Adaptive Parameter Selection Scheme for PSO: A Learning Automata Approach

Ali B. Hashemi and M.R. Meybodi

Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
{a_hashemi, mmeybodi}@aut.ac.ir

Abstract

PSO, like many stochastic search methods, is very sensitive to efficient parameter setting. As modifying a single parameter may result in a large effect. In this paper, we propose a new a new learning automata-based approach for adaptive PSO parameter selection. In this approach three learning automata are utilized to determine values of each parameter for updating particles velocity namely inertia weight, cognitive and social components. Experimental results show that the proposed algorithms compared to other schemes such as SPSO, PSO-IW, PSO TVAC, PSO-LP, DAPSO, GPSO, and DCPSO have the same or even higher ability to find better local minima. In addition, proposed algorithms converge to stopping criteria significantly faster than most of the PSO algorithms.

1. Introduction

The particle swarm optimization (PSO) algorithm is introduced by Kennedy and Eberhart [1] based on the social behavior metaphor. In particle swarm optimization, a group of particles, without quality and volume, fly through a D dimensional space, adjusting their positions in search space according to their own experience and their neighbors. The ith particle is represented as $P_i=(p_{i1}, p_{i2},..., p_{Di})$ in the D-dimensional space. The velocity for particle i is represented as $V_i=(v_{i1}, v_{i2},..., v_{Di})$, which is usually clamped to a maximum velocity V_{max}, specified by the user. In each time step t, the particles are manipulated according to the following equations:

$$v_i(t+1) = v_i(t)+c_1r_1(pbest_i - p_i(t))+c_2r_2(gbest - p_i(t)) \quad (1)$$

$$p_i(t+1) = p_i(t)+v_i(t+1) \quad (2)$$

Where c_1 and c_2 are positive acceleration constants used to scale the contribution of cognitive and social components respectively. r_1 and r_2 are uniform random variables in range [0,1]. $pbest_i$ is the best personal position of particle i which has been visited during the lifetime of the particle. *gbest* is the global best position that is the best position of all particles in the swarm.

Stochastic search techniques are usually problem dependent. Thus, an efficient parameter setting forms an important part of the algorithm. PSO is no exception, modifying a single parameter may result in a large effect[2] [3]. Many researchers tried to improve the performance by tuning the parameters of basic PSO [4], [5], and [6], or by adding new parameters such as mutation[7]. However the combination of PSO with other evolutionary algorithms has introduced new parameters and increased the computational effort[8], [9]. We follow the opposite direction of research with the idea to add a mechanism for particle swarm to adapt its velocity step size to an appropriate value.

Learning automata are adaptive decision-making devices that operating in an unknown random environment and progressively improve their performance via a learning process. It has been used successfully in many applications such as call admission control in cellular networks [10] and [11], capacity assignment problems[12], Adaptation of back propagation parameter[13], and Determination of the number of hidden units for three layers neural networks [14]. In this paper, we propose a new approach for adaptive adjusting parameters of PSO, inertia weight, cognitive and social parameters of PSO (w, c_1, and c_2 respectively). In the proposed approach, parameters of a PSO are set by means of three learning automata, one learning automata for each parameter. To adjust value of a parameter we take two approaches. The first approach is somehow adventurous, where learning automata select value of a parameter among a finite set. While In the second

approach learning automata conservatively decide to increase, decrease or not to change previous value of a parameter.
The rest of the paper is organized as follows. Section 2 reviews the parameter selection methods in literature. Section 3 gives a brief introduction to learning automata. The proposed algorithms are given in section 4. Experiments settings and results are presented in section 5. Section 6 concludes the paper.

2. Related Work

PSO Parameter selection schemes introduced in literature can be classifies in three categories. In the first category, all parameters of PSO are selected empirically. For example assigning different values to c_1 and c_2 sometimes leads to faster convergence and improved performance [15]. Typically, these are set to a value of 2.0 [1], but experimental results indicate that alternative configurations, depending on the problem, may produce superior performance. An extended study of the acceleration coefficients in the first version of PSO, is given in [16]. Although this approach may lead the suboptimal results of a single problem with constant parameters, it would not result the optimal parameters because of different optimal values for various stages of swarm, i.e. more exploration rather than exploitation at the beginning. To address this problem, algorithms in the second category try to choose time variant parameter values.

Generally, in population-based optimization methods, considerably high diversity is necessary during the early stages of the search to allow use of the full range of the search space. On the other hand, during the latter stages of the search, when the algorithm is converging to the optimal solution, fine-tuning of the solution is important in order to find the global optima efficiently. Considering these concerns, Shi and Eberhart [4], [3], and [17] have found a significant improvement in the performance of the PSO method with a linearly varying inertia weight over the generations. They experimentally show that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions [4] [3]. Moreover Ratnaweera et al. [7] proposed a PSO with time-varying acceleration coefficients (PSO-TAVC), in which c1 decreases linearly over time while c2 increases linearly. In their method in case of no improvement, the velocity is mutated by a mutation probability to the maximum allowed velocity multiplies by the mutation step size. Chatterjee and Siarry [5] proposed DAPSO, a PSO model with dynamic adaptation that concentrates on the adjustability of the decreasing rate for the inertia weight by using a set of exponential-like curves to create a complete set of free parameters for any given problem.

Although it helps the swarm to have different behavior during its lifetime, it could vary in different applications. Therefore the third category algorithms change PSO parameters over time by looking on the swarm state. Tawdross and König [18] has proposed a new approach in PSO in which , like real world, each individual has its own character, which means each particle has different values for w, c_1, and c_2. These parameters are updated after each iteration by a controller in order to improve itself for each particle individually. If they are not improving anymore, start to fly more toward their own fitness and mutate its velocity with a mutation probability. Pasupuleti and Battiti [19] introduced GPSO, in which the particle's previous velocity is not taken into account. In G-PSO, the population is attracted by the global best position and each particle is re-initialized with a random velocity if it is stuck close to the global best position. In this manner, the algorithm proceeds by greedily scouting the local minima whereas basic PSO proceeds by trying to avoid them. In each iteration, a particle either will take a step along the direction towards the global best position or re-initialized if it gets very close to the global best position. Yanga et al. [20], proposed a PSO algorithm with dynamic adaptation (DAPSO). In their method, velocity update formula of the particle is modified so that the randomness in updating particle velocity is relatively decreased over time. Moreover, DAPSO introduced two parameters describing the state of the algorithm, the evolution speed factor and aggregation degree factor which the inertia weight of each particle is dynamically adjusted according to the evolution speed and aggregation degree.

3. Learning Automata

The automata approach to learning can be considered as the determining of an optimal action from a set of actions. An automaton can be regarded as an abstract object that has finite number of actions. It selects an action from its finite set of actions. This action is applied to an environment. The environment evaluates the applied action and sends a reinforcement signal to automata (i.e. favorable or unfavorable in a P-model environment). The response from environment is used by learning algorithm of automata to update its internal state. By continuing this process, the automaton learns to select an action, which leads to a favorable response.

Variable structure learning automata are represented by the sextuple $<\beta, \Phi, \alpha, P, G, T>$, where β is a set of input actions, $\Phi$is a set of internal states, α is a set of outputs, P denotes state probability vector governing the choice of the state at each stage k; G is the output mapping, and T is learning algorithm. The learning algorithm is a recurrence relation and is used to modify the state probability vector.

It is evident that the crucial factor affecting the performance of the variable structure learning automata is the learning algorithm for updating the action probabilities. The linear reward-penalty algorithm (L_{R-P}) is one of the various learning algorithms reported in the literature [21]. Let α_i be the action chosen at time k as a sample realization from distribution $p(k)$. In a P-model environment where $\beta \in \{0,1\}$, an L_{R-P} scheme for updating P is defined as eq. (3) when $\beta=0$ (favorable input) and eq. (4) when $\beta=1$ (unfavorable input).

$$p_j(k+1) = \begin{cases} p_j(k) + a\left(1 - p_j(k)\right) & if \ \ i = j \\ p_j(k)(1-a) & if \ \ i \neq j \end{cases} \tag{3}$$

$$p_j(k+1) = \begin{cases} p_j(k)(1-b) & if \ \ i = j \\ \dfrac{b}{r-1} + (1-b)p_j(k) & if \ \ i \neq j \end{cases} \tag{4}$$

Where a and b represent reward and penalty step length respectively.

4. Proposed Algorithm

In the proposed algorithms, we use three learning automata to set three parameters of PSO velocity update equation, namely w, c_1, and c_2. To adjust a parameter value we utilized both adventurous and conservative approach. In the first approach, a learning automaton adjusts value of each parameter. In the second approach, learning automata of each parameter decides how to change value of the parameter, whether increase/decrease it or does not change it.

PSO-APLA$_1$ uses three learning automata LA_w, LA_{c1}, and LA_{c2} for adjusting value of swarm parameters w, c_1, and c_2 respectively. Each automaton has N_p actions, where each action corresponds to a value in range $[P_{min}, P_{max}]$ where P_{min}, and P_{max} are boundaries of parameter P. For each of the parameters w, c_1, and c_2 their corresponding learning automata select an action at each iteration. Afterwards, the parameters are set to the value selected by learning automata. The pseudocode of this algorithm is presented in Figure 1.

```
Procedure PSO-APLA$_1$
begin
  Initialize a D-dimensional PSO
  repeat
    for each parameters $\xi = \{w, c_1, \text{and } c_2\}$ do
      Select an action of LA$_\xi$ as $\alpha_\xi$, where $\alpha_\xi = \{0,1,\ldots N_\xi\}$
        $\xi(t) = \xi_{\min} + \alpha.\left(\frac{\xi_{\max} - \xi_{\min}}{N_\xi}\right)$
    end
    for each particle $p_i$, $i \in [1 \ldots N]$
      Update particle's velocity with eq. (1)
      Update particle's positions with eq. (2)
      Update $pbest_i$ and $gbest$
      if $pbest_i$ was updated then
        $n_{imp} = n_{imp} + 1$
      end
    end
    Calculate environment response $\beta = \begin{cases} 0 & if \ \frac{n_{imp}}{N} > \tau \\ 1 & otherwise \end{cases}$
    Update Learning Automata with $\beta$
  until a terminate condition is met
end
```

Figure 1. Pseudocode of PSO-APLA$_1$

Since in many PSO algorithms c_1 and c_2 are set equally, we also consider another version of the above algorithm with one learning automata to set one value for both c_1 and c_2 and consider them equal in all iterations ($\text{PSO} - \text{APLA}_1^2$).

In PSO-APLA$_2$, parameter adaptation is done in conservative approach. In this model decisions are changing value of a parameter, which can be increasing, decreasing, or not changing it at all. At each iteration, three learning automata decide how to change current value of the three parameters of swarm w, c_1, and c_2. Then all particles update their new velocities and positions according to new values for the parameters of the swarm. When all particles update their position, learning automata receive a response from environment indicating the selected action was a good choice or not. The pseudo code of PSO-APLA$_2$ is like PSO-APLA$_1$ but actions of leaning automata and parameter adaptation is different as shown in eq.5.

$$\alpha_\xi = \{increase, decrease, no-change\}$$

$$\xi(t+1) = \begin{cases} \xi(t) + \theta_\xi & if \ \ \alpha_\xi = increase \\ \xi(t) & if \ \ \alpha_\xi = no\text{-}chane \\ \xi(t) - \theta_\xi & if \ \ \alpha_\xi = decrease \end{cases} \tag{5}$$

5. Experimental Results

Proposed PSO algorithms were tested on a suite of benchmark functions used in literature and compare with well known PSO algorithms: SPSO [22], PSOIW[4], GPSO[19], PSOLP[18], DCPSO[23], DAPSO[20], and PSO-TVAC[7].

5.1 Benchmarks functions

The performance of the proposed PSO model is tested for a number of benchmark functions which have been extensively used to compare both PSO-type and non-PSO type metaheuristic algorithms. This paper utilizes the benchmark function set, shown in Table 1.

Table 1 Parameters for benchmark functions

Function	D	Range	Initial Range	Goal
Rosenbrock : $f_1 = \sum_{i=1}^{n-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right)$	30	$[-30,30]^n$	$[15,30]^n$	100
Sphere: $f_2 = \sum_{i=1}^{n} x_i^2$	30	$[-5.12,5.12]^n$	$[50,100]^n$	0.01
Rastrigin: $f_3 = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos\left(2\pi x_i\right)\right)$	30	$[-5.12,5.12]^n$	$[2.56,5.12]^n$	100
Ackley: $f_4 = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)$	30	$[-32,32]^n$	$[16,32]^n$	0.1
Griewank : $f_5 = 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	30	$[-600,600]^n$	$[300,600]^n$	0.1

Functions in real world can be unimodal or multimodal, with correlated or uncorrelated variables. De Jong's Sphere function is the most basic problem. Having no local optima, it provides a smooth gradient towards its global optimum. Rastrigin, Griewank, and Ackley functions are highly complex multimodal problems with many local minima. The Rastrigin function has many local optima around the global optima, and no correlation among variables. The Ackley function is also multimodal at low resolution. Griewank function is the only function, which introduces correlation between its variables.

Table 1 shows the values that have been used for the dimension of these functions, feasible bounds, the range of the corresponding initial position of the particles, and the goals of each function that have to be achieved by the algorithms.

5.2 PSO Algorithms Configuration

As suggested in the literature [24][25-27] particles are initialized in regions that expressly do not include the optima to verify results obtained for symmetric initialization schemes. Also, The particles have been initialized with a random velocity where the values in every dimension have been randomly chosen according to a uniform distribution over the initial range [$-x_{max}$,x_{max}]. During a run of an algorithm, the position and velocity of a particle have not been restricted to the defined intervals, but whenever a particle passes the defined domain, fitness evaluation step is then skipped, thereby preventing the infeasible position from being set as a personal and/or global best [27].

Eberhart and Shi [22] showed that population size has almost no significant effect on the performance of the PSO. However, van den Bergh and Engelbrecht [28] suggested that even though there is a slight improvement of the optimal value with increasing swarm size, it increases the number of function evaluations to converge to an error limit. However, in PSO literature it is quite common to limit the number of particles to the range 20 to 60. Therefore, in all experiments the swarm size *s* is set to 40 particles.

Proposed algorithms are compared with the following PSO algorithms:

- SPSO: Standard PSO with w=0.72 and $c_1=c_2=\{1.49,2.0,2.05\}$ as suggested in [22].
- PSOIW: PSO with a linearly varying inertia $_{weight}$ PSO (PSOIW) [4]. The inertia weight w is varied from w_1=0.9 at the beginning of the search to w_2=0.4 at the end [3].
- DAPSO: Dynamic adaptation PSO [20]. α=0.8, $\beta=0.4$, and $w_{ini}=1.0$ are set as suggested in [20].
- GPSO: Gregarious particle swarm optimizer. γ_{init}=3.0, $\gamma\in[2,4]$, δ=0.5, and $\varepsilon=10^{-8}$ are set as suggested in [19].
- PSO-LP: PSO with Local parameter [18]. α_1=1.05,α_2=0.975,β_1=1.07, β_1=0.833, and *MutationProbability*=0.007 as suggested in [18].
- PSO-TVAC: PSO with Time-Varying Acceleration Coefficients [7]. The best results were achieved when c_1 starts with 2.5 and decreases to 0.5 and c_2 starts by 0.5 increases linearly to reach 2.5 in the last iteration. In addition, w_1 is set to 0.9 at the beginning of the search and decreases to 0.4 at the end.

Since all possible permutation of parameters of the proposed algorithms creates an enormous list, only one configuration of each algorithm, as its representative, is presented. The representatives are selected from the best configurations of each algorithm. To determine the best configuration of each algorithm, we used

Holm-Bonferroni multiple comparison test as suggested in [27] on following range of parameters: w_{max}={1.0,2.0}, θ_w={0.1,0.2,0.5} or N_w={5,10}, c_{max}={2.0,3.0,4.0}, θ_c={0.1,0.2,0.5,1.0,1.5,2.0} or N_c={5,10}, τ={0.1,0.2,…,0.9}

In all proposed algorithm, learning automata uses $L_{R\text{-}P}$ learning scheme with *a=b=0.01*.

A configuration that is superior to others is selected as the representative of its algorithm. Following configuration is selected as the representatives of the proposed algorithms.

- **PSO-APLA$^1_1$**:w_{max}=2.0,N_w=10,$c_{1,2\max}$=3.0,N_c=10,τ=0.5
- **PSO-APLA$^2_1$**:w_{max}=1.0,N_w=5, $c_{1,2\max}$=4.0,N_c=10,τ=0.5
- **PSO-APLA$_2$**:w_{max}=2.0,θ_w=0.2,$c_{1,2\max}$=3.0,$\theta_{c1,c2}$=1.5,τ=0.7

5.3 Experiments Settings

For all PSO algorithms and for all 10 and 30 dimensions functions, two experiments were carried out 300 times for 5000 iterations (2×10^5 function evaluations), or until the best particle's fitness reaches the goal specified in Table 1, depending on the type of experiment being performed. However, results for 10,000 iterations (4×10^5 function evaluations) for 30-D functions were included to show behavior of proposed algorithms when more computational resources are available.

For each of 10 and 30 dimension space experimental results are presented in two tables, which are described as follows.

In the first tables, mean error values and 95% confidence interval for 300 runs for all PSO algorithms on the test functions were presented. In order to determine whether the obtained results are statistically different, Holm-Bonferroni multiple comparison test [27, 29] are conducted for each function.

In the second tables, for each function success rate of each algorithm and average number of fitness evaluations required to reach the specified goal with a 95% confidence interval for the successful runs are presented. If the algorithm could achieve the specified goal for a function (Table 1) within the maximum number of iterations (5000 iteration or 2×10^5 function evaluation), the run was considered successful.

In addition, in order to have an overall comparison of tested PSO algorithms, for each experiments, we present cumulative results of pair comparison of all PSO algorithms over all functions using Holm-Bonferroni multiple comparison tests [27] in the last table. These tests produce some intermediate results like paired comparisons over each function, which are omitted here due limited space. However, for each PSO algorithm (PSO_i) number of PSO algorithms which PSO_i is significantly superior to and number of PSO algorithms which PSO_i is significantly inferior to are reported.

5.5 Description of results

10 Dimensions:

In 10-Dimenstion space, PSO-TAVC was the only algorithm could not defeated by any other tested PSO algorithms followed by PSO-APLA$^1_1$ which were only worse than PSO-TAVC considering all functions (Table 2). Moreover, PSO-APLA$_2$ followed by PSO-APLA$^1_1$ resulted the least error for Rosenbrock function.

In addition, all proposed algorithms were tataly successful in reaching the specified goal for all function except Griewank (Table 3). However, PSO-APLA$^1_1$ was the second successful algorithm after GPSO to find the global optima in Griewank(Table 3). Also, proposed algorithms could reach specified goal very fast, specially compare to their competitor algorithm PSO-TVAC and GPSO. For example, for all functions, PSO-APLA$^1_1$ and PSO-APLA$^2_1$ could the goal with significantly less iterations than PSO-TVAC.

Finally, in Table 7, comparison of all tested PSO algorithms considering all benchmark functions is presented. For example, PSO-APLA$^1_1$ was significant-ly better than 7 of tested PSO algorithms and worse than one algorithm. Although in PSO-TVAC was inferior to no algorithm, PSO-APLA$^1_1$ which was only worse than PSO-TVAC has shown faster convergence to the global optima (Table 3).

30 Dimensions:

For 30-D Rastrigin function (Table 4), proposed algorithms performed better than all PSOIW In addition, with increasing maximum iteration to 10^4, APLA$_2$ could defeat successful algorithms like PSO-TAVC and GPSO as well and became inferior to no PSO algorithm for Rastrigin (Table 7). For other benchmark functions, proposed algorithm does not perform noteworthy. Yet, all proposed algorithm could successfully find the goal of all functions but Ackley. However, all proposed algorithm could reach the goal for all function faster than PSOIW, PSO-TVAC and GPSO (except Ackley and Rastrigin). This was of great importance since PSO-TVAC and GPSO were the closest competitors of the proposed algorithms.

In a different experiment, we tested all algorithms in case of availability of more computational resources by increasing the maximum number of iterations to 10,000 iterations (Table 7). Although there is no noticeable result for any of proposed algorithms on the benchmark functions separately, when comparing the results of PSO algorithms for all functions (Table 7), PSO-APLA$_2$ can be called the best algorithm since not only it was worse than no algorithm, but also it defeated 8 of 10 algorithms.

Table 2 Average best fitness value for 10-D function, MaxIteration=5000

	f_1	f_2	f_3	f_4	f_5
SPSO, $c_{1,2}$=1.49	7.60E-01±1.78E-01	5.42E-273±0.00E+00	7.13E+00±4.35E-01	3.19E-01±2.46E-01	7.42E-02±4.61E-03
SPSO, $c_{1,2}=2.0$	3.97E+00±3.85E-01	6.60E-35±5.16E-35	5.52E-01±8.43E-02	**3.68E-15**±7.60E-17	7.62E-02±4.00E-03
SPSO, $c_{1,2}=2.05$	5.72E+00±1.27E+00	1.59E-19±9.46E-20	**4.67E-01**±8.65E-02	1.56E-10±1.20E-10	1.04E-01±6.81E-03
PSOIW, $c_{1,2}$=1.49	6.46E-01±1.61E-01	0.00E+00±0.00E+00	2.10E+00±1.41E-01	3.56E-15±4.04E-17	6.23E-02±3.34E-03
PSOIW, $c_{1,2}$=2.0	1.36E+00±1.84E-01	1.17E-130±1.28E-130	1.43E+00±1.22E-01	**3.53E-15**±3.29E-17	**5.89E-02**±2.98E-03
PSOIW, $c_{1,2}$=2.05	2.05E+00±2.37E-01	7.59E-107±5.25E-107	1.48E+00±1.19E-01	**3.54E-15**±4.04E-17	6.53E-02±3.30E-03
DCPSO	1.55E+01±3.19E+00	8.45E-10±6.54E-10	1.37E+00±1.62E-01	1.54E+00±6.08E-01	6.67E-02±4.03E-03
GPSO	3.39E+00±4.45E-01	1.68E-18±1.06E-19	5.92E-18±1.17E-17	1.51E-09±5.10E-11	**5.16E-02**±2.64E-03
DAPSO	1.70E+02±3.20E+01	4.01E-01±2.88E-02	6.55E+00±2.34E-01	1.96E+00±6.69E-02	6.71E-01±1.80E-02
PSO-LP	7.66E-01±1.71E-01	**0.00E+00**±0.00E+00	2.63E+01±1.58E+00	1.95E+01±2.56E-01	1.78E-01±1.96E-01
PSO-TAVC	4.43E-01±1.40E-01	**0.00E+00**±0.00E+00	1.83E+00±1.36E-01	**3.60E-15**±5.69E-17	**5.36E-02**±2.83E-03
PSO-APLA$^{1}_{1}$	1.75E+00±2.89E-01	1.02E-28±1.97E-28	1.19E-01±4.22E-02	1.94E-12±3.78E-12	**5.84E-02**±2.93E-03
PSO-APLA$^{2}_{1}$	5.60E-01±1.43E-01	7.35E-46±1.41E-45	**1.19E+00**±1.84E-01	**3.78E-15**±9.85E-17	6.46E-02±3.68E-03
PSO-APLA$_2$	**1.79E-01**±6.57E-02	**2.20E-81**±3.95E-81	3.06E+00±4.69E-01	9.25E-12±1.82E-11	9.33E-02±5.83E-03

Table 3 Success rate and average number of FEs to reach goal in for 10-D functions, MaxIteration=5000

	Success Rate					***Average Number of FEs to Reach Goal***				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
SPSO, $c_{1,2}$=1.49	**100%**	**100%**	**100%**	96.7%	79.3%	**4,450**±519	4,388±32	477±23	**4,375**±302	**9,853**±1,725
SPSO, $c_{1,2}=2.0$	**100%**	**100%**	**100%**	**100%**	77.0%	18,569±1,526	22,499±323	828±46	20,199±342	84,316±6,470
SPSO, $c_{1,2}=2.05$	99.3%	**100%**	**100%**	**100%**	58.0%	26,160±1,889	33,642±583	904±53	29,977±561	104,425±7,092
PSOIW,$c_{1,2}$=1.49	**100%**	**100%**	**100%**	**100%**	90.0%	26,923±801	30,112±167	865±89	28,884±189	46,505±2,099
PSOIW, $c_{1,2}$=2.0	**100%**	**100%**	**100%**	**100%**	94.3%	70,421±1,214	76,688±260	2,904±283	74,422±282	107,723±2,757
PSOIW,$c_{1,2}$=2.05	**100%**	**100%**	**100%**	**100%**	87.0%	78,260±1,264	84,351±297	3,133±313	82,052±332	117,227±2,917
DCPSO	97.7%	**100%**	**100%**	92.3%	85.0%	21,169±3,981	4,988±187	**417**±14	9,541±2,085	49,122±5,660
GPSO	**100%**	**100%**	**100%**	**100%**	98.0%	10,637±622	16,945±214	531±28	15,225±269	24,082±2,152
DAPSO	53.0%	0.0%	**100%**	0.0%	0.0%	85,809±6,452	-	1,347±97	-	-
PSO-LP	**100%**	**100%**	**100%**	1.0%	74.0%	7,124±1,447	4,785±45	2,555±91	18,840±54,993	23,852±5,696
PSO-TAVC	**100%**	**100%**	**100%**	**100%**	95.3%	33,340±701	36,708±159	3,746±234	35,881±173	52,917±1,665
PSO-APLA$^{1}_{1}$	**100%**	**100%**	**100%**	**100%**	95.0%	12,942±1,940	11,322±741	812±55	14,628±845	46,805±4,536
PSO-APLA$^{2}_{1}$	**100%**	**100%**	**100%**	**100%**	86.7%	7,355±1,146	5,682±364	629±41	10,046±1,363	28,341±4,391
PSO-APLA$_2$	**100%**	**100%**	**100%**	**100%**	66.7%	8,434±1,028	8,190±406	1,017±82	12,490±1,609	23,920±4,825

Table 4 Average best fitness value for 30-D functions, MaxIteration=5000

	f_1	f_2	f_3	f_4	f_5
SPSO, $c_{1,2}$=1.49	8.55E+00±1.74E+00	2.24E-106±1.72E-106	1.62E+02±4.87E+00	1.74E+01±7.79E-01	2.07E-02±4.30E-03
SPSO, $c_{1,2}=2.0$	5.16E+03±7.32E+02	9.67E+00±1.46E+00	1.11E+02±4.62E+00	6.52E+00±9.09E-01	1.07E+00±1.80E-02
SPSO, $c_{1,2}=2.05$	1.99E+05±3.47E+04	3.71E+02±3.41E+01	1.76E+02±4.44E+00	9.57E+00±6.84E-01	4.60E+00±3.23E-01
PSOIW, $c_{1,2}$=1.49	1.48E+01±3.06E+00	2.83E-112±5.58E-112	7.94E+01±2.55E+00	1.09E+01±1.31E+00	1.74E-02±2.85E-03
PSOIW, $c_{1,2}$=2.0	4.45E+01±4.98E+00	6.86E-29±6.57E-29	3.60E+01±1.47E+00	4.87E+00±1.15E+00	1.44E-02±2.19E-03
PSOIW, $c_{1,2}$=2.05	5.14E+01±5.59E+00	1.00E-20±3.97E-21	3.43E+01±1.38E+00	4.29E+00±1.12E+00	1.67E-02±2.83E-03
DCPSO	1.71E+02±2.43E+01	6.54E-04±1.29E-04	4.47E+01±2.09E+00	1.25E+01±1.13E+00	4.34E-02±5.03E-03
GPSO	7.12E+01±5.15E+00	4.90E-06±9.17E-07	5.33E-04±2.43E-04	5.51E-04±4.91E-05	3.38E-02±3.69E-03
DAPSO	8.28E+03±6.51E+02	1.11E+02±3.99E+00	1.39E+02±4.21E+00	1.72E+01±6.34E-01	1.96E+00±3.37E-02
PSO-LP	8.22E+00±2.38E+00	4.33E-97±8.51E-97	1.12E+02±4.51E+00	1.98E+01±1.29E-02	4.56E-02±7.24E-03
PSO-TAVC	1.10E+01±2.42E+00	2.01E-86±2.73E-86	7.71E+01±2.04E+00	6.34E+00±9.41E-01	3.55E-02±4.17E-03
PSO-APLA$^{1}_{1}$	5.59E+01±5.90E+00	4.15E-06±6.78E-06	1.53E+01±9.23E-01	1.08E+01±1.11E+00	2.02E-02±2.64E-03
PSO-APLA$^{2}_{1}$	6.05E+01±7.89E+01	1.04E+02±1.06E+02	3.18E+01±2.86E+00	1.47E+01±9.42E-01	5.44E-02±2.78E-02
PSO-APLA$_2$	3.40E+01±3.94E+00	4.09E-18±6.35E-18	6.88E+00±3.84E-01	1.51E+01±8.54E-01	2.90E-02±3.69E-03

Table 5 Success rate and average number of FEs to reach goal for 30-D functions, MaxIteration=5000

	Success Rate					***Average Number of FEs to Reach Goal***				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
SPSO, $c_{1,2}$=1.49	**100%**	**100%**	3.5%	1.5%	97.0%	20,053±2,196	13,940±138	10,697±2,115	15,280±1,292	12,655±123
SPSO, $c_{1,2}=2.0$	0.0%	0.0%	41.0%	0.0%	0.0%	-	-	143,952±7,307	-	-
SPSO, $c_{1,2}=2.05$	0.0%	0.0%	1.0%	0.0%	0.0%	-	-	165,300±397,196	-	-
PSOIW,$c_{1,2}$=1.49	**100%**	**100%**	87.0%	25.0%	99.5%	63,943±1,925	57,572±193	54,780±969	59,516±857	55,987±224
PSOIW,$c_{1,2}$=2.0	95.0%	**100%**	**100%**	72.5%	**100%**	138,047±2,069	127,846±321	113,680±920	132,513±1,812	125,664±357
PSOIW,$c_{1,2}$=2.05	92.5%	**100%**	**100%**	77.0%	99.5%	150,108±1,632	141,160±344	125,156±1,199	144,382±1,230	138,223±494
DCPSO	55.7%	**100%**	99.0%	34.0%	89.7%	92,024±6,635	72,875±3,526	39,356±3,497	116,696±5,531	64,661±4,012
GPSO	87.7%	**100%**	**100%**	**100%**	95.7%	104,276±3,889	114,360±922	15,948±274	94,754±1,206	102,180±1,364
DAPSO	0.0%	0.0%	12.0%	0.0%	0.0%	-	-	168,478±7,968	-	-
PSO-LP	99.3%	**100%**	39.3%	0.0%	89.3%	23,726±3,038	17,789±273	104,400±6,184	-	12,336±347
PSO-TAVC	99.3%	**100%**	91.3%	19.3%	95.3%	69,716±1,756	61,719±219	60,896±639	67,529±1,871	60,825±410
PSO-APLA$_1^1$	90.0%	**100%**	**100%**	43.7%	98.7%	72,933±4,637	54,528±2,781	47,468±2,529	82,313±5,666	54,098±2,851
PSO-APLA$_1^2$	97.7%	96.7%	97.0%	21.3%	90.3%	33,916±3,056	20,284±1,012	54,772±4,274	72,056±11,772	23,675±2,354
PSO-APLA$_2$	97.0%	**100%**	**100%**	12.7%	95.7%	57,497±4,094	40,822±1,078	48,014±1,579	132,405±18,088	36,481±1,169

Table 6 Average best fitness value for 30-D functions, MaxIteration=10,000

	f_1	f_2	f_3	f_4	f_5
SPSO, $c_{1,2}$=1.49	3.45E+00±1.27E+00	1.07E-217±0.00E+00	1.58E+02±4.81E+00	1.70E+01±8.47E-01	2.52E-02±6.06E-03
SPSO, $c_{1,2}=2.0$	4.89E+02±8.08E+01	6.59E-02±2.95E-02	7.11E+01±3.61E+00	1.80E+00±6.63E-01	2.69E-01±3.29E-02
SPSO, $c_{1,2}=2.05$	2.98E+04±7.25E+03	6.95E+01±8.36E+00	1.35E+02±4.14E+00	5.39E+00±4.20E-01	1.62E+00±7.49E-02
PSOIW, $c_{1,2}$=1.49	7.47E+00±1.99E+00	2.43E-237±0.00E+00	5.37E+01±2.13E+00	8.69E+00±1.32E+00	1.85E-02±2.86E-03
PSOIW, $c_{1,2}$=2.0	2.80E+01±4.24E+00	4.52E-63±3.15E-63	2.62E+01±1.06E+00	2.00E+00±7.95E-01	1.74E-02±2.95E-03
PSOIW, $c_{1,2}$=2.05	3.66E+01±4.88E+00	6.11E-45±8.38E-45	2.50E+01±9.78E-01	1.36E+00±6.73E-01	1.62E-02±2.68E-03
DCPSO	1.40E+02±2.91E+01	7.15E-05±1.72E-05	3.16E+01±2.16E+00	1.31E+01±1.37E+00	3.63E-02±5.19E-03
GPSO	5.30E+01±4.56E+00	1.49E-13±3.96E-14	7.95E-12±3.09E-12	1.08E-07±2.20E-08	2.80E-02±3.76E-03
DAPSO	1.47E+03±1.34E+02	1.89E+01±8.97E-01	8.77E+01±3.05E+00	1.12E+01±9.70E-01	1.16E+00±7.62E-03
PSO-LP	1.49E+00±7.86E-01	8.28E-241±0.00E+00	9.54E+01±5.52E+00	1.98E+01±1.12E-02	5.96E-02±1.01E-02
PSO-TAVC	2.55E+00±8.59E-01	5.45E-183±0.00E+00	5.09E+01±1.96E+00	4.64E+00±1.09E+00	3.38E-02±4.66E-03
PSO-APLA$_1^1$	3.02E+01±4.36E+00	2.69E-14±4.30E-14	3.20E+00±3.81E-01	8.29E+00±1.34E+00	1.87E-02±3.17E-03
PSO-APLA$_1^2$	9.84E+00±3.92E+00	3.46E+01±6.31E+01	1.16E+01±1.33E+00	1.11E+01±1.33E+00	3.30E-02±1.06E-02
PSO-APLA$_2$	1.12E+01±2.71E+00	3.67E-42±6.26E-42	2.49E+00±3.96E-01	6.64E-01±4.30E-01	2.40E-02±4.17E-03

Table 7 Cumulative Comparison Results of PSO algorithms

Dim	**Dim=10,MaxIteration=5000**		**Dim=30,MaxIteration=5000**		**Dim=30,MaxIteration=10000**	
	superior to #	**inferior to #**	**superior to #**	**inferior to #**	**superior to #**	**inferior to #**
SPSO, $c_{1,2}$=1.49	4	8	6	5	4	6
SPSO, $c_{1,2}=2.0$	3	6	2	10	2	10
SPSO, $c_{1,2}=2.05$	2	10	0	12	0	12
PSOIW, $c_{1,2}$=1.49	9	1	8	1	8	1
PSOIW, $c_{1,2}$=2.0	9	0	**12**	**0**	9	0
PSOIW, $c_{1,2}$=2.05	7	5	8	1	9	1
DCPSO	1	11	3	8	4	8
GPSO	4	7	6	4	6	4
DAPSO	0	13	1	11	1	11
PSO-LP	1	7	3	8	4	8
PSO-TAVC	**12**	**0**	6	2	6	2
PSO-APLA$_1^1$	10	1	7	1	8	0
PSO-APLA$_1^2$	9	2	6	5	6	4
PSO-APLA$_2$	6	6	6	1	**11**	**0**

6. Conclusion

In this paper, we proposed LA-based algorithms to adjust parameters of PSO. Two approaches for adjusting value of parameters of PSO were adopted. In first approach, value of a parameter is selected from a finite set while in the second approach, value of a parameter either changes conservatively by a fixed amount or does not change at all. Experimental results show that having the same error (and sometimes less) as SPSO, PSOIW, DCPSO, DAPSO, GPSO, PSOLP, and PSO-TAVC for all benchmark functions, proposed algorithms can reach the goal of test functions faster than most of other PSO algorithms with the same, or even better, success rate. Moreover, considering all benchmark functions to compare algorithms, the proposed algorithms can only be defeated by PSO-TVAC in terms of error but significantly converge faster than it. Also, when there is more computational power available, the proposed algorithm, PSO-APLA$_2$, performs better than all tested PSO algorithms considering all benchmark functions. Although the proposed algorithms do not need any priori knowledge about the search space, they require some parameter tuning like GPSO and PSO-TAVC.

10. References

[1] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942-1948.

[2] A. Carlisle and G. Dozier, "An Off-the-Shelf Pso," in Workshop on Particle Swarm Optimization, 2001, pp. 1-6.

[3] Y. Shi and R. C. Eberhart, "Parameter Selection in Particle Swarm Optimization," in 7th International Conference on Evolutionary Programming Vii Lecture Notes in Computer Science, vol. 1447, 1998, pp. 591–600.

[4] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer," in IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998, pp. 69-73.

[5] A. Chatterjee and P. Siarry, "Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization," Computers & Operations Research, vol. 33, no. 3, pp. 859-871, 2006.

[6] M. Clerc and J. Kennedy, "The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space," IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, 2002.

[7] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients," IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 240-255, 2004.

[8] M. Løvbjerg, T. K. Rasmussen, and T. Krink, "Hybrid Particle Swarm Optimiser with Breeding and Subpopulations," in Third Genetic and Evolutionary Computation Conference, 2001, pp. 469-476.

[9] R. Poli, C. D. Chio, and W. B. Langdon, "Exploring Extended Particle Swarms: A Genetic Programming Approach," in 2005 Conference on Genetic and evolutionary computation, Washington DC, USA, 2005, pp. 169-176.

[10] H. Beigy and M. R. Meybodi, "An Adaptive Call Admission Algorithm for Cellular Networks," Journal of Computer and Electrical Engineering, vol. 31, no. 2, pp. 132-151, 2005.

[11] H. Beigy and M. R. Meybodi, "Call Admission Control in Cellular Mobile Networks: A Learning Automata Approach," in Eurasia-Ict 2002: Information and Communication Technology, Lecture Notes in Computer Science, vol. 2510, 2002, pp. 450-457.

[12] B. J. Oommen and T. D. Roberts, "Continuous Learning Automata Solutions to the Capacity Assignment Problem," IEEE Transactions on Computers, vol. 49, no. 6, pp. 608-620, 2000.

[13] M. R. Meybodi and H. Beigy, "A Note on Learning Automata Based Schemes for Adaptation of Bp Parameters," Journal of Neurocomputing, vol. 48, no. 4, pp. 957-974, 2002.

[14] H. Beigy and M. R. Meybodi, "A Learning Automata Based Algorithm for Determination of the Number of Hidden Units for Three Layers Neural Networks," International Journal of Systems Science, to appear.

[15] P. N. Suganthan, "Particle Swarm Optimizer with Neighborhood Operator," in Congress of Evolutionary Computation, Washington D.C, USA, 1999, pp. 1958-1962.

[16] J. Kennedy, "The Behavior of Particles," in 7th International Conference on Evolutionary Programming VII, 1998, pp. 581-589.

[17] Y. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization," in IEEE Congress on Evolutionary Computation, Carmel, IN, U.S.A., 1999, pp. 101-106.

[18] P. Tawdross and A. Konig, "Local Parameters Particle Swarm Optimization," in Sixth International Conference on Hybrid Intelligent Systems (HIS'06), 2006, pp. 52-55.

[19] S. Pasupuleti and R. Battiti, "The Gregarious Particle Swarm Optimizer (G-Pso)," in 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, 2006, pp. 67-74.

[20] X. Yanga, J. Yuana, J. Yuana, and H. Maob, "A Modified Particle Swarm Optimizer with Dynamic Adaptation," Applied Mathematics and Computation, vol. 189, no. 2, pp. 1205-1213, 2007.

[21] M. Thathachar and P. Sastry, "Varieties of Learning Automata: An Overview," IEEE Transactions on Systems, Man and Cybernetics, vol. 32, no. 6, pp. 711-722, 2002.

[22] R. C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization," in Congress on Evolutionary Computation, San Diego, CA., 2000, pp. 84 -88.

[23] J. Jie, J. Zeng, and C. Han, "Adaptive Particle Swarm Optimization with Feedback Control of Diversity," in Computational Intelligence and Bioinformatics, Lecture Notes in Computer Science, vol. 4115, vol. 4115, 2006, p. 81.

[24] B. F. David and B. Hans-Georg, "A Note on the Empirical Evaluation of Intermediate Recombination," Evolutionary Computation, vol. 3, no. 4, pp. 491-495, 1995.

[25] P. J. Angeline, "Using Selection to Improve Particle Swarm Optimization," in IEEE International Conference on Evolutionary Computation, 1998, pp. 84-89.

[26] P. Angeline, "Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences," in Evolutionary Programming Vii, 1998, pp. 601-610.

[27] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, USA, 2007, pp. 120-127.

[28] F. van den Bergh and A. P. Engelbrecht, "Effects of Swarm Size on Cooperative Particle Swarm Optimizers," in Genetic and Evolutionary Computation Conference, San Francisco, USA, 2001, pp. 892-899.

[29] S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," Scandinavian Journal of Statistics, vol. 6, pp. 65-70, 1979.