Memetic-CLA-PSO: A Hybrid Model for Optimization

Mojdeh Akhtari , Mohammad Reza Meybodi

Computer Engineering and Information Technology Department, Azad Islamic University Qazvin, Iran
Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
m_akhtari@qiau.ac.ir, mmeybodi@aut.ac.ir

Abstract. In this paper a hybrid model which is a combination of Memetic algorithm, cellular learning automata (CLA) and PSO is proposed. The proposed algorithm in addition to maintaining diversity; largely reduces the probability of getting trap in local optima. Experimental results on eight optimization problems show the superiority of the proposed algorithm.

Keywords: Memetic Algorithm, Particle Swarm Optimization, Cellular Learning Automata, Meme, Optimization.

1 Introduction

Although most EAs are capable of detecting the region of attraction of the global optimizer fast, once there, they cannot perform a refined local search to compute the optimum with high accuracy, unless specific procedures are incorporated in their operators. Some versions of PSO also exhibit this deficiency. The aforementioned drawback of EAs triggered the development of Memetic Algorithms (MAs), which incorporate local search components. MAs constitute a class of metaheuristics that combines population-based optimization algorithms with local search procedures (Dawkins 1976; Moscato 1989, 1999). More specifically, MAs consist of a global component, which is responsible for a rough search of the search space and the detection of the most promising regions, and a local search component, which is used for probing the detected promising regions, in order to obtain solutions with high accuracy. EAs have been used as the global component in MAs with Simulated Annealing and random local search (Moscato, 1999).

The advantage of PSO algorithm over other EA algorithms is easy to implement and few parameters to adjust and is an optimize probability algorithm that extracted a population of particle to search the space. In PSO, particles try adjusting their path and move toward the best group experience to move into the ultimate solution. Since the particles in the algorithm gradually move toward the best solution found so far, if this solution is a local optima, all the particles move toward it and the PSO, does not offer approach for removal of local optima. This is the biggest problem of the standard pso that causes incapable in solving multiple peaks problems, especially with a large state space. One of the actions taken to address this problem is use a meme as a local search around the best solution and exit out of the local optima.

On the other hand, in the PSO algorithm, inertia weight is the coefficient of particle velocity that is used to determine the particle velocity in the next step. Setting this parameter can be established the balance between global search and local search. In some applications, inertia

weight is considered fixed. But often the inertia weight must have a higher value at first and over time decreases value linearly. Thus, particles at the beginning of the search process attempted to identify new locations of the search space but over time have more desire to following. Decries inertia weight in linear way is very simple method that doesn't have ability to complete apply basic knowledge about the problem and also couldn't use the feedback system.

In order to solve this problem, CLAPSO algorithm the combination of cellular learning automata and particle swarm optimization, presented by Sheibani, Meybodi at first. Learning Automata of each cell has two actions "following" and "continue the current path" that regulate the behavior of the particles of that cell. In this algorithm using Learning Automata has two important advantages: First, the existing knowledge on the problem can be used to determine process of change inertia weight and secondly, this process is corrected by getting feedback from implementation of algorithm. To set the behavior of particles, each cell learning automata, use the experiences of themselves and learning automata cells neighboring. Thus a kind of cooperation between populations of a cell and neighboring cells formed which make them better performance. Figure.1 shows a general view of CLAPSO with two cells.

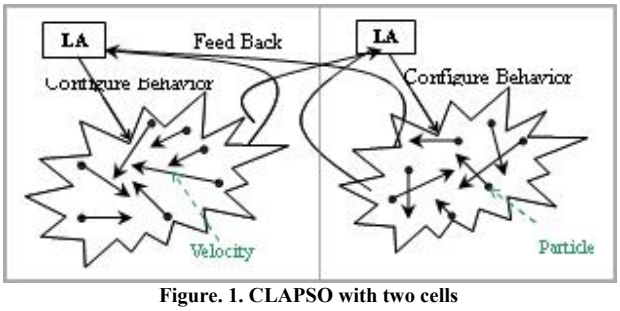


Figure. 1. CLAPSO with two cells

Premature convergence is one of the problems of this method. In this method the group is better than other groups, and has a better global best will be encouraged to follow and the particles of this group, regardless of their initial velocity follow the gbest and pbest and quickly converge. To resolve this problem Hamidi, Meybodi offered a new model which significantly improved CLA-PSO algorithm.

In this paper, a new structure of PSO algorithm is presented which hybridize the features of memetic algorithm by CLA-PSO so in addition to maintaining diversity; largely can reduce the probability of getting trap in local optima solutions.

2 CLA-PSO

CLA-PSO algorithm is the combination of cellular learning automata and particle swarm optimization, which presented by Sheibani, Meybodi at first. After that Hamidi, Meybodi offered a new model which improved CLA-PSO algorithm as described in introduction. In this paper we use this model for proposed algorithm. This model has n particles which each particle having a position and velocity vector. The search space, particles are located in is D-dimensional. In this model each cell of cellular learning automata includes one dimension of one particle. So D learning automates is considered. The neighboring between cells is considered Moore neighborhood. This neighborhood structure is shown in Figure 2.

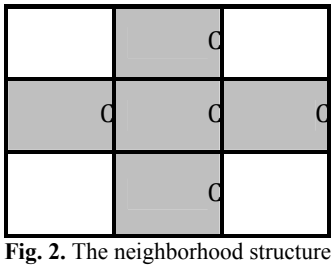


Fig. 2. The neighborhood structure

The learning automaton in each cell of CLA is responsible for adjusting behavior of a specific particle dimension. Learning Automata of each cell has two actions "following" and "to continue the current path". Adjusting the behavior of a specific particle dimension means, LA in each step, determines that if the particles should continue their current path or should follow the best particle found so far. The action that LA associated to a cell selected in each step, determines how to update velocity of specific particle dimension, on that step. If “following” action selected, only following the best personal experience and the best group experience, will be considered to update velocity of a specific dimension of particle and current velocity of the particles will be eliminated. If “to continue the current path” action selected, the previous velocity is also considered to calculate new particle velocity.

Premature convergence is one of the major problems of PSO algorithm. Even if the diversity of particles at a specific dimension reduced, the particles are stopped at that specific dimension. Therefore, if particles at a specific dimension continue “following” action, they will be converged quickly.

The global response to learning automates is calculated as follows. If particle position is better than previous particle position in a step, action chosen by all learning automata dedicated to the particle dimensions is evaluated positively and otherwise negatively. This global response is called βg.

The local response or βl which a LA receives after select own action is calculated as follow. To evaluate the selected action, the variance of neighboring cells position is calculated. Whenever this variance is much less than certain limit it means the particles close to each other on that

dimension and go towards convergence. To prevent premature convergence, the particle shouldn't continue "following" action. But continue current path. In this condition if the action chosen by the learning automata is "following", it is evaluated negatively and if selected action is "to continue the current path" it is evaluated positively. Whenever variance of particles is much more than certain limit it means the diversity between particles is high and selected action doesn't cause premature convergence, so "following" action is evaluated positively and "to continue the current path" action is evaluated negatively.

This algorithm describe as follow:

1. Initial position and velocity of particles randomly placed in the search space.
2. A learning automaton dedicated to each dimension of particle.
3. Probability vectors of select learning automata Initialized.
4. Until the maximum number of steps to be performed or the desired goal is achieved, steps 5 to 10 will be repeated :
5. LA selects one of its actions using its probability vector.
6. If "following" action is selected by LA of dth dimension of ith particle, velocity of particle is updated as follows (the inertia of particle velocity is considered zero):

   $V_i^d(t+1) = c_1 rand_1(t)(pbest_i(t) - X_i^d(t)) + c_2 rand_2(t)(gbest_i(t) - X_i^d(t))\ \ d \in 1,....,N_d$

   If "continue current path" action is selected, the particle velocity is updated as follows:

   $$w(t) = w_0 - \frac{t * w_1}{MaxGen}$$

   $$V_i^d(t+1) = w(t) * V_i^d(t) + c_1 * rand_{1i}^d * (pbest_i^d(t) - X_i^d(t)) + c_2 * rand_{2i}^d * (gbest^d(t) - X_i^d(t))$$
7. Global response taken from environment by LA is calculated as follows:

   $$\beta g_i = \begin{cases} 0 & if\ \ fitness(X_i(t+1)) < fitness(X_i(t)) \\ 1 & otherwise \end{cases}$$
8. Local response taken from environment by LA is calculated as follows:

   if var *iance* (*neighbor* ($X^d{}_i(t+1)$)) < *vTreshold*

   if $LA_i^d.action = 1$

   $\beta l_i^d = 1$

   else

   $\beta l_i^d = 0;$

   else

   if $LA_i^d.action = 1$

   $\beta l_i^d = 0$

   else

   $\beta l_i^d = 1;$

   end
9. Reward and punishment of learning automata in any dimension are evaluated based on global and local response as follows:

$$\beta_i^d = \begin{cases} \beta l_i^d & if\ \ \beta g_i = 1 \\ 0 & if\ \ \beta g_i = 0 \end{cases}$$

10. LA's probability vector for selecting actions is updated.

3 Memetic Algorithm

Memetic algorithms constitute a class of metaheuristics that combines population-based optimization algorithms with local search procedures [1]. These methods are inspired by models of natural systems that combine the evolutionary adaptation of a population with individual learning within the lifetimes of its members. Additionally, MAs are inspired by Richard Dawkin's concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement [2]. From an optimization point of view, MAs have been shown to be both more efficient (i.e. requiring orders of magnitude fewer evaluations to find optima) and more effective (i.e. identifying higher quality solutions) than traditional EAs for some problem domains. As a result, MAs are gaining wide acceptance, in particular in well-known combinatorial optimization problems where large instances have been solved to optimality and where other meta-heuristics have failed to produce comparable results (see for example [3] for a comparison of MAs against other approaches for the Quadratic Assignment Problem). [4]

Metaheuristics such as GA or PSO amazing in the last iterations and couldn't continue their path correctly. One of the reasons is to reduce diversity. In GA mutation solve this problem somehow. One solution in this way is using Memetic algorithms. The idea is to imitate the effect of learning and social interaction during the life span of an individual. Researcher use local search as a meme to learning. Another problem is premature convergence. Since the particles in the algorithm gradually move toward the best solution found so far, if this solution is a local optima, all the particles move toward it and the PSO, does not offer approach for removal of local optima. One of the actions taken to address this problem is use a meme as a local search around the best solution and exit out of the local optima.

Petalas et al. introduced an algorithm named Memetic Particle Swarm Optimization consists of two main components, a global one that is responsible for the global search of the search space, and a local one, which performs more refined search around potential solutions of the problem at hand. They employed a stochastic iterative LS technique in their MA, called random walk with direction exploitation (RWDE), where a sequence of approximations of the optimizer are generated by assuming a random vector as a search velocity. [5] In that paper, the reason for choosing RWDE was described as its simplicity and its relative efficiency. RWDE does not make any continuity or differentiability assumptions on the objective function, thus, it is consistent with the PSO framework that requires function values solely. Therefore, it was preferred over gradient-based local search algorithms. Moreover, it is easily implemented and it can be modified with minor effort to suit different problems. [5]

3.1 Random walk with direction exploitation

RWDE is an iterative, stochastic optimization method that generates a sequence of approximations of the optimizer by assuming a random vector as a search direction. In this method at each steps the fitness value of $x^{(t)} + \lambda z^{(t)}$ is calculated as F' which λ is a prescribed scalar step-length and $z^{(t)}$is a unit-length random vector. Then the values of F' and F is compared. If better performance achieved, this is continued otherwise the scalar step is reduced as $\lambda = \lambda/2$. For equal performance x doesn't change. [6]

3.2 Proposed Algorithm

Inspired by these researches, we hybrid CLA-PSO with local search technique and use RWDE as a meme. So, two schemes are planned, local search on pbest and local search on gbest. The classification of Ong et al. describes adaptation of operators and parameters in EAs. They categorize algorithms according to One way is to use a fixed strategy; this is termed "static." Another is to use feedback of which operators have provided the best improvement recently. This is termed "adaptive" and they note that LS operators may be linked to candidate solutions (self-adaptive) [8], [9]. Essentially, all of these approaches maintain a pool of LS operators available to be used by the algorithm and, at each decision point, make a choice of which to apply. Another approach lies in the population based nature of MAs so we have multiple candidate solutions, which makes the task of deciding which LS operator to apply to any given one more complex. [7] So in proposed algorithm, we consider two type candidates to apply local search as a meme. One is local search on the overall best position per cell (gbest) and another, for each pbest a local search is considered. Krasnogor and Smith and Ong and Keane have suggested that multiple LS operators should be executed simultaneously on those individuals that are selected for local improvements and that a certain learning mechanism should be adopted to give the efficient LS methods greater chances to be chosen in the later stage. However, Neri et al. have also proposed a multiple LS based MA with a non-competitive scheme, where different LS methods can be activated during different population evolution periods. [10]

From the above discussion, we use a learning automaton, to control the local search on gbest and local search on pbest of particles. In fact this LA has tow actions .we use LA, may not only cooperate to improve the quality of individuals, but also compete with each other to achieve a greater selection probability. LA in each step selects an action as a meme from its probability vector. To promote competition between them, the selection probability of LS operators can be re-calculated according to learning automata rule where the LS operator with a higher fitness improvement is rewarded with more chance of being. Additionally, another solution we proposed to experiment as follows. Variance of particle with its neighbors is calculated. If improvement is achieved, again variance of particle with its neighbors is calculated (the variance before and after local search is calculated), if the variance increase or does not change, meme is applied

otherwise, doesn't apply. In addition we experiment the algorithm by only local search on pbest as a meme and another, experiment the algorithm only by local search on gbest as a meme.

The proposed algorithm used Meme for all populations at each iteration. In this paper local search as a meme describe as follows:

In each stage, meme apply on pbest of each particle as follows,

Calculate the following objective function value:

$$F' = F\big(pbest(t) + \lambda z(t)\big)$$

Compare the value of F' with $F(pbest)$:

- If F' was better than $F(pbest)$ then :
  - $pbest(t+1) = pbest(t) + \lambda z(t)$
  - $\lambda = \lambda_{initialize}$
  - $F\big(pbest(t)\big) = F'$
- If F' was worse than $F(pbest)$ then :
  - $pbest(t+1) = pbest(t)$
  - $\lambda = \lambda/2$
- If F' was equal to $F(pbest)$ then :
  - $pbest(t+1) = pbest(t)$

For another method in each stage, meme apply on pbest of each particle as follows,

Pbest of each particle with its neighbors, poured into a box and their variance obtained for all dimensions. (Var_pre) Then , $F' = F\big(pbest(t) + \lambda z(t)\big)$. If F' was better than $F(pbest)$ then $pbest(t) + \lambda z(t)$ of each particle with its neighbors, poured into a box and their variance obtained for all dimensions.(Var_after) If Var_after for all dimensions is larger than Var_pre, then apply changes.

In each stage, gbest is obtained from the best pbest of neighbors, which meme apply on it as follows,

Calculate the following objective function value:

$$F' = F\big(gbest(t) + \lambda z(t)\big)$$

Compare the value of F' with $F(gbest)$:

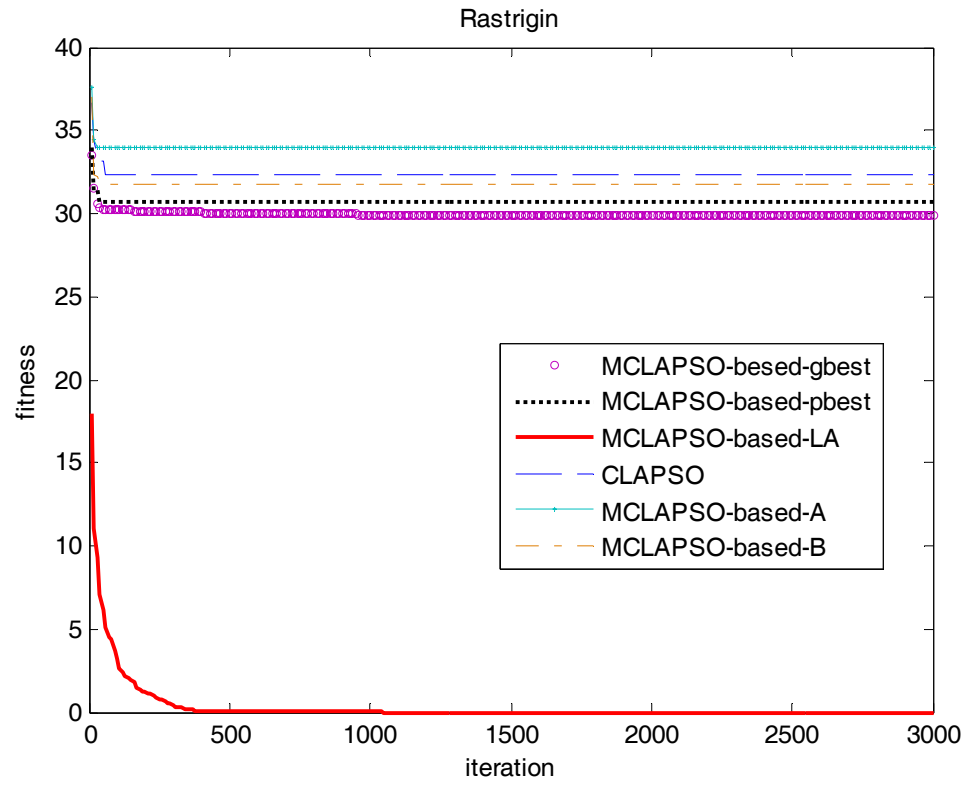
- If F' was better than $F(gbest)$ then :
  - $gbest(t+1) = gbest(t) + \lambda z(t)$
  - $\lambda = \lambda_{initialize}$
  - $F\big(gbest(t)\big) = F'$

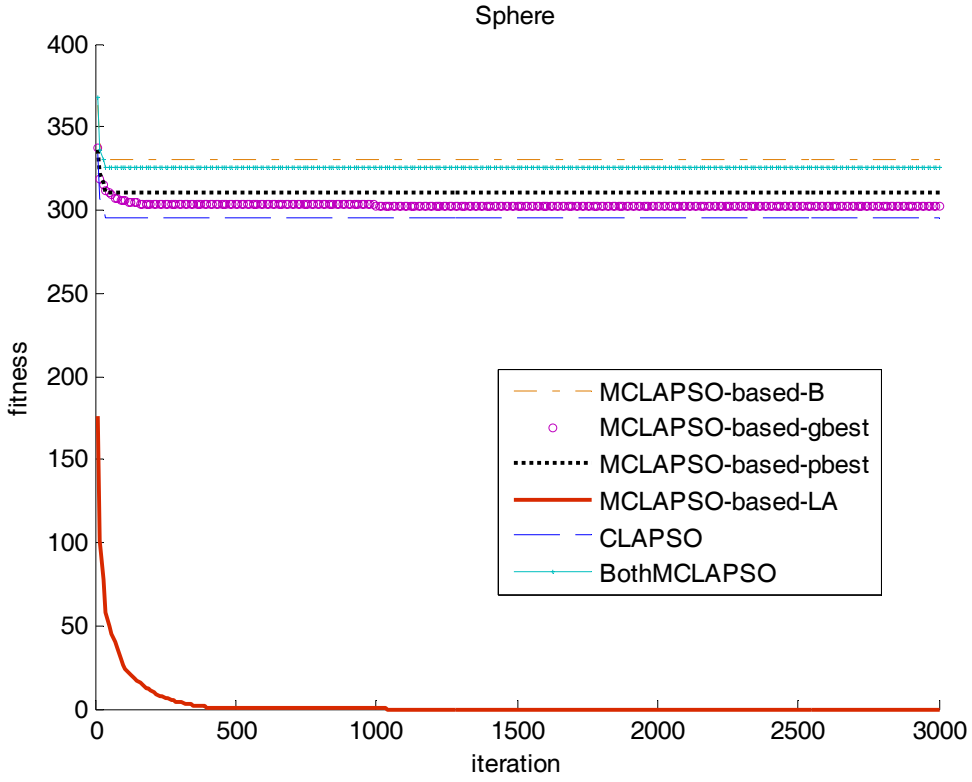
- If F' was worse than $F(gbest)$ then :
    - $gbest(t+1) = gbest(t)$
    - $\lambda = \lambda/2$
- If F' was equal to $F(pbest)$ then :
- If F' was equal to $F(gbest)$ then :
    - $gbest(t+1) = gbest(t)$

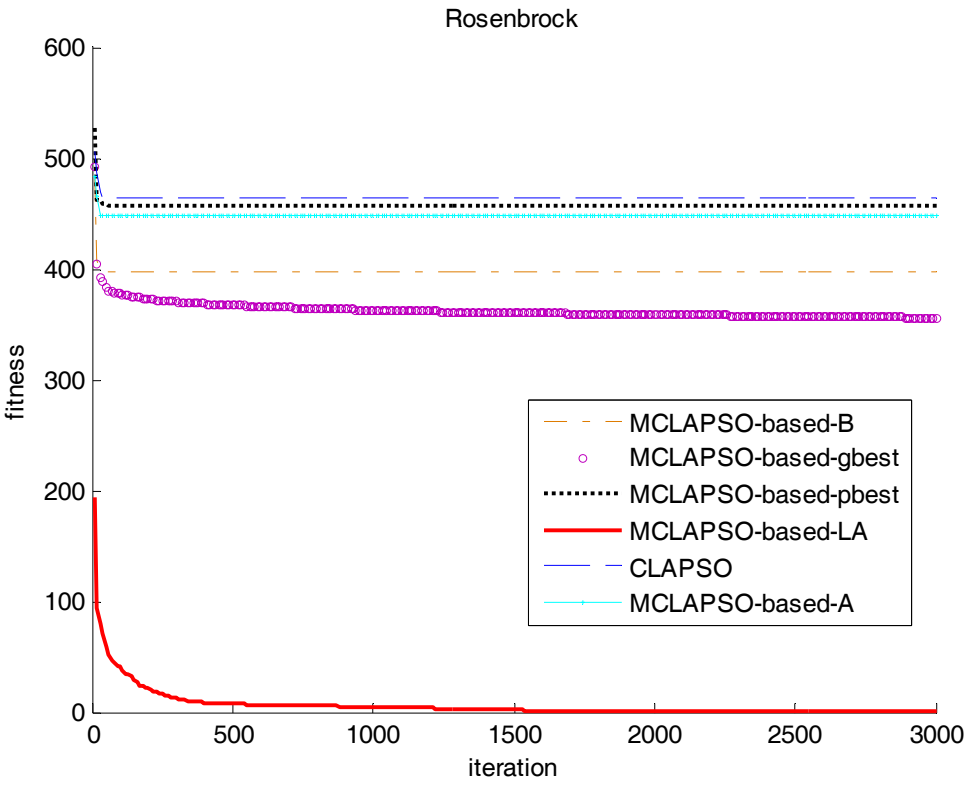
4 Experimental Results

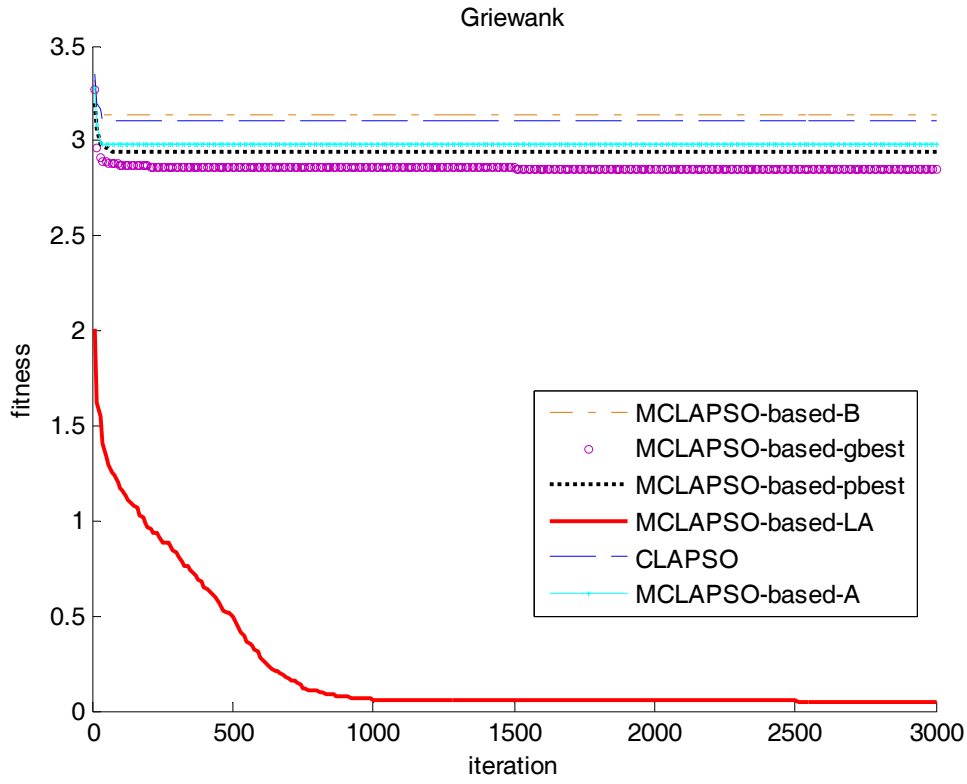
The experiments performed on eight standard functions which usually use as scale of evaluation in optimization algorithms.

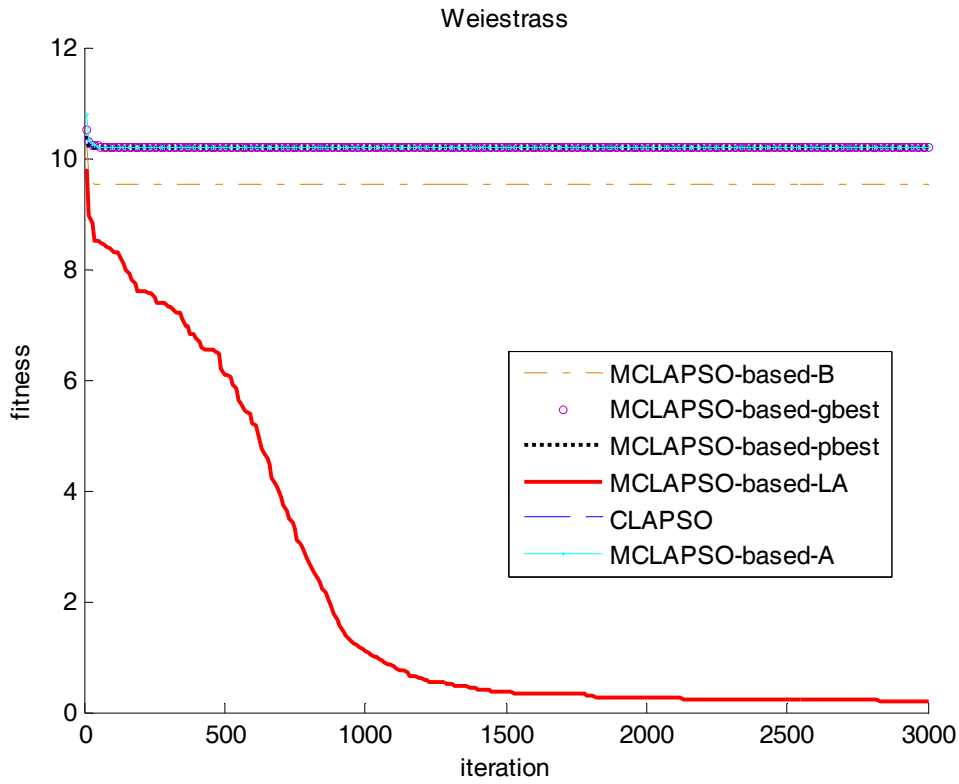
For all algorithms the value of *w0, w1, c1, c2* was set to 0.9, 0.5, 1.45, 1.45 respectively, for LRP, the parameters **α , β = 0.01** and $\lambda = 1$ in the RWDE local search. All the algorithms that we experimented described as follows. Memetic-CLA-PSO which performed local search on only gbest, named as MCLAPSO-based-gbest. Memetic-CLA-PSO which performed local search on only pbest of particles, named as MCLAPSO-based-pbest. Memetic-CLA-PSO which performed local search on both gbest and pbest, named as MCLAPSO-based-A. Memetic-CLA-PSO which performed local search on both gbest and pbest with consideration of variance, named as MCLAPSO-based-B. Memetic-CLA-PSO which performed local search on both gbest and pbest and using LA to control both, named as MCLAPSO-based-LA. For this algorithm we use LRP method for LA with parameters **α , β = 0.01.**

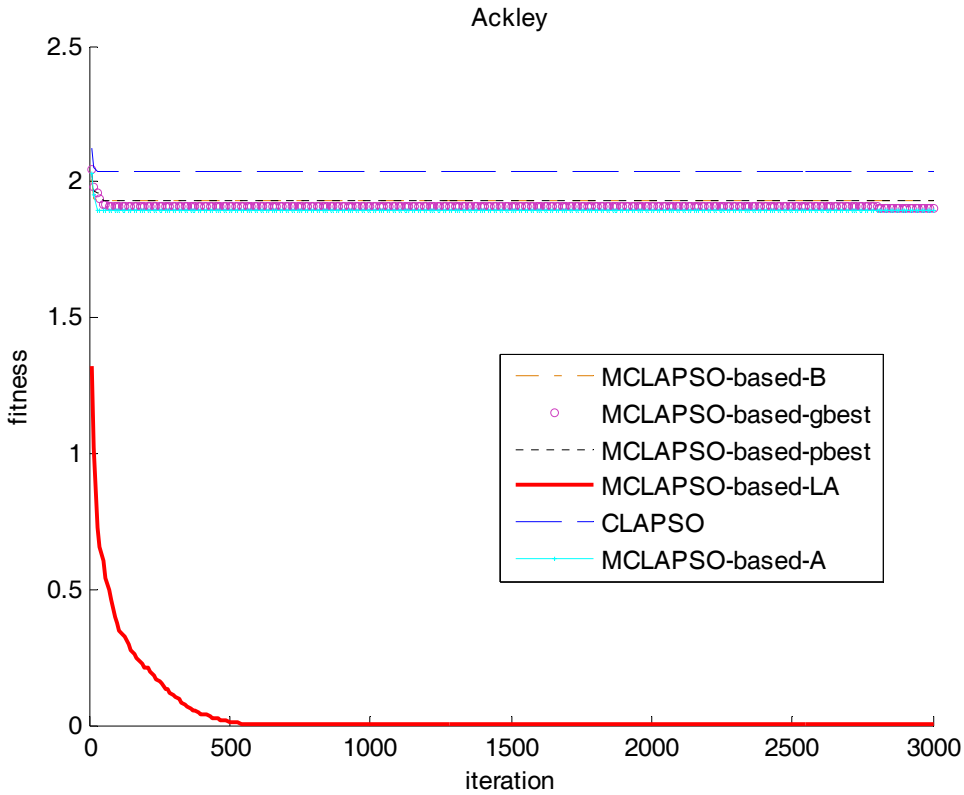

Rastrigin
fitness
iteration
MCLAPSO-besed-gbest
MCLAPSO-based-pbest
MCLAPSO-based-LA
CLAPSO
MCLAPSO-based-A
MCLAPSO-based-B
0
5
10
15
20
25
30
35
40
0
500
1000
1500
2000
2500
3000


Sphere
fitness
iteration
MCLAPSO-based-B
MCLAPSO-based-gbest
MCLAPSO-based-pbest
MCLAPSO-based-LA
CLAPSO
BothMCLAPSO
0
50
100
150
200
250
300
350
400
0
500
1000
1500
2000
2500
3000


Rosenbrock
600
500
400
300
200
100
0
fitness
0
500
1000
1500
2000
2500
3000
iteration
MCLAPSO-based-B
MCLAPSO-based-gbest
MCLAPSO-based-pbest
MCLAPSO-based-LA
CLAPSO
MCLAPSO-based-A


Griewank
3.5
3
2.5
2
1.5
1
0.5
0
fitness
0
500
1000
1500
2000
2500
3000
iteration
MCLAPSO-based-B
MCLAPSO-based-gbest
MCLAPSO-based-pbest
MCLAPSO-based-LA
CLAPSO
MCLAPSO-based-A


Weiestrass
12
10
8
6
4
2
0
fitness
0
500
1000
1500
2000
2500
3000
iteration
MCLAPSO-based-B
MCLAPSO-based-gbest
MCLAPSO-based-pbest
MCLAPSO-based-LA
CLAPSO
MCLAPSO-based-A


Ackley
2.5
2
1.5
1
0.5
0
fitness
0
500
1000
1500
2000
2500
3000
iteration
MCLAPSO-based-B
MCLAPSO-based-gbest
MCLAPSO-based-pbest
MCLAPSO-based-LA
CLAPSO
MCLAPSO-based-A

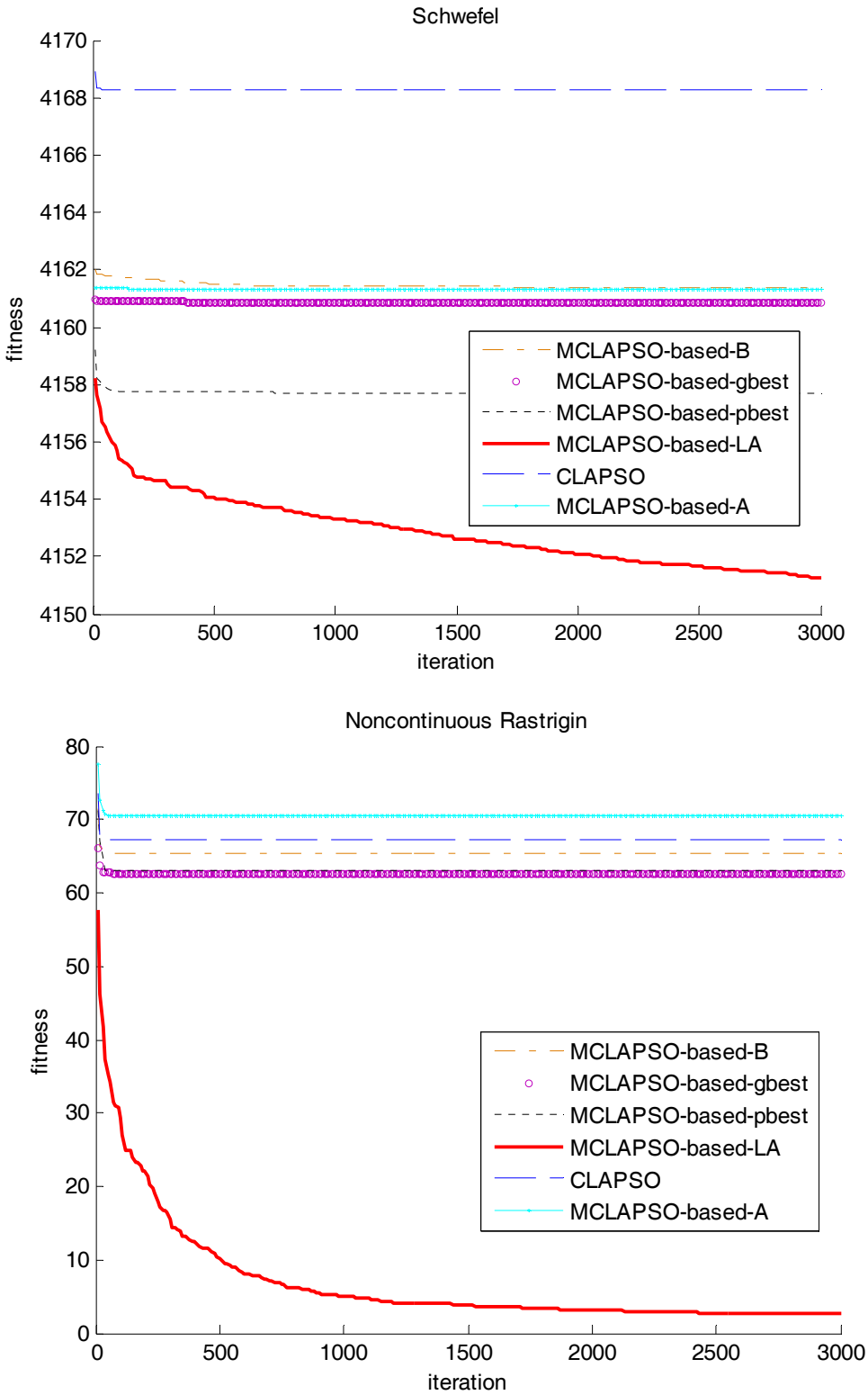


Fig. 1. The mean of best fitness of 20 runs, 3000 iteration in each run, 30 particles on Functions: Sphere, Rosenbrock, Ackley, Greiwank, Weierstrass, Rastrigin, Noncontinuous Rastrigin, Schwefel.

5 Conclusion

In this paper, we proposed an algorithm based CLA-PSO model which hybridize by memetic algorithm. In this way we introduced some algorithms which have different memes or different candidates to apply local search. As experiment results we can found that, some MCLA-PSO algorithms has better performance than CLA-Pso, but MCLA-PSO-based-LA achieved very high performance than others. The learning automatons in each dimension is responsible for adjusting behavior of a specific particle on that dimension of search space and using the schemes, local search performed on gbest and pbest of particles and apply LA to balance between themes, maintaining diversity and largely reduces the probability of getting trap in local optima. So we proposed an adaptive version of Memetic-CLA-PSO that could further enhance the algorithm's performance.

6 References

[1] P. Moscato, "Memetic Algorithms. A short introduction", In D. Corne, M. Dorigo & F. and Glover (Eds.), New ideas in optimization (pp. 219–235). London: McGraw-Hill. 1999)

[2] R. Dawkins, The Selfish Gene. Oxford University Press, New York, 1976.

[3] P. Merz and B. Freisleben, "A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem," in Proceedings of the 1999 International Congress of Evolutionary Computation (CEC'99), Washington DC, USA, 1999, pp. 2063–2070.

[4] Krasnogor N, Smith JE (2005) A tutorial for competent memetic algorithms: model, taxonomy and design issues. IEEE Trans Evol Comput 9(5):474–488

[5] Petalas YG, Parsopoulos KE, Vrahatis MN (2007) Memetic particle swarm optimization. Ann Oper Res 156:99–127

[6] Rao, S. S. (1992). Optimization: theory and applications. New Dehli: Wiley Eastern.

[7] J. E. Smith, "Coevolving Memetic Algorithms: A Review and Progress Report", IEEE Transactions on Systems, vol. 37, no. 1, February. 2007.

[8] J. Smith and T. Fogarty, "Operator and parameter adaptation in genetic algorithms," Soft Comput., vol. 1, no. 2, pp. 81–87, 1997.

[9] R. Hinterding, Z. Michalewicz, and A. Eiben, "Adaptation in evolutionary computation: A survey," in Proc. IEEE Conf. Evol. Comput., Piscataway, NJ, 1997, pp. 65–69.

[10] H. Wang, D. Wang and S. Yang, "A Memetic Algorithm with Adaptive Hill Climbing Strategy for Dynamic Optimization Problems", Soft Computing, vol. 13, pp. 763-780, 2009.

.

.