CellularDE: A Cellular Based Differential Evolution for Dynamic Optimization Problems

Vahid Noroozi, Ali B. Hashemi, and Mohammad Reza Meybodi

Computer Engineering and Information Technology Department,
Amirkabir University of Technology, Tehran, Iran
{vnoroozi,a_hashemi,mmeybodi}@aut.ac.ir

Abstract. In real life we are often confronted with dynamic optimization problems whose optima change over time. These problems challenge traditional optimization methods as well as conventional evolutionary optimization algorithms. In this paper, we propose an evolutionary model that combines the differential evolution algorithm with cellular automata to address dynamic optimization problems. In the proposed model, called CellularDE, a cellular automaton partitions the search space into cells. Individuals in each cell, which implicitly create a subpopulation, are evolved by the differential evolution algorithm to find the local optimum in the cell neighborhood. Experimental results on the moving peaks benchmark show that CellularDE outperforms DynDE, cellular PSO, FMSO, and mQSO in most tested dynamic environments.

Keywords: Dynamic environments, Differential evolution, Cellular Automata.

1 Introduction

Many optimization problems in the real-world are dynamic in which the fitness function changes over time. These problems can be modeled with multiple local optimums that one of them is the global optimum. When a change is occurred in the environment, the global optimum may change. Hence the optimization algorithm has to track the changes in the environment and find the new global optimum quickly.

The dynamic optimization problems have challenged traditional optimization algorithms as well as conventional evolutionary algorithms, which were designed for non-stationary problems. This is due to the goal of conventional evolutionary algorithms, converging to the fixed global optimum, which results in losing diversity and decreases the power of algorithm to search for the new optimum after the environment changes.

Many approaches have been introduced to address dynamic optimization problems that can be categorized into four groups: (1) Diversity maintenance approaches [1-2]. (2) Memory based approaches [3-4]. (3) Increasing diversity approaches [5-6]. (4) Multi-population approaches [7-8]. A comprehensive survey on evolutionary algorithms that are developed and applied to dynamic environments can be found in [9]. Most of the recent researches on evolutionary algorithms for dynamic optimization problems are focused on the fourth category which also has been shown the most outstanding results in dynamic optimization among other proposed approaches [10].

It has been shown that Differential evolution algorithm (DE) [11] is be a simple and powerful algorithm for continuous function optimization, not even in static [12] but also in dynamic environments [7, 13-15]. Moreover, DynDE [7], which to the best of our knowledge is the best-performing differential evolution algorithm for dynamic optimization problems, produces competitive results compared to other dynamic optimization algorithms. Although an improved version of DynDE [13] is presented, it only improves the performance of a few environments a little, while it significantly increases the complexity of the algorithm.

In [16-17], Hashemi and Meybodi proposed cellular PSO, a hybrid model of particle swarm optimization and cellular automata and showed it produces promising results. In cellular PSO, a cellular automaton is embedded into the search space and partitions the search space, such that each partition corresponds to a cell. At any time, in some cells of the cellular automaton, a group of particles are present and search for an optimum. In the search for an optimum, particles in a cell use their best personal experience and the best solution found in their neighborhood cells. Moreover, in order to prevent losing diversity of particles, a limit on the number of particles searching in each cell is imposed.

In this paper, we combine DE and cellular automata in order to maintain diversity of the population in the search space as well as to perform effective local search at the same time. In the proposed model, named CellularDE, like cellular PSO, a cellular automaton is embedded into the search space and partitions the search space such that each partition corresponds to a cell. The population implicitly is divided into sub-populations, each residing in a cell. Individuals in each cell search for a local optimum using local information in that cell and its neighbors. In moving towards dynamic optimum, preserving individual's density in each cell below a specified threshold is of great concern. To keep the individuals density below this threshold, a portion of individuals may be reinitialized at the end of each iteration. This mechanism makes the individuals spread out over the highest peaks across the environment, meanwhile it helps converging to each peak in a short time. Moreover, after detecting a change in the environment, individuals temporarily perform random local search around the local optimum in their neighborhood, which helps the algorithm to quickly find new peaks.

The performance of the proposed CellularDE is compared with DynDE [7], cellular PSO [16-17], FMSO [18] and mQSO [19-20] in various dynamic environments modeled by the moving peaks benchmark [21-22]. The results of the experiments show that CellularDE outperforms other algorithms in most tested dynamic environments. In addition, it is more robust to the number of peaks in the environment. Moreover, CellularDE takes the advantage of local information exchange and decentralize control of cellular automaton which makes the algorithm scalable.

The rest of this paper is organized as follows. The next section describes the differential evolution algorithm. Section 3 provides detailed specification of the proposed algorithm with a brief introduction to cellular automata as the foundations of our approach. Section 4 gives out experimental results of the proposed model. Finally, Section 5 concludes this paper.

2 Differential Evolution

Differential Evolution (DE) [11] is a population based evolutionary algorithm which differs from other evolutionary algorithms in its mechanism of generating offspring. In evolutionary algorithms, an individual plays the role of a parent to generate an offspring. However, in DE an offspring is generated using vector differences among individuals in the population [12]. The canonical DE algorithm works as follows.

1. Initialize population with uniform distribution.
2. Until a termination condition is met, for each individual $\vec{x}_i$ in the population do in parallel:

    i. Select three different individuals $\vec{x}_{r1}$, $\vec{x}_{r2}$, and $\vec{x}_{r3}$ from the current population randomly.

    ii. Generate a trial vector $\vec{v}_i$ using eq. (1).

$$\vec{v}_i = \vec{x}_{r1} + F.(\vec{x}_{r2} - \vec{x}_{r3}) \tag{1}$$

where $F \in [0,1]$ is a scale factor.

    iii. Create a new vector $\vec{u}_i$ using binomial crossover according to eq.(2).

$$u_{ij} = \begin{cases} v_{ij} & \mathit{if}\ (U(0,1) \le C_r\ or\ j = r_j) \\ x_{ij} & otherwise \end{cases} \tag{2}$$

where $C_r \in (0,1)$ is known as crossover probability and $r_j \in [1,D]$ is a random integer and D is number of dimensions.

    iv. Evaluate the candidate $\vec{u}_i$.

    v. Replace individual $\vec{x}_i$ with $\vec{u}_i$, if fitness of the newly create vector $\vec{u}_i$ is better than fitness of individual $\vec{x}_i$.

In this algorithm F and C_r are control parameters which play the same role as mutation and crossover probability in the evolutionary algorithms such as Genetic algorithms.

Many typical variants of DE, called schemes, have been proposed in literatures. Each scheme varies with respect to the number of random individuals that are used to construct a new trial vector, as well as with respect to whether or not the current individual or the global best individual is used as part of that computation.

3 The Proposed Algorithm

In the proposed algorithm, we follow the partitioning approach using cellular automata [16-17]. Cellular automaton (CA) is a mathematical model for systems consisting of large number of simple identical components with local interactions. 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 automaton because it follows a simple rule [23]. Each individual cell is in a specific state. The cells synchronously update their states on discrete steps according to a local rule which depends on the previous states of the cell itself and its neighborhood. Fig. 1 (a, b) shows Neighborhood in a 2-dimensional cellular automaton and two well-known neighborhoods, Moore and von Neumann. The overall structure can be viewed as a parallel processing device.

In CellularDE, a cellular automaton is embedded into the search space and divides it into a number of partitions each of which corresponds to one cell in the cellular automaton. Fig. 1 (c) illustrates a 2-D cellular automaton which is embedded into a two-dimensional search space and partitions it into 5^5 cells. Cellular automaton implicitly divides the population into a number of sub-populations each of which resides in a cell and is responsible for finding the highest peak in that cell and its neighborhood if there is any peak. In the following, first we introduce components of the CellularDE, and then detailed steps of the proposed algorithm are described.

3.1 Memory of the Cells

In CellularDE, each cell has two memories, *cellBestMem$_i$* and *gBestMem$_i$*. *cellBestMem$_i$* is the best position found in cell i since the last change in the environment (eq. (3)). *gBestMem$_i$* represents the best position found in neighborhood of cell i, which include cell i and its neighbors, since the last change in the environment (eq. (4)).

$$cellBestMem_i \leftarrow \underset{\forall k\text{ ,individual}_k\text{ is in cell }i}{argmax} \left\{f\left(individual_k\right), f\left(cellBestMem_i\right)\right\} \tag{3}$$

$$gBestMem_i \leftarrow \underset{\forall j\text{ ,cell}_j\text{ is a neighbor of cell}_i}{argmax} \left\{f\left(cellBestMem_i\right), f\left(cellBestMem_j\right)\right\} \tag{4}$$

3.2 Cellular Differential Evolution

In CellularDE, individuals within a cell and its neighbors implicitly create a sub-population. Each sub-population aims to search for the local optimum in its neighborhood using differential evolution algorithm. Therefore, differential evolution in CellularDE should provide local exploration capability rather than global exploration of canonical DE. Hence, CellularDE utilizes the DE/rand-to-best/1/bin scheme, in which the mutant vector $\vec{v}_i$ for individual $\vec{x}_i$ residing in cell$_j$ is calculated using eq. (5).

$$\vec{v}_i = gBestMem_j + F.(\vec{x}_i - \vec{x}_{r1}) + \lambda_{rand}.(\vec{x}_{r2} - \vec{x}_{r3}) \tag{5}$$

where $\vec{x}_{r1}$, $\vec{x}_{r2}$, and $\vec{x}_{r3}$ are three different individuals that are randomly selected among the individuals residing in cell j. F specifies the greediness of the search algorithm towards the current best point and λ_{rand} is a uniform random variable in range [0,1].

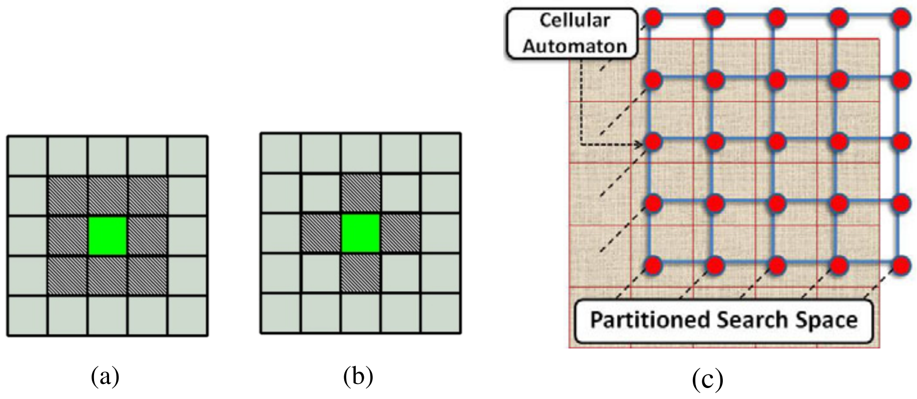

Fig. 1. (a) Moore neighborhood (b) Von Neumann neighborhood (c) A 2-D search space partitioned by a cellular automaton with 5^5 cells

3.3 Maintaining Diversity of Population

In CellularDE, to prevent convergence of many individuals to a region and maintain diversity of the population in the search space, the number of individuals in each cell is limited to a specific value, referred to as cell capacity (θ). When the number of individuals in a cell exceeds the cell capacity, the worst individuals in the cell are moved to new randomly selected cells such that the size of the population in the cell reduces to the cell capacity. Since the connections in the cellular automata are local, an individual, which is selected to move to a random cell, can only move one cell at a time. Until this individual arrives to its destination cell, it will be inactive, i.e. it will not participate in the evolution process, and hence it does not require any fitness evaluation.

3.4 Dealing with the Change

CellularDE detects a change in the environment by re-evaluating *cellBestMem* of each cell. If the fitness of *cellBestMem* has been changed since its last evaluation, a change is detected.

Upon detecting a change in environment, the fitness of every individual will be re-evaluated and *gBestMem* of all cells will be cleared. However, in order to use the previous search efforts of the population, each cell preserves *cellBestMem* after re-evaluation.

Moreover, after a change is detected in the environment, the population perform local search for the next LS_{num} iterations. To perform the local search, corresponding to each individual in cell *i*, a new individual is generated and positioned randomly in a hyper sphere with the radius of LS_r centered at $gBestMem_i$. If the fitness of the new individual is better than the old one, it will replace the old individual. This local search helps the individuals around each peak to follow the peak and find the new position of peak quickly. The detailed steps of the proposed CellullarDE are summarized in Algorithm 1.

Algorithm 1. CellularDE

```
Initialize a cellular automaton with $C^D$ equal-sized cells
Initialize population  of size $P_{num}$ randomly in the cellular automaton

do
    for all cell$_i$ in cellular automaton do
        Update $cellBestMem_i$ according to eq. (3)
        Update $gBestMem_i$ according to eq. (4)
        if there was a change in the past $LS_{num}$ iterations Then
            for all $individual_m$ in cell $i$ do
                Set $individual_m$ to a random position in a hyper-sphere with radius $LS_r$ cen-
                    tered at $gbestMem_i$.
            end-for
        else
            Evolve active population in $cell_i$ by DE using the mutant vector eq. (5)
        end-if

        while population of $cell_i > \theta$
          Re-initialize the worst active individual to a random cell in the cellular automata
        end-while

        if a change is detected Then
            Re-evaluate $cellMem_i$
            Clear $gBestMem_i$
            for all $individual_m$ in $cell_i$ do
                    Re-evaluate $individual_m$
            end-for
        end-if
    end-for
until a termination condition is met
```

4 Experiments

4.1 Moving Peaks Benchmark

Moving Peaks Benchmark (MPB) [21-22] is widely used in the literature to evaluate the performance of optimization algorithms in dynamic environments [10]. This benchmark defines several moving peaks in a multi-dimensional space which the height, width, and position of the peaks can be changed periodically.

The default parameter setting of MPB, known as the scenario II, is presented in Table 1. In MPB, 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 heights of the peaks. Parameter A denotes the range of the search space for all dimensions.

Table 1. Parameters of Moving Peaks Benchmark (MPB) for scenario II

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

In order to measure performance of an algorithm, offline error is used which is defined as the average fitness error of the best position found by the algorithm at every point in time.

$$Offline\ Error = \frac{1}{FEs}\sum_{t=1}^{FEs}\left(f\ (bestSolution(t)) - f\ (globalOptimum(t))\right) \tag{6}$$

where *FEs* is the maximum fitness evaluation, and *bestSolution*(*t*) and *globalOptimum*(*t*) are the best position found by the algorithm and the global optimum at the t^{th} fitness evaluation, respectively.

4.2 Experimental Settings

The results of the proposed algorithm is compared with DynDE [7], cellular PSO [16-17], FMSO [18] and mQSO [19-20]. To the best of our knowledge, DynDE is the best-performing DE optimization algorithms introduced for dynamic environments and cellular PSO [16-17] which shares the idea of embedding a cellular automaton in the search space. FMSO [18] and mQSO [19-20] are dynamic optimization algorithms recently introduced and have shown good performance in dynamic environments [10, 20].

For all experiments, parameters of FMSO, MQSO, DynDE, cellular PSO are set to the values reported in [18], [19], [7] and [16-17], respectively. The value of the parameters of CellularDE is selected so that the proposed algorithm performs best for the scenario II of MPB, introduced above. In CellularDE, the cellular automata partition the search space into 10^5 cells, i.e. each dimension is divided into 10 partitions to maximize the probability of cells containing one peak. Moreover, in the cellular automata, the Moore neighborhood with radius of 2 cells is used, which is large enough to maintain the exploration of the sub-populations in a neighborhood and is small enough to prevent the convergence of the population to a local optimum. The population size (P_{num}) is considered 100 and the capacity of the cells (θ) is set to 10 individuals per cell. This way, each of the 10 peaks in an environment defined by the scenario II of MPB can be equally exploited by 10 individuals, if they are located in 10 different cells. Mutation (F) and crossover (C_r) parameter of DE are empirically set to 0.2 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 6 and 1.0, respectively.

4.3 Experimental Results

All experiments were performed for 500,000 fitness function evaluations. The average offline errors of the algorithms in 100 runs with 95% confidence interval for various dynamic environments are depicted in tables 3 to 5. For each environment, 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.

Table 2. Offline errors for different number of peaks (*f* = 1000)

m	CellularDE	DynDE	Cellular PSO	FMSO	mQSO
1	**4.98±0.35**	16.84±9.39	7.90±0.47	14.42±0.4	13.30±0.40
5	**3.96±0.04**	5.32±0.44	5.81±0.21	10.59±0.2	5.76±0.15
10	**3.98±0.03**	4.25±0.05	5.70±0.15	10.40±0.1	5.43±0.10
20	**4.53±0.02**	5.34±0.02	5.92±0.18	10.33±0.1	5.84±0.09
30	**4.77±0.02**	5.80±0.04	6.07±0.16	10.06±0.1	6.12±0.11
40	**4.87±0.02**	6.09±0.02	5.95±0.15	9.85±0.11	6.36±0.11
50	**4.87±0.02**	6.22±0.02	6.01±0.15	9.54±0.11	6.61±0.15
100	**4.85±0.02**	6.55±0.03	6.04±0.13	8.77±0.09	6.62±0.11
200	**4.46±0.01**	6.49±0.02	5.90±0.13	8.06±0.07	6.59±0.10

Table 3. Offline errors for different number of peaks (*f* = 2500)

m	CellularDE	DynDE	Cellular PSO	FMSO	mQSO
1	**2.38±0.78**	7.08±2.08	4.91±0.28	6.29±0.20	6.03±0.22
5	**2.12±0.02**	3.14±0.11	2.95±0.20	5.03±0.12	2.98±0.09
10	**2.42±0.02**	2.81±0.05	2.97±0.15	5.09±0.09	3.05±0.09
20	**3.05±0.04**	3.83±0.05	3.51±0.14	5.32±0.08	4.05±0.08
30	**3.29±0.03**	4.32±0.05	3.87±0.12	5.22±0.08	4.53±0.17
40	**3.43±0.03**	4.54±0.05	3.89±0.10	5.09±0.06	4.61±0.13
50	**3.44±0.02**	4.71±0.05	4.16±0.15	4.99±0.06	4.86±0.12
100	**3.36±0.01**	4.90±0.05	4.18±0.11	4.60±0.05	5.13±0.13
200	**3.13±0.01**	4.91±0.04	4.04±0.09	4.34±0.04	5.10±0.15

Table 4. Offline errors for different number of peaks (*f* = 5000)

m	CellularDE	DynDE	Cellular PSO	FMSO	mQSO
1	**1.53±0.07**	4.06±0.49	3.46±0.22	3.44±0.11	3.00±0.09
5	**1.50±0.04**	2.22±0.15	1.79±0.12	2.94±0.07	1.70±0.09
10	**1.64±0.03**	2.26±0.05	1.84±0.08	3.11±0.06	1.96±0.08
20	**2.46±0.05**	3.14±0.07	**2.63±0.11**	3.36±0.06	3.11±0.07
30	**2.62±0.05**	3.49±0.60	2.91±0.10	3.28±0.05	3.61±0.08
40	**2.76±0.05**	3.82±0.09	3.16±0.11	3.26±0.04	3.88±0.07
50	**2.75±0.05**	4.05±0.07	3.23±0.11	3.22±0.05	3.99±0.13
100	**2.73±0.03**	4.18±0.09	3.43±0.10	3.06±0.04	4.30±0.08
200	**2.61±0.02**	4.06±0.05	3.38±0.09	2.84±0.03	4.32±0.09

The results of the experiments show that CellularDE outperforms all other tested algorithms in all experiments except one case. Moreover, as depicted in Fig. 2 for an environment with 50 peaks, CellularDE can find better solutions after most environment changes. This is because CellularDE, by imposing a limit on the number of

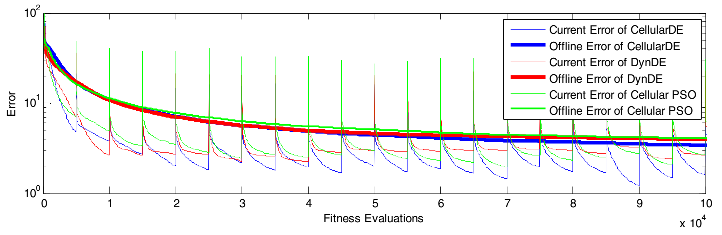

Fig. 2. The current error and the offline error for a dynamic environment with 50 peaks, *f*=5000

individuals in each cell, maintains the diversity of the individuals better than DynDE, thus performing a better exploration in the environment. In addition, CellularDE takes the advantage of local search capability of DE, which results in more powerful exploitation compared to cellular PSO.

Moreover, as the number of peaks in the environment increases, the offline error of all algorithms, except CellularDE, increases significantly. This is because in DynDE, mQSO, and FMSO the numbers of sub-populations are limited, hence they can only scout a few peaks in the environments. But CellularDE is able to cover many peaks simultaneously. Furthermore, the binomial crossover operation in CellularDE brings about a good exploration in a neighborhood, which leads to outperforming cellular PSO in the environments with many peaks.

5 Conclusion

In this paper, we proposed CellularDE, a cellular automata based differential evolution algorithm to tackle dynamic optimization problems. In CellularDE a cellular automaton is embedded into the search space and partitions the search space into cells. By imposing a limit on the number of individuals searching in each cell, the proposed algorithm maintains a balance between exploration and exploitation in the environment. Moreover, individuals in each neighborhood search for a peak together using the local information exchange between cells. In addition, in order to track the local optima after a change occurred in the environment, a local random search is performed in the few iterations after the change is detected.

Extensive experiments in various dynamic environments modeled by the moving peaks benchmarks were conducted to evaluate the performance of CellularDE. The results of the experiments show that CellularDE outperforms DynDE [7], cellular PSO [16-17], FMSO [18], and mQSO [19-20] in most tested environments which contain many peaks. In addition, it has been shown that CellularDE is more robust to the number of peaks in the environment than other tested algorithms.

References

1. Yang, S.: Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. Evolutionary Computation 16, 385–416 (2008)
2. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Männer, R., Manderick, B. (eds.) Proc. of the 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137–144 (1992)

3. Goldberg, D., Smith, R.: Nonstationary function optimization using genetic algorithms with dominance and diploidy, pp. 59–68. L. Erlbaum Associates Inc., Mahwah (1987)
4. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm, vol. 3, pp. 2246–2253. IEEE, Los Alamitos (2004)
5. Cobb, H.G., Grefenstette, J.J.: Genetic Algorithms for Tracking Changing Environments. In: Proceedings of the 5th International Conference on Genetic Algorithms, June 01, pp. 523–530 (1993)
6. Morrison, R., De Jong, K.: Triggered hypermutation revisited, vol. 2, pp. 1025–1032. IEEE, Los Alamitos (2002)
7. Mendes, R., Mohais, A.: DynDE: a differential evolution for dynamic optimization problems, vol. 3, pp. 2808–2815. IEEE, Los Alamitos (2005)
8. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Transactions on Evolutionary Computation 10, 440–458 (2006)
9. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation 9, 303–317 (2005)
10. Moser, I., Chiong, R.: Dynamic function optimisation with hybridised extremal dynamics. Memetic Computing 2, 137–148 (2010)
11. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11, 341–359 (1997)
12. Price, K., Storn, R., Lampinen, J.: Differential evolution: a practical approach to global optimization. Springer, Heidelberg (2005)
13. du Plessis, M., Engelbrecht, A.: Improved differential evolution for dynamic optimization problems, pp. 229–234. IEEE, Los Alamitos (2008)
14. Brest, J., Zamuda, A., Boskovic, B., Maucec, M., Zumer, V.: Dynamic optimization using self-adaptive differential evolution, pp. 415–422. IEEE, Los Alamitos (2009)
15. Kanlikilicer, A., Keles, A., Uyar, A.: Experimental analysis of binary differential evolution in dynamic environments, pp. 2509–2514. ACM, New York (2007)
16. Hashemi, A.B., Meybodi, M.: A multi-role cellular PSO for dynamic environments, pp. 412–417. IEEE, Los Alamitos (2009), `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5349615`
17. Hashemi, A.B., Meybodi, M.: Cellular PSO: A PSO for dynamic environments. In: Cai, Z., Li, Z., Kang, Z., Liu, Y. (eds.) ISICA 2009. LNCS, vol. 5821, pp. 422–433. Springer, Heidelberg (2009)
18. Li, C., Yang, S.: Fast multi-swarm optimization for dynamic optimization problems, vol. 7, pp. 624–628. IEEE, Los Alamitos (2008)
19. Blackwell, T., Branke, J.: Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10, 459–472 (2006)
20. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, pp. 29–49 (2007)
21. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems, vol. 3. IEEE, Los Alamitos (2002)
22. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer Academic Publishers, Norwell (2001)
23. Fredkin, E.: An informational process based on reversible universal cellular automata. Physica D: Nonlinear Phenomena 45, 254–270 (1990)