Two Phased Cellular PSO: A New Collaborative Cellular Algorithm for Optimization in Dynamic Environments

Ali Sharifi, Vahid Noroozi, Masoud Bashiri, Ali Hashemi, Mohammad Reza Meybodi
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
E-mails: alish@aut.ac.ir, vnoroozi@aut.ac.ir, ashkan_bashiri@aut.ac.ir, a_hashemi@aut.ac.ir, mmeybodi@aut.ac.ir

Abstract— **Many real world optimization problems are dynamic in which the landscape is time dependent and the optimums may change over time such as dynamic economic modeling, dynamic resource scheduling and dynamic vehicle routing. These problems challenge traditional optimization methods as well as conventional evolutionary optimization algorithms. In these environments, optimization algorithms must not just find the optima but also closely track the optima's trajectory. In this paper we propose a two phased and collaborative version of Cellular PSO, named Two Phased Cellular PSO (TP-CPSO), which introduces two search phases in order to create a more efficient balance between the exploration and exploitation of the optimums. We address the weaknesses of Cellular PSO and propose some modifications and ideas to tackle them including a modified PSO update rule and an efficient local search. Moreover, the cell capacity threshold which is a key parameter of Cellular PSO is eliminated due to these modifications. To demonstrate the performance and robustness of the proposed algorithm, Moving Peaks Benchmark (MPB) has been adopted. For all the experimented dynamic environments, TP-CPSO outperformed all compared evolutionary algorithms including Cellular PSO.**

Keywords-Particle Swarm Optimizer (PSO); Dynamic Environment; Cellular Automata; Space Partotioning; Cellular PSO; TP-CPSO

I. INTRODUCTION

In dynamic environments, the fitness function changes over time and consequently the optimum points may change. Hence the optimization algorithm has to track the changes in the environment and find the new optimums quickly. The dynamic optimization problems have challenged traditional optimization algorithms which were designed for non-stationary problems. These algorithms converge to the fixed global optimum, which results in losing diversity and decreasing the algorithm's ability to search for the new optimum after the environment changes.

Many nature-inspired algorithms have been proposed for optimization in dynamic environments. Particle Swarm Optimization (PSO) algorithm is considered as one of the major algorithms that is widely used for optimization in stationary environments, yet it still needs some modifications to better cope with dynamic environments.

The disadvantages of PSO in dynamic environments come from two major sources: the outdated memory or experiences of particles after the changes due to the dynamism of environment; and loss of diversity due to convergence. The outdated memory problem occur when the optima move(s) and/or optimum value changes. Particle memory which conveys the best place visited and its relevant fitness may not be valid after the change which leads to a misdirected search. The outdated memory problem can be solved either by clearing or by re-evaluating all the memories. The loss of diversity problem is rather serious. The time spent for swarm convergence and re-diversification, finding the moved optimums and re-convergence lead to severe decrease in the performance of the algorithm.

In this paper, a two phased version of Cellular PSO (TP-CPSO) is proposed for optimization in dynamic environments. The proposed algorithm addresses the disadvantages of Cellular PSO algorithm and introduces solutions to these issues. Two different phases of exploration and exploitation are defined for each cell which leads to eliminating the cell capacity threshold which is a key parameter of Cellular PSO. A powerful local search (Naïve Direct Search) is also introduced in order to follow the moving peaks more efficiently.

The rest of this paper is organized as follows. Next section provides an overview of previous state of the art PSO approaches proposed for dynamic optimization problems. Sections 3 and 4 provide brief introductions to Particle Swarm Optimization and Cellular Automata as the foundations of our approach followed by a detailed explanation of the proposed algorithm in section 5. Section 6 provides the experimental results of the proposed model. Finally, Section 7 concludes the paper with some suggestions on relevant future works.

II. PREVIOUS WORKS

All the PSO based optimization algorithms for dynamic environments can be divided into five categories: 1) Randomization, 2) Mutual Repulsion, 3) Dynamic Information Networks, 4) Multi-Population and 5) Hybrid.

A. Randomization

One of the simplest approaches to address dynamic problems by PSO is reinitializing. Hu and Eberhart used re-diversification of the population after the change [1]. Their proposed method included randomly relocating all or part of the memories by calculating the fitness in one or more points of particles' memories after detecting the environment change. Since randomization leads to missing information, there is a chance that a large amount of information goes missing as if the algorithm runs from scratch with no previous memory. Also, lack of randomization may decrease diversity to an extent which is not enough to track the optimum.

B. Mutual Repulsion

Applying mutual repulsion between particles, swarms or extracted optimums can maintain a desirable amount of diversity during the search process. For instance, Vesterstrom and Krink [2] studied particles with limited size to avoid premature convergence. Parsopoulos and Vrahatis [3] introduced a repulsive operator inside a found optimum point to repel the swarm from that point which allows the swarm to extract other unfound peaks. The atomic model [4-7] is another example of using repulsion for dynamic problems. This model defines a swarm of particles as two sub-swarms of charged and neutral particles. Atomic model can be pictured as a cloud of charged particles orbiting around a neutral nucleus. Charged particles increase the diversity around the converging neutral sub-swarm (nucleus).

C. Dynamic Information Networks

Another approach is applying modifications to the topology of information sharing between the particles of the swarm. This can lead to a temporal decrease in tendency to move towards the best found positions which maintains the diversity. Li and Dum [8] proposed a neighborhood model with a four-cell grid structure and Johnson and Middendorf [9] studied a hierarchical structure and reported improvements in compare to canonical PSO algorithm.

D. Multi-Population

One of the most successful approaches to tackle dynamic environments is multi-population methods. For multi-population algorithms it is usually desired to divide the particles into sub-swarms some designed to converge to the best positions found while others designed to look for new optimum points. This becomes more useful when global optimum alters between local optimums. This method increases the odd that a sub-swarm would always be around the new global optimum after the change. Niching-PSO introduced by Brits [10] has performed well for many static multi-modal benchmarks though have not shown promise for dynamic environments.

Lunge and Dumitrescu [11] use two collaborative populations with the same as a solution to avoid premature convergence and effectively track the optimum. One population is responsible for maintaining diversity using Crowding Differential Evolution algorithm while the other population tracks the optimum using PSO algorithm. The collaboration system is activated every time an environment change is detected or the best member of the second population is too close to the best solution found, that is the second population in reinitialized with the particles from the first population.

In [12] Du and Li suggested that particles should be divided into two parts, one part uses the canonical PSO equipped with a local Gaussian search and the other uses a differential mutation to do the search around the first swarm in order to expand the search regions of the algorithm and reaching the moving peak. These two parts improve the convergence and local optima avoidance of the algorithm respectively.

E. Hybrid

Blackwell and Branke proposed an algorithm that combines the advantages of repulsion and multi-population methods. Inspired by multi-population methods, the algorithm divides the population into multiple sub-swarms. Similar to the atomic model each sub-swarm includes charged particles, and applies the repulsion method by employing repulsion and anti-convergence operators. Charged particles maintain a suitable amount of diversity around found optimums. Repulsion operator tunes the mutual interaction between the swarms. If the distance between two swarms falls below a predetermined threshold, positions of the weaker swarm's particles are reinitialized randomly. Repulsion operator on the other hand, helps the algorithm maintain diversity during the optimization process by avoiding over-convergence to one particular point. However, given the fact that the number of peaks may be larger than the number of swarms, it is necessary to put a number of particles in charge of finding new and better peaks. An anti-convergence operator is embedded in the algorithm to do this task. In [13] Blackwell and Branke introduced the basic version of the algorithm and later proposed the two self-adaptive variants of the algorithm [14]. The first variant starts with one swarm and more swarms will be added based on the needs with regard to a predetermined number of particles. The second variant starts with a random number of swarms and slowly converges to a number of swarms desirable to cover the peaks. This method uses a prefixed number of particles and dynamically assigns particles to swarms.

Hashemi and Meybodi [15] proposed an algorithm, named Cellular PSO, that combines the features from multi-populations, information networks and randomization methods. Cellular PSO is based on the idea of partitioning the search space and mapping a Cellular Automata (CA) onto this divided space in a way that each cell is assigned to one part of the partitioned space. A threshold on the maximum number of particles in each cell is applied which prevents from the crowding of many particles on a single optimum. Extra particles are moved to random cells to explore other points of the search space.

Noroozi et al. [16] also proposed another cellular based optimization algorithm named, CellularDE, which used the similar idea of space partitioning but instead of PSO it utilized Differential Evolution (DE) with a new proposed DE scheme suitable for dynamic optimization. They reported that CellularDE outperforms compared algorithms including Cellular PSO in most tested environments.

III. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) algorithm was first introduced by Eberhart and Kennedy [17] which is based on the social behavior of self-organizing swarms. In PSO, a potential solution for a problem is considered as a bird without quality and volume, which is called a particle. This so called bird flies through a *d*-dimensional space, adjusts its position in search space according to its own experience and its neighbors. A set of particles is called swarm.

Each particle in the *d*-dimensional space is represented as $p_i = (p_{i1}, p_{i2}, \ldots, p_{id})$, where p_i is the *i*th particle. The velocity of the *i*th particle is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ that is in the range of $(0, V_{max})$, where V_{max} is a parameter set by the user. At each time step, particles calculate their new velocities from (1) and update their position using (2).

$$v_i(t+1) = wv_k(t) + c_1 r_1(t)\left(pBest_i - p_i(t)\right) + c_2 r_2(t)\left(lBest_i - p_i(t)\right) \quad (1)$$

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

In above equations, $w < 1$ is the constriction factor which slows down the particles. c_1 and c_2 are positive constants known as acceleration constants that determine the amount of contribution that social and cognitional factors make to the particle's navigation. r_1 and r_2 are *d*-dimensional vectors filled with values generated from a random uniform distribution over [0,1]. $pBest_i$ is the best so far place that the *i*th particle visited and $lBest_i$ is the best visited place by particles that are considered as the *i*th particle's neighbors. The term ($pBest_i - p_i(t)$) is the cognitive component of the *i*th particle's velocity, and the term ($lBest_i - p_i(t)$) conveys the social component of the particle's velocity. In other words, the particle tends to move based on two attractors: its own best memory and the best memory of its neighbors. The coefficient *w* is the inertial weight that indicates the importance of the particle's past velocity to the particle.

IV. CELLULAR AUTOMATA

Cellular Automata (CA) are mathematical models for systems consisting of large number of simple identical components with local interactions in which space and time are discrete. It is called cellular because it is made up of cells like points of a lattice or squares of checker boards, and it is called automata because each cell follows a simple rule [18]. Informally, a *d*-dimensional CA consists of a *d*-dimensional lattice of identical cells. The simple components act together to produce complicated patterns of behavior. 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. Moore and Von Newman are the two common neighborhood styles which are illustrated in Figure 1. for a 2-dimensional space with the neighborhood size of one.

V. PROPOSED ALGORITHM

The proposed algorithm is based on the idea of partitioning the search space into finer areas then performing the search in each of them locally. A CA is mapped into the partitioned space where each cell is responsible to control the search process in its corresponding partition. Particles are initialized randomly and assigned to their corresponding cells according to their locations. These particles are considered as a part of the state of their cells. Each cell performs the search process within its predetermined region using a new version of PSO that we name it Mutative PSO (MPSO). This structure implicitly distributes particles into sub-swarms wherein particles are responsible for finding the most promising optimum in the neighborhood of their corresponding cells. All the procedures of the proposed algorithm such as controlling the search process, interaction between neighboring cells and controlling swarm diversity are defined as rules of the CA.

A. Initialization

If the search landscape is a *d*-dimensional space and each dimension is partitioned equally into *Np* segments, then each cell in a d-dimensional CA belongs to cells collection $C = \{cell_i | 1 \le i \le (N_p)^d\}$. Figure 2. illustrates a 2-dimensional search space where each dimension is partitioned into four equal segments and a CA of the same size is embedded into the partitioned space. The function Ψ, which calculates the *d*-dimensional location of the $cell_k$, is defined as in (3), where z_i can be calculated by (4) and (5).

$$\Psi\{cell_k\} = (z_1, z_1, \cdots, z_d) \quad (3)$$

$$z_1 = \left\lfloor k / (N_p)^{d-1} \right\rfloor \quad (4)$$

$$z_{i+1} = \left\lfloor k - \sum_{j=1}^{i} \frac{z_j * (N_p)^{d-j}}{(N_p)^{d-(i+1)}} \right\rfloor \quad (5)$$

In the proposed algorithm, Moore neighborhood is used as the neighborhood relation. If the size of the neighborhood is to be notated by S_N, then the neighbor cell collection of $cell_{z_1, z_2, \cdots, z_d}$ is calculated by (3).

$$Z(cell_{z_1, z_2, \cdots, z_d}) = \{cell_{m_1, m_2, \cdots, m_d} \mid m_k = z_1 \pm g, \; k = 1 \ldots d, -S_N \le g \le S_N, g \in Z\} \quad (6)$$

(a) (b)

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

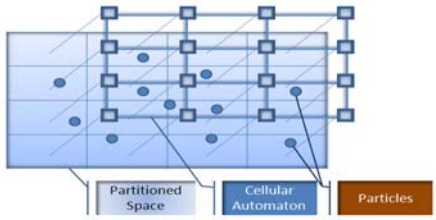

Figure 2. Embedding a CA in a two-dimensional space which is partitioned into four equal sections.

Initially, a swarm of M particles, $P=\{p_1, p_2, \ldots, p_M\}$ are randomly generated and each particle is assigned to its corresponding cell. The ith particle (p_i) is assigned to the $cell_{z_1,z_2,\cdots,z_d}$ where z_i is calculated by (7).

$$z_i = \lfloor x_{ki} / N_p \rfloor + 1 \tag{7}$$

B. Cell Memory

To reduce unnecessary searches in the regions which are already searched and also to use the results of previous searches, a memory, Mem_i is defined for each $cell_i$ as in (8) which memorizes the best individual found in the cell from the time of the last change up to the time step t. These cell memories are used to calculate the best visited place in each cell and its neighborhood from the last change until the current iteration. Equation (9) shows this calculation where f is the fitness function of the optimization problem. $LB_i(t)$, is considered as the global best experience, *lBest*, for all the particles in $cell_i$.

$$Mem_i(t) \leftarrow \underset{\forall k, \text{particle } p_k \text{ is in } cell_i}{argmax} \left\{ f(p_k), f(Mem_i(t-1)) \right\} \tag{8}$$

$$LB_i(t) \leftarrow \underset{\forall j, cell_j \text{ is a neighbor of } cell_i}{argmax} \{ f(Mem_i(t-1)), f(Mem_j(t-1)) \} \tag{9}$$

C. Evolution by Mutative PSO

In Cellular PSO, the conventional version of PSO is used to update the particles, but it causes stagnation or premature convergence for small sized swarms which may constantly occur in the proposed framework. This is due to the fact that in small-sized swarms, most particles may come too close to their attractors before convergence which drastically slows down their progress toward the local optima. In order to tackle the aforementioned problem, a modified PSO algorithm is proposed that uses a new velocity updating equation. A mutation term is introduced that adds a random vector to the velocity vector which increases the diversity of velocities. The proposed velocity update equation for the particle p_i, which is now located in $cell_j$, is showed in (10) where the mutation term (N_i) is calculated by (11) and (12).

$$v_i(t+1) = wv_k(t) + c_1 r_1(t)\left(pBest_i - p_i(t)\right) + c_2 r_2(t)\left(LB_j - p_i(t)\right) + \sigma_i N(0,1) \tag{10}$$

$$\sigma_i = \frac{2e^{-ds_i}}{1+e^{-ds_i}} \tag{11}$$

$$ds_i = \left\| p_i - pBest_i \right\| + \left\| p_i - LB_j \right\| \tag{12}$$

Note that for all particles in $cell_j$, LB_j is used as their *lBest*. The proposed update equation eliminates the stagnation problem or premature convergence to points other than the actual local optima, but when a sub-swarm in a cell is located to the local optima, the added mutation term slows down the convergence of particles which is unnecessary. In addition, when the particles in a cell are converged to the local optima, any further searching would have a little influence on the performance and only force many futile fitness evaluations to the algorithm. The proposed PSO algorithm is able to explore the search space though lacks necessary exploitation abilities, so it's necessary to use another search method after the convergence in order to perform exploitation appropriately.

D. Exploration and Exploitation Phases

In contrast to Cellular PSO where all the search process is always conducted by PSO, the proposed method defines two phase of exploration, exploitation for each cell. Each cell has an operational status variable that determines which phase the cell is functioning in. At first, all cells are in exploration phase in which the search is performed by the proposed PSO algorithm until a convergence to local optima is detected.

If a cell includes some particles for more than k consecutive fitness evaluations and the best position found in the cell is the same as the best position in the neighborhood of the cell, then the cell goes to the exploitation state. When a cell enters the exploitation state, all the particles other than the best one will be reinitialized to random cells. In the cells which are in exploitation phase, the search process is conducted by a proposed local search with a high capability of exploitation instead of PSO that we name it Naïve Direct Search (NDS).The NDS algorithm is based on directed movements in all dimensions with a defined step size. The step size is reduced if no improvement is achieved in any dimension by the current step size.

For $cell_i$ which is in exploitation state, a step size ss_i is defined that specifies the absolute amplitude of movements and is initialized by a defined value ss_{init}. A direction vector $dir_i=(dir_i^1, dir_i^2, \ldots, dir_i^d)$ is defined which defines the current direction of movement in each dimension. Direction in each dimension may be *up* or *down* and are initially assigned randomly. At each iteration, the current $LB_i(t)$ is considered as the base point of the local search and a consequent move is done on all the dimensions of the base point one by one. At each dimension, a directed move is applied by the amplitude of the current step size ss_i in the current direction of that dimension and the new position is evaluated. If the resulted position had a better fitness, the base point is updated to the new position; otherwise the direction in that dimension is flipped to the opposite direction. The obtained movements are preserved to be used in the next iteration. In case of a movement failure, if there has been at least one success in the current direction for the current step size, movement stops for its relative particle, otherwise it tries the reverse direction for the next movement. A counter sn_i is also defined. When the search is stopped in all dimensions, the search process starts again in all dimensions and the counter is increased by one. At each iteration, the step size of movement (ss_i) is defined by (13), where the discount factor (b) defines the speed of reduction. All the failure counters are reset to their initial values after any change detection in the environment.

$$ss_i = ss_{init} \times b^{sn_i} \tag{13}$$

Since the proposed local search method is able to perform exploitation using only one particle, cell capacity is set to one during the exploitation state. Until the cell is in this state, if any particle enters the cell in the next iterations, all particles other than best one are reinitialized to random cells. As a result, a number of particles that were previously converged to

these cells will be free and ready to help explore the search space. To put it differently, PSO is in charge of exploring the search space then the proposed local search is assigned to exploit the converged region. A cell would remain in the state of exploitation until the base point of local search exceeds the cell bounds. Afterwards the cell would go back to the exploration state.

In Cellular PSO, a threshold on the maximum number of allowed particles in each cell is defined which is so dependent and sensitive to the environment. Selection of an appropriate value for this threshold has a great influence on the result of the algorithm. But in the proposed algorithm, capacity threshold is defined unlimited for cells in exploration state and one in exploitation state. In some way, there is no need to define capacity threshold according to the environment. It's resulted from the definition of two separate phases for the cells which is considered as one of the main advantages of the proposed algorithm in compare to Cellular PSO.

Decreasing the swarm size leads to severe decrease in performance for Cellular PSO. The reason is that when using small swarms, canonical PSO suffers from premature convergence problem. In Proposed algorithm due to employing proposed PSO which can avoid stagnation and unlimited cell capacity in exploration phase and then using the proposed local search algorithm which provides necessary exploitation capabilities only with one particle for the algorithm.

E. Particle's movement in CA

Due to the limit of access to other cells, which is originated from the locality of CA rules, it is not possible to move the particles freely between cells. A propagation mechanism is used to move particles between different cells. Particles which are marked to move are made inactive and will not participate in the search process. A counter, Hop_j, is defined for each inactive particle, p_j, which is initialized to the distance of the source, $cell_{s_1,s_2,\cdots,s_d}$, and destination cell, $cell_{q_1,q_2,\cdots,q_d}$, as in (14). At each iteration, all cells copy the inactive particles with non-zero counters which are in their neighborhood cells. Their counters are decreased by one and the ones with zero counters are checked to have the same destination as the current cell. Inactive particles which reach their destination become active again in order to participate in the search process.

$$Hop_j = \underset{k=1}{\overset{d}{Max}}\left\{\left|q_k - s_k\right|\right\} / S_N \tag{14}$$

F. Dealing with Change

Each time a change is detected in the environment, all particle memories are re-evaluated and cell memories are cleared. Also the failure counters for the local search are reset to their initial values. As a result of utilizing the combination of this local search and the proposed PSO, in contrast with Cellular PSO, there is no need to perform any other additional local search for tracking moving optima after any change detection in the environment.

VI. EXPERIMENTS

A. Moving Peaks Benchmark

Moving Peaks Benchmark (MPB) [19] is adopted for experiments. MPB has been widely used in the literature to study the performance of optimization algorithms in dynamic environments. MPB generates functions in multidimensional landscapes consisting of several peaks where the height, the width and the position of the peaks are altered every time a change in the environment occurs. The MPB function with *m* peaks in an *n*-dimensional environment.

$$F(\vec{x},t) = \max(B(\vec{x}), \max_{i=1\ldots m} P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t))) \tag{15}$$

Where $B(\vec{x})$ is a time-invariant basis landscape, P is a function defining a peak shape, $h_i(t)$, $w_i(t)$, and $p_i(t)$ are the height, width, and position of the respective peaks respectively and are defined as:

$\sigma \in \mathrm{N}(0,1)$

$$h_i(t) = h_i(t-1) + height\_severity \cdot \sigma \tag{16}$$

$$w_i(t) = w_i(t-1) + width\_serverity \cdot \sigma \tag{17}$$

$$\vec{p}_i(t) = \vec{p}_i(t-1) + \vec{v}_i(t) \tag{18}$$

In above equations, h_s and w_s are the maximum amount with which the width and height of peaks can change and $\vec{v}_i(t)$ is the amount of peak movement defined in (18).

$$\vec{v}_i(t) = \frac{s}{\left|\vec{r} + \vec{v}_i(t-1)\right|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)) \tag{19}$$

Where $\vec{r}$ is a random vector, s controls the severity of the shift, and is the degree of correlation to previous shifts.

TABLE I. THE DEFAULT PARAMETERS OF MPB FOR SCENARIO II

Parameter	Value
Number of peaks (m)	10
Change frequency (*f*)	5000
Height severity (h_s)	0.7
Width severity (w_s)	0.1
Peak shape	Cone
Shift length (*s*)	1.0
Number of dimensions (*d*)	5
A	[0, 100]
H	[30, 70]
W	[1, 12]
I	50

We consider the parameters from a well-known scenario known as Scenario II as default values in our experiments. The settings defined in Scenario II are represented in TABLE I. . Shift length *s* is the radius of peak movement after environment changes. *m* is the number of peaks. *f* is the frequency of the changes in environment as number of fitness evaluations. *H* and *W* denote range of the height and width of peaks which change by height severity (h_s) and width severity (w_s), respectively. *I* is the initial height of the peaks. Parameter *A* denotes the range of the search space for all dimensions.

B. Evaluation Criterion

Since there are no specific constant optimums for dynamic optimization problems, the goal is not just to find the

optimums, but to track the trajectories of optimum points in the search space. A common method for illustrating an algorithm's prominence over others is to demonstrate convergence diagrams for best or average fitness and visually comparing them. The most common measurement criterion in dynamic optimization problems is offline-error which can be calculated by (20).

$$offline-error = \frac{1}{T}\sum_{t=1}^{T} e_t^{'} \quad (20)$$

In above equation T is the total number of fitness evaluations, $e_t^{'} = \min\{e_\tau, e_{\tau+1}, \ldots, e_t\}$ and τ represents the last time step at which the last change in the environment occurred.

C. Experimental setting

For all experiments the default values for parameters of Two Phased Cellular PSO are set according to Table 2. The search space is partitioned into 12^5 cells, i.e. each dimension is divided into 12 partitions. Moreover, in the cellular automata, the Moore neighborhood with radius of two cells is used, which is large enough to handle the exploration of the sub-populations in a neighborhood and is small enough to prevent the convergence of the population to a local optimum. Initial step size (ss_{init}), discount factor (b) and NDS are empirically set to 0.5 and 0.4, respectively. The number of local search iterations (LS_{num}) and the radius of the local search (LS_r) are empirically set to 3 and 1.0, respectively.

D. Experiment: Comparison with other algorithms

The results of the proposed algorithm are compared with Cellular PSO [18, 19], CellularDE [16], HmSO [20], and Adaptive mQSO [14]. To the best of our knowledge, Adaptive mQSO and HmSO are two of the best-performing PSO-based optimization algorithms introduced for dynamic environments. Cellular PSO [18, 19] and CellularDE share the idea of embedding a cellular automaton in the search space.

For all experiments, parameters of Cellular PSO, CellularDE, HmSO, and Adaptive mQSO are set to the values reported in [21], [16], [20], and [14], respectively. All experiments were performed for 100 changes in environment. The average offline errors of the algorithms in 100 runs with 95% confidence interval for various dynamic environments are depicted in TABLE II. to TABLE V. . For each environment, student t-tests with a significance level of 0.05 have been applied and the result of the best performing algorithm(s) is printed in bold. When the offline errors of the best performing algorithms are not significantly different, all are printed in bold.

The results of the experiments show that Two Phased Cellular PSO outperforms all other compared algorithms, for most of the tested dynamic environments. As depicted in Figure 3. for an environment with 50 peaks, TP-CPSO can find better solutions with a higher speed of convergence compared to Cellular PSO. This is due to the fact that TP-CPSO, takes advantage of two separate phases which impose an efficient balance between exploration and exploitation in the search process. Moreover the local search capability of NDS results in powerful exploitation, and eliminating of cell threshold capacity leads to impose no limit on particle absorption for cells in exploration phase.

VII. CONCLUSION

In this paper, we proposed a two phased version of Cellular PSO algorithm, designed to tackle dynamic optimization problems. In TP-CPSO the disadvantages of Cellular PSO were addressed and some strategies were proposed as solutions to these issues. Extensive experiments in various dynamic environments modeled by the moving peaks benchmark were conducted. The results show significant prominence of the proposed algorithm over the best PSO-based algorithms known in the literature.

Moreover, the proposed algorithm eliminates the cell capacity threshold which is a key parameter of Cellular PSO and the performance of Cellular PSO is sensitive to its value.

TABLE II. OFFLINE ERRORS FOR DIFFERENT NUMBER OF PEAKS (F = 500)

m	Two Phased Cellular PSO	CellularDE	Cellular PSO	HmSO	Adaptive mQSO
1	5.96±0.22	8.20±0.19	11.62±0.77	8.53±0.49	**5.08±0.27**
5	**5.12±0.11**	6.06±0.05	8.59±0.36	7.40±0.31	**5.14±0.09**
10	**5.56±0.11**	5.93±0.04	8.78±0.28	7.56±0.27	6.20±0.11
20	**5.67±0.10**	**5.60±0.03**	8.67±0.25	7.81±0.20	6.94±0.18
30	**5.59±0.08**	**5.56±0.03**	8.24±0.22	8.33±0.18	7.23±0.16
40	**5.53±0.07**	**5.48±0.02**	8.50±0.20	8.45±0.18	7.43±0.17
50	5.57±0.08	**5.47±0.02**	8.37±0.20	8.83±0.17	7.49±0.09
100	5.36±0.06	**5.29±0.02**	7.91±0.18	8.85±0.16	7.29±0.15
200	5.33±0.05	**5.07±0.02**	7.71±0.14	8.85±0.16	6.82±0.14

TABLE III. OFFLINE ERRORS FOR DIFFERENT NUMBER OF PEAKS (F = 1000)

m	Two Phased Cellular PSO	CellularDE	Cellular PSO	HmSO	Adaptive mQSO
1	**2.66±0.08**	4.98±0.35	5.86±0.42	4.46±0.26	**2.68±0.14**
5	**2.67±0.09**	3.96±0.04	5.26±0.26	4.27±0.08	3.22±0.07
10	**3.17±0.06**	3.98±0.03	5.75±0.23	4.61±0.07	4.11±0.08
20	**3.33±0.06**	4.53±0.02	5.74±0.19	4.66±0.12	4.75±0.14
30	**3.45±0.06**	4.77±0.02	5.84±0.16	4.83±0.09	4.98±0.10
40	**3.58±0.05**	4.87±0.02	5.84±0.17	4.82±0.09	5.10±0.11
50	**3.57±0.05**	4.87±0.02	5.84±0.14	4.96±0.03	5.12±0.05
100	**3.52±0.04**	4.85±0.02	5.73±0.11	5.14±0.08	5.03±0.09
200	**3.51±0.04**	4.46±0.01	5.48±0.11	5.25±0.08	4.65±0.09

TABLE IV. OFFLINE ERRORS FOR DIFFERENT NUMBER OF PEAKS (F = 2500)

m	Two Phased Cellular PSO	CellularDE	Cellular PSO	HmSO	Adaptive mQSO
1	**0.92±0.03**	2.38±0.78	3.78±0.25	1.75±0.10	**1.09±0.06**
5	**1.17±0.05**	2.12±0.02	2.91±0.14	1.92±0.11	1.58±0.13
10	**1.59±0.06**	2.42±0.02	3.18±0.16	2.39±0.16	2.33±0.11
20	**1.82±0.04**	3.05±0.04	3.65±0.13	2.46±0.09	2.84±0.09
30	**1.99±0.04**	3.29±0.03	3.90±0.11	2.57±0.05	3.13±0.09
40	**1.96±0.03**	3.43±0.03	4.20±0.13	2.56±0.06	3.23±0.08
50	**2.01±0.03**	3.44±0.02	4.08±0.11	2.65±0.05	3.24±0.07
100	**2.09±0.03**	3.36±0.01	4.23±0.09	2.72±0.04	3.20±0.06
200	**2.06±0.02**	3.13±0.01	4.09±0.10	2.81±0.04	3.00±0.05

TABLE V. OFFLINE ERRORS FOR DIFFERENT NUMBER OF PEAKS (F = 5000)

m	Two Phased Cellular PSO	CellularDE	Cellular PSO	HmSO	Adaptive mQSO
1	**0.40±0.01**	1.53±0.07	2.79±0.18	0.87±0.05	0.55±0.02
5	**0.75±0.07**	1.50±0.04	2.03±0.17	1.18±0.04	1.00±0.04
10	**0.96±0.05**	1.64±0.03	2.06±0.12	1.42±0.04	1.43±0.04
20	**1.18±0.03**	2.46±0.05	2.99±0.13	1.50±0.06	1.95±0.05
30	**1.32±0.03**	2.62±0.05	3.21±0.11	1.65±0.04	2.15±0.05
40	**1.36±0.03**	2.76±0.05	3.35±0.11	1.65±0.05	2.28±0.04
50	**1.40±0.03**	2.75±0.05	3.37±0.12	1.66±0.02	2.28±0.02
100	**1.50±0.02**	2.73±0.03	3.35±0.10	1.68±0.03	2.31±0.03
200	**1.56±0.02**	2.61±0.02	3.32±0.09	1.71±0.02	2.11±0.03

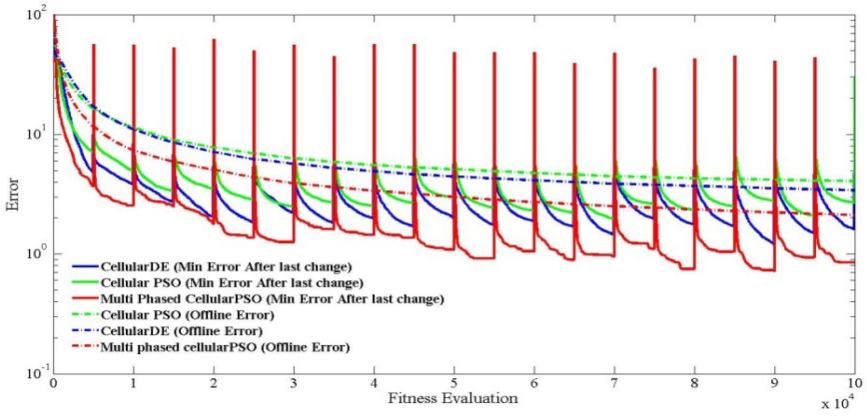

Figure 3. The current error and the offline error for a dynamic environment with 50 peaks and *f*=5000.

REFERENCES

[1] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimisation: detection and response to dynamic systems," presented at the Congress on Evolutionary Computation, 2002.

[2] J. Vesterstrom, T. Krink, and J. Riget, "Particle swarm optimisation with spatial particle extension," presented at the Congress on Evolutionary Computation, 2002.

[3] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimizer in noisy and continuously changing environments," in *IASTED International Conference on Artificial Intelligence and Soft Computing*, Cancun, Mexico, 2001, pp. 289-294.

[4] T. M. Blackwell and P.Bentley, "Don't Push me! collesion avoiding swarms," 2002.

[5] T. M. Blackwell and J. Branke, "Dynamic search with charged swarms," presented at the Genetic and Evolutionary Computation Conference, 2002.

[6] T. M. Blackwell and J. Brank, "Multi-swarm optimization in dynamic ennvironments," presented at the Application of Evolutionary Computing, 2004.

[7] T. M. Blackwell, "Swarms in dynamiic environments," presented at the Genetic and Evolotionary Computation Conference, 2003.

[8] X. Li and K. H. Dam, "Comparing particle swarm for tracking exterma in dynamic environments," 2003.

[9] S. Janson and M. Middendorf, "A hierachical particle swarm optimizer for dynamic optimization problems," presented at the Applications of evolutionary computating, 2004.

[10] R. Brits, A. P. Engelbrecht, and F. v. d. Bergh, "A niching particle swarm optimizer," in *In Fourth Asia-Pacific conference on simulated evolution and learning*, 2002, pp. 692-696.

[11] R. I. Lung and D. Dumitrescu, "A Collaborative Model for Tracking Optima in Dynamic Environments," presented at the IEEE Congress on Evolutionary Computation, 2007.

[12] W. Du and B. Li, "Multi-Strategy Ensemble Particle Swarm Optimization for Dynamic Optimization," *Information Sciences: an International Journal,* vol. 178, pp. 3096-3109, 2008.

[13] T. M. Blackwell and J. Branke, "Multi-swarm, exlusion and anti-convergence in dynamic environments," *IEEE transactions on Evolutionary Computation,* 2004.

[14] T. Blackwell, "Particle Swarm Optimization in Dynamic Environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*. vol. 51, ed: Springer Berlin, 2007, pp. 29-49.

[15] A. B. Hashemi and M. R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments," *Advances in Computation and Intelligence*, pp. 422-433, 2009.

[16] V. Noroozi, A. B. Hashemi, and M. R. Meybodi, "CellularDE: A Cellular Based Deifferential Evolution for Dynamic Optimization Problems," presented at the Adaptive and Natural Computing Algorithms, Lecture Notes in Computer Science, 2011.

[17] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in *Proceeding of IEEE International Conference of Neural Networks*, 1995, pp. 1942-1948.

[18] E. Fredkin, "Digital Mechanics: An Information Process Based on Reversible Universal Cellular Automata," *Physica,* vol. D45, pp. 254-270, 1990.

[19] J. Branke, "Memory enhanced evolutionary algorethms for changing optimization problems," presented at the Congress on Evolutionary Computation CEC99, 1999.

[20] M. Kamosi, A. B. Hashemi, and M. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," in *Proceeding of the Second World Congress on Nature and Biologically Inspired Computing(NaBIC)*, 2010, pp. 363-369.

[21] M. R. Meybodi and A. B. Hashemi, "A Multi-role cellular PSO for dynamic environments," presented at the 14th International CSI Computer Conference, Tehran, Iran, 2009.