

A Multi-Role Cellular PSO for Dynamic Environments

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

In real world, optimization problems are usually dynamic in which local optima of the problem change. Hence, in these optimization problems goal is not only to find global optimum but also to track its changes. In this paper, we propose a variant of cellular PSO, a new hybrid model of particle swarm optimization and cellular automata, which addresses dynamic optimization. In the proposed model, population is split among cells of cellular automata embedded in the search space. Each cell of cellular automata can contain a specified number of particles in order to keep the diversity of swarm. Moreover, we utilize the exploration capability of quantum particles in order to find position of new local optima quickly. To do so, after a change in environment is detected, some of the particles in the cell change their role from standard particles to quantum for few iterations. Experimental results on moving peaks benchmark show that the proposed algorithm outperforms mQSO, a well-known multi swarm model for dynamic optimization, in many environments.

1. Introduction

Many practical optimization problems are dynamic. In these problems there are usually multiple local optima, one of them is the global optimum. When the environment changes the local optima may change vastly. Hence, an optimization algorithm has to track the changes in the environment, and find the new global optimum quickly.

Particle swarm optimization is a population-based method, a variant of evolutionary algorithms which moves towards the target rather than evolution, through the search space. It has been shown to perform well in many static [1] and dynamic problems [2-6]. Although it is possible to utilized PSO algorithms designed for static environments in dynamic

environments, they result poor performance. There are two main reasons, outdated memory due to the changes in environment and diversity loss due to convergence of particles to one or few local optima. Outdated memory usually solved by clearing the memory when a change is detected [7] though it results loosing the information collected from the environment. The most effective approaches to counterbalance the effect of diversity loss are multiswarm approaches [4]. Blackwell and Branke [3] introduced a multiswarm PSO, with the aim of maintaining a multitude of swarms on different peaks (local optima) and improved their approach by adding an anti-convergence operator in mQSO [4]. Moreover, Hashemi and Meybodi introduced cellular PSO, a new hybrid model of particle swarm optimization and cellular automata in [8]. The idea of cellular PSO was to utilized local interactions in cellular automata and split the population of particles into different groups across cells of cellular automata by imposing a restriction on number of particles in each cell.

In this paper, we propose an enhancement on our previous model, Cellular PSO[8], by temporarily changing the role of some particles to quantum particle after environment changes. The quantum particles increase exploration capability around the previous found peaks hence new peaks can be found more quickly. Extensive experiments show that the proposed model not only improves basic cellular PSO [8], but also can performs significantly better than mQSO[4] in many different dynamic environments.

The rest of this paper is organized as follows: Section 2 provides a brief introduction on PSO and an overview of the previous works on adapting PSO into dynamic environments. Section 3 introduces cellular automata as the foundations of our approach following by detailed specification of the proposed algorithm in section 4. Section 5 gives out the experimental result of the proposed model along with its comparison to

best results gained in previous works. Finally in section 6 we conclude our paper.

2. Particle swarm optimization

The particle swarm optimization (PSO) algorithm is introduced by Kennedy and Eberhart [9] based on the social behavior metaphor. In PSO a potential solution for a problem is considered as a bird without quality and volume, which is called a particle, flying through a D-dimensional space, adjusting its position in search space according to its own experience and its neighbors.

In PSO, the i th particle is represented by its position vector p_i in the D-dimensional space and its velocity vector v_i . In each time step t , the particles calculate their new velocity then update their position according to eq. (1) and eq. (2) respectively.

$$v_i(t+1) = v_i(t) + c_1 r_1 (p_i^{best} - p_i(t)) + c_2 r_2 (lbest_i - p_i(t)) \quad (1)$$

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

Where p_i^{best} is the best personal position of particle i which has been visited during the lifetime of the particle. $lbest_i$ is the local best position that is the best position of all neighboring particles of particle i . 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 $[0,1]$.

2.1. PSO in Dynamic Environment

Many researches have been done on using particle swarms to address dynamic optimization problems. Blackwell and Branke [3, 4] has suggested dividing the whole population into several sub swarms incorporating three concepts, namely quantum particles, exclusion and anti-convergence. Quantum particles [3] appear at random positions around the sub-swarm's best ensuring a close chase of moving peaks. In order to keep diversity at a desired level, exclusion operator reinitializes the worse swarm when two swarms collide. Moreover taking into account the possibly of existing more peaks than swarms, anti-convergence operator has been utilized to keep constantly patrolling for new and better peaks.

Lung and Dumitrescu have used two collaborating populations of equal size to avoid premature convergence and to efficiently trace the moving optimum [10]. One is responsible for preserving the diversity of the search by using crowding differential evolutionary algorithm while the other keeps track of global optimum with a PSO algorithm. The collaboration mechanism is applied whenever a change

is detected in the search space, or if the best individual in second population is too close to the best solution. The search of the second population is restarted by re-initializing it with the positions of individuals in the first population.

In [6] Li and Yang proposed a multi-swarm method which maintains the diversity through the run. To meet this end two types of swarms are used. A parent swarm which maintains the diversity and detects the promising search area in the whole search space using a fast evolutionary algorithm, and a group of child swarms which explore the local area for the local optima found by the parent using a fast PSO algorithm. This mechanism makes the child swarms spread out over the highest multiple peaks, as many as possible, and guarantees to converge to a local optimum in a short time.

Du and Li [11] suggested dividing particles into two parts, of which the first uses a standard PSO enhanced a Gaussian local search and the second uses differential mutation acting as a patrol team around first to extend the search area of algorithm, and catch up with the moving optimum.

3. Cellular Automata

Cellular automata are mathematical models for systems consisting of large number of simple identical components with local interactions. CA is a non-linear dynamical system in which space and time are discrete. It is called cellular because it is made up of cells like points in a lattice or like squares of checker boards, and it is called automata because it follows a simple rule[12]. Cellular automata perform complex computations with a high degree of efficiency and robustness. Which makes them especially suitable for modeling natural systems that can be described as massive collections of simple objects interacting locally with each other [13, 14]. Informally, a d-dimensional CA consists of an infinite d-dimensional lattice of identical cells. Each cell can assume a state from a finite set of states. The cells update their states synchronously on discrete steps according to a local rule. The new state of each cell depends on the previous states of a set of cells, including the cell itself, and constitutes its neighborhood (Figure 1).

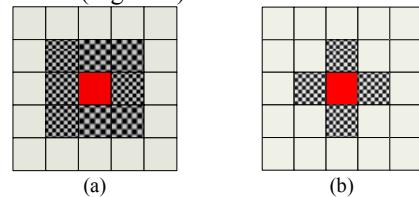


Figure 1. Neighborhood in a 2-D cellular automata: a. Moore, b. von Neumann Neighborhood

4. Proposed model

In previous multi-swarm methods [4-6] in order to prevent sitting of two or more swarm on a peak, every pair of swarms have calculate their distance so that if they were too close, the worse swarm set to be reinitialized. In the cellular PSO [8] we omitted the burden of exhaustive distance calculations by embedding cellular automata into the search space as shown in Figure 2.

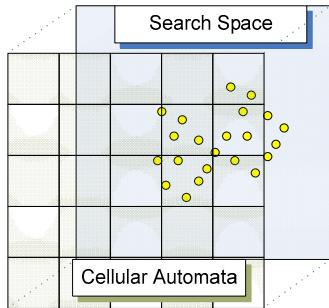


Figure 2. Embedding cellular automata in a 2-D Search Space

In cellular PSO, D-dimensional cellular automata with C^D identical cells are embedded into in the D-dimensional search space. Therefore each particle in search space is in boundaries of a cell. This concept helps preserving diversity in the search space. We define *particle density* of a cell c at time t , $\rho_c(t)$, to be the number of particles which are positioned in boundaries of cell c at the specified time t . By keeping the particle density of all cells bellow a specified threshold θ , we prevent loosing the diversity of whole population and converging to a cell. Therefore only a fraction of population can investigate sub-space of a cell while there are particles available to search other parts. In this model when the automata of cell i find outs particle density in cell i is greater than the specified threshold θ , it randomly selected some of the particles in that cell and reinitialize them in search space as follows.

First, status of a selected particle is changed to *inactive* then a randomly selected cell is assigned as destination for each particle. Afterwards, distance between current cell and the destination cell is calculated and is set as the hop count for the selected particle. Then, at the next iteration neighbors of cell i receive this particle's information as part of state of cell i and add this inactive particle to their state while decreasing the particle's hop counter. This will continue until the destination cell receives this particle as a piece of its neighbors' state. At this time, particle will be activated and will begin searching in the new neighborhood.

Moreover, in order to keep a good diversity among search space and preventing a crowded neighborhood, we use cellular automata neighborhood to determine a particle's neighbors in order to calculate $lbest$ in velocity update equation. Therefore all particles located in cell i uses the same $lbest$.

After detecting a change [15, 16], memory of all cells and all particles' history is reset. Furthermore, half of active particles in all cells take the role of quantum particles for the next τ iterations. On the $\tau+1$ th iteration after the change, all quantum particles take the role of standard particles again.

In our cellular PSO, state of cell i is consists of following information.

1. Best position found in cell i . ($cbest_i$)
2. Best position found in cell i and neighborhood of cell i since last environment change. ($lbest_i$)
3. Information of all particles currently exists in cell i , including particle position, velocity, state, destination, and hop count. Where particle state can be active or inactive. When a particle is inactive it does no function evaluation. Particle's destination and hop count are only meaningful for an inactive particle which is headed to another cell as mentioned above.

At iteration t , automata of cell i calculates its next state as a function of its current state and its neighbors as follows.

1. $lbest_i$ is updated by reading state of neighbors.
2. All standard particles p_k in cell i updates their velocity according to eq. (3).
3. Next positions for standard particles are calculated by eq. (2).
4. If particle p_k is leaving cell i to a neighbor e.g. cell j where $i \neq j$, then the state of particle p_k is set to *inactive*. Later at time $t+1$ when cell j reads state of cell i , it will activate particle p_k and will add it to its state. Next positions for quantum particles are randomly selected within a ball of radius r_{cloud} centered on $cbest_i$ [3].
5. $cbest_i$ updates on finding a better position in cell i .
6. If the particle density in cell i (ρ_i) is greater than the specified threshold (θ) the cell is saturated. In this case $\rho_i - \theta$ particles of particles located in cell i are randomly selected and reinitialize in search space as mentioned above.

5. Experimental study

5.1 Dynamic Test Function

We used “moving peaks” benchmark (MPB) problem introduced by Branke [17] to evaluate performance of our proposed model. In this problem, there are some peaks in a multi-dimensional space, where the height, width and position of each peak alter when the environment change. This function is widely used in literature [18].

The default parameter setting of MPB used in the experiments is presented in Table 1. In MPB, shift length s is the radius of peak movement after an environment change. m is the number of peaks. f is the frequency of environment change as number of fitness evaluations. H and W denote range of height and width of peaks which will change after a change in environment by height severity and width severity respectively. I is the initial heights for all peaks. Parameter A denotes minimum and maximum value on all dimensions.

Table 1. Parameters of Moving Peaks Benchmark

Parameter	Value
number of peaks m	10
f	every 5000 evaluations
height severity	7.0
width severity	1.0
peak shape	cone
shift length s	1.0
number of dimensions D	5
A	[0, 100]
H	[30.0, 70.0]
W	[1, 12]
I	50.0

5.2 Experimental Settings

Blackwell and Brank have shown that mQSO [4] outperforms RPSO [16], multi-QSO, multi-CPSO [3], and Hierarchical Swarms [19]. Thus, here we only present comparison results of our proposed model (mRCPSO) with basic cellular PSO (CPSO) [8] and, mQSO10(5+5^q) [4].

For basic cellular PSO (CPSO) [8] and proposed multi role cellular PSO (mRCPSO), the acceleration constants C1 and C2 were both set to 1.496180, and the inertia weight $w = 0.729844$ [20]. The maximum velocity for a particle was set equal to the neighborhood distance on the cellular automata. Hence, a particle could not move further than neighboring cells at each movement. Both CPSO and proposed

mRCPSO has had cellular automata with 10^5 cells in a 5-Dimensional space. The neighborhood was a Moore neighborhood with a neighborhood distance of 2 cells. Moreover, the maximum allowed particle density, θ , was set equally to 10 for all cells. Also, in mRCPSO 50% of particles in each cell change their role to quantum particles (with $r_{cloud}=0.5$) for 5 iterations after environment change ($\tau=5$).

Moreover, the size of swarm was set to 100 particles for all the experiments. All parameters of mQSO are the same as [4]. We presented the results of mQSO with 10 swarms, each has 5 standard and 5 quantum particles as suggested in [4] ($r_{exec}=31.5$, $r_{conv}=0.0$, $r_{cloud}=1.0$).

In order to evaluate the efficiency of the algorithms, we have used offline error measure[17], the average deviation of the currently best individual from the optimum in all iterations.

5.3 Experimental Results

For all algorithms we reported the average offline error and 95% confidence interval for 100 runs. Table 2 through Table 6 present offline error of mQSO[4], CPSO[8], and proposed multi role cellular PSO (mRCPSO) for different frequency of environment change(f) and the number of peaks(m). For each environment configuration, result of the best performing algorithm(s) with 95% confidence is printed in bold. Where the results of two algorithms do not have a significant difference, both are highlighted.

The proposed model introduced two new parameters to cellular PSO. The first parameter specifies how many standard particles in each cell should change their role to become quantum particle and the second parameter indicates life time of quantum particles (τ). Experimental results show that none of these parameters significantly change the offline error (Figure 3 and Figure 4 respectively).

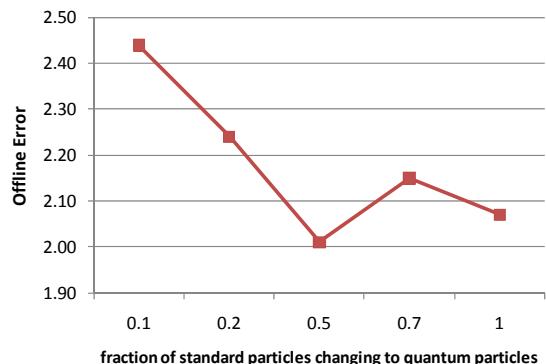


Figure 3. Influence of fraction of standard particles changing to quantum particles on offline error

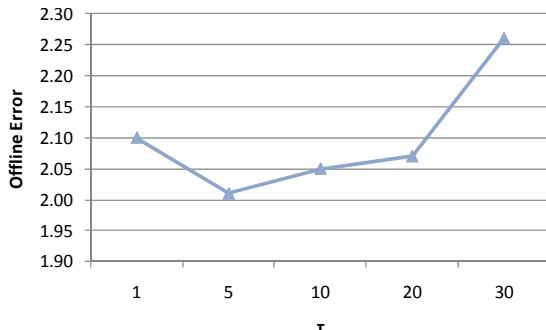


Figure 4. Influence of quantum particle life time τ on offline error

For highly dynamic environment where the peaks change every 2500 FEs or less (Table 2, Table 3, and Table 4), the proposed model mRCPSO, performs significantly better than mQSO for any number of peaks except 10 peaks environments. The superiority of mQSO on 10 peak environments could be because of specific tuning of mQSO 10(5+5^q) which considered 10 swarms that is the optimum number of swarms for a 10 peaks environment as discussed in [4]. Although this specific superiority of mQSO is interesting, it remarks parameter sensitivity of mQSO. Moreover, mRCPSO outperforms mQSO in different configurations with different number of peaks and change frequency, without specific parameter tuning.

By increasing the duration between environment changes to 5000 FEs and 10000 FEs, mRCPSO gradually loose its superiority to mQSO for environments which have few peaks. This can be seen in the environments with $f=5000$ (Table 5) where there is no significant difference between mRCPSO and mQSO for 1, 5, and 10 peaks environment. Moreover, when the duration between environment changes increases to 10000 FEs (Table 6), mQSO performs significantly better than mRCPSO for 1, 5, 10 peaks environments and there are no significant difference between mQSO and mRCPSO for 20 and 30 peaks environments.

Table 2. Offline error for $f=500$

m	mQSO 10(5+5 ^q)	CPSO	mRCPSO
1	28.38±1.89	11.41±0.68	7.68±0.37
5	8.80±0.45	6.83±0.18	6.06±0.15
10	5.74±0.08	6.81±0.17	6.00±0.10
20	6.80±0.04	7.73±0.07	6.47±0.06
30	7.28±0.04	8.39±0.11	6.83±0.06
40	7.52±0.04	8.64±0.11	7.05±0.06
50	7.67±0.04	8.63±0.11	7.15±0.05
100	7.92±0.04	8.93±0.12	7.42±0.05
200	7.91±0.04	8.68±0.11	7.37±0.05

Table 3. Offline error for $f=1000$

m	mQSO 10(5+5 ^q)	CPSO	mRCPSO
1	15.02±0.99	9.05±0.53	7.54±0.44
5	5.49±0.22	4.84±0.25	4.44±0.15
10	4.00±0.06	4.73±0.13	4.32±0.10
20	5.11±0.05	5.96±0.12	4.96±0.07
30	5.71±0.05	6.61±0.12	5.34±0.06
40	5.95±0.05	7.01±0.11	5.56±0.07
50	6.12±0.05	7.11±0.12	5.64±0.06
100	6.37±0.05	7.35±0.13	5.81±0.06
200	6.39±0.05	7.39±0.14	5.81±0.06

Table 4. Offline error for $f=2500$

m	mQSO 10(5+5 ^q)	CPSO	mRCPSO
1	7.42±0.47	6.42±0.59	5.46±0.30
5	3.07±0.11	3.22±0.23	2.75±0.13
10	2.71±0.06	3.26±0.12	2.80±0.11
20	3.81±0.06	4.58±0.13	3.51±0.08
30	4.34±0.07	5.20±0.18	3.78±0.07
40	4.55±0.08	5.59±0.20	4.10±0.07
50	4.76±0.07	5.67±0.18	4.13±0.07
100	5.00±0.07	5.91±0.16	4.30±0.07
200	5.00±0.07	6.21±0.19	4.29±0.07

Table 5. Offline error for $f=5000$

m	mQSO 10(5+5 ^q)	CPSO	mRCPSO
1	4.20±0.25	5.06±0.55	4.46±0.27
5	2.09±0.10	2.13±0.17	1.98±0.12
10	1.94±0.04	2.71±0.13	2.01±0.08
20	2.98±0.08	4.48±0.25	2.86±0.08
30	3.57±0.08	4.93±0.27	3.19±0.09
40	3.97±0.10	5.35±0.32	3.38±0.08
50	3.94±0.09	5.67±0.32	3.49±0.08
100	4.27±0.11	6.14±0.25	3.59±0.08
200	4.29±0.09	5.80±0.23	3.59±0.09

Table 6. Offline error for $f=10000$

m	mQSO 10(5+5 ^q)	CPSO	mRCPSO
1	2.07±0.12	4.51±0.45	3.37±0.27
5	1.07±0.06	1.59±0.16	1.38±0.14
10	1.16±0.05	2.65±0.19	1.52±0.11
20	2.36±0.07	4.05±0.25	2.49±0.13
30	3.05±0.08	4.91±0.35	2.91±0.14
40	3.39±0.10	5.26±0.30	3.01±0.15
50	3.44±0.09	5.48±0.30	3.19±0.14
100	3.72±0.10	6.17±0.30	3.29±0.13
200	3.76±0.09	6.10±0.36	3.25±0.11

6. Conclusion

In this paper, we proposed a new variant of cellular PSO which utilized quantum particles. Cellular PSO first introduced in [8], combines the power of particle swarm optimization with cellular automata concepts by embedding cellular automata into the search space and preserving diversity by keeping particle density in each cell below a specified threshold. The proposed model takes the advantage of exploration capability of quantum particles around a promising area in order to improve its performance. Compared to cellular PSO[8], and mQSO[4], a well-known approach in the literature, the proposed model results more accurate solutions in most dynamic environments without specific parameter tuning for each environment. Furthermore, similar to original cellular PSO, the proposed model requires less computational effort than mQSO, since unlike mQSO it does not need to compute distance of every pair of swarms on each iteration.

References

- [1] K. E. Parsopoulos and M. N. Vrahatis, "Recent Approaches to Global Optimization Problems through Particle Swarm Optimization," *Natural Computing*, vol. 1, no. 2, pp. 235-306, 2002.
- [2] T. Blackwell, "Swarms in Dynamic Environments," in *Genetic and Evolutionary Computation - Gecco 2003, Lecture Notes in Computer Science*, vol. 2723, 2003, pp. 200-200.
- [3] T. Blackwell and J. Branke, "Multi-Swarm Optimization in Dynamic Environments," in *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, vol. 3005, 2004, pp. 489-500.
- [4] T. Blackwell and J. Branke, "Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459-472, 2006.
- [5] T. Blackwell, J. Branke, and X. Li, "Particle Swarms for Dynamic Optimization Problems," in *Swarm Intelligence, Natural Computing Series*, vol. Part II, 2008, pp. 193-217.
- [6] C. Li and S. Yang, "Fast Multi-Swarm Optimization for Dynamic Optimization Problems," in *Fourth International Conference on Natural Computation, Jinan, Shandong, China*, 2008, pp. 624-628.
- [7] R. C. Eberhart and Y. Shi, "Tracking and Optimizing Dynamic Systems with Particle Swarms," in *IEEE Congress on Evolutionary Computation*, 2001.
- [8] A. B. Hashemi and M. R. Meybodi, "Cellular Pso: A Pso for Dynamic Environments," in *To Be Appear in The 4th International Symposium on Intelligence Computation and Applications (ISICA 2009)*, Huangshi, China, 2009.
- [9] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks, Piscataway, NJ*, 1995, pp. 1942-1948.
- [10] R. I. Lung and D. Dumitrescu, "A Collaborative Model for Tracking Optima in Dynamic Environments," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 564-567.
- [11] W. Du and B. Li, "Multi-Strategy Ensemble Particle Swarm Optimization for Dynamic Optimization," *Information Sciences: an International Journal*, vol. 178, no. 15, pp. 3096-3109, 2008.
- [12] E. Fredkin, "Digital Mechanics: An Informational Process Based on Reversible Universal Cellular Automata," *Physica D*, vol. 45, no. 1-3, pp. 254-270, 1990.
- [13] M. Mitchell, "Computation in Cellular Automata: A Selected Review," in *Nonstandard Computation*, T. Gramss, S. Bornholdt, M. Gross, M. Mitchell, and T. Pellizzari, Eds., 1996, pp. 95-140.
- [14] N. H. P. a. S. Wolfram, "Two-Dimensional Cellular Automata," *Journal of Statistical Physics*, vol. 38, pp. 901-946, 1985.
- [15] A. Carlisle and G. Dozier, "Adapting Particle Swarm Optimization to Dynamic Environments," in *International Conference on Artificial Intelligence, Las Vegas, NV, USA*, 2000, pp. 429-434.
- [16] X. Hu and R. C. Eberhart, "Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems," in *IEEE Congress on Evolutionary Computation, Honolulu, HI, USA*, 2002, pp. 1666-1670.
- [17] J. Branke, "Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems," in *1999 Congress on Evolutionary Computation, Washington D.C., USA*, 1999, pp. 1875-1882.
- [18] I. Moser, "All Currently Known Publications on Approaches Which Solve the Moving Peaks Problem," *Swinburne University of Technology, Melbourne, Australia*, 2007.
- [19] S. Janson and M. Middendorf, "A Hierarchical Particle Swarm Optimizer for Dynamic Optimization Problems," in *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, vol. 3005, 2004, pp. 513-524.
- [20] F. van den Bergh, "An Analysis of Particle Swarm Optimizers," *PhD Dissertation, Department of Computer Science, University of Pretoria, Pretoria, South Africa*, 2002.