A New Discrete Binary Particle Swarm Optimization based on Learning Automata

R. Rastegar
Soft Computing Lab
Computer Eng. Department
Amirkabir University
Tehran, Iran
rrastegar@ce.aut.ac.ir

M. R. Meybodi
Soft Computing Lab
Computer Eng. Department
Amirkabir University
Tehran, Iran
meybodi@ce.aut.ac.ir

K. Badie
Information Technology Department
Iran Telecom. Research Center
Tehran, Iran
k_badie@itrc.ac.ir

Abstract: The particle swarm is one of the most powerful methods for solving global optimization problems. This method is an adaptive algorithm based on social-psychological metaphor. A population of particle adapts by returning stochastically toward previously successful regions in the search space and is influenced by the successes of their topological neighbors. In this paper we propose a learning automata based discrete binary particle swarm algorithm. In the proposed algorithm the set of learning automata assigned to a particle may be viewed as the brain of the particle determining its position from its own and other particles past experience. Simulation results show that the proposed algorithm is a good candidate for solving optimization problems.

1. Introduction

Particle Swarm Optimization (PSO) technique was first proposed by Kennedy and Eberhart [6] in 1995. This technique is inspired by choreography of bird flock and can be regarded as a distributed behavior that perform multidimensional search. According to PSO, the behavior of each particle is affected by either the best local or the best global particle to help it fly through a search space. Moreover, a particle can learn from its past experience to adjust its flying speed and direction. Therefore, by observing the behavior of the flock and memorizing their flying histories all particle in swarm can quickly converge to near optimal geographical with a well preserved population density distribution [7]. PSO is considered as an evolutionary computation approach in that it possesses many characteristics that is used by evolutionary algorithms such as, initializing with a population of random solutions, searching for optima by updating generations, the adjustment of particles and evaluating particles by a fitness function. However unlike evolutionary algorithms, the updates of particles are not accomplished by crossover or mutation. The particle swarm algorithms reported in the literatures are classified into two groups: discrete PSO and continuous PSO [10][4][6]. In continuous PSO the particles operate in continuous search space, where the trajectories are defined as changes in position on some number of dimensions. But in discrete PSO the particles operates on discrete search space, and the trajectories are defined as changes in the probability that a coordinate will take on a value from feasible discrete values [9].

Learning Automaton (LA) is a general-purpose stochastic optimization tool, which has been developed as a model for learning systems. They are typically used as the basis of learning systems, which through interactions with a stochastic unknown environment learn the optimal action for that environment. The learning automaton tries to determine, iteratively, the optimal action to apply to environment from a finite number of actions that are available to it. The environment returns a reinforcement signal that shows the relative quality of action of the learning automaton. This signal is given to learning automaton and learning automaton adjusts itself by a learning algorithm [12][14].

In this paper, a new discrete PSO algorithm will be proposed. In the proposed algorithm, learning automata are used by the particles to model the dynamics of the group to which the particles belong. The set of leaning automata associated to a particle, by observing the behavior of the group help the particle in searching for optimal geographical with a well preserved population density distribution. To show the effectiveness of the proposed algorithm we test the algorithm on several function optimization problems. The results of computer simulations show that the proposed algorithm attains better solutions in a faster way for most of the problems.

The rest of the paper is organized as follows; Section 2 describes the Particle Swarm Optimization method. Section 3 briefly reviews the learning automata. Section 4 presents the proposed algorithm and section 5 demonstrates simulations results. The last section is the conclusion.

2. Particle Swarm Optimization

The particle swarm optimization simulates the behaviors of bird flocking. Suppose the following scenario; a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. The effective solution to find food is to follow the bird, which is nearest to the food. PSO learn from the scenario and use it to solve the optimization problems [6]. In PSO, each single solution is a bird in the search space that is called particle. All of the particles have fitness values, which are evaluated by a fitness function to be optimized. Particles have velocities, which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optimal solutions by updating generations. In each iteration, the velocity and the position of each particle *i* is updated using two quantities: the best solution obtained by particle *i* (***LB****i*.) and the global best solution (***GB***) obtained by the group of particles. After finding these two quantities, particle *i* updates its velocity and its position according to the following equations.

$$v_{ij} = v_{ij} + rnd \times c_1 (LB_{ij} - x_{ij}) + rnd \times c_2 (GB_j - x_{ij}) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

where $\boldsymbol{v_i}=(v_{i1}...v_{in})$ is the velocity of particle *i*, $\boldsymbol{x_i}=(x_{i1},..., x_{in})$ is the current position (solution). *rnd* is a random number in the range (0,1). c_1 and c_2 are learning factors. Usually c_1 is equal to c_2. Particles velocities on each dimension are limited to a maximum velocity of V_{max}. If the sum of the accelerations causes the velocity on that dimension to exceed V_{max}, a parameter specified by the user, then the velocity on that dimension is set to V_{max}.

In past several years, PSO has been successfully applied to many research and application areas such as minimax problems [11], binary constraint satisfaction problem [13], constrained nonlinear optimization problems [8], Improvised music [2], optimization in electromagnetic [3], and multimachine power system stabilizer design [1] to mention a few. It is demonstrated that PSO for optimization gets better results in a faster, cheaper way compared to other methods such as genetic algorithm. Another reason that PSO is attractive is that it has fewer parameters to adjust comparing to other methods of optimization [10][4].

The first version of particle swarm algorithm reported by Kennedy in [6] operates in continuous search space. But many optimization problems are set in discrete space. For this reason in [9], Kennedy proposed a discrete binary version of the particle swarm (DPSO). DPSO is obtained by modify equations in continuous PSO to be adapted to discrete binary space. In discrete binary space the search space can be viewed as a hypercube. Viewing the search space as a hypercube, the meanings of the concepts such as trajectory, velocity between and beyond used in continuous version of PSO will be changed. A particle may be seen to move nearer or farther from corners of hypercube by flipping various numbers of bits; in this way, velocity of the particles can be described by the number of bits changed per iteration. A particle with zeros bits changed does not move. A particle moves the farthest if all of its binary coordinates are changed. In this DPSO a dilemma occurs. What is the velocity or rate of change of a single bit or coordinate? Kennedy and Eberhart solved this dilemma by defining velocities and trajectories in term of changes in the probabilities that a bit will be 0 or 1.

For binary discrete search spaces, DPSO updates the velocity according to equation (1) but computes the new position component to be 1 with a probability which is obtained by applying a sigmoid transformation *(1/(1+exp (-v))* to the velocity component [9]. The pseudo code for DPSO is given in figure 1.

```
Repeat
 For each particle i
   Calculate the fitness value
   If the fitness value is better than the best fitness value $LB_i$
      Set current value as the new $LB_i$
   End if
   Choose the particle with the best fitness value and call it GB
  End for
  For each particle i
    Calculate the particle velocity according to equation (1)
    Update the particle position as follows,
         If $rnd<(1/(1+exp(-v_{ij}))$
             $x_{ij}=1$
         Else
             $x_{ij}=0$
         End if
   End for
Until maximum iterations or minimum error criteria is attained
```

Figure 1. Pseudo code of DPSO

3. Learning Automata

Learning Automata are adaptive decision-making devices operating on unknown random environments. The Learning Automaton has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action. Figure 1 illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA) [12][14]. In the following, the variable structure learning automata is described.

A VSLA is a quintuple $<\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{p},T(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{p})>$, where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{p}$ are an action set with *s* actions, an environment response set and the probability set $\boldsymbol{p}$ containing *s* probabilities, each being the probability of performing every action in the current internal automaton state, respectively. The function of *T* is the reinforcement algorithm, which modifies the action probability vector $\boldsymbol{p}$ with respect to the performed action and received response. Let a VSLA operate in an environment with *β={0,1}*. Let $t \in N$ be the set of nonnegative integers. A general linear schema for updating action probabilities can be represented as follows. Let action *i* be performed. If *β(t)=0* (Reward),

$$\begin{aligned} p_i(t+1) &= p_i(t) + a[1-p_i(t)] \\ p_{j\neq i}(t+1) &= (1-a)p_j(t) \end{aligned} \tag{3}$$

If *β(t)=1* (Penalty),

$$\begin{aligned} p_i(t+1) &= (1-b)p_i(t) \\ p_{j\neq i}(t+1) &= (b/s-1) + (1-b)p_j(t) \end{aligned} \tag{4}$$

Where *a* and *b* are reward and penalty parameters. When *a=b*, automaton is called L_{RP}. If *b=0* the automaton is called L_{RI} and if *0<b<<a<1* the automaton is called $L_{R\varepsilon P}$.

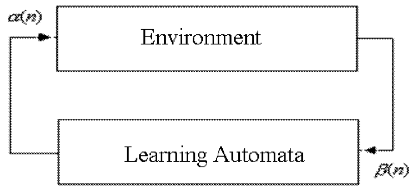


Figure 2. The interaction between learning automata and environment

```
Initialize p to [1/s,1/s,…,1/s] where s is the number of actions
While not done
    Select an action i a sample realization of distribution p
    Evaluate action and return a reinforcement signal β
    Update probability vector according to learning algorithm
End While
```

Figure 3. Pseudocode of variable-structure learning automaton

4. Learning Automata based Discrete Particle Swarm Optimization

In this section, a new DPSO algorithm is proposed. In the proposed algorithm, learning automata are used by the particles to model the dynamics of the group of which the particle is a member. The set of leaning automata associated to a particle by observing the behavior of the group leads the particle to search the optimal geographical with a well preserved population density distribution.

Similar to the DPSO, the velocities and trajectories concepts are defined in terms of changes of probabilities that a bit will become 0 or 1. Instead of using equation (1) and the sigmoid transformation, learning automata are used to determine the position of the particles. In other words, the set of learning automata associated to a particle can be viewed as the components of the brain of the particle. These components collectively lead the particle to search the place where the food can be found with high probability. The number of learning automata assigned to each particle is equal to the dimension of the search space. Each automaton has two actions 0 and 1. The LA based DPSO algorithm works as follows:

While some condition is not reached, iterate the following steps,

1- all the particles do steps I and II simultaneously,

I -every automaton associated to particle *i* chooses one of their actions (0 or 1) according to their action probability vectors.

II- the particle *i* generates a new position (a corner of the hypercube) by (concatenating) combining the actions chosen by its set of learning automata. The particle then moves to that position. If the fitness value of new position is better than the best fitness value LB_i, LB_i will be set to the new position.

2- the position of the particle with the best fitness, *GB,* is computed.

3-All the particles perform the following simultaneously

- Based on *GB*, LB_i and the position of the particle, each particle *i* generates a reinforcement vector $\beta_i=(\beta_{i1},\ldots,\beta_{in})$ which becomes the input to the set of learning automata associated to particle *i.*

The *j*th element of the reinforcement vector for particle *i*, β_{ij}, *is* computed as follows,

$$\beta_{ij} = \begin{cases} 0 & if\ \ LB_{ij} = GB_{ij} = x_{ij} \\ 1 & otherwise \end{cases} \tag{5}$$

The reinforcement vector β_i will be used to update the action probabilities vectors of the learning automata associated to particle *i.*

The set of learning automata associated to a particle helps the particle to gradually move to a position that confirms with the global goal of the group of the particles. This happens by updating the actions probabilities of the set of learning automata associated to the particle in such a manner that the group goal will be achieved.

5. Simulations

This section presents simulation results and compares the LAPSO with DPSO, in terms of solution quality, the number of function evaluations taken and the speed of finding the best solution for a given population size. Each quantity of the results reported is the average taken over 20 runs. The parameters used for DPSO are the same as those used in [9], i.e. V_{max} is set to 6, and c1 and c2 are equal to 1. The population size varies from 2 to 50 with increments of two. The proposed Algorithm is tested for different learning algorithms: L_{RI}, and L_{RP}. For the sake of convenience in presentation, we use LA(*automata)*-PSO to refer to the LA-PSO algorithm when it uses Learning automata *automata.* The algorithm terminates when the number of iterations succeeds 500. The proposed algorithms are tested on five different standard functions to be minimized. These functions are given below are and are taken from De Jong's dissertation [5].

$$F1(X) = \sum_{i=1}^{3} x_i^2 \quad -5.12 \leq x_i \leq 5.12$$

$$F2(X) = 100\,(x_1^2 - x_2)^2 + (1 - x_1)^2 \quad -2.048 \leq x_i \leq 2.048$$

$$F4(X) = \sum_{i=1}^{30} i x_i^4 + Gauss(0,1) \quad -1.28 \leq x_i \leq 1.28$$

$$F_5(X) = (0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1} \quad -65.536 \leq x_i \leq 65.536$$

Experiment 1: In experiment 1, we compare the proposed algorithms with respect to the quality of the solution produced and the number of function evaluations needed when the algorithms use different population sizes or learning algorithms. Experimentation has also been conducted to study the effect of the parameters of the learning algorithm on the effectiveness of the proposed algorithms. By careful inspection of the results reported in Figures 4 through 6, it is found that as the number of particles increase, the quality of the solution produced and the number of function evaluations needed by the algorithm increases. Also, it has been noted that, better solutions are obtained when L_{RP} automata with penalty and reward parameters near zero (i.e. 0.01) is used.

Experiment 2: In this experiment algorithms LA(L_{RI})-PSO with *a=0.01* and LA(L_{RP})PSO with *a=b=0.01* are used and compared with DPSO with respect to the quality of solution produced and the number of function evaluations needed for different population size (Figure 7 through 10). For all the problems except problem F2, algorithm LA(L_{RI})-PSO obtains the worst result with respect to the quality of the solution produced. For problem F2, both LA(L_{RI})-PSO and LA(L_{RP})-PSO algorithms perform better than DPSO. For problems F1, F4 and F5, algorithm LA(L_{RP})-PSO performs nearly the same as DPSO with respect to the quality of solution, but the number of function evaluations needed by DPSO is higher.

6. Conclusion

In this paper a learning automata based discrete particle swarm optimization algorithm (LA-PSO) was proposed. In the proposed algorithm the learning automata is used to determine the position of a particle that confirms with the global goal of the group of the particles. This happens by updating the actions probabilities of the set of learning automata associated to the particle in such a manner that the group goal will be achieved. Simulation results showed the effectiveness of the proposed algorithm in solving the optimization problems.

References

[1] Abido, M. A., "Particle swarm optimization for multimachine power system stabilizer design", Power Engineering Society Summer Meeting, Vol. 3, PP. 1346-1359, 2001.

[2] Blackwell, T. and Bentley, P. J., "Improvised music with swarms", Proceedings of the IEEE Congress on Evolutionary Computation 2002 Honolulu, Hawaii USA, 2002b.

[3] Ciuprina, G., Ioan, D., and Munteanu, I., "Use of intelligent-particle swarms optimization in electromagnetics", IEEE Transactions on Magnetics, Vol. 38, No. 2, PP. 1037-1040. 2002.

[4] Clerc, M., and Kennedy, J., "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space", IEEE Transaction on Evolutionary Computation, Vol. 6, No. 1, PP. 58-73, February 2002.

[5] De Jong, K. A., "The Analysis of the behavior of a class of genetic adaptive systems" Ph.D. dissertation, University of Michigan, Ann Arbor, 1975.

[6] Eberhart, R. C. and Kennedy, J., "Particle Swarm Optimization", in Proceedings of IEEE International Conference on Neural Networks, Vol 4, PP. 1942-1948, IEEE Service Center, Piscataway, NJ, 1995.

[7] Gary, G, Y., and Haiming, Lu,. "Dynamical Population Strategy assisted Particle Swarm Optimization", Proceedings of the 2003 IEEE International Symposium on Intelligent Control, Houston, Texas, PP. 697-702, October 2003.

[8] Hu, X., and Eberhart, R. C., "Solving constrained nonlinear optimization problems with particle swarm optimization", Proceedings of the Sixth World Multiconference on Systemic, Cybernetics and Informatics 2002 (SCI 2002), Orlando, USA, 2002.

[9] Kennedy, J., and Eberhart, R. C., "A Discrete Binary Version of The Particle Swarm Algorithm", in Proceedings of Conference on Systems, Man, and Cybernetics, PP. 4104-4108, IEEE Service Center, Piscataway, NJ, 1997.

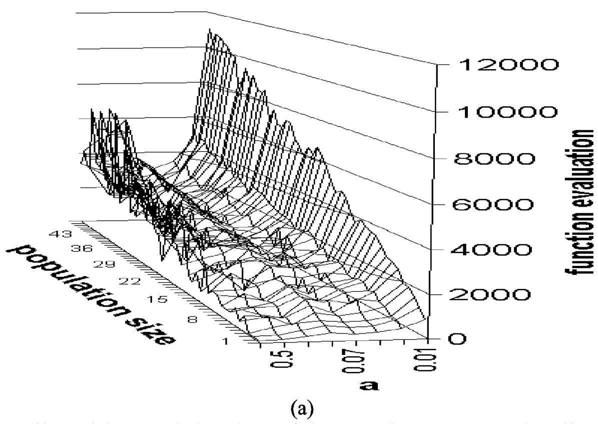
[10] Kennedy, J., "The Particle Swarm: Social Adaptation of Knowledge", in Proceedings of IEEE International Conference on Neural Networks (Indianapolis, Indiana), pp. 303-308, IEEE Service Center, Piscataway, NJ, 1997.

[11] Laskari, E. C., Parsopoulos, K. E., and Vrahatis, M. N., "Particle swarm optimization for minimax Problems", Proceedings of the IEEE Congress on Evolutionary computation 2002 Honolulu, Hawaii USA, 2002b.

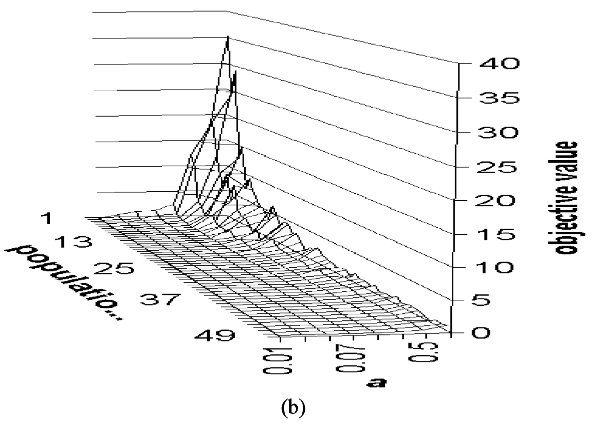
[12] Narendra, K. S., and Thathachar, M. A. L., *Learning Automata: An Introduction*, Printice-Hall Inc, 1989.

[13] Schoofs, L., and Naudts, B., "Swarm Intelligence on the Binary Constraint Satisfaction problem", Proceedings of the IEEE Congress on Evolutionary Computation 2002 Honolulu, Hawaii USA, 2002.

[14] Thathachar, M. A. L., Sastry, P. S., "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 32, No. 6, PP. 711-722, 2002.

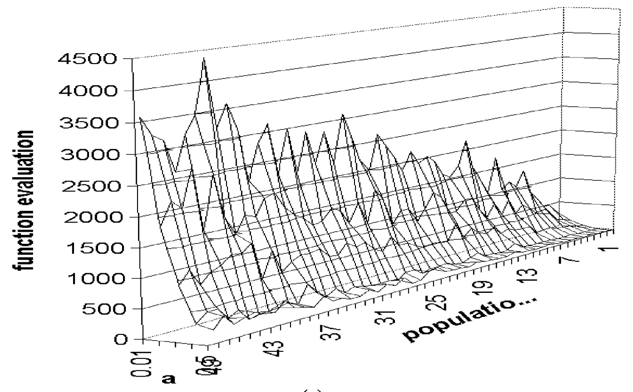


(a)

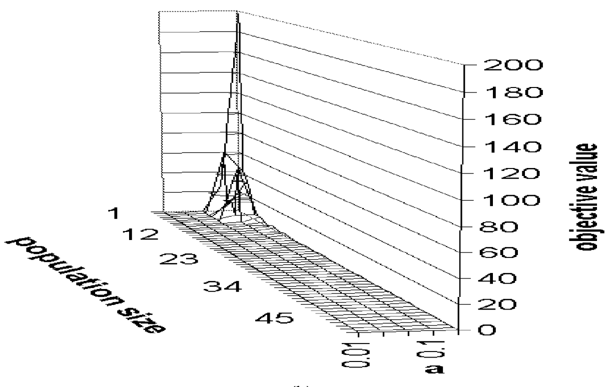


(b)

Figure 4. Effect of the population size and the reward parameter on the effectiveness of LA(L_{RI})-PSO for function F1 (a) the number of function evaluations taken in 500 iterations to obtain the best solution. (b) the average of the best solutions obtained.

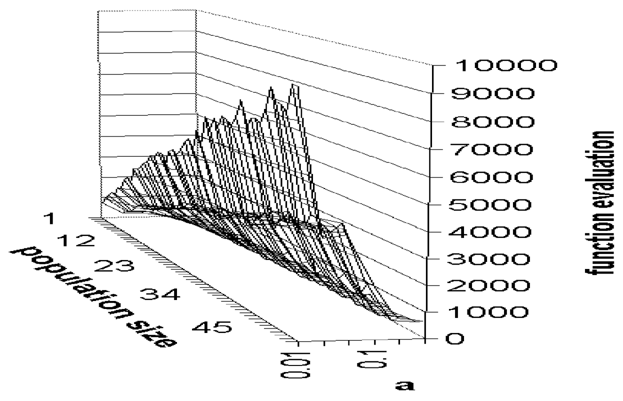


(a)

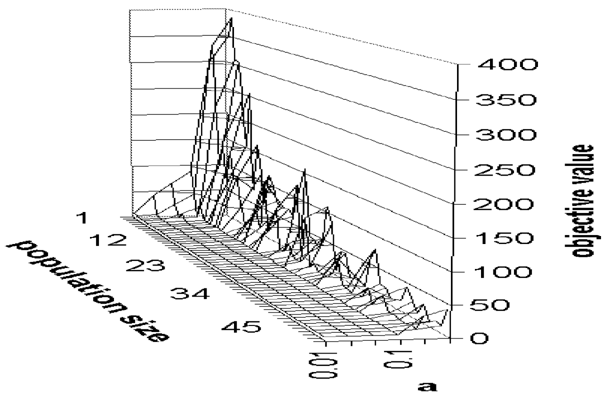


(b)

Figure 5. Effect of the population size and the reward parameter on the effectiveness of LA(L_{RP})-PSO for function F2 (a) the number of function evaluations taken in 500 iterations to obtain the best solution. (b) the average of the best solutions obtained.



(a)



(b)

Figure 6. Effect of the population size and the reward parameter on the effectiveness of LA(L_{RP})-PSO for function F5 (a) the number of function evaluations taken in 500 iterations to obtain the best solution. (b) the average of the best solutions obtained.

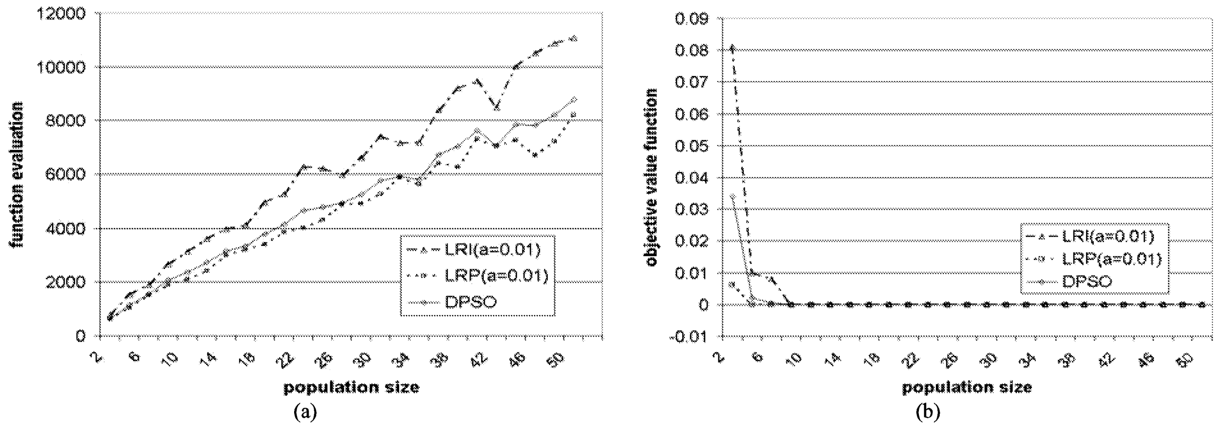


Figure 7. Comparison of the DPSO, LA(L_{RI})-PSO, and LA(L_{RP})-PSO algorithms for function F1 (a) the number of function evaluations taken to obtain the best solution in 500 iterations. (b) the solution quality obtained at the end of the runs.

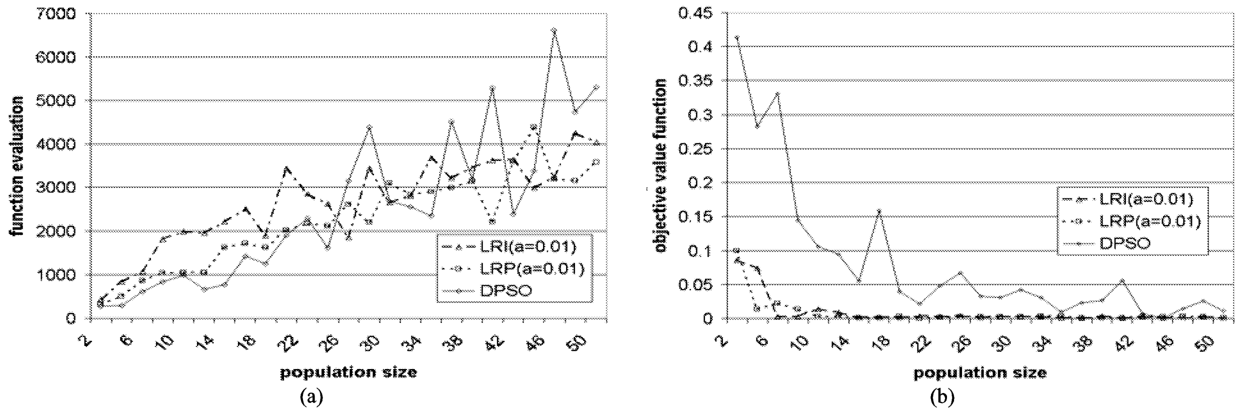


Figure 8. Comparison of the DPSO, LA(L_{RI})-PSO, and LA(L_{RP})-PSO algorithms for function F2 (a) the number of function evaluations taken to obtain the best solution in 500 iterations. (b) the solution quality obtained at the end of the runs.

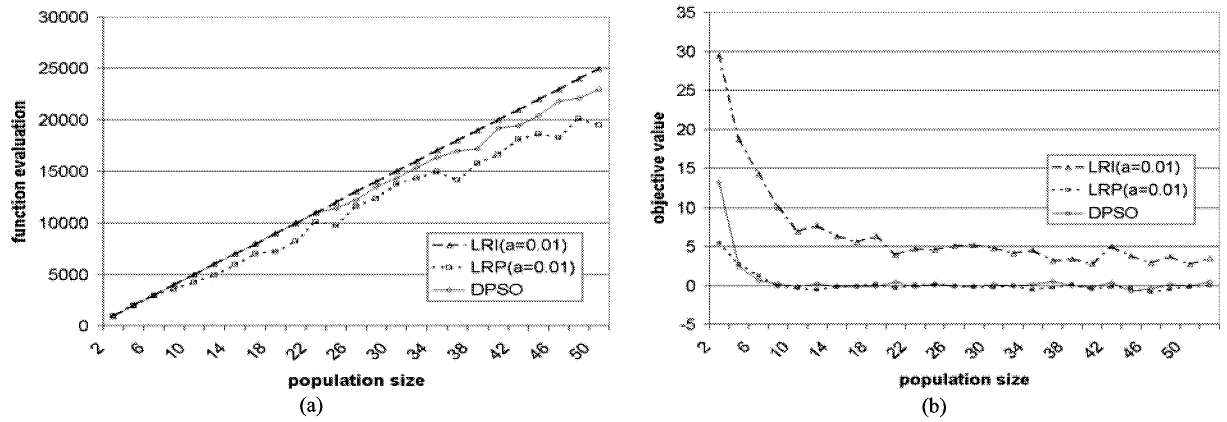
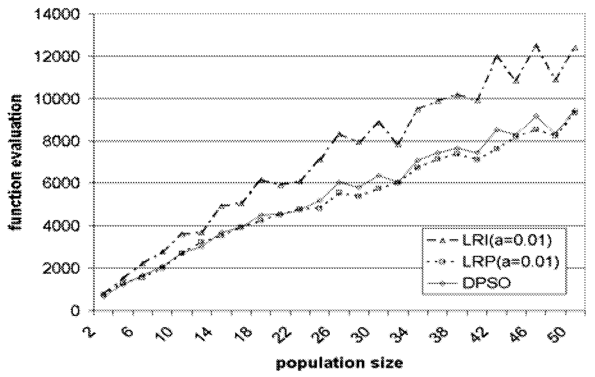
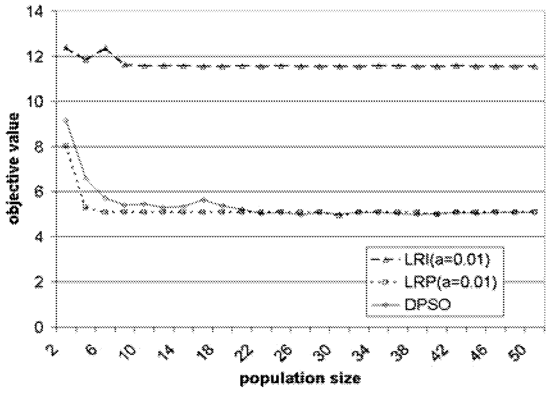


Figure 9. Comparison of the DPSO, LA(L_{RI})-PSO, and LA(L_{RP})-PSO algorithms for function F4 (a) the number of function evaluations taken to obtain the best solution in 500 iterations. (b) the solution quality obtained at the end of the runs.



(a)



(b)

Figure 10. Comparison of the DPSO, LA(L_{RI})-PSO, and LA(L_{RP})-PSO algorithms for function F5 (a) the number of function evaluations taken to obtain the best solution in 500 iterations. (b) the solution quality obtained at the end of the runs.