International Journal of Intelligence Science, **2011, 1, 1-7**
doi:10.4236/ijis.2011.11001 Published Online July 2011 (http://www.SciRP.org/journal/ijis)

Scientific Research

Using Intelligent Computational Methods for Optimizing Niching Method

Mohsen Jahanshahi
Computer Engineering Department, Central Tehran Branch, Islamic Azad University, Tehran, Iran
E-mail: *mjahanshahi@iauctb.ac.ir*
Received June 22, 2011; *revised July* 18, 2011; *accepted July* 25, 2011

Abstract

Optimization implies the minimization or maximization of an objective function. Some problems have several optimum points which all, should be computed. Niching method is presented to do so. However, its efficiency can be improved via combining it with Memetic algorithm. Therefore, in this paper, Memetic method is used to improve this method in terms of convergence rate and diversity. In the proposed methods, genetic algorithm, PSO, and learning automata are used as a local search algorithm of Memetic method. The result of simulations demonstrates that proposed methods are more effective compared with Niching in terms of convergence and diversity.

Keywords: PSO, Niching, Genetic Algorithm, Learning Automata

1. Introduction

Optimization is the minimization or maximization of an objective function that normally is done with consideration of limitations identifying conditions of a problem. In other words, it means the finding of the best solution for a given problem. Some problems have several local optimums which may be computed using Niching method [1]. In this paper, memetic method is used for increase of convergence speed and also the increasing rate of particles' diversity in Niching method, as two assessment factors of search methods. In this research, genetic algorithms (GA) [2], particle swarm optimization (PSO) [3], and learning automata (LA) [4] are used as three local search algorithms in memetic method.

The rest of the paper is organized as follows. In section 2, PSO method is briefly introduced. In section 3, NichePSO is discussed in summary. Section 4 explains used local search methods and then proposed methods are discussed in section 5. The results of simulations are included in section 6. Section 7 concludes the manuscript.

2. Particle Swarm Optimization

Particle swarm optimization (PSO) was discussed in [3] and then has been improved for many years. It is combined by GA in [5] and modified in [6-8] work on discrete PSO as a new direction. PSO has been successfully utilized for aircraft transportation [9] and optimal path discovery in automated drilling operations [10], to name a few. There are also some other works which concentrated on its functionality which are not in the scope of this manuscript [11,12].

The main idea of PSO had been taken from Birds or fishes swarm behavior which are searching for meal. Some birds are searching for meal randomly. Only a piece of meal may be found in the mentioned space. None of them knows about the real place of the meal. One of the best strategies is to follow a bird that is placed in minimum distance to a meal. In fact, this strategy is basis of PSO algorithm.

This method is an effective technique for solution of optimization problems based on a swarm behavior. In this method, each member of a swarm is called a particle who attempts to achieve a final solution with adjustment of its route and movement to the best personal and swarm experiences. In PSO algorithm, a solution is called a particle the same as a bird in swarm scheme. Each particle is described using a quality factor is given by a fitness function. The more nearness to the goal, the more qualification is obtained. Each particle also moves with a specified speed which conducts its movement. Each particle that follows the optimum particles in cur-

rent position will continue its movement in the space of problem.

PSO starts in this way that a swarm of particles (solutions) are generated randomly and are updated during the generations and attempt to find an optimum solution. In each step, every particle is updated using two best values. First, is the best situation that a particle has ever been reached. It is known and kept by personal best (*pbest*). Another best value is used by the algorithm is the best position that has ever been obtained by a swarm of particles. It is known by global best (*gbest*). In some PSO editions, a particle chooses parts of populations that are its topological neighbors and only involves those in its behavior. In this case, the best local solution is shown by *lbest* (local best) and is used instead of *gbest*. After finding the best values, the velocity and position of particles are updated by formula (1) and (2) given below:

$$v[\,] = v[\,] + c1 * rand(\,) * (pbest[\,] - position[\,]) + c2 * rand(\,) * (gbest[\,] - position[\,]) \quad (1)$$

$$position[\,] = position[\,] + V[\,] \quad (2)$$

In (1) and (2) equations, $V[\,]$ is the particle velocity and $position[\,]$ is current position of a particle. Both are arrays as long as the number of problem dimensions. $Rand(\,)$ is a random variable in the random domain. (0, 1) $c1$ and $c2$ are learning parameters, normally both are the same values as $c1 = c2 = c$. In each dimension, velocity of particles is limited to a *V*max. In case, the sum of accelerations cause that the velocity in one dimension exceeds the maximum value, it is considered as *V*max. The right hand side of Equation (1) is composed of three components, the first part, is the current velocity of a particle, the second and third parts take responsibility for variation of the velocity and its rotation towards the best personal and swarm experiences. Combining these two factors in Equation (1) helps create a balance between local and global searches. Let us we don't consider the first part of the equation then the particles velocity is determined only with consideration of current position and the best single and swarm experiences of particles. Therefore, the best particle is fixed in its position and the rest of the particles move towards it. In fact, if we ignore the first part of Equation (1), PSO will be a process in which, search space gradually becomes smaller and local search is performed around the best particle. In contrast, if we consider only first part of the Equation (1), then the particles will continue their normal route up to border and it is said those are doing global search. Pseudo code of PSO algorithm is shown in **Figure 1**.

Since in this algorithm, particles gradually tend to current optimum solution so if this is a local optimum solution then whole particles move towards it consequently PSO is not a practical way to leave this local optimization. Meanwhile, some problems have more than one general optimum solution that all have to be computed. These are the greatest problems of PSO algorithm that make it unable to solve multi peak problems particularly with a large state space.

```
For each particle
   Initialize particle
End For
Do
   For each particle
      Calculate fitness value of the particle f_p
      /*updating particle's best fitness value so far)*/
      If f_p is better than pBest
         set current value as the new pBest
   End For
   /*updating population's best fitness value so far)*/
   Set gBest to the best fitness value of all particle
   For each particle
      Calculate particle velocity according Equation (1)
      Update particle position according Equation (2)
   End For
While maximum iterations OR
      minimum error criteria is not attained
```

Figure 1. Pseudo code of PSO algorithm.

3. Niche PSO

Niche PSO is presented to find all the solutions of problems with more than one general optimum solution [1]. In this algorithm, niches are parts of the environment and main operation of each niche is to self-organize the particles to independent sub-swarm. Each sub-swarm determines the position of one niche and keeps it. The task of each sub-swarm is to find one of the optimum solutions. No information is exchanged amongst sub-swarms. This independency lets sub-swarms to keep niches. In summary, it can be said that performance of sub-swarms is stable and independent of the other swarms.

Niche PSO begins its operation with one swarm that is called main swarm that includes whole particles. As soon as, a particle gets close to an optimum solution, one swarm will be formed with classification of particles. Then these particles are put out of the main swarm and continue the operations for finding the solution in their own sub swarms. In fact the main swarm is broken into some sub-swarms. Niche PSO is convergent when smaller sub-swarms improve their presented solutions, and then in continue, the best global position for each sub-swarms is accepted as a solution. Pseudo code for Niche PSO is shown in **Figure 2**. Different steps of this algorithm are explained in details in section below.

3.1. Training Main Swarm

Here, congnition-model (PSO) is used to update velocity

Create and initialize a n_x *– demensioal main swarm, S*;
Repeat
Train the main swarm, S, for one iteration using the cognition-only model;
Update the fitness of each main swarm particle, $S \cdot x_i$;
for *each sub-swarm* S_k *do*
Train sub-swarm particle, $S_k \cdot x_i$ *, using a full model PSO*;
Update each particle's fitness;
Update the swarm radius $S_k \cdot R$;
End for
If possible, merge sub-swarms;
Allow sub-swarms to absorb any particle from the main swarm that moved in to the sub-swarm;
If possible, create new sub-swarm;
Until *stopping condition is true*;
Return $S_k \cdot \hat{y}$ *for each sub-swarm* S_k *as a solution*;

Figure 2. Niche PSO algorithm.

and position of the particles.

3.2. Training Sub-Swarms

Sub-swarms are independent swarms. In this paper, for training those, local search methods are used such as genetic algorithms, memetic and PSO. Use of methods above for training sub-swarms improves this method.

3.3. Identification of Niches

When a particle is getting close to a local optimum, a swarm is formed. If the acceptability rate of a particle indicates tiny variation of some repetitions, then the swarm will be formed by this particle and its nearest neighbors. In simpler word, standard deviation (σ_i) occurs in some repetition in objective function ($f(x_i)$) for each particle. If ($\sigma_i < \varepsilon$), the swarm is formed. To prevent the dependency problem, (σ_i) is normalized based on domain. The nearest neighbor of for position of (x_i) of particle *i*, is given by Euclidean distance as follow:

$$l = \arg\min_a \left\{ \left\| x_i - x_a \right\| \right\} \tag{3}$$

The process of swarm generation or niche identification is summarized in **Figure 3**. In this algorithm, the symbol Q indicates set of sub-swarms ($Q = \{S_1, \cdots, S_K\}$) where, ($|Q| = K$). Each sub-swarm has the number of ($S_k \cdot n_s$) particles. At the time of Niche PSO generation, the value of *K* is zero and *Q* is an empty set.

3.4. Absorbing Particles in Sub-Swarms

Particles of main swarm move towards an area that is covered by the sub-swarms (S_k). These particles combine with sub-swarm for reasons below:

if $\sigma_i < \varepsilon$ then
$k = k + 1$
create sub-swarm $S_k = \{x_i, x_l\}$;
Let $Q \leftarrow Q \cup S_k$;
Let $S \leftarrow S / S_k$;

Figure 3. Generation algorithm of sub-swarms in Niche PSO.

- The particles move in search area for creation of a sub-swarm may improve the diversity of sub-swarms.
- These particles in a sub-swarm increase the dispersion of those to an optimum space via increase of general information.

If for particle *i* we have:

$$\left\| x_i - S_k \cdot \hat{y} \right\| \le S_k \cdot R \tag{4}$$

Then the particle *i* is absorbed to sub-swarm (S_k) in such a way

$$\begin{aligned} S_k &\leftarrow S_k \cup \{x_i\} \\ S &\leftarrow S / \{x_i\} \end{aligned} \tag{5}$$

In equation above, ($S_k \cdot R$) points to radius of sub-swarm (S_k) and is defined as follow:

$$S_k \cdot R = \max\left\{ \left\| S_k . \hat{y} - S_k . x_i \right\| \right\} \qquad \forall_i = 1, \cdots, S_k \cdot n_s \tag{6}$$

Here, ($S_k \cdot \hat{y}$) is the best general position of sub-swarm (S_k).

3.5. Merging of Sub-Swarms

Perhaps, more than one sub-swarms points to an optimum point. Here, it is possible that a particle moves to a solution is not absorbed to a sub-swarms. As a result, a new sub-swarm will be generated and it leads to a problem that a solution is followed by several sub-swarms in form of redundancy. For solving this problem, similar sub-swarms must merge together. The sub-swarms are the same if their space is considered in such a way that radiuses of particles converge in sub-swarms. The new merged sub-swarm has more general information and uses experiences of both old sub-swarms. The results of new sub-swarm are normally more accurate than the smaller old sub-swarm. In simpler word, two sub-swarms (S_k) and (S_{k+1}) merge together if:

$$\left\| S_{k1} \cdot \hat{y} - S_{k2} \cdot \hat{y} \right\| < \left(S_{k1} \cdot R + S_{k2} \cdot R \right) \tag{7}$$

If ($S_{k1} \cdot R = S_{k2} \cdot R = 0$) the following formula will be replaced with equation 8.

$$\left\| S_{k1} \cdot \hat{y} - S_{k2} \cdot \hat{y} \right\| < \mu \tag{8}$$

where μ is a tiny value tends to zero (e.g. $\mu = 10^{-3}$). If μ is a very large value, perhaps unsuitable sub-swarms merge together and lead to failure of finding some solu-

tions. In order to keep μ in the domain, ($\|S_{k1}\cdot\hat{y}-S_{k2}\cdot\hat{y}\|$) is normalized in (0, 1).

3.6. Stop Conditions

Several stop conditions may be used to finish the search of solutions. Note that each sub-swarm has founded a unique solution.

4. Used Local Search Method

We applied the PSO algorithm as a local search method in Memetic which was introduced in section 2. We have also utilized genetic algorithm and learning automata which are briefly introduced in the next sub-section.

4.1. Genetic Search Algorithm

Genetic algorithm was introduced by John Haland for the first time in 1975 and since then has been used for solving optimization problems [2]. Genetic algorithms have great applications in random searches and optimization techniques. Today's those are known more as type of evolutional calculations. This algorithm is similar to natural evolution process and unlike the other methods, uses a population for search in solution space and always applies the principle "survival based on eligibility" to population. Based on this principle, after creation of every generation, unqualified chromosomes will be killed and eliminated from populations, only suitable chromosomes will be remained and create next generation and make suitable solutions from those.

If genetic algorithm is designed well, all the population will converge to a unique global optimum solution. Genetic algorithms are so powerful and effective and practical for problems with no systematic solutions.

4.2. Theory of Learning Automata

In this section the Learning Automata for the proposed framework will be briefly reviewed;

Learning automata (LA) is an abstract model that chooses an action from a finite set of its actions randomly and takes it [4,13]. In this case, environment evaluates this taken action and responses by a reinforcement signal. Then, learning automata updates its internal information regarding both the taken action and received reinforcement signal. After that, learning automata chooses another action again. **Figure 4** depicts the relationship between learning automata and environment. Every environment is represented by $E=\{\alpha,\beta,c\}$, where $\alpha=\{\alpha_1,\alpha_2,\cdots,\alpha_r\}$ is a set of inputs, $\beta=\{\beta_1,\beta_2,\cdots,\beta_r\}$ is a set of outputs, and $c=\{c_1,c_2,\cdots,c_r\}$ is a set of penalty probabilities. Whenever set β has just two members, model of environment is *P*-model. In this environment $\beta_1=1$, $\beta_2=0$ are considered as penalty and reward respectively. Similarly, *Q*-model of environment contains a finite set of members. Also, *S*-model of environment has infinite number of members. c_i is the penalty probability of taken action α_i.

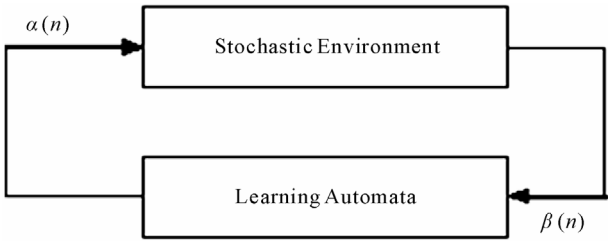

Figure 4. Interaction between learning automata and environment.

Learning automata is classified into fixed structure and variable structure. Learning automata with variable structure is introduced as follows; Learning automata with variable structure is represented by $\{\alpha,\beta,p,T\}$, where $\alpha=\{\alpha_1,\alpha_2,\cdots,\alpha_r\}$ is a set of actions, $\beta=\{\beta_1,\beta_2,\cdots,\beta_r\}$ is a set of inputs, $p=\{p_1,p_2,\cdots,p_r\}$ is the action probability vector, and $p(n+1)=T[\alpha(n),\beta(n),p(n)]$ is learning algorithm. Learning automata operates as follows; learning automata chooses an action from its probability vector randomly (P_i) and takes it. Suppose that the chosen action is α_i. Learning automata after receiving reinforcement signal from environment updates its action probability vector according to formulas 9 and 10 in case of desirable and undesirable received signals respectively. In formulas 9 and 10, *a* and *b* are reward and penalty parameters respectively. If *a* = *b* then algorithm is named L_{R-P}. Also, if $b \square a$ then the algorithm is named $L_{R\varepsilon P}$. Similarly, if *b* = 0 then the algorithm is called L_{R-I}.

$$\begin{aligned} &p_i(n+1)=p_i(n)+a\cdot(1-p_i(n))\\ &p_j(n+1)=p_j(n)-a\cdot p_j(n) \qquad (9)\\ &\forall j\ \ j\neq i \end{aligned}$$

$$\begin{aligned} &p_i(n+1)=(1-b)\cdot p_i(n)\\ &p_j(n+1)=\frac{b}{r-1}+(1-b)\,p_j(n) \qquad (10)\\ &\forall j\ \ j\neq i \end{aligned}$$

5. Proposed Methods

In this paper, to make Niche PSO, more effective, memetic method has been used for training the particles in each sub-swarm. This paper introduces three different methods resulting from combination of intelligent search

methods by Niche PSO for improvement of this method. In proposed methods in each sub-swarm, a set of particles is chosen and improved by one of local search methods. Improved particles are added to set of whole particles and from those, enough sub-swarms will be chosen. The structure of memetic method and proposed method are shown in order, in **Figures 5** and **6**.

For training the particles inside each niche, genetic algorithm by training single particle, genetic by training several particles and PSO have been used.

In continue, details of used algorithms are presented as a local search in memetic method.

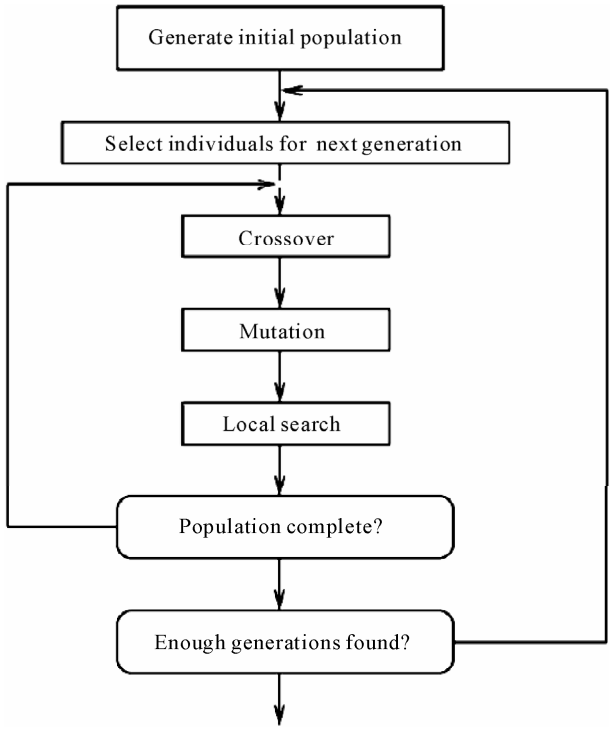

Figure 5. Memetic algorithm.

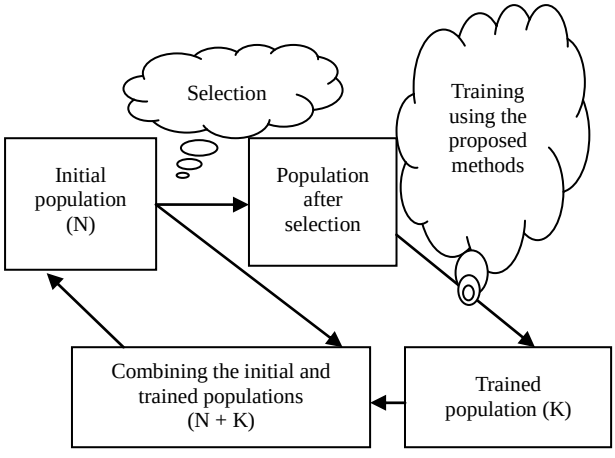

Figure 6. Proposed framework.

5.1. GA-Based Local Search Method

As described above, we have used GA as local search in Memetic method. Mutation operator of GA was done as follows. The best particle of sub-swarm is chosen and its value is converted to a binary number. Then Mutation is applied to it once, if the value of new particle is getting better the old one, new particle will be replaced with the old one. We name this method as GA1.

We also propose another GA-based method in which the Mutation operator was accomplished as follows. In each sub-swarm, one quarter of the particles are chosen randomly. Then single point Mutation is applied to each particle up to maximum three times. Each time if the generated particle is better than the old one, it will be replaced with the old one and next go to another particle. We call this method as GA2

5.2. PSO-Based Local Search Method

In this method, PSO algorithm is used as local search. This method is done in two ways: one global best (all the swarms) and another local best (inside the sub-swarm) that by global best method poor results are obtained.

5.3. LA (L_{R-P})-Based Method

This method uses learning automata to update the velocity of particles. In this method two functions are considered to update particles velocity by learning automata. The first operation means to consider the effect of global best of subgroup for updating a particle and the second operation means to update velocity of the particle without consideration of the global best of subgroup. After applying these functions to whole particles of subgroup if the total values of particles is less than the old total value of the particles then automata is rewarded otherwise is penalized.

5.4. LA (L_{R-I})-Based Method

This method is similar to method above. The only difference is that after doing Automata functions on whole particles of subgroup, if the total values of current particles is less than the old particles then the automata is rewarded but if the total value of particles is more than old total values the automata is not penalized.

6. Simulation Results

In this section, the results of simulations are presented compared with original Niche PSO method to be run in Matlab 7.04. The aim of running all the methods in all

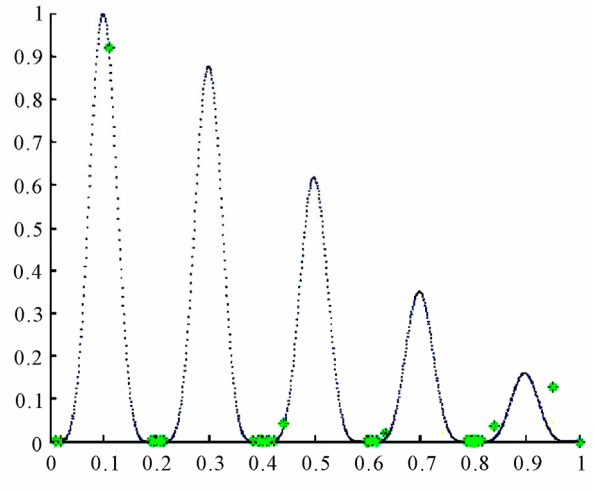

Figure 7. Approaching the particles to the local optima.

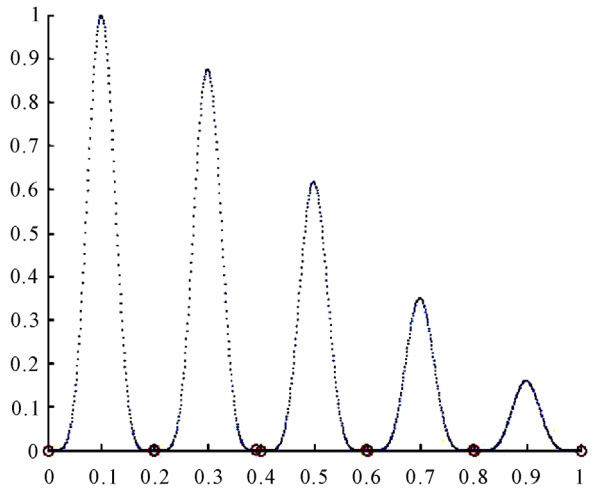

Figure 8. Forming sub-swarms.

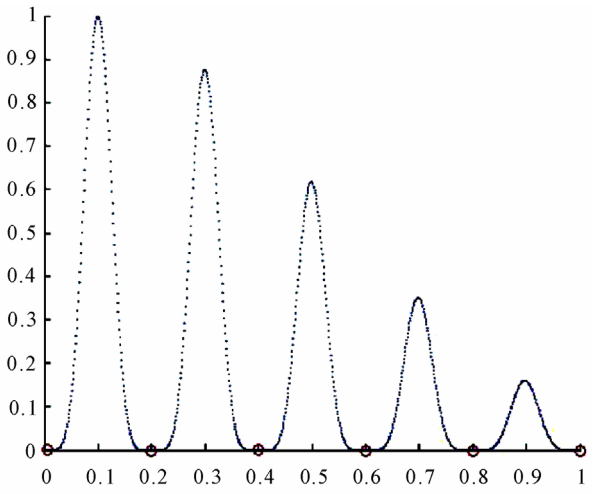

Figure 9. Full running of algorithm and combination of sub-swarms and reaching all the optimum solutions.

simulations is to find optimum points of a following standard function.

$$y = \left(\exp\left(\left(-2*\log 2(2)\right)*\left(\left(x-0.08\right)/\left(0.854\right)\right)\wedge 2\right)\right) * \left(\sin\left(5*pi*x\right)\right)\wedge 6 \quad (9)$$

In all simulations, PSO parameters are initially valued as follow:

$pop = 40$	$c1 = 1.2$	$a = 0.01$	$r1 = a + (b-a)*rand$
	$c2 = 1.2$	$b = 1$	$r2 = a + (b-a)*rand$

Also the value of inertia weight w is given by the following formula where, T is the maximum repetition and t is current time of simulation.

$$w = 0.5 - \left(0.3 * \frac{1-t}{1-T}\right) \quad (10)$$

Figures 7 to **9** shows the result of a sample of running algorithm and the process of reaching optimum solution for particles during simulation.

Figure 10, depicts the diversity rate of particles during different simulations. This results are gained from 10 times repetition of an algorithm that its accurate results is listed in **Table 1** in which the values are gained by taking the mean of variance for whole particles during the period of each step of simulation. As depicted in this figure, the method LA (L_{R-P})-based has the most diversity rate and hence outperforms the others schemes. In this evaluation, LA (L_{R-I})-based method stands in the second place. **Table 2** lists the divergences behavior of the proposed designs over the 50 runs. From the results, LA (L_{R-I})-based method has the least divergence rate and hence outperforms the other methods in terms of convergence rate.

7. Conclusions and Future works

In this paper, five combinational methods were presented

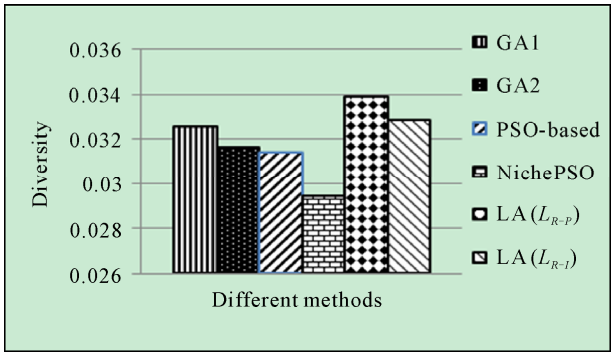

Figure 10. Diversity of particles in various algorithms.

Table 1. Diversity rate of particles in various algorithms in 10 times of simulations.

GA1	GA2	PSO-based	Niche PSO	LA (L_{R-P})	LA (L_{R-I})
0.0391	0.0309	0.0302	0.0296	0.0331	0.0331
0.0363	0.032	0.0359	0.0284	0.0315	0.0315
0.0311	0.0377	0.0319	0.0321	0.0397	0.0315
0.0299	0.0269	0.031	0.0357	0.0314	0.0314
0.0299	0.0297	0.0315	0.0265	0.0391	0.0329
0.0413	0.0316	0.0267	0.0268	0.0301	0.0301
0.0292	0.0384	0.0329	0.0256	0.0336	0.0336
0.0292	0.0327	0.0267	0.0272	0.0311	0.0397
0.0285	0.0279	0.0272	0.0292	0.0299	0.0329
0.0308	0.0283	0.0397	0.0329	0.0391	0.0317

Table 2. Divergence rate of different algorithms.

LA (L_{R-I})	LA (L_{R-P})	Niche PSO	PSO-based	GA2	GA1
5	11	12	15	8	12

to raise the diversity of particles and improvement of convergence of NichePSO as two assessment factors of search methods. From the results, LA (L_{R-P})- and LA (L_{R-I})-based methods outperform other methods in terms of diversity and convergence rates, respectively.

8. References

[1] R. Brits, A. P. Engelbrecht and F. van den Bergh, "A Niching Particle Swarm Optimizer," *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, 24 - 26 April 2003, pp. 54-59,

[2] P. Moscato, "Genetic Algorithms and Martial Arts towards Memetic Algorithm," *Calteth Concurrent Computation Program Report* 826, Moscato, 1989.

[3] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, 27 November - 1 December 1995, pp. 1942-1948.

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

[5] A. Stacey, M. Jancic and I. Grundy, "Particle Swarm Optimization with Mutation," *Proceedings of the Congress on Evolutionary Computation*, Canberra, 8 - 12 December 2003, pp. 1425-1430.

[6] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer," *IEEE International conference on Evolutionary Computation*, Anchorage, 4 - 9 May 1998.

[7] M. Clerc, "Discrete Particle Swarm Optimization," New Optimization Techniques in Engineering, Springer-Verlag, New York, 2004.

[8] P. Yin, "A Discrete Particle Swarm Algorithm for Optimal Polygonal Approximation of Digital Curves," *Journal of Visual Communication and Image Representation*, Vol. 12, No. 2, 2004, 241-260.

[9] G. Venter and J. Sobieszczanski-Sobieski, "Multidisciplinary Optimization of a Transport Aircraft Wing Using Particle Swarm Optimization," *Structural and Multidisciplinary Optimization*, Vol. 26, No. 1, 2004, pp. 121-131.

[10] G. C. Onwubolu and M. Clerc, "Optimal Path for Automated Drilling Operations by a New Heuristic Approach Using Particle Swarm Optimization," *International Journal of Production Research*, Vol. 4, No. 3, 2004, pp. 473-491. doi:10.1080/00207540310001614150

[11] F. Bergh and A. P. Engelbrecht, "A New Locally Convergent Particle Swarm Optimizer," *IEEE International Conference on Systems, Man and Cybernetics*, Tunisia, 2002.

[12] T. I. Zohdi, "Computational Design of Swarms," *International Journal for Numerical Methods in Engineering*, Vol. 57, No. 15, 2003, pp. 2205-2219. doi:10.1002/nme.762

[13] M. Jahanshahi, M. R. Meybodi and M. Dehghan, "Cellular Learning Automata Based Scheduling Method for Wireless Sensor Networks," *Proceedings of the* 14*th International CSI Computer Conference* (*CSICC*'09), Amirkabir University of Technology, Tehran, October 20-21, 2009, pp. 646-651.

