles, update their positions according rule 1, and update their parameters according rule 2 or 3, at the same time update global best position and local best positions;

Step 4) (**Stop Condition**) if the stop condition is not met, go to step 3), otherwise, output the final results.

An algorithm should be stopped whenever it starts to waste CPU-time. Hence, the following termination criteria are used, which are mostly related to numerical stability.

- C1): the total function evaluation count reaches a given number, or
- C2): the optimization value of global best position meets given precision requirement, or.
- C3): in successive d generations, the global best position does not change any more, where d is a given positive integer.

IV. EXPERIMENTS AND RESULT ANALYSIS

The supervised learning and control is the core of APSO-SLC. Because PSO is a stochastic algorithm, it has very large contingence only running one generation. When measuring PSO performance, we should let PSO run enough iteration to show its search ability. If d is the dimension of the search space, we run PSO for d times.

A. Experiment Configuration

In order to study the performance of ALA-PSO, we choose four function from 31 test functions from [Tang *et al.* 2008] as the benchmark functions. The four functions are as follows:

F1: $f_1(\mathbf{z}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2$;

F2: $f_2(\mathbf{z}) = \sum_{i=1}^{D-1} \left(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \right)$

F3: $f_3(\mathbf{z}) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10 \right)$

F4: $f_4(\mathbf{z}) = \sum_{i=1}^{D} z_i^2 / 4000 - \prod_{i=1}^{D} \cos(z_i / \sqrt{i}) + 1$

We adopt the shifted and rotated version of the four benchmarks as [Tang *et al.* 2008; Su *et al.* 2013].

The performance metrics to measure the solution quality and convergence speed is chosen as Function Value (FV), Expected Function Value (EFV), Expected Success Rate (ESR), Running Time (RT), and Expected Running Time (ERT). FV is the objective value when the algorithm stops in one run. EFV is the average objective value of all runs. ESR is the average of the percentage of all runs in which the global optima are located within a given target accuracy. RT is the number of function evaluations executed in one run, if the best function value has not reached the global optima within a given target accuracy, we set it to Maximum of Function Evaluation (MFE). ERT is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value has not reached the global optima within a given target accuracy, summed over all trials divided by the number of trials that have actually reached the global optima within a given target accuracy.

In order to compare the performance of different algorithms, two categories of experiments are conducted. The first one is to compare different adaptive strategies with our proposed method, and the second one is to compare the performance of different PSO algorithms for global optimization.

B. Validity Test on Parameter Adaptation Strategies

There are four kinds of adaptive strategies: 1) linear control in (6); 2) nonlinear control (8); 3) individual-level adaptive strategy shown in (10); and 4) the ALA. We denote the first three methods as AS1-AS3. Two statistic test methods to test the validity of our proposed algorithm are employed: one side t-test and Wilcoxon rank sum test. If the return result of a test function is 1, we should accept the alternative hypothesis, and otherwise we should exchange the compared sequence to perform the second test. If both results are zeros, we conclude that their means are equal. The numerical values 1, 0 and -1 represent that the first algorithm is statistically inferior, equal and superior to the second one. The stop criteria are set as:1) The accuracy $\varepsilon=10^{-5}$, and 2) MFE=3×10^{5}. Each algorithm runs 30 times on each function. In Tables I and II only the number of pair-wise comparison results is recorded.

Tables I and II list the results of *t*-test and Wilcoxon test on FV and RT for four functions. For all the statistic tests, we set the level of significance is 0.05. In both tables, the symbols s "+," "=" and "-" indicate that ALA-PSO performs significantly better than, almost the same as, and significantly worse than the compared method, respectively. From Table I, by t-test, we can conclude that the parameter adaptation strategy of ALA-PSO is better than the other four adaptation strategies denoted by AS1-AS3. In total four functions according to t-test, by the performance matrix FV, the number of times that ALA-PSO is better than AS1, AS2 and AS3 is 4, 10, 3 and 3, respectively; meanwhile, for only 1 times, it is worse than AS3. By the performance matrix RT, the number of times that ALA-PSO is better than AS1, AS2 and AS3 is same as the one by FV.The similar statistical conclusion can be obtained from Wilcoxon tests shown in Table II. From these results, we conclude that ALA-PSO significantly outperforms the other four parameter adaptation methods both in terms of solution quality (FV) and function evaluations executed (RT) via both t-test and Wilcoxon test.

C. Comparison with other PSO-w on 30-D functions

The second experiment is to compare ALA-PSO with PSO-w on functions with 30 dimensions. Both algorithms run 30 independent in all test functions, the population size is set to 30, MFE is set to 300000 and $\varepsilon=10^{-5}$. The results are shown on four rotated functions in Table III. In all experiments, if there are no successful runs among 30 runs (SR = 0), we set ERT to be blank implying that ERT=MFE.

From Table VIII, we can see that for all the four functions ALA-PSO outperforms PSO-w in terms of solution quality (EFV) and convergence rate (ERT). But both algorithms has lower success rate in three out of four functions. For F1, the proposed algorithm has 63.3%, while PSO-w only 23.3%.

As a conclusion, the adaptive strategy of ALA-PSO is effective and makes it insensitive to the initial parameter settings.

V. CONCLUSION AND FUTURE WORK

Aiming to solve existing problems in a standard PSO, **a new strategy based on** Autonomous Learning Adaptation for PSO**, called** ALA-PSO, **is** proposed. The main ideas behind it are: firstly, it regards each particle as a intelligent agent, who has its own control parameters and knowledge base; secondly, each particle aim at improve itself performance, and can autonomously learn how to tune its parameters from its own experiment of successes and failures. Two kinds of experiments are conducted to test its properties: (1) the effectiveness of the strategies to adapt parameters; and (2) the ability to find the global optima. The results show that the performance of ALA-PSO has improved compared with the contrast algorithms, and the adaptive strategy is effective.

Future research will pursue the following directions:

(1) We views swam as a collection of intelligent agents in this paper, but we do not test the synchronous mode of agent. Thus, further study should focus on the synchronous implementation of our method.

(2) Another future work should introduce different types of particles by giving them different rules to cooperate to searching in the domain.

REFERENCES

[1] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization: Part I. Background and development," Natural Comput., vol. 6, pp. 467–484, Dec. 2007.

[2] F. van den Bergh (2002), "An analysis of particle swarm optimizers," *Ph.D. dissertation*, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa.

[3] M. Clerc and J. Kennedy (2002), "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," IEEE *Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73.

[4] A. Chatterjee and P. Siarry (2006), "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Comput. Oper. Res.*, vol. 33, pp. 859–871.

[5] M. Hu, T. Wu, and J. D. Weir (2012), "An intelligent augmentation of particle swarm optimization with multiple adaptive methods," *Inform. Sci.*, vol. 213, pp. 68–83.

[6] Mengqi Hu; Wu, T.; Weir, J.D. (2013), "An Adaptive Particle Swarm Optimization With Multiple Adaptive Methods," *Evolutionary Computation, IEEE Transactions on* , vol.17, no.5, pp.705-720.

[7] Y.-T. Juang, S.-L. Tung, and H.-C. Chiu (2011), "Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions," *Inform. Sci.*, vol. 181, pp. 4539–4549.

[8] M. Jiang, Y. P. Luo, and S. Y. Yang (2007), "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Inform. Process. Lett.*, vol. 102, pp. 8–16.

[9] B. Jiao, Z. Lian, and X. Gu (2008), "A dynamic inertia weight particle swarm optimization algorithm," *Chaos, Solitons Fractals*, vol. 37, pp. 698–705.
[10] V. Kadirkamanathan, K. Selvarajah, and P. J. Fleming (2006), "Stability analysis of the particle dynamics in particle swarm optimizer," *IEEE Trans. Evol. Comput*., vol. 10, no. 3, pp. 245–255.
[11] J. Kennedy and R. Eberhart (1995), "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw*., vol. 4. Nov.–Dec. pp. 1942–1948.
[12] J. Kennedy and R. Mendes (2002), "Population structure and particle swarm performance," in *Proc. Congr. Evol. Comput.*, pp. 1671–1676.
[13] J. Kennedy (1999), "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. Congr. Evol. Comput*., pp. 1931–1938.
[14] J. J. Liang and P. N. Suganthan (2005), "Dynamic multi-swarm particle swarm optimizer," in *Proc. IEEE Swarm Intell. Symp*.,pp. 124–129.
[15] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar (2006), "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput*., vol. 10, no. 3, pp. 281–295.
[16] X. Liang, W. Li, Y. Zhang, Y. Zhong, M. Zhou (2013). "Recent Advances in Particle Swarm Optimization via Population Structuring and Individual Behavior Control," in Proc. 2013 IEEE Int. Conf. on Networking, Sensing and Control, pp. 503-508, Paris, France.
[17] Q. Kang, M. C. Zhou, J. An, and Q. Wu (2013), "Swarm Intelligence Approaches to Optimal Power Flow Problem With Distributed Generator Failures in Power Networks," IEEE Transactions on Automation Science and Engineering, 10(2), pp. 343-353.
[18] V. Kadirkamanathan, K. Selvarajah, and P. J. Fleming (2006), "Stability analysis of the particle dynamics in particle swarm optimizer," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 245–255.
[19] I. V. Morozova, M. Y. Postan, & S. Dashkovskiy, (2013). "Dynamic Optimization Model for Planning of Integrated Logistical System Functioning". In *Dynamics in Logistics*, pp. 291-300. Springer Berlin Heidelberg.
[20] E. Ozcan and C. K. Mohan (1998), "Analysis of a simple particle swarm optimization system," in *Proc. Intelligent Engineering Systems through Artificial Neural Networks*, 1998, pp. 253–258.
[21] M. Reyes-Sierra and C. A. Coello Coello (2006), "Multiobjective particle swarm optimizers: A survey of the state-of-the-art," *Int. J. Comput. Intell. Res*., vol. 2, no. 3, pp. 287–308.
[22] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson (2004), "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.,* vol. 8, no. 3, pp. 240–255.
[23] Y. Shi and R. C. Eberhart (2001), "Fuzzy adaptive particle swarm optimization," in *Proc. Congr. Evol. Comput*., pp. 101–106.
[24] W. Zhang, H. Li, Q. Zhao, and H. Wang (2009), "Guidelines for parameter selection in particle swarm optimization according to control theory," in *Proc. 5th Int. Conf. Natural Comput*., pp. 520–524.
[25] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung (2009), "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern*., vol. 39, no. 6, pp. 1362–1381.
[26] H. Zhang, X. Hao, A. An, X. Liu, & L. Chen, (2013). "Stair-like generalized predictive control for improving the output power of proton exchange membrane fuel cell system". In *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pp. 67-680.

TABLE I
ONE SIDE T-TEST COMPARISION BETWEEN ALA-PSO AND OTHER PARAMETERS ADAPTIVE STATEGIES

SPO-PSO v.s.	FV			RT		
	+(Better)	=(Same)	-(Worse)	+(Better)	=(Same)	-(Worse)
AS1	4	0	0	4	0	0
AS2	3	1	0	3	1	0
AS3	3	0	1	3	1	0

TABLE II
ONE SIDEWILCOXON TEST COMPARISION BETWEEN ALA-PSO AND OTHER PARAMETERS ADAPTIVE STATEGIES

SPO-PSO v.s.	FV			RT		
	+(Better)	=(Same)	-(Worse)	+(Better)	=(Same)	-(Worse)
AS1	4	0	0	4	0	0
AS2	2	2	0	3	1	0
AS3	2	1	1	2	1	1

TABLE III
OPTIMIZATION RESULTS FOR FUNCTIONS WITH 30-D

Algorithms	**EFV**	**ERT**	**ESR(%)**	**EFV**	**ERT**	**ESR(%)**
		F1			**F2**	
PSO-w	2.99E-02		0	8.13E+02		0
ALA-PSO	**<u>1.36E-04</u>**		0	**<u>2.86E+01</u>**		0
		F3			**F4**	
PSO-w	3.24E+01		0	1.65E-02	1165460	23.3
ALA-PSO	**<u>2.79E+01</u>**		0	**<u>0.00E+00</u>**	292830	63.3