Improved speciation-based Firefly Algorithm in dynamic and uncertain environment

Babak Nasiri[1], MohammadReza Meybodi[2]

[1] *Department of Computer Engineering and Information Technology, Islamic Azad University, Qazvin Branch, Qazvin, Iran, Nasiri.babak@qiau.ac.ir*

[2] *Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran, meybodi@aut.ac.ir*

Abstract: Many real-world optimization problems are dynamic in nature. The applied algorithm in this environment can pose serious challenges, especially when the search space is multimodal with multiple, time-varying optima. To address these challenges, this paper investigates a speciation-based firefly algorithm to enhance the population diversity with aim of generating several populations in different areas in the landscape without knowing the number of optima in it. To improve the performance of the algorithm, multiple adaptation techniques have been used: automatically adapting the number of sub-swarm in environment with speciation-based firefly algorithm, adapting number of fireflies in each sub-swarm and also adapting number of active fireflies during the run based on the domain knowledge of fitness landscape. An experimental study is presented with a well-known Moving Peaks Benchmark (MPB) problem to evaluate the performance of the proposed algorithm with other leading algorithms in dynamic environments. The results show the efficiency of the proposed algorithm in comparison with other state-of-the-art algorithms on different configurations and scenarios of this benchmark.

Keywords: Firefly Algorithm, Multi-swarm, Dynamic Optimization, speciation, dynamic environments.

1. INTRODUCTION

Many real-world optimization problems are dynamic in the nature. That's why, in the past decade, there has been a growing interest in proposing new approaches for optimization in dynamic environment[1]. Generally, the goals and challenges are completely different for optimization in dynamic environments rather than the static ones. The only goal in static optimization problems is finding the global optima as close as possible. In dynamic ones, however, tracking optima over the time is other goal of optimization, as well. Furthermore, there are some more challenges in dynamic environments due to happening changes in environments compared with the static ones. However, out-dated memory and diversity loss are two of the most important ones.

Generally speaking, so far, different methods have been proposed for dynamic optimization. A great category of such methods is Meta-heuristic ones, more specifically evolutionary computing and swarm intelligence methods. In this category, in each of iterations, the population changes to move toward a better one. Due to this intrinsic capability, these methods are more suitable for optimization in dynamic environments. Therefore, various methods based on evolutionary computing techniques and swarm intelligence techniques have been proposed for dynamic optimization in the last two decades, such as GA[2-5], AC[6, 7], EDA[8], PSO[9-17], and their composition.

Also, the authors proposed a speciation-based firefly algorithm in [18] by using the intrinsic capability of firefly algorithm in converging to more than one optima at a time, and creating the sub-swarms dynamically the same as speciation based methods. They proposed two different types of swarms with different search behaviour in the fitness landscape. One swarm is responsible for global search and the other one for the local search. Although, this approach seems ideal, it does suffer some shortcomings. First, some redundant fitness evaluations around the local optima which have not any influence on decreasing error of the optimization at all. Second, inability of the algorithm to cover a number of local optima due to the fixed number of fireflies in each speciation and also the weakness of the algorithm in global search.

In this paper, a new speciation method based on firefly algorithm has been proposed which suggests a set of adaptation techniques with the problem space for improving the performance of dynamic optimization. In this approach, all the mentioned shortcomings of the previous one are resolved completely. To evaluate the proposed method like the previous one, MPB benchmark has been used which is the best-known benchmark for evaluating dynamic optimization approaches in literature. To assess the proposed method more accurately, different scenarios and configurations of this benchmark have been considered and the efficiency of the proposed method for optimization has been evaluated

and compared with state-of-the-art methods in this domain.

The rest of the paper is organized as follows: In section two, the background about dynamic optimization and firefly algorithm will be provided. Section three explains the proposed method for solving optimization problems in dynamic environments. Section four is dedicated to experimental setup which includes benchmark function, performance metric and parameters setting. In section five, the results and the comparison between the proposed approach and other approaches will be reviewed. The last section includes the conclusion of the paper and presents future works.

2. BACKGROUND

2.1. Dynamic optimization

Using meta-heuristic methods for dynamic optimization has some challenges, which do not exist in static environments. Among these challenges, two of the most important ones are outdated memory and diversity loss. The outdated memory happens when the environment changes and the fitness value of the obtained solutions will no longer correspond to the stored value in the memory. In fact, the fitness value stored for the obtained results will not be valid in such conditions.

Diversity loss occurs because of the intrinsic nature of meta-heuristic methods for convergence. Convergence to the goal decreases diversity and the algorithm loses its ability to converge to the optimum. In other words, the reason for this challenge is the inherent characteristic of these algorithms for converging to the previous optimum position and the exorbitant proximity of the solutions to each other. The easiest way to solve these two issues is re-initialization[2]. In re-initialization, we look at the changed environment as a new problem and restart the optimization method using changed environment. It is only recommended when the changes in the environment are major and numerous. Since the efficiency of the optimization process in the changed environment could be improved using the knowledge acquired from the previous environment, the re-initialization methods imply the loss of all the knowledge obtained so far from the problem space.

In the following, the presented solutions for resolving these two main challenges are studied in details; the challenges that arise as a result of using meta-heuristic methods for optimization in dynamic environments.

The first challenge, i.e. the outdated memory, is of less importance compared with the other one, and two solutions have been proposed for it, which are forgetting memory and re-evaluating memory [19]. These two solutions are also used for optimization methods which use the memory for storing information obtained from the problem space. In forgetting memory method, the position stored for each solution will be replaced by its position in the new environment. In re-evaluating memory method, the stored positions in the memory are re-evaluated.

The second challenge, i.e. diversity loss, has observed many solutions, which have been classified into three main categories.

2.1.1. Presenting Diversity Method: In this category, the algorithms allow the diversity loss to occur, and afterward they try to solve it. They are divided into two subcategories as well:

Mutation and Self-Adaptation: In this subcategory, the algorithms try to generate the lost diversity in the environment by performing mutation and self-adaptation [20-24].

Other Approaches: There are other methods proposed for creating diversity in the environment after discovering a change in the environment. In [25], a method called “RPSO” has been proposed for randomizing some or all of the solutions after detecting changes in the environment. In[26], another algorithm called “PBIL” (Population-Based Incremental Learning) is presented which uses a flexible probability vector for generating solutions. The vector in this algorithm is utilized for adjusting the learning rate after the change.

2.1.2. Diversity Maintenance Method: In this method, it is trying to keep diversity in the environment at all times (before and after the change). The proposed algorithms in this category are divided into three subcategories:

Dynamic Topology: Algorithms in this subcategory try to decrease the rate of algorithm convergence into a global optimum by limiting the existing communications among solutions, and also keep the

diversity in the environment. In[27], a method called "FGPSO" with a proximity structure similar to a grid is proposed for maintaining diversity, which illustrates the higher efficiency compared with RPSO [25] in dynamic environments with higher dimensions. In [28], HPSO has been proposed for preserving diversity, with its hierarchical and pseudo-tree structure. Another method which is called "CellularPSO", [29] has been suggested for dynamic optimization, and this method utilizes the local information exchange and cellular automata distribution features. In[30], the presented model in [29] has been improved by changing the roles of some particles to quantum particles just after the environment changes.

***Memory-Based methods*:** When the environment changes are periodical or recurrent, it will be very useful to store previous optimal solutions for using in the future. Memory-based methods try to store such information. These methods have been usually suggested for evolutionary methods such as GA and EDA which are genetic-based. The Memory-based methods can be divided into two main subclasses of implicit and explicit. In implicit methods, memory is integrated with meta-heuristic methods as a redundant representation [31, 32]. However, in explicit methods, memory is created explicitly [23, 33-37]

Other approaches: Other methods have also been presented for maintaining the diversity in the environment after detecting the changes in the environment. In [37], a Sentinel Placement method is used for diversity maintenance. In this method, a series of sentinels distributed in the search space are utilized in order to generate a new population. In [3], using Random Immigrant method has been proposed in every generation, in which a number of random solutions are added to the population to preserve diversity. In [38], a multi-objective method is proposed in which two goal functions are implemented. The first one is the main goal function, and the second one is a goal function for keeping diversity in the environment.

2.1.3. Hybrid Methods: This category is a hybrid of both presenting diversity and diversity maintenance methods. This means that diversity is maintained during execution, and also diversity creation is considered in the environment, in case of changes in the environment. Multi-population methods are the most used subcategory of hybrid methods. In this type of algorithms, a number of subpopulations are used for covering different regions of the search space. All subpopulations usually have the similar tasks, but they may also have different tasks. In designing algorithms based on multi-population, some points must be considered. The first point is the manner in which subpopulations are generated, and on the other hand the equality or inequality of the number of the existing solutions in each one. The second point is determining the task of each subpopulation in the problem space. Provided that subpopulations have different tasks, another important point will be the manner of task transition. The third point is to provide a policy in order to prevent two subpopulations from local searches in a common region of the solution space (on the same peak). The fourth point is to determine how diversity can be created after a change or during the execution time.

In [39], a method called "Shifting Balance Genetic Algorithm (SBGA) "has been proposed in which a number of small subpopulations are responsible for global search in the problem space, and a large subpopulation is responsible for tracking the changing peaks. Another approach is suggested in [34] and it is called "Self Organizing Scouts (SOS)", which contrarily uses a big subpopulation for global search and a number of small subpopulations for tracking changes. This strategy has also been proposed with other meta-heuristic methods such as Genetic Algorithm in [40] and Differential Evolution in [41]. In [42], a population is first used to global search, and when an optimum is discovered, the population is divided into two subpopulations, one responsible for tracking optimum changes and another responsible for global search. In [11, 14, 43] , speciation based on PSO approaches are proposed for optimization in dynamic environments. Also in [44], a regression-based approach named "RSPSO" is suggested for enhancing the convergence rate using speciation based methods. In this approach, every subpopulation is considered as a hypersphere and is created with a certain radius of the best solution. In [45], a technique named "SPSO" is proposed in which every cluster is divided into two. The first cluster is responsible for exploitation and the second one is in charge of exploration. In this article, Gaussian local search and differential mutation have been used in order to improve diversity in the environment. In [46], a method based on clustering is proposed for creating subpopulations, and in [47], this method has been improved and has been named "PSO-CP". It uses simplifications such as removing the learning procedure, and reducing the number of phases for

clustering from two phases to only one phase. In [13] , two multi-population methods, named " MQSO "and "MCPSO" , are proposed. In the former one, quantum particles and in the latter one, charged particles are used to generate diversity. The number of solutions in this technique is equal for every subpopulation, and the number of subpopulations is also constant from the beginning. In[48], an approach for enhancing this method is proposed by making the number of subpopulations adaptive, and it is named "AMQSO". It has significantly improved the performance of the algorithm. Finally, in [49] ,a method for dynamic optimization is proposed and it is based on composite particles. This method has an acceptable efficiency.

2.2. Firefly Algorithm

Firefly algorithm was introduced by Yang in 2008[50] and inspired from flashing behaviour of fireflies in nature. It uses three particular idealized rules based on some of the basic characteristics of real fireflies which are as follows:1) All fireflies are unisex and they move towards more attractive and brighter ones regardless of their sex.2) The degree of attractiveness of a firefly is proportional to its brightness. 3)The brightness or light intensity of a firefly is determined by the value of the objective function of a given problem.

In this algorithm, each firefly has a location $X=(x_1, x_2, x_3, \dots, x_d)^t$ in a D-dimensional space and a light intensity $I(x)$ or attractiveness $\beta(x)$ which are proportional to the objective function $f(x)$. Attractiveness $\beta(x)$ and light intensity $I(x)$ are relative and these should be judged by other fireflies. Thus, it will vary with the distance r_{ij} between firefly i and firefly j. So the attractiveness β of a firefly can be defined by Equation 1.

$$\beta = \beta_0 e^{-\gamma r^2} \quad (1)$$

where r, the Cartesian distance is the distance between any two fireflies i and j at x_i and x_j, respectively. β_0 is attractiveness at $r = 0$ and γ is the light absorption coefficient in the environment.

Algorithm 1: Firefly Algorithm

Algorithm 1: Firefly Algorithm

```
Begin
    Objective function f(x), x = (x1, ..., xd)^T
    Generate initial population of fireflies xi (i = 1, 2, ..., n)
    Light intensity Ii at xi is determined by f(xi)
    Define light absorption coefficient γ
    while (t <MaxGeneration)
        for i = 1 : n all n fireflies
            for j = 1 : i all n fireflies
                if (Ij > Ii)
                    Move firefly i towards j based Equation 2
                end if
                Evaluate new solutions and update light intensity
            end for j
            if firefly i does not move at all
                move firefly i randomly
            end if
        end for i
        Rank the fireflies and find the current best
    end while
    Post-process results and visualization
End
```

Also, each firefly i can move toward another more attractive (brighter) firefly j by Equation 2.

$$x_i^{(t+1)} = x_i^{(t)} + \beta_0 e^{-\gamma r_{i,j}^2}(x_j - x_i) + \alpha\varepsilon \quad (2)$$

where, α is a significance factor of randomization parameter, and ε is a random number obtained

from a uniform distribution.

Recently, some studies have revealed efficiency of this algorithm [51] by outperforming other meta-heuristic algorithms including some variation of particle swarm optimization [52] and also, in dealing with stochastic test functions and noisy environments [53]. A more comprehensive description of this algorithm is given in [50]. Also, a pseudo code of firefly algorithm is presented in Algorithm 1.

2.3. Speciation Based Firefly Algorithm

Firefly algorithm is a meta-heuristic algorithm with a great potential for discovering multiple optima simultaneously. The author in [18] has proposed an algorithm using the mentioned potential capability of firefly algorithm well in converging to multiple optima and creating dynamically the sub-swarm the same as speciation based approach.

In [18], some modifications are made on firefly algorithm as follows for better performance in dynamic environment. First, the best personal location of each firefly (*Pbest*) is added to the algorithm by inspiration from particle swarm optimization [54-56]. The best personal location of each firefly can only help algorithm to move toward the best position. By having *Pbest* for each firefly, after fitness evaluation, algorithm can move to a better position if the fitness value is better than the fitness value of the *Pbest*.

Second, randomization parameter of the firefly (α) is changed in an adaptive manner. The value of α increases if new location of firefly has better fitness value than previous fitness value of the Pbest of that firefly, otherwise it decreases.

This approach has some advantages and disadvantages. Among the advantages, we can mention the following points: No need to define a mechanism for dividing fireflies into sub-populations, and no need to define a mechanism to make sub-populations away from each other.

Also, we can mention the following disadvantages for this approach. First, some redundant fitness evaluations near the local optima which have not any influence in reduction of the overall error of the optimization. As the most performance metrics in dynamic environments calculate the error only based on the distance between the best solution founded by the algorithm and the global optima, then finding better solution near the local optima is wasting the number of fitness evaluations. Second, inability of the algorithm to cover a high number of local optima due to the fixed number of fireflies in each speciation and also the weakness of algorithm in global search. In dynamic environment, each local optima can be converted to a global optima by happening changes in environment. So, covering high number of local optima during the run can be a benefit for an algorithm in such environment. Because, the global optima can be found more rapidly by doing some local search around the best swarms.

In this paper, a new approach is suggested by the authors to improve the performance of their previous algorithm in dynamic environments by using some adaptation techniques. The details of them are given in the next section.

3. PROPOSED ALGORITHM

In this section, a novel algorithm based on the firefly algorithm with speciation is proposed for dynamic optimization. The proposed algorithm applies various mechanisms in order to handle the dynamic environment challenges, and to improve performance. In the following, the mechanisms used in the proposed algorithm will be discussed.

So far, the algorithms which have been designed for optimization in Unknown and Dynamic Environments have some of the properties listed below for better function in these environments.

a) Covering all local optima with minimum fitness evaluation.
b) Disabling and enabling individuals for improving performance of the algorithm.
c) Discovering or predicating Change in the environment.
d) Appropriate local search.
e) Appropriate global search.
f) Appropriate trade-off between local and global search.
g) Knowing the minimum information about the environment.

The last one (g) is a property which makes the algorithm more general than the other algorithms. In the real world, most environments are also dynamic and unknown. So, knowing minimum knowledge about the environment is a great feature for an algorithm.

Up to now, most algorithms designed for optimization in dynamic environments have some prior knowledge about the environment, such as the number of local optima, the time between two changes in the environment, the minimum and the maximum of severity, the minimum and the maximum of each designing variable and so on. In this paper, this approach tries to use minimum knowledge about the problem.

The proposed approach has two types of swarm for better covering the environment. The follower swarm is responsible for local search and the discoverer swarm is responsible for global search and finds a promising area in the environment. Also, this approach has a solution for each aforementioned property which is defined as below:

a) For covering all local optima with minimum fitness evaluation, this approach changes the way that the discoverer swarm does the discovery process in [18] and has provided a better global search. The Details for this will be provided later in this section.

b) For disabling and enabling individuals, this approach uses a mechanism for activating and inactivating fireflies in the follower swarm during the run.

c) For discovering or predicating the change in the environment, this approach uses a test point for identifying the change in the environment. The test point starts evaluating in the beginning of the algorithm. If the fitness value is different from the last evaluation, then the change happens to the environment and the algorithm performs the change process in the environment.

d) This approach uses a cloud around the global optimum and reduces the radius of the cloud adaptively for the follower swarm during the run.

e) For global search also this approach uses the discoverer swarm.

f) This approach uses a mechanism for trading off between local search and global search provided in detail in the following.

g) This approach does not use any knowledge about the number of local optima and also the time between two changes in the environment.

The pseudo code of the proposed algorithm is provided in Algorithm 2.

Algorithm 2: Proposed Algorithm (Adaptive-SFA)

Algorithm 2: Proposed Algorithm(Adaptive-SFA)

```
1  begin
2    Initialize Parameters.
3    Initialize fireflies in discoverer swarm and follower swarm Randomly.
4    Calculate Fitness of all fireflies and set Pbest and Gbest.
5    while stop criteria is not satisfied do
6      call TestForChange()
7      If fitness of Gbest_value > minimum_for_activating_discoverer then
8        Activate discoverer swarm
9      end if
10     if discoverer swarm is active
11       call StartDiscovey()
12       if distance between Gbest_finder and any fireflies is less than r_excl then
13         reinitialize the discoverer_swarm
14       end if
15       if discoverer_swarm is converged then
16         add k best fireflies to the follower swarm
17       end if
18     end if
19     Move Fireflies In Follower Swarm based Equation 2
20     Call GbestLocalSearch()
21   end
22 end
```

In the proposed algorithm, after initialization, fitness for all fireflies in follower and discoverer

swarm is calculated. Then, a test will be done to check whether a change has happened to the environment or not. The pseudo code of the test for the change routine is provided in Algorithm3. In the next step, the algorithm checks if discoverer swarm must be active or not. If the discoverer swarm is active then *start_discovery* routine will be recalled. The pseudo code of this routine is given in algorithm 4, too. In the next step, the algorithm checks whether there is any conflict between the discoverer swarm and follower swarm. If it finds some conflicts, the discoverer swarm will be reinitialized.

The r_{excl} parameter calculated by Equation 3.

$$r_{excl} = 0.5 * \left(\frac{peaks\_location\_range}{number\_of\_peaks^{\frac{1}{number\_of\_dimensions}}} \right) \quad (3)$$

Next, the algorithm checks the convergence of discoverer swarm to local optima. Also, the convergence condition in this approach is that the mean of *Gbest* of the discoverer swarm in three sequential iterations is less than *num_seq_iteration*. If the discoverer swarm converges, the best *k_selected* firefly of the discoverer swarm would be added to the follower swarm. After that, the follower swarm will move according to Equation 2 and after that Gbest Local Search routine will be recalled. The pseudo code of Gbest local search routine is given in Algorithm 5. The algorithm will do the mentioned step until it reaches to the maximum number of fitness evaluations in the problem.

Algorithm 3: TestForChange ()

Algorithm 3: TestForChange()

```
1  Begin
2      Evaluate Gbest as a test point.
3      if the Fitness(Gbest) is different from previous one then
4          Re-evaluate all the fireflies in both swarm
5          Reset Pbest and Gbest.
6          Re-initialize r_cloud and α.
7          if the fireflies number n is more than max_num_of_fireflies then
8              // Cluster follower swarm for finding the all sub-swarms
9              ClusterFirefly()
10             remove the worse sub-swarm
11         end
12     end
13 end
```

In Algorithm 3, the best firefly (*Gbest*) is considered as the test point and at the beginning of each iteration, it checks whether there are any differences between fitness value of Gbest firefly and its previous value. If there are any changes in the environment, some parameters will be adjusted for diversifying the environment. Also, if the number of fireflies in the swarm is more than *max_num_of_fireflies* parameter, the fireflies in the follower swarm will be clustered by the algorithm which is given in [57] for finding multiple species seeds and then by using these seeds, data clustering will be done. After finding the clusters, removing the worst swarm (cluster) from the follower swarm is very simple.

In algorithm4, the discovery routine occurs. Discovery routine uses firefly algorithm like follower routine. But the values of parameters for $\alpha_{discoverer}$ and $\gamma_{discoverer}$ are completely different from the follower swarm. The $\gamma_{discoverer}$ for the discovery routine is set to zero. Thus, the firefly algorithm can be executed similar to particle swarm optimization. Moreover, $\alpha_{discoverer}$ decreases during the run according to Equation 4.

$$\alpha_{discoverer} = \alpha_{discoverer} \cdot f_{dec} \quad (4)$$

where f_{dec} is a predefined factor for decreasing $\alpha_{discoverer}$.

Algorithm 4: StartDiscovey ()

```
Algorithm 4: StartDiscovey ()
1  begin
2    for each firefly i in discoverer swarm
3      for each firefly j in discoverer swarm
4        Move firefly i toward firefly j if the fitness (firefly_j) is better than
5        fitness(firefly_i) based Equation 2.
6      end for
7      if firefly i does not move at all
8        move firefly i randomly
9      end if
10   end for
11 end
```

If discoverer swarm converges, k best fireflies of the discoverer swarm would be added to follower swarm. In addition, Gbest local search is recalled for improving the results as much as possible based on Algorithm 5.

Algorithm 5: GbestLocalSearch ()

```
Algorithm 5: GbestLocalSearch()
1  begin
2    for counter =1 to try_number
3      Update Quantum_firefly based on Equation 5.
4      if fitness(Quantum_firefly) is better than fitness(Global best) then
5        Gbest = Quantum_firefly
6      end
7    end for
8    update r_cloud based on Equation 6.
9  End
```

In this algorithm, a quantum firefly is calculated with respect to Equation 5.

$$Quantum\_firefly = Gbest\_position + (r_{cloud} \times R_j) \quad (5)$$

where R_j is a random number with uniform distribution in *[-1,1]*. Hence, the *quantum_firefly* is located in the radius of r_{cloud} from Gbest. And r_{cloud} will decrease based on Equation 6.

$$r_{cloud} = r_{cloud} \times (w_{\min} + (rand \times (w_{\max} - w_{\min})) \quad (6)$$

where W_{min} , W_{max} are the lower and the upper limits for multiplying in r_{cloud}.

In the next section, the simulation setup and the result of applying the proposed algorithm on well-known benchmark will be evaluated.

4. EXPERIMENTAL SETUP

In this section, first, the benchmark function and the performance metric are described and then, the results of applying the proposed algorithm on benchmark function is compared with some other similar algorithms.

4.1. MPB Function

We used the Moving Peaks Benchmark (MPB) originally proposed by Brank [58] as a benchmark function in our simulations. It is widely used in the literature for evaluating the performance of dynamic optimization algorithms. In this benchmark, there are some peaks in a multi-dimensional space, where height, width and position of peaks alter when a change occurs in the environment. The parameter setting for MPB function in this paper can be found in Table 1.

Table 1: MPB Parameter Setting

PARAMETER	VALUE
Number of peaks, M	Variable between 1 to 200
Change frequency, CF	500,1000,2500,5000,10000
Height change, H	7.0
Width change, W	1.0
Peaks shape	Cone
Basic function	No
Shift length, s	1.0
Number of dimensions, D	5
Correlation Coefficient, λ	0
peaks location range	[0 – 100]
Peak height	[30.0 – 70.0]
Peak width	[1 – 12]
Initial value of peaks	50.0
Number of environmental change in each run, NCR	100

4.2. Performance Metric

The performance metric used in this paper is the offline error, which is the average of difference between the fitness of best solution found by the algorithm and the fitness of the global optima in all fitness evaluations [58, 59] .

$$offline \quad error = \frac{1}{FEs} \sum_{t=1}^{FEs} \left(fitness(gbest(t)) - fitness(globalOptima(t)) \right) \qquad (7)$$

Where FEs is the maximum fitness evaluation, and gbest(t) and globalOptima(t) are the best position found by the algorithm and the global optimum at the t^{th} fitness evaluation, respectively.

It is worth mentioning that in [17, 47], offline error is calculated using different approach which is the average of current errors exactly before each environmental change. Also in [60], the authors proposed new name for this type of offline error as best_before_change error. In this paper, we calculate offline error based on Equation 7 for comparing the algorithms' results.

4.3. Configurations of Adaptive-SFA

In Adaptive-SFA, the number of sub-swarm, number of fireflies in each sub-swarm and number of active fireflies are adaptive. However, in order to make the algorithm adaptive, some non-adaptive parameters are introduced, as well. Table 2, shows all the parameters' settings for Adaptive-SFA. Most of the values for non-adaptive parameters are selected by doing some sensitivity analysis on a simple MPB scenario (default parameter setting except CF=5000, M=10, NCR=5).

The zero is considered for $\gamma_{discoverer}$ parameter. It means that a flashing firefly in discoverer swarm can be seen anywhere in the fitness landscape. Also, as the follower swarm is responsible for local search in the fitness landscape, 0.01 is considered for α parameter.

Table 2: Adaptive-SFA parameters settings

PARAMETER	VALUE
α	0.01
γ	1
$\alpha_{discoverer}$	1
$\gamma_{discoverer}$	0
max_num_of_fireflies	100
num_seq_iteration	2
k_selected	5
f_{dec}	0.99
r_{cloud}	0.2 * severity
minimum_for_activating	2

_discoverer	
w_{min}	0.6
w_{max}	0.9

4.4. Comparison with Other state-of-the-art approaches

In this section, the proposed algorithm is tested on MPB with various configurations. The obtained results are the outcomes of 30 times execution of the proposed algorithm with different random seed. Also, for each environment, t-tests with a significance level of 0.05 have been applied and the result of the best performing algorithms is printed in bold. When the results for the best performing algorithms are not significantly different, all are printed in bold.

Table 3: Comparison of the offline error (standard error) of seven algorithms on MPB function with different number of peaks and change frequencies.

Algorithm	**C.F.**	**NUMBER OF PEAKS**							
		1	**5**	**10**	**20**	**30**	**50**	**100**	**200**
mQSO(5,5q)	500	33.67(3.42)	11.91(0.76)	9.62(0.34)	9.07(0.25)	8.80(0.21)	8.72(0.20)	8.54(0.16)	8.19(0.17)
AmQSO		**3.02(0.32)**	5.77(0.56)	5.37(0.42)	6.82(0.34)	7.10(0.39)	7.57(0.32)	7.34(0.31)	7.48(0.19)
mPSO		8.71(0.48)	6.69(0.26)	7.19(0.23)	8.01(0.19)	8.43(0.17)	8.76(0.18)	8.91(0.17)	8.88(0.14)
HmPSO		8.53(0.49)	7.40(0.31)	7.56(0.27)	7.81(0.20)	8.33(0.18)	8.83(0.17)	8.85(0.16)	8.85(0.16)
APSO		4.81(0.14)	**4.95(0.11)**	**5.16(0.11)**	**5.81(0.08)**	6.03(0.07)	**5.95(0.06)**	**6.08(0.06)**	**6.20(0.04)**
SFA		4.72(0.12)	**4.88(0.12)**	**5.11(0.14)**	**5.72(0.13)**	**5.97(0.12)**	**5.94(0.15)**	**6.15(0.08)**	**6.18(0.11)**
Adaptive-SFA		4.80(0.08)	**4.98(0.11)**	**5.09(0.10)**	**5.67(0.14)**	**5.84(0.11)**	**5.86(0.10)**	**6.04(0.11)**	**6.08(0.17)**
mQSO(5,5q)	1000	18.60(1.63)	6.56(0.38)	5.71(0.22)	5.85(0.15)	5.81(0.15)	5.87(0.13)	5.83(0.13)	5.54(0.11)
AmQSO		**2.33(0.31)**	2.90(0.32)	4.56(0.40)	5.36(0.47)	5.20(0.38)	6.06(0.14)	4.77(0.45)	5.75(0.26)
mPSO		4.44(0.24)	3.93(0.16)	4.57(0.18)	4.97(0.13)	5.15(0.12)	5.33(0.10)	5.60(0.09)	5.78(0.09)
HmPSO		4.46(0.26)	4.27(0.08)	4.61(0.07)	4.66(0.12)	4.83(0.09)	4.96(0.03)	5.14(0.08)	5.25(0.08)
APSO		2.72(0.04)	2.99(0.09)	3.87(0.08)	4.13(0.06)	4.12(0.04)	**4.11(0.03)**	**4.26(0.04)**	**4.21(0.02)**
SFA		**2.45(0.12)**	**2.71(0.06)**	3.64(0.04)	**4.01(0.07)**	**4.02(0.08)**	4.12(0.07)	4.40(0.07)	4.43(0.07)
Adaptive-SFA		2.74(0.14)	2.89(0.09)	**3.51(0.08)**	**3.92(0.12)**	**3.97(0.07)**	**4.02(0.09)**	4.34(0.09)	4.38(0.08)
mQSO(5,5q)	2500	7.64(0.64)	3.26(0.21)	3.12(0.14)	3.58(0.13)	3.63(0.10)	3.63(0.10)	3.58(0.08)	3.30(0.06)
AmQSO		**0.87(0.11)**	2.16(0.19)	2.49(0.10)	2.73(0.11)	3.24(0.18)	3.68(0.15)	3.53(0.14)	3.07(0.12)
mPSO		1.79(0.10)	2.04(0.12)	2.66(0.16)	3.07(0.11)	3.15(0.08)	3.26(0.07)	3.31(0.05)	3.36(0.05)
HmPSO		1.75(0.10)	1.92(0.11)	2.39(0.16)	2.46(0.09)	2.57(0.05)	2.65(0.05)	2.72(0.04)	2.81(0.04)
APSO		1.06(0.03)	1.55(0.05)	2.17(0.07)	2.51(0.05)	2.61(0.02)	2.66(0.02)	**2.62(0.02)**	**2.64(0.01)**
SFA		**0.85(0.06)**	1.49(0.07)	1.95(0.04)	2.28(0.05)	2.46(0.05)	2.57(0.05)	2.74(0.04)	2.76(0.04)
Adaptive-SFA		1.12(0.08)	**1.32(0.08)**	**1.77(0.07)**	**2.09(0.06)**	**2.28(0.07)**	**2.42(0.06)**	**2.64(0.05)**	**2.65(0.06)**
mQSO(5,5q)	5000	4.92(0.21)	1.82(0.08)	1.85(0.08)	2.48(0.09)	2.51(0.10)	2.53(0.08)	2.35(0.06)	2.24(0.05)
AmQSO		0.51 (0.04)	**1.01(0.09)**	1.51(0.10)	2.00(0.15)	2.19(0.17)	2.43(0.13)	2.68(0.12)	2.62(0.10)
mPSO		0.90(0.05)	1.21(0.12)	1.61(0.12)	2.05(0.08)	2.18(0.06)	2.34(0.06)	2.32(0.04)	2.34(0.03)
HmPSO		0.87(0.05)	**1.18 (0.04)**	1.42(0.04)	1.50(0.06)	1.65(0.04)	1.66(0.02)	1.68(0.03)	1.71(0.02)
APSO		0.53(0.01)	1.05(0.06)	1.31(0.03)	1.69(0.05)	1.78(0.02)	1.95(0.02)	1.95(0.01)	**1.90(0.01)**
CPSO		7.80(1.00)	4.20(0.32)	4.50(0.26)	4.00(0.16)	3.50(0.16)	3.50(0.13)	3.20(0.13)	2.50(0.091)
CPSOR		6.10(0.84)	2.90(0.34)	2.60(0.20)	2.60(0.30)	2.00(0.14)	2.40(0.12)	2.50(0.10)	2.30(0.097)
SFA		**0.42(0.03)**	**0.89(0.07)**	1.05(0.04)	1.48(0.05)	1.56(0.06)	1.87(0.05)	2.01(0.04)	1.99(0.06)
Adaptive-SFA		0.66(0.05)	**0.79(0.06)**	**0.90(0.05)**	**1.24(0.08)**	**1.41(0.06)**	**1.71(0.04)**	**1.84(0.04)**	**1.90(0.05)**
mQSO(5,5q)	10000	1.90(0.18)	1.03(0.06)	1.10(0.07)	1.84(0.08)	2.00(0.09)	1.99(0.07)	1.85(0.05)	1.71(0.04)
AmQSO		**0.19(0.02)**	**0.45(0.04)**	0.76(0.06)	1.28(0.12)	1.78(0.09)	1.55(0.08)	1.89(0.14)	2.52(0.10)
mPSO		0.27(0.02)	0.70(0.10)	0.97(0.04)	1.34(0.08)	1.43(0.05)	1.47(0.04)	1.50(0.03)	1.48(0.02)
HmPSO		-	-	-	-	-	-	-	-
APSO		0.25(0.01)	0.57(0.03)	0.82(0.02)	1.23(0.02)	1.39(0.02)	1.46(0.01)	1.38(0.01)	1.36(0.01)
SFA		0.26(0.03)	0.53(0.04)	0.72(0.02)	0.91(0.03)	0.99(0.04)	1.19(0.04)	1.44(0.04)	1.52(0.03)
Adaptive-SFA		0.35(0.03)	**0.49(0.03)**	**0.63(0.03)**	**0.77(0.03)**	**0.84(0.03)**	**0.97 (0.03)**	**1.27(0.05)**	**1.31(0.03)**

4.4.1. Effect of varying the Number of peaks and Change frequencies

Table 3 presents the comparison of proposed algorithm and all other peer algorithms on MPB function with different number of peaks and change frequencies.

The results of mQSO[13], Amqso [48] and SFA[18] algorithms are obtained from their execution which implemented by us, and the results of other algorithms are obtained from their respective papers. For CPSO[47] and CPSOR[17] the results are obtained from [60] which are provided based on defined offline error in this paper instead of best_before_change error in their relevant papers. From the results, Adaptive-SFA achieves the best offline error in most cases. The rank of the algorithms on MPB with different number of peaks and change frequencies are presented in Tables 4, 5. As we can see in Table 5, AmQSO is the best algorithm when the peak number is equal to 1. Also, Adaptive-SFA and SFA are in the second and third place. This superiority is due to use of only one swarm during the run in AmQSO and also using at most two swarms in Adaptive-SFA and SFA. In this way, the number of wasted fitness evaluation decrease very much.

From Table 3, it is clear that the algorithms' performances declined by increasing the number of peaks. However, in some algorithms the results are improved when the number of peaks is remarkable because the difference between local and global fitness values decreases. Furthermore, increasing the frequency change leads to more chance for the algorithms to find better values and discover the peaks which results in their improved efficiency. In effect, when the change frequency is low, it will be possible to lose the optimums or never to reach them because of fast environment changes, which reduces the algorithms efficiency. The results in Table 3 show that Adaptive-SFA is of a very good efficiency in different environments using different peaks and different high and low frequency changes.

Table 4: Rank of nine algorithms on MPB function with different change frequency.

Algorithm	**C.F.**					
	500	**1000**	**2500**	**5000**	**10000**	**All**
mQSO(5,5q)	7	7	7	7	6	7
AmQSO	4	4	5	5	4	5
mPSO	6	6	6	6	5	6
HmPSO	5	5	4	3	3	4
APSO	3	3	3	4	-	3
CPSO	-	-	-	9	-	9
CPSOR	-	-	-	8	-	8
SFA	2	2	2	2	2	2
Adaptive-SFA	1	1	1	1	1	1

Table 5: Rank of nine algorithms on MPB function with different number of peaks.

Algorithm	**Peak Numbers**								
	1	**5**	**10**	**20**	**30**	**50**	**100**	**200**	**All**
mQSO(5,5q)	6	6	6	6	6	6	6	4	6
AmQSO	1	4	4	4	4	5	4	5	4
mPSO	5	5	5	5	5	4	5	6	5
HmPSO	7	7	7	7	7	7	7	7	7
APSO	3	3	3	3	3	3	2	1	3
CPSO	9	9	9	9	9	9	9	9	9
CPSOR	8	8	8	8	8	8	8	8	8
SFA	2	2	2	2	2	2	3	3	2
Adaptive-SFA	4	1	1	1	1	1	1	2	1

From Tables 3-5, it is clear that the proposed algorithm is significantly better than the other peer algorithms in most cases. In second and third position, SFA which proposed by the authors and APSO is located.

4.4.2. Effect of varying shift severity

Table 6 presents the comparison of proposed algorithm and all other peer algorithms on MPB function with different shift severity, change frequency = 5000 and number of peaks = 10.

Table 6, Comparison of offline error (standard error) of seven algorithms on MPB function with CF =5000, M=10 and different shift severity.

Algorithm	Shift length			
	1	**2**	**3**	**5**
mQSO(5,5q)	1.85(0.08)	2.40(0.06)	3.00(0.06)	4.24(0.10)
AmQSO	1.51(0.10)	2.09(0.08)	2.72(0.09)	3.71(0.11)
CPSO	4.50(0.26)	5.40(0.24)	6.10(0.19)	7.20(0.15)
CPSOR	2.60(0.20)	3.7(0.19)	4.50(0.24)	6.10(0.21)
SFA	1.05(0.04)	1.44(0.06)	2.06(0.07)	2.89(0.13)
Adaptive-SFA	**0.90(0.05)**	**1.31(0.07)**	**1.76(0.09)**	**2.64 (0.08)**

In Table 6, the efficiency of the proposed algorithm on MPB with different shift severities has been illustrated and has been compared with six other algorithms. MPB configuration has been done according to Table 1, and just the value of shift length parameter was changed in it. By increasing shift length, it becomes more difficult for algorithms to follow the peaks because after each environment changes, the peaks move into further distances. As the results in Table 6 represented, the efficiency of all the algorithms dramatically decreases by increasing shift length, but the efficiency of the proposed Adaptive-SFA shows less degradation and more robustness in these conditions in comparison to other algorithms.

As the results of the experiments show, the efficiency of the proposed method is very high because of utilizing different mechanisms for improving its performance and eliminating the challenges in dynamic environments. The peaks are rapidly covered by the discoverer swarms in the proposed algorithm, and therefore, the algorithm can quickly discover the position of the global optimum after each environment change.

5. CONCLUSION

In this paper, a novel speciation based [18] firefly algorithm was proposed for optimization in Unknown and Dynamic Environments, in which several mechanisms were used to conquer challenges of dynamic environments. In the proposed algorithm, two types of swarm were used, which were follower and discoverer. The discoverer swarm tries to help the main swarm for global search in the environment. On the other hand, the follower swarm is used for appropriately performing local optimization around the covered peaks and following them after an environment change. The performance of the proposed method was evaluated utilizing different configurations of Moving Peak Benchmark problem which is the most well-known benchmark in the literature and results were compared with those of the state-of-the art methods. The results showed the superiority and performance of the proposed method in terms of efficiency and accuracy.

The proposed approach already used some knowledge about the problem such as severity of change in the environment and range of designing variable for better acting in a dynamic environment. Hence, for applying the proposed algorithm for a real dynamic application, it needs to remove its sensitivity to severity and other parameters. Also, proposing a new approach with minimum number of parameters is always needed for any problems.

REFRENCES

[1] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1-24, 2012.

[2] K. Krishnakumar, "Micro genetic algorithms for stationary and nonstationary function optimization," *Proceedings of SPIE International Conference Adaptative Systems*, pp. 289-296, 1989.

[3] J. Grefenstette, "Genetic algorithms for changing environments," *Parallel Problem Solving from Nature*, vol. 2, pp. 137-144, 1992.
[4] N. Mori and H. Kita, "Genetic algorithms for adaptation to dynamic environments - a survey," in *Proceedings of the 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation*, Nagoya, Japan, 2000, pp. 2947-2952.
[5] L. Lili, W. Dingwei, and W. Hongfeng, "A new dual scheme for genetic algorithm in dynamic environments," in *Proceeding of 2008 Chinese Control and Decision Conference*, 2008, pp. 135-138.
[6] V. Ramos, C. Fernandes, A. C. Rosa, and A. Abraham, "Computational chemotaxis in ants and bacteria over dynamic environments," in *IEEE Congress on Evolutionary Computation, CEC*, 2007, pp. 1109-1117.
[7] X. Wang, X. Z. Gao, and S. J. Ovaska, "An immune-based ant colony algorithm for static and dynamic optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, 2007, pp. 1249-1255.
[8] L. Xiaoxiong, W. Yan, and Y. Jimin, "An Improved Estimation of Distribution Algorithm in Dynamic Environments," in *Proceedings of IEEE Fourth International Conference on Natural Computation, ICNC*, 2008, pp. 269-272.
[9] J. Branke, *Evolutionary Optimization in Dynamic Environments*: Kluwer Academic Publishers, 2001.
[10] L. Xiaodong and D. Khanh Hoa, "Comparing particle swarms for tracking extrema in dynamic environments," in *IEEE Congress on Evolutionary Computation, CEC*, 2003, pp. 1772-1779.
[11] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *IEEE Congress on Evolutionary Computation, CEC*, 2004, pp. 98-103.
[12] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," *Applications of Evolutionary Computing*, pp. 489-500, 2004.
[13] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 459-472, 2006.
[14] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 51-58.
[15] M. Kamosi, A. Hashemi, and M. Meybodi, "A New Particle Swarm Optimization Algorithm for Dynamic Environments," *Swarm, Evolutionary, and Memetic Computing*, pp. 129-138, 2010.
[16] M. Kamosi, Hashemi, A. B. and Meybodi, M. R., "A Hibernating Multi-Swarm Optimization Algorithm for Dynamic Environments," in *Proceedings of World Congress on Nature and Biologically Inspired Computing, NaBIC*, Kitakyushu, Japan, 2010, pp. 370-376.
[17] C. Li and S. Yang, "A general framework of multi-population methods with clustering in undetectable dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 16, pp. 556 - 577, 2011.
[18] B. Nasiri and M. Meybodi, "Speciation based firefly algorithm for optimization in dynamic environments," *International Journal of Artificial Intelligence*, vol. 8, pp. 118-132, 2012.
[19] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," presented at the International Conference on Artificial Intelligence, 2000.
[20] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," *NRL Memorandum Report*, vol. 6760, pp. 523-529, 1990.
[21] T. Nanayakkara, K. Watanabe, and K. Izumi, "Evolving in dynamic environments through adaptive chaotic mutation," in *Proceedings of the Fourth International Symposium on Artificial Life and Robotics*, Oita, Japan, 1999, pp. 520-523.
[22] F. Vavak, K. Jukes, and T. Fogarty, "Performance of a genetic algorithm with variable local search range relative to frequency of the environmental changes," *Proceedings of the Third Annual Conference on Genetic Programming*, pp. 22-25, 1998.
[23] E. L. Yu and P. N. Suganthan, "Evolutionary programming with ensemble of explicit memories for dynamic optimization," in *IEEE Congress on Evolutionary Computation, CEC*, 2009, pp. 431-438.

[24] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 500-513, 2009.
[25] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *IEEE Congress on Evolutionary Computation,CEC*, 2002, pp. 1666-1670.
[26] S. Yang and X. Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 9, pp. 815-834, 2005.
[27] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *IEEE Congress on Evolutionary Computation, CEC*, 2002, pp. 1671-1676.
[28] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," *Applications of evolutionary computing*, pp. 513-524, 2004.
[29] A. Hashemi and M. Meybodi, "Cellular Pso: A Pso for Dynamic Environments," *Advances in Computation and Intelligence*, pp. 422-433, 2009.
[30] A. B. Hashemi and M. R. Meybodi, "A multi-role cellular PSO for dynamic environments," in *14th International CSI Computer Conference, CSICC*, 2009, pp. 412-417.
[31] Y. Shengxiang, "On the Design of Diploid Genetic Algorithms for Problem Optimization in Dynamic Environments," in *IEEE Congress on Evolutionary Computation, CEC*, 2006, pp. 1362-1369.
[32] A. Uyar and A. E. Harmanci, "A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 9, pp. 803-814, 2005.
[33] S. Yang, "Memory-based immigrants for genetic algorithms in dynamic environments," in *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, 2005, pp. 1115-1122.
[34] J. Branke, T. Kaussler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," *Adaptive computing in design and manufacturing*, vol. 6, 2000.
[35] H. Richter and S. Yang, "Memory based on abstraction for dynamic fitness functions," in *Applications of Evolutionary Computing*, ed: Springer-Verlag, 2008, pp. 596-605.
[36] S. Yang and X. Yao, "Population-Based Incremental Learning With Associative Memory for Dynamic Environments," *IEEE Transactions on Evolutionary Computation,* vol. 12, pp. 542-561, 2008.
[37] R. W. Morrison, *Designing evolutionary algorithms for dynamic environments*: Springer-Verlag, 2004.
[38] L. T. Bui, H. A. Abbass, and J. Branke, "Multiobjective optimization for dynamic environments," in *IEEE Congress on Evolutionary Computation, CEC*, 2005, pp. 2349-2356 Vol. 3.
[39] F. Oppacher and M. Wineberg, "The shifting balance genetic algorithm: Improving the GA in a dynamic environment," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO*, 1999, pp. 504–510.
[40] H. Cheng and S. Yang, "Multi-population Genetic Algorithms with Immigrants Scheme for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks," *Applications of Evolutionary Computation*, pp. 562-571, 2010.
[41] R. I. Lung and D. Dumitrescu, "A new collaborative evolutionary-swarm optimization technique," in *Proceedings of the 9th annual conference companion on Genetic and Evolutionary Computation*, 2007, pp. 2817-2820.
[42] R. K. Ursem, "Multinational GAs: Multimodal optimization techniques in dynamic environments," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2000, pp. 19-26.
[43] D. Parrott and L. Xiaodong, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 440-458, 2006.
[44] S. Bird and X. Li, "Using regression to improve local convergence," in *IEEE Congress on Evolutionary Computation, CEC*, 2008, pp. 592-599.
[45] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information Sciences*, vol. 178, pp. 3096-3109, 2008.

[46] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *IEEE Congress on Evolutionary Computation, CEC*, 2009, pp. 439-446.
[47] S. Yang and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 959-974, 2010.
[48] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," *Swarm Intelligence*, pp. 193-217, 2008.
[49] L. Liu, S. Yang, and D. Wang, "Particle swarm optimization with composite particles in dynamic environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, pp. 1634-1648, 2010.
[50] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*: Luniver Press, 2008.
[51] X. S. Yang, "Firefly algorithms for multimodal optimization," *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, pp. 169-178, 2009.
[52] S. Lukasik and S. Zak, "Firefly algorithm for continuous constrained optimization tasks," in *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, ed, 2009, pp. 97-100.
[53] P. Aungkulanon, N. Chai-ead, and P. Luangpaiboon, "Simulated manufacturing process improvement via particle swarm optimisation and firefly algorithms," in *Proceedings of the International MultiConference of Engineerings and Computer Scientists*, 2011.
[54] M. Hasanzadeh, M. Meybodi, and M. Ebadzadeh, "Adaptive cooperative particle swarm optimizer," *Applied Intelligence*, pp. 1-24, 2013.
[55] H. Masoud, S. Jalili, and S. Hasheminejad, "Dynamic clustering using combinatorial particle swarm optimization," *Applied Intelligence*, vol. 38, pp. 289-314, 2013.
[56] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
[57] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *IEEE Congress on Evolutionary Computation, CEC*, 1996, pp. 798-803.
[58] J. Branke. (1999). *The Moving Peaks Benchmark*. Available: http://web.archive.org/web/20130906140931/http:/people.aifb.kit.edu/jbr/MovPeaks/
[59] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *IEEE Congress on Evolutionary Computation, CEC*, 1999.
[60] C. Li, S. Yang, and M. Yang, "An Adaptive Multi-Swarm Optimizer for Dynamic Optimization Problems," *Evolutionary Computation*, vol. 22, pp. 559-594, 2014.

Babak Nasiri received the B.S. and M.S. degrees in Computer science in Iran, in 2002 and 2004, respectively. He has been a Ph.D. candidate in computer science from Qazvin Islamic Azad University, Qazvin, Iran from 2010. Prior that, he was the faculty member of computer engineering and IT department at Qazvin Islamic Azad University from 2008. His research areas include soft computing, machine learning, nature-inspired algorithms, data mining and web mining.

MohammadReza Meybodi received the B.S. and M.S. degrees in Economics from National University in Iran, in 1974 and 1977, respectively. He also received the M.S. and Ph.D. degree from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include soft computing, learning system, wireless networks, parallel algorithms, sensor networks, grid computing and software development.