A New Particle Swarm Optimization Algorithm for Dynamic Environments

Masoud Kamosi[1], Ali B. Hashemi[2], and M.R. Meybodi[2]

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran
masoud.kamosi@gmail.com, {a_hashemi,mmeybodi}@aut.ac.ir

Abstract. Many real world optimization problems are dynamic in which global optimum and local optima change over time. Particle swarm optimization has performed well to find and track optima in dynamic environments. In this paper, we propose a new particle swarm optimization algorithm for dynamic environments. The proposed algorithm utilizes a parent swarm to explore the search space and some child swarms to exploit promising areas found by the parent swarm. To improve the search performance, when the search areas of two child swarms overlap, the worse child swarms will be removed. Moreover, in order to quickly track the changes in the environment, all particles in a child swarm perform a random local search around the best position found by the child swarm after a change in the environment is detected. Experimental results on different dynamic environments modelled by moving peaks benchmark show that the proposed algorithm outperforms other PSO algorithms, including FMSO, a similar particle swarm algorithm for dynamic environments, for all tested environments.

Keywords: Particle Swarm Optimization, Dynamic Environments.

1 Introduction

The particle swarm optimization algorithm (PSO) is introduced by Kennedy and Eberhart [1]. In PSO, a potential solution for a problem is considered as a bird, which is called a particle, flies through a D-dimensional space and adjusts its position according to its own experience and other particles'. In PSO, a particle is represented by its position vector *p* and its velocity vector *v*. In time step *t*, particle *i* calculates its new velocity then updates its position according to Equation (1) and Equation (2), respectively.

$$v_i(t+1) = wv_i(t) + c_1r_1(pbest_i - p_i(t)) + c_2r_2(gbest - p_i(t)) \quad (1)$$

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

where w is the inertial weight, and c_1 and c_2 are positive acceleration coefficients used to scale the contribution of cognitive and social components, respectively. $pbest_i$ is the best position that particle i has been visited. $gbest$ is the best position found by all particles in the swarm. r_1 and r_2 are uniform random variables in range [0,1].

The standard particle swarm optimization algorithms and its variants [2, 3] have been performed well for static environment. Yet, it is shown that PSO, like evolutionary algorithms, must be modified to not only find the optimal solution in a short time but also to be capable of tracking the solution after a change in the environment occurred. In order to have these capabilities, two important problems should be addressed for designing a particle swarm optimization algorithm for dynamic environments: outdated memory and diversity loss[4]. Outdated memory refers to the condition in which memory of the particles, that is the best location visited in the past and its corresponding fitness, may no longer be valid after a change in the environment [5]. Outdated memory problem is usually solved in one of these two ways: re-evaluating the memory [6] or forgetting the memory [7]. Diversity loss occurs when the swarm converges on a few peaks in the landscape and loses its ability to find new peaks, which is required after the environment changes. There are two approaches to deal with diversity loss problem. In the first approach, a diversity maintenance mechanism runs periodically (or when a change is detected) and re-distributes the particles if the diversity falls below a threshold[8]. In the second approach, diversity is always monitored and as soon as it falls below a threshold, the swarm will be re-diversified.

In this paper, we propose a new multi-swarm algorithm for dynamic environments which address the diversity loss problem by introducing two types of swarm: a parent swarm, which explores the search space to find promising area containing local optima and several non-overlapping child swarms, each of which is responsible for exploiting a promising area found by the parent swarm. In the proposed algorithm, both parent and child swarms use the standard PSO[9] with the local neighborhood [10], in which a particle is affected by the best experience of its swarm rather than the best experience of all particles in the environment. Moreover, in order to track the changing local optima, after a change is detected in the environment particles in each child swarms perform a random search around the best position of that child swarm. Extensive experiments show that for all tested dynamic environments the proposed algorithm outperforms all tested PSO algorithms, including FMSO[11], with which shares the idea of utilizing a parent swarms and child swarms.

The rest of this paper is organized as follows: Section 2 briefly reviews particle swarm optimization algorithms for dynamic environments introduced in the literature. The proposed algorithm is presented in section 3. Section 4 presents the experimental results of the proposed algorithm along with comparison with alternative approaches from the literature. Finally, section 5 concludes this paper.

2 PSO in Dynamic Environments

Hu and Eberhart proposed re-randomization PSO for optimization in dynamic environments[8] in which some particles randomly are relocated after a change is detected or when the diversity is lost, to prevent losing the diversity. Li and Dam [12] showed that a grid-like neighborhood structure used in FGPSO[13] can perform better than RPSO in high dimensional dynamic environments by restricting the information sharing and preventing the convergence of particles to the global best position, thereby enhancing population diversity. Janson and Middendorf proposed HPSO, a tree-like structure hierarchical PSO [14], and reported improvements over standard PSO for dynamic environments. They also suggested Partitioned Hierarchical PSO in which a hierarchy of particles is partitioned into several sub-swarms for a limited number of generations after a change in the environment is detected [15]. Lung and Dumitresc [16] used two collaborating populations of equal size, one swarm is responsible for preserving the diversity of the particles by using a crowding differential evolutionary algorithm [17] while the other keeps track of global optimum with a PSO algorithm.

Blackwell and Bentley presented a repulsion mechanism in using the analogy of atom particles [18, 19]. In their model, a swarm is comprised of charged and neutral particles. The charged particles repel each other, leading to a cloud of charged particles orbiting a contracting, neutral, PSO nucleus. Moreover, Blackwell et al. extended the idea of charged particles to a quantum model and presented a multi-swarm method [4, 20, 21].

Du and Li [22] suggested an algorithm which divides particles into two parts. The first part uses a standard PSO enhanced by a Gaussian local search to find the global optimum quickly and the second part extends the searching area of the algorithm and patrols around the first part to track the changed global optimum which possibly escaped from the coverage of the first part. Although their algorithm performs well in the environments with one or two local optima, it cannot find the global optimum when the environment has more local optima.

Li and Yang proposed a fast multi-swarm method (FMSO) which maintains the diversity through the run [11]. To meet this end two type 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 programming 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. Moreover, In [23], the authors introduced a clustering particle swarm optimizer in which a clustering algorithm partitions the swarm into several sub-swarms each searching for a local optimum.

Liu et al. [24] introduced compound particle swarm optimization (CPSO) utilizing a new type of particles which helps explore the search space more comprehensively after a change occurred in the environment. In another work, they used composite particles which help quickly find the promising optima in the search space while maintaining the diversity by a scattering operator[25].

Hashemi and Meybodi introduced cellular PSO, a hybrid model of cellular automata and PSO [26]. In cellular PSO, a cellular automaton partitions the search space into cells. At any time, in some cells of the cellular automaton a group of particles search for a local optimum using their best personal experiences and the best solution found in their neighborhood cells. To prevent losing the diversity, a limit on the number of particles in each cell is imposed. Furthermore, to track the changes in the environment, in [27] particles in cellular PSO change their role to quantum particles and perform a random search around the previously found optima for a few iterations after a change is detected in the environment.

3 The Proposed Algorithm

The proposed multi-swarm algorithm consists of a parent swarm and some child swarms. The parent swarm is responsible for finding promising area in the search space upon which a child swarm is created to exploit the new found promising area. The proposed algorithm works as follows.

After initializing the parent swarm, the particles in the parent swarm begin searching in the search space. At each iteration, velocity and position of a particle *i* in the parent swarm is updated using its best personal position ($pbest_i$) and the best position found by the parent swarm ($cbest_{parent}$) according to (1) and (2), respectively. If the fitness of the new position of particle *i* is better than its best personal position ($pbest_i$), $pbest_i$ will be updated to the new position. Likewise, the best position found in the parent swarm ($cbest_{parent}$) will be updated. Afterwards, the distance between particle *i* and the attractor of each child swarm *c*, i.e. the best position found by a child swarm *c* ($cbest_c$), is calculated. If the distance between particle *i* and the attractor of a child swarm *c* is less than *r*, the attractor of the child swarm *c* will be updated to the position of particle *i*. Then, particle *i* will be reinitialized. When all particles in the parent swarm are updated, if the best position found in the parent swarm ($cbest_{parent}$) is improved, a new child swarm will be created with $cbest_{parent}$ as its attractor. If there are *m* particles in the parent swarm whose distances to the attractor of the newly created child swarm are less than *r*, these particles will be moved to the new child swarm, at the same time *m* new particles will be created and initialized in the parent swarm. If the number of particles moved to the newly created child swarm (*m*) is less than the number of particles determined for a child swarm (π), π-*m* particles for the child swarm will be created and initialized in a hypersphere with radius *r*/3 centered at the child swarm's attractor. Afterwards, all particles in every child swarm *c* update their velocity and position according to (3) and (2), respectively. Then, the personal best position (*pbest*) for all child particles and the child swarms' best position ($cbest_j$) will be updated.

$$v_i(t+1) = wv_i(t) + c_1r_1(pbest_i - p_i(t)) + c_2r_2(cbest_c - p_i(t)) \tag{3}$$

Since searching an area with more than one child swarm is not very useful, at the end of each iteration every two child swarms are checked whether they are searching in the same area or not. Two child swarms are searching in the same area or they are colliding, if the Euclidian distance between their attractors is less than a specified threshold r_{excl}. If a collision between two child swarms is detected, the worse child swarm whose attractor is less fit than the other's, will be destroyed.

In the proposed algorithm, when an environment change is detected, particles in the parent swarm re-evaluate their positions and reset their best personal position (*pbest*) to their current position. However, the particles in the child swarms change their behaviors in the following iteration after a change is detected in the environment. They will set their new positions to a random location in a hypersphere with radius r_s centered at their swarm's attractor. Then they will reset their best personal positions (*pbest*) to their new position and update the child swarms attractor.

4 Experimental Study

In this section, we first describe moving peaks benchmark [28] on which the proposed algorithms is evaluated. Then, experimental settings are described. Finally, experimental results of the proposed algorithm are presented and compared with alternative approaches from the literature.

Algorithm 1. Proposed multi-swarm optimization algorithm

```
Initialization the parent swarm
repeat
  if a change is detected in the environment then
    for each particle $i$ in the parent swarm do
      $pbest_i = p_i$
    end-for
    Set $cbest_{parent}$ to the best position found by the particles in the parent swarm
    for each child swarm $c$ do
      for each particle $i$ in the child swarm $c$ do
        $p_i$=a random position in a hypersphere with radius $r_s$ centered at $cbest_c$
        $pbest_i = p_i$
      end-for
      Set $cbest_c$ to the best position found by the particles in child swarm $c$
    end-for
    continue
  end-if
  for each particle $i$ in the parent swarm do
    Update position of particle $i$ according to eq. 1 and eq. 2.
    Update $pbest_i$
    for each swarm $c$ do
      if $distance(p_i, cbest_c) < r$ then
        if $f(p_i) > f(cbest_c)$ then
          $cbest_c = p_i$
        end-if
        Reinitialize particle $i$
      end-if
    end-for
  end-for
  Set $cbest_{parent}$ to the best position found by the particles in the parent swarm
  if $cbest_{parent}$ is updated then
    // create a new child swarm $\nu$ around the new position found by the parent swarm
    $cbest_\nu = cbest_{parent}$
    for each particle $i$ in parent swarm do
      if $distance(p_i, cbest_\nu) < r$ then
        if $|\nu| < \pi$ then
          copy particle $i$ to the child swarm $\nu$
        end-if
        Initialize particle $i$
      end-if
    end-for
    while $|\nu| < \pi$
      Create a new particle in the child swarm $\nu$ within a radius $r/3$ centered at
          $cbest_\nu$
    end-while
  end-if
  for child swarm $c$ do
    for each particle $i$ in the child swarm $c$ do
      Update position of particle $i$ according to eq. (3) and eq. (2)
      Update $pbest_i$
    end-for
    Set $cbest_c$ to the best position found by the particles in child swarm $c$
  end-for
  for each pair of child swarms $(k, l)$, $k \neq l$ do
    if $distance(cbest_k, cbest_l) < r_{excl}$ then
        Destroy the child swarm whose $cbest$ has a less fitness value.
    end-if
  end-for
until a maximum number of fitness evaluations is reached
```

Moving peaks benchmark (Fig. 1) [28] is widely used in the literature to evaluate the performance of optimization algorithms in dynamic environments [29]. In this benchmark, there are some peaks in a multi-dimensional space, where the height, width, and position of each peak alter when the environment changes. Unless stated otherwise, the parameters of the moving peaks benchmark are set to the values presented in Table 1.

In order to measure the efficiency of the algorithms, offline error that is the average fitness of the best position found by the swarm at every point in time is used [30].

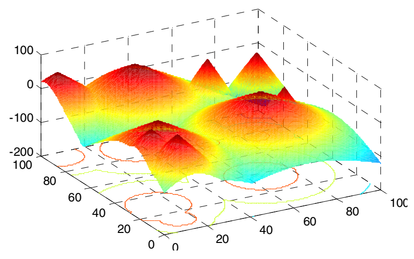

Fig. 1. Moving peaks benchmark

Table 1. Default settings of moving peaks benchmark

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

4.1 Experimental Settings

For the proposed algorithms the acceleration coefficients c_1 and c_2 are set to 1.496180 and the inertial weight w is set to 0.729844 [31]. The number of particles in the parent swarm and the child swarms (π) are set to 5 and 10 particles, respectively. The radius of the child swarms (r), the minimum allowed distance between two child swarm (r_{excl}) and the radius of quantum particles (r_s) are set to 30.0, 30.0, and 0.5, respectively. The proposed algorithm is compared with mQSO[4] and FMSO [11], and cellular PSO [26, 27]. For mQSO we adapted a configuration $10(5+5^q)$ which creates 10 swarms with 5 neutral (standard) particles and 5 quantum particles with r_{cloud}=0.5 and r_{excl}= r_{conv} =31.5, as suggested in [4, 21]. For FMSO, there are at most 10 child swarms each has a radius of 25.0. The size of the parent and the child swarms are set to 100 and 10 particles, respectively[11]. For cellular PSO, a 5-Dimensional cellular automaton with 10^5 cells and Moore neighborhood with radius of two cells is embedded into the search space. The maximum velocity of particles is set to the neighborhood radius of the cellular automaton and the radius for the random local search (r) is set to 0.5 for all experiments. The cell capacity θ is set to 10 particles for every cell[26, 27].

4.2 Experimental Results

For all algorithms we reported the average offline error and 95% confidence interval for 100 runs. Offline error of the proposed algorithm, mQSO10($5+5^q$) [4], FMSO [11], and cellular PSO[26, 27] for different dynamic environment is presented in table 2 to table 5. For each environment, result of the best performing algorithm(s) with 95% confidence is printed in bold. When the offline errors of the best two (or more) algorithms are not statistically different, all are printed in bold.

As depicted in the table 2 to table 5, the proposed algorithm outperforms other tested PSO algorithms, including FMSO, for all environments. Moreover, the difference between offline error of the proposed algorithm and the next best algorithm decreases as the environment changes less frequently from *f*=500 (table 2) to *f*=10000 (table 5). This is because the proposed algorithm quickly finds better solutions than other algorithms after a change occurs in the environment, especially at the early iterations (Fig. 2).

Furthermore, in the proposed algorithm the number of child swarms converges to the number of peaks in the environment (Fig. 3). This will help the proposed algorithm to track the changes more effectively since there will be a child swarm on each peak.

Table 2. Offline error ±Standard Error for f =500

m	Proposed algorithm	mQSO10	FMSO	CellularPSO
1	**5.46**±0.30	33.67±3.4	27.58±0.9	13.4±0.74
5	**5.48**±0.19	11.91±0.7	19.45±0.4	9.63±0.49
10	**5.95**±0.09	9.62±0.34	18.26±0.3	9.42±0.21
20	**6.45**±0.16	9.07±0.25	17.34±0.3	8.84±0.28
30	**6.60**±0.14	8.80±0.21	16.39±0.4	8.81±0.24
40	**6.85**±0.13	8.55±0.21	15.34±0.4	8.94±0.24
50	**7.04**±0.10	8.72±0.20	15.54±0.2	8.62±0.23
100	**7.39**±0.13	8.54±0.16	12.87±0.6	8.54±0.21
200	**7.52**±0.12	8.19±0.17	11.52±0.6	8.28±0.18

Table 3. Offline error ±Standard Error for f =1000

m	Proposed algorithm	mQSO10	FMSO	CellularPSO
1	**2.90**±0.18	18.6±1.6	14.42±0.4	6.77±0.38
5	**3.35**±0.18	6.56±0.38	10.59±0.2	5.30±0.32
10	**3.94**±0.08	5.71±0.22	10.40±0.1	5.15±0.13
20	**4.33**±0.12	5.85±0.15	10.33±0.1	5.23±0.18
30	**4.41**±0.11	5.81±0.15	10.06±0.1	5.33±0.16
40	**4.52**±0.09	5.70±0.14	9.85±0.11	5.61±0.16
50	**4.57**±0.08	5.87±0.13	9.54±0.11	5.55±0.14
100	**4.77**±0.08	5.83±0.13	8.77±0.09	5.57±0.12
200	**4.76**±0.07	5.54±0.11	8.06±0.07	5.50±0.12

Table 4. Offline error ±Standard Error for *f* =5000

m	Proposed algorithm	mQSO10	FMSO	CellularPSO
1	**0.56**±0.04	3.82±0.35	3.44±0.11	2.55±0.12
5	**1.06**±0.06	1.90±0.08	2.94±0.07	1.68±0.11
10	**1.51**±0.04	1.91±0.08	3.11±0.06	1.78±0.05
20	**1.89**±0.04	2.56±0.10	3.36±0.06	2.61±0.07
30	**2.03**±0.06	2.68±0.10	3.28±0.05	2.93±0.08
40	**2.04**±0.06	2.65±0.08	3.26±0.04	3.14±0.08
50	**2.08**±0.02	2.63±0.08	3.22±0.05	3.26±0.08
100	**2.14**±0.02	2.52±0.06	3.06±0.04	3.41±0.07
200	**2.11**±0.03	2.36±0.05	2.84±0.03	3.40±0.06

Table 5. Offline error ±Standard Error for *f* =10000

m	Proposed algorithm	mQSO10	FMSO	CellularPSO
1	**0.27**±0.02	1.90±0.18	1.90±0.06	1.53±0.12
5	**0.70**±0.10	1.03±0.06	1.75±0.06	0.92±0.10
10	**0.97**±0.04	1.10±0.07	1.91±0.04	1.19±0.07
20	**1.34**±0.08	1.84±0.08	2.16±0.04	2.20±0.10
30	**1.43**±0.05	2.00±0.09	2.18±0.04	2.60±0.13
40	**1.47**±0.06	1.99±0.07	2.21±0.03	2.73±0.11
50	**1.47**±0.04	1.99±0.07	2.60±0.08	2.84±0.12
100	**1.50**±0.03	1.85±0.05	2.20±0.03	2.93±0.09
200	**1.48**±0.02	1.71±0.04	2.00±0.02	2.88±0.07

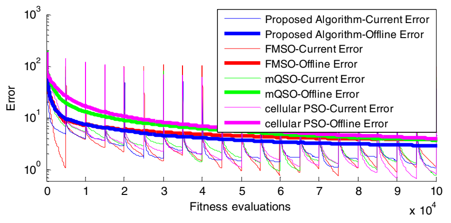

Fig. 2. Convergence of the offline error

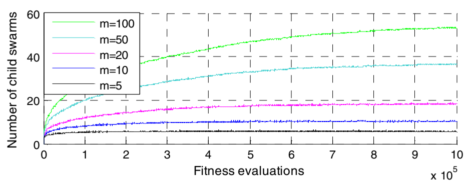

Fig. 3. Convergence of the number of child swarms

4.3 Effect of Varying the Size of the Parent Swarm

This experiment examines the effect of the size of the parent swarm, i.e. the number of particles in the parent swarm, on offline error. The visualized results in Fig. 4 and Fig. 5 depicts that when there are many peaks in the environment (*m*=50 or *m*=100) or when the environment changes slowly (*f*=5000 or *f*=10000) the size of the parent swarm does not affect offline error significantly. For other environments, offline error escalates by increasing the size of the parent swarm. The less peaks exist in the environment or the more frequently the changes occur, the more offline error will be affected by increasing the size of the parent swarm. However, if there are 5 peaks in the environment and *f*=500 (Fig. 5), the offline error slightly decreases when the size of the parent swarm increases from 1 particle to 5 particles.

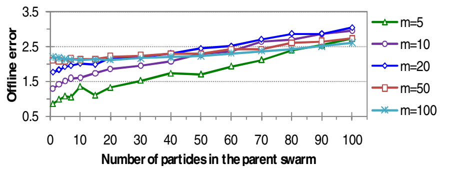

Fig. 4. The effect of the size of the parent swarm on the offline error

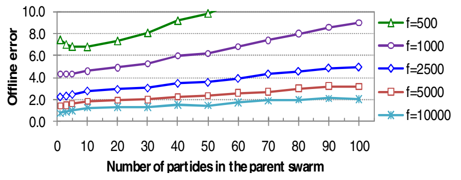

Fig. 5. The effect of the size of the parent swarm on the offline error

4.4 Effect of Varying the Size of the Child Swarms

This experiment examines the effect of the size of the child swarms (π), i.e. the number of particles in each child swarm, on offline error. As depicted in Fig. 6 and Fig. 7, the optimal value for the number of particles in the child swarms for different environments is 10 particles. In addition, either increasing or decreasing the number of particles in the child swarms from its optimal value (π=10) increases offline error monotonically. The reason is that existence of many particles in the child swarms not only does not help finding better solutions but also consumes precious function evaluations. Conversely, when there are too few particles in the child swarms, they cannot find the peaks quickly enough thereby increasing offline error.

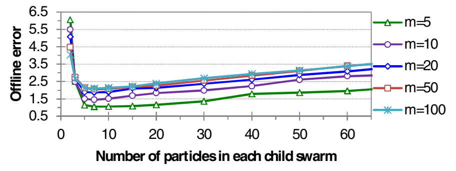

Fig. 6. The effect of the number of particles in each child swarm on the offline error

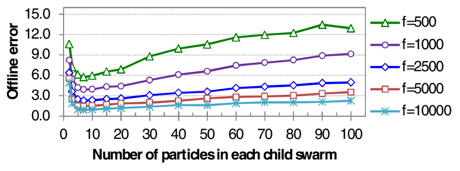

Fig. 7. The effect of the number of particles in each child swarm on the offline error

5 Conclusion

In this paper, we proposed a new multi-swarm PSO algorithm for dynamic environment environments. The proposed PSO consists of a parent swarm and some child swarms. The parent swarm is responsible for exploring the search space and finding promising regions containing a local optimum. Child swarms exploit the promising regions found by the parent swarm. To prevent redundant search in the same area, two mechanisms have been adapted. If a parent particle collides with a child swarm, it will be reinitialized. If two child swarms collide the one with the least fitness will be removed. In addition, to track the local optima after detecting a change in the environment, particles in each child swarm temporarily change their behavior to quantum particles and perform a random search around the child swarm's attractor. Results of the experiments show that for all tested environments the proposed algorithm outperforms all tested PSO algorithms, including FMSO, a previously presented multi-swarm algorithm with the similar approach.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International conference on neural networks, Piscataway, NJ, USA, vol. IV, pp. 1942–1948 (1995)
2. del Valle, Y., Venayagamoorthy, G.K., Mohagheghi, S., Hernandez, J.C., Harley, R.G.: Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. IEEE Transactions on Evolutionary Computation 12, 171–195 (2008)
3. Hashemi, A.B., Meybodi, M.R.: A note on the learning automata based algorithms for adaptive parameter selection in PSO. Applied Soft Computing 11, 689–705 (2011)
4. Blackwell, T., Branke, J.: Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10, 459–472 (2006)
5. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, pp. 29–49 (2007)
6. Eberhart, R.C., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: IEEE Congress on Evolutionary Computation, Seoul, Korea, vol. 1, pp. 94–100 (2001)
7. Carlisle, A., Dozier, G.: Adapting particle swarm optimization to dynamic environments. In: International Conference on Artificial Intelligence, Las Vegas, NV, USA, vol. 1, pp. 429–434 (2000)
8. Hu, X., Eberhart, R.C.: Adaptive particle swarm optimization: detection and response to dynamic systems. In: IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, vol. 2, pp. 1666–1670 (2002)
9. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, USA, pp. 120–127 (2007)
10. Eberhart, R.C., Dobbins, R., Simpson, P. (eds.): Evolutionary Computation Implementations. Computational Intelligence PC Tools, pp. 212–226. Morgan Kaufmann, San Francisco (1996)
11. Li, C., Yang, S.: Fast Multi-Swarm Optimization for Dynamic Optimization Problems. In: Fourth International Conference on Natural Computation, Jinan, Shandong, China, vol. 7, pp. 624–628 (2008)

12. Li, X., Dam, K.H.: Comparing particle swarms for tracking extrema in dynamic environments. In: IEEE Congress on Evolutionary Computation, Canberra, Australia, pp. 1772–1779 (2003)
13. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Evolutionary Computation Congress, Honolulu, Hawaii, USA, pp. 1671–1676 (2002)
14. Janson, S., Middendorf, M.: A Hierarchical Particle Swarm Optimizer for Dynamic Optimization Problems. Applications of Evolutionary Computing, 513–524 (2004)
15. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer for noisy and dynamic environments. Genetic Programming and Evolvable Machines 7, 329–354 (2006)
16. Lung, R.I., Dumitrescu, D.: A collaborative model for tracking optima in dynamic environments. In: IEEE Congress on Evolutionary Computation, Singapore, pp. 564–567 (2007)
17. Thomsen, R.: Multimodal optimization using crowding-based differential evolution. In: IEEE Congress on Evolutionary Computation, Portland, Oregon, USA, pp. 1382–1389 (2004)
18. Blackwell, T.: Swarms in Dynamic Environments. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, p. 200. Springer, Heidelberg (2003)
19. Blackwell, T.M., Bentley, P.J.: Dynamic search with charged swarms. In: Genetic and evolutionary computation conference, pp. 19–26. Morgan Kaufmann Publishers Inc., New York (2002)
20. Blackwell, T., Branke, J.: Multi-swarm Optimization in Dynamic Environments. In: Applications of Evolutionary Computing, pp. 489–500. Springer, Heidelberg (2004)
21. Blackwell, T., Branke, J., Li, X.: Particle Swarms for Dynamic Optimization Problems. Swarm Intelligence, 193–217 (2008)
22. Du, W., Li, B.: Multi-strategy ensemble particle swarm optimization for dynamic optimization. Information Sciences 178, 3096–3109 (2008)
23. Li, C., Yang, S.: A clustering particle swarm optimizer for dynamic optimization. In: IEEE Congress on Evolutionary Computation, pp. 439–446 (2009)
24. Liu, L., Wang, D., Yang, S.: Compound Particle Swarm Optimization in Dynamic Environments. Applications of Evolutionary Computing, 616–625 (2008)
25. Liu, L., Yang, S., Wang, D.: Particle Swarm Optimization With Composite Particles in Dynamic Environments. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 1–15 (2010)
26. Hashemi, A.B., Meybodi, M.R.: Cellular PSO: A PSO for Dynamic Environments. Advances in Computation and Intelligence, 422–433 (2009)
27. Hashemi, A.B., Meybodi, M.R.: A multi-role cellular PSO for dynamic environments. In: 14th International CSI Computer Conference, Tehran, Iran, pp. 412–417 (2009)
28. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problem. In: 1999 Congress on Evolutionary Computation, Washington D.C., USA, vol. 3, pp. 1875–1882 (1999)
29. Moser, I.: All Currently Known Publications On Approaches Which Solve the Moving Peaks Problem. Swinburne University of Technology, Melbourne, Australia (2007)
30. Branke, J., Schmeck, H.: Designing evolutionary algorithms for dynamic optimization problems. Advances in evolutionary computing: theory and applications, 239–262 (2003)
31. van den Bergh, F.: An Analysis of Particle Swarm Optimizers, Ph.D. dissertation, Department of Computer Science, University of Pretoria, Pretoria, South Africa (2002)