Hindawi
Discrete Dynamics in Nature and Society
Volume 2017, Article ID 5678393, 27 pages
https://doi.org/10.1155/2017/5678393

Research Article

The Patch-Levy-Based Bees Algorithm Applied to Dynamic Optimization Problems

Wasim A. Hussein,[1,2] Siti Norul Huda Sheikh Abdullah,[3] and Shahnorbanun Sahran[1]

[1] *Pattern Recognition Research Group, Center of Artificial Intelligence Technology, Faculty of Information Systems and Technology, Universiti Kebangsaan Malaysia, 43650 Bandar Baru Bangi, Malaysia*
[2] *Department of Computer Science, College of Science and Arts, University of Bisha, Balqarn 61985, Saudi Arabia*
[3] *Digital Forensic Lab, Cyber Security Center, Faculty of Information Systems and Technology, Universiti Kebangsaan Malaysia, 43650 Bandar Baru Bangi, Malaysia*

Correspondence should be addressed to Wasim A. Hussein; wassimahmed@yahoo.com

Received 19 December 2016; Revised 26 February 2017; Accepted 22 March 2017; Published 10 May 2017

Academic Editor: Seenith Sivasundaram

Many real-world optimization problems are actually of dynamic nature. These problems change over time in terms of the objective function, decision variables, constraints, and so forth. Therefore, it is very important to study the performance of a metaheuristic algorithm in dynamic environments to assess the robustness of the algorithm to deal with real-word problems. In addition, it is important to adapt the existing metaheuristic algorithms to perform well in dynamic environments. This paper investigates a recently proposed version of Bees Algorithm, which is called Patch-Levy-based Bees Algorithm (PLBA), on solving dynamic problems, and adapts it to deal with such problems. The performance of the PLBA is compared with other BA versions and other state-of-the-art algorithms on a set of dynamic multimodal benchmark problems of different degrees of difficulties. The results of the experiments show that PLBA achieves better results than the other BA variants. The obtained results also indicate that PLBA significantly outperforms some of the other state-of-the-art algorithms and is competitive with others.

1. Introduction

Many population-based metaheuristic algorithms have been used to solve stationary optimization problems where the fitness landscape is fixed during the course of optimization. However, most of the real-world optimization problems may face some uncertainties that may come from sources such as dynamic change of the attributes or the goal of the optimization problem [1]. For instance, in the Travelling Salesman Problem (TSP), if traffic jams are faced on some roads, the time between cities associated with those roads is no longer fixed and is increased [2]. The job scheduling problem can also face some uncertainties such as changes of due dates, orders, arrival of new jobs, and faults in the work of some machines [3, 4]. Thus, proposing metaheuristic algorithms that can deal with such problems is very important. Additionally, the existing metaheuristic algorithms should be adapted to deal with such dynamic optimization problems (DOPs).

Therefore, in recent years, there has been growing concern from researchers in the optimization community in the dynamic optimization domain. As a result, in the literature, many metaheuristic algorithms have been applied or modified to handle the DOPs. One of the first metaheuristic algorithms used to explore the DOPs was the genetic algorithm (GA) [4]. Ursem [5] proposed a multinational GA to deal with the DOPs by evolving the algorithm parameters during the search process. Grefenstette [6] adopted a self-adaptive GA for dynamic environments. The proposed algorithm can select different mutation or crossover operators based on the agent idea to control the selection process. Yang [7] proposed a memory-based GA to solve DOPs. Simões and Costa [8] investigated a GA based on an immune system for DOPs. In addition, the differential evolution (DE) has been applied to solve the DOPs as can be found in Mendes and Mohais [9], Brest et al. [10], and Mukherjee et al. [11]. Mendes and Mohais [9] introduced a multipopulation DE for the DOPs.

Brest et al. [10] proposed a self-adaptive multipopulation DE integrated with aging mechanism for DOPs. Mukherjee et al. [11] presented a variant of DE that modified the genetic operators of the DE to suit the dynamic optimization. The proposed algorithm utilized locality-induced mutation and crossover operators that acted as a retention strategy by employing the previously stored information to adaptively deal with the DOPs.

Swam intelligence-based metaheuristics have also been investigated in the dynamic environments. Eberhart and Shi [12] investigated particle swarm optimization (PSO) in tracking a single spatially changing peak. Hu and Eberhart [13] presented an adaptive PSO to automatically track a wide variety of changes in a dynamic system. Parrott and Li [14] investigated PSO with multiple parallel subpopulations to track multiple peaks simultaneously. Blackwell and Branke [15, 16] proposed some improvements on PSO to work well on DOPs by building interacting multiswarms. Wang et al. [17] introduced a memory scheme to PSO that is triggered whenever the exploration of the population results in a peak to deal with dynamic environments. Recently, Kordestani et al. [18] presented an oscillating triangular inertia weight in PSO, which is a time-varying inertia weight parameter, and investigated its performance on tracking optima in dynamic environments. The Ant Colony Optimization (ACO) algorithm has also been studied and adapted to work in dynamic environments. Eyckelhof and Snoek [2] presented a new ant system technique to a dynamic TSP problem. Tfaili et al. [19] used a multiagent ant colony algorithm, called Dynamic Hybrid Continuous Interacting Ant Colony (DHCIAC track), which hybridized between ant colony and dynamic simplex methods to optimize a set of dynamic test functions.

The Artificial Bee Colony (ABC) has also been applied to solve the DOPs. Raziuddin et al. [20] proposed a variant of ABC for DOPs based on a differential update strategy and an external archive or memory. In this variant, the good solutions through the generations are retained in a memory and a number of these solutions are randomly selected for the differential update strategy. In this update strategy, the weighted difference between the bee and its neighbor is added to the elite bee. Jiang [21] proposed a modified version of ABC for solving DOPs. The proposed ABC divides the population into two types: the sensitive bees and the optimizing bees. The sensitive bees work as monitors to detect the environmental changes and the optimizing bees act as respondents to changes by searching the changing optimal solution. Kojima et al. [22] proposed an improved version over Basic ABC for solving DOPs with some modifications to the procedures of the Basic ABC. Nakano et al. [23] have further modified the improved ABC in [22] for DOPs by incorporating a detection scheme and a memory scheme of the best solutions. Nseef et al. [24] presented an adaptive multipopulation ABC for dynamic optimization. The number of subpopulations in the proposed ABC changes over time based on the environmental changes strength to adapt to these changes.

In addition, an immune-based algorithm that is called Artificial Immune Network for Optimization (opt-aiNet) [25] was proposed for static optimization and then extended to deal with DOPs [1, 26]. This extended dynamic version is called Artificial Immune Network for Dynamic Optimization (dopt-aiNet). Recently, Turky and Abdullah [27] proposed a multipopulation harmony search with an external archive for dynamic optimization. Other swarm intelligence-based algorithms have been also adapted to solve DOPs such as the Cuckoo Search (CS) [28] and the artificial fish swarm algorithm (AFSA) [29]. A comprehensive survey on swarm intelligence-based algorithms for dynamic optimization can be found in [30].

Over the years, various dynamic optimization problems have been proposed to investigate the population-based metaheuristic approaches in the dynamic environments. Among these problems is the Moving peaks benchmark by Branke [31] that was employed in many works such as those studies done by Blackwell and Branke [15, 16], Turky and Abdullah [27], and Kordestani et al. [18] and the DF1 generator developed by Morrison and De Jong [32] that was used for testing metaheuristics in dynamic environments in works like that performed by Tfaili et al. [19]. However, there was no unified method for constructing dynamic optimization problems across the real space, combinatorial space, and binary space. Thus, recently, a Generalized Dynamic Benchmark Generator (GDBG) has been proposed for all the three spaces [33]. Using this generator, six dynamic benchmark problems were generated in the real solution space [34, 35]. These benchmarks can be included under two benchmark instances: Dynamic Composition Benchmark Generator (DCBG) and Dynamic Rotation Peak Benchmark Generator (DRPBG) [34, 35].

Then, many researchers were motivated to employ these new benchmark problems to study the performance of various metaheuristic algorithms in the dynamic environments. De França and Von Zuben [1] used these challenging benchmarks to test the performance of an adapted variant of the dopt-aiNet algorithm [26] in changing environments. Yu and Suganthan [36] adopted an evolutionary programming (EP) version based on a set of explicit memories to deal with these challenging dynamic benchmark problems. Korošec and Šilc [37] applied the differential ant-stigmergy algorithm (DASA) on the newly proposed dynamic benchmark problems. Li and Yang [38] proposed a variant of PSO (CPSO) that employed a hierarchical clustering method and a fast local search to address the dynamic environments constructed by the benchmarks. Mukherjee et al. [11] presented a modified variant of DE (MDE-LiGO) that utilized locality-induced genetic operators and tested it on the same benchmarks. Good surveys regarding the problems and metaheuristic algorithms in the dynamic environments were conducted by Mori and Kita [4], Blackwell et al. [3], Branke [39], Cruz et al. [40], Nguyen et al. [41], and Mavrovouniotis et al. [30].

A recently developed swarm intelligence algorithm is the Bees Algorithm (BA). The Bees Algorithm (BA) is a bee swarm-based algorithm proposed by Pham et al. [42] and inspired by the foraging behavior of swarm of honeybees. A few works have been done to investigate the performance of BA on DOPs [43, 44]. These problems were optimization benchmarks in chemical engineering. Recently, a modified version of Bees Algorithm, which is called Patch-Levy-based

Bees Algorithm (PLBA), has been proposed by Hussein et al. [45, 46]. The PLBA has been adopted to solve challenging static real-parameter optimization problems [45]. The experimental results showed that the PLBA outperformed other BA variants and some of the other state-of the-art algorithms on a set of challenging static real-parameter optimization problems. Additionally, PLBA was competitive with other state-of-the-art algorithms. The PLBA has also been proposed for multilevel image thresholding [46]. The experiments indicated that the PLBA significantly outperformed Basic BA and other sate-of-the-art algorithms. Encouraged by these promising results of the PLBA in solving these types of static problems, we validate the performance of PLBA and other variants of BA on the set of recently proposed dynamic multimodal benchmarks [34, 35] mentioned above.

The dynamic version of an optimization algorithm should be able to dynamically adapt to the environmental changes [24]. A dynamic optimization problem requires an optimization algorithm that is able not only to find the global optimum but also to detect the environmental change and track the changing global optimal solution [36, 38]. Some of the following features should characterize an optimization algorithm to deal with dynamic environments [3, 30, 39]: diversity creation, diversity maintenance, memory of old solutions, and multipopulation feature. The memory of old solutions can be advantageous to act as the initial population for a new change in the environment, where a new change is considered as the arrival of a new problem [36]. The memory can be beneficial especially in the case that the new optimal solution is not far from its previous positions [36]. The multipopulation strategy has been applied to track many peaks in the search space [24, 36, 38]. The memory and multipopulation are widely proposed and integrated with the optimization algorithms to enhance the diversity of the populations. Thus, it can be stated that the most important task in dealing with dynamic optimization problems is maintaining the diversity of the solutions [36, 38] to ensure that the population are not stuck into a single optimum where it cannot make further progress [36].

Several strategies have been adopted to maintain diversity in dynamic optimization problems such as hypermutation in GAs [11], the random immigrant scheme [11], prediction scheme and the memory-based methods, which are considered a special case of it [24], self-adaptive strategy [24], and multipopulation strategy [11, 24]. The hypermutation strategy maintains the diversity by increasing the mutation rate for some generations after the dynamic environmental change [11]. The random immigrant strategy maintains the diversity by replacing a part of the population with randomly generated solutions in each generation [11]. The hypermutation and random immigrant have been used for GA and evolutionary algorithms. Predictions schemes make the algorithm able to learn patterns from the previous search history and predict the upcoming changes. Examples of prediction-based techniques can be found in [47–49]. Many memory-based methods can be found in the literature such as Branke [31], Eggermont and Lenaerts [50], Branke [51], Yu and Suganthan [36], Yang [52], Daneshyari and Yen [53], and Wang et al. [17]. Self-adaptive schemes maintain the diversity by adaptively improving the search behavior of the algorithm, thus reducing the need for the manual tuning of the algorithm parameters [24]. Examples of methods based on this scheme are available in [5, 6, 18]. The multipopulation strategies have been widely applied to maintain the diversity of the population in the dynamic optimization problems. The methods based on the multipopulation strategy divide the population into subpopulations and distribute them over the search space to track multiple peaks in the search space [11, 24]. Many multipopulation methods can be found in the literature such as Branke et al. [54], Blackwell and Branke [15], Li and Yang [38], Li et al. [55], Turky and Abdullah [27], and Nseef et al. [24]. Li et al. [56] conducted a comprehensive experimental analysis on the performance of multipopulation methods and investigated the difficulties related to the multipopulation strategy. One of the challenging issues to apply the multipopulation strategy to DOPs is the identification of the suitable number of subpopulations [30]. Thus, researchers have been motivated to propose adaptive multipopulation algorithms to deal with DOPs.

Blackwell [57] proposed a self-adapting multiswarm optimizer based on a simple rule for generating and removing subswarms, which helps in the dynamic change of the number of subswarms. This optimizer was one of the first adaptive methods concerning the number of populations [30]. Li et al. [55] proposed an adaptive multiswarm optimizer (AMSO), which employs a single-linkage hierarchical clustering method to generate the proper number of subpopulations. In the proposed optimizer, the diversity of the population is maintained based on the differences of the number of subpopulations between two successive diversity increasing points. Li et al. [56] proposed an adaptive multipopulation framework for solving the DOPs, in which the number of populations is adjusted based on a database storing historical information of the changes in the algorithm behavior. Ali et al. [58] proposed an adaptive multipopulation version of DE, in which each population has its own life cycle and size. The life cycle and the size of each population in each generation are controlled by a success-based scheme, which adaptively changes them based on the previous success of the population. Yazdani et al. [29] proposed a multiswarm AFSA, where the swarms are categorized into parent and child swarms. The parent swarms are employed to find uncovered peaks and the child swarms are responsible for covering and tracking the located peaks. Whenever a parent swarm converges to a new peak, it generates a child swarm to cover that peak and track it. This parent-child mechanism was proposed to address the challenging issue of the unknown number of peaks.

The diversity in BA-based algorithms is created and maintained by keeping a large portion of the population $(n - m)$ scouting for new promising solutions. In the PLBA, features such as the patch concept and Levy flights help the PLBA to track more than one peak in the sense of maximization problems. The patch environment in the initialization part helps in spreading the solutions out along the search space. The Levy flights in the initialization and global search parts with a suitable search size enhance PLBA to maintain diversity because of the rare long jumps of these flights. In

addition, at the same time, the greedy local search based on Levy flight with a suitable small search size reduces the length of the long steps of Levy flight that work together with the frequent short steps on exploiting the new regions and thus finding the new optimum.

The key objectives of this paper are as follows:

(1) To validate the performance of the recently proposed PLBA and other variants of BA on the set of recently proposed dynamic multimodal benchmarks [34, 35] mentioned above and compare among them.

(2) To apply the PLBA to challenging DOPs and compare it with other state-of-the-art algorithms.

(3) To show the advantage of modelling additional natural aspects in nature-inspired metaheuristics such as BA, which are the patch concept and Levy motion.

The remainder of this paper is organized as follows. Section 2 provides a brief description of the Basic BA, Shrinking-based BA, Patch-Levy-based Bees Algorithm (PLBA), and other state-of-the-art algorithms. Section 3 describes the adaptation of the PLBA algorithm and other BA versions to deal with dynamic environments. Section 4 presents the results of performance evaluations and experiments obtained for the PLBA and compares them with those obtained using other BA variants and other state-of-the-art algorithms. Finally, Section 5 concludes this paper.

2. Brief Description of the Compared Algorithms

2.1. Basic BA. The Bees Algorithm (BA) is a bee-based optimization algorithm inspired by the foraging behavior of a swarm of honeybees. Basic BA performs a kind of exploitative local search combined with an exploratory global search [59]. Both search modes implement a uniform random search. In the global search, the scout bees are uniformly distributed at random to different areas of the search space to scout for potential solutions. In the local search, follower bees are recruited to exploit patches that scout bees have found to be more promising. Two processes are required to conduct the local search, namely, the selection and recruitment processes. In the selection process, the patches found to be more promising are chosen, whereas in the recruitment operation, follower bees are recruited for the promising patches, while more bees are recruited for the best patches out of those selected.

2.2. Shrinking-Based BA. Shrinking-based BA is an improved version over Basic BA, which includes the neighborhood shrinking step as an additional step over Basic BA [59]. In this paper, the shrinking procedure is implemented to all of the patches simultaneously, using a global memory at each iteration of the BA, after the recruitment stage [59].

2.3. Patch-Levy-Based Bees Algorithm (PLBA). PLBA is an enhanced variant of BA, in which the population initialization, local search, and global search are performed according to the patch concept and Levy motion [45, 46]. In the initialization part of PLBA, the search space that represents the environment is divided into clear segments, which represent patches, such that the food sources are clearly distributed in patches. The centres of the patches are used to represent areas inside the hive. Then, the bees are distributed randomly from these hive areas according to the Levy flight distribution, which is believed to approximate the natural flight patterns of bees. In the global search, the scout bees are distributed from the hive areas, which are chosen to be the same areas of the hive from which they are initially distributed. Then, the bees start to scout according to the Levy flight as in the initial step. The local search inside a patch is performed based on Levy looping flights. The patch and Levy concepts in the PLBA algorithm are modelled in the initialization, local, and global parts. The pattern in Levy flights can be described by many relatively short steps (corresponding to the detection range of the searcher) that are separated by occasional longer jumps. Thus, generating step sizes according to a Levy distribution can be advantageous since the frequent short steps help in making more population members exploit the most promising regions and at the same time the rarely occurred long jumps help in keeping a portion of the members exploring the distant regions of solution space [46]. This can help in accelerating the convergence to the optimal solutions.

2.4. Other State-of-the-Art Algorithms. As stated in the introduction (Section 1), the optimization algorithms that were tested on the recently proposed dynamic benchmarks [34, 35] include dopt-aiNet [1], PSO [1], rPSO [38], rGA [38], Memory-based EP [36], CPSO [38], DASA [10, 37], and MDE-LiGO [11]. Therefore, these algorithms were employed in the comparisons of this paper as the other state-of-the-art algorithms.

The dopt-aiNet is the artificial immune network algorithm designed for dynamic optimization. The dopt-aiNet maintained the diversity by detecting the redundant solutions and removing the worst ones, replacing them and inserting newly generated solutions. To rapidly locate the nearest local optimum, the Gaussian mutation in opt-aiNet was modified, where the step size is automatically calculated using a Golden Section Procedure. The dopt-aiNet employed a small and constantly changing population.

The PSO is the standard PSO without restart. The rPSO (PSO with restart) is the standard PSO, in which the population is reinitialized when an environmental change is detected. The rGA is the standard genetic algorithm which reinitializes the population when a dynamic change is detected.

The Memory-based EP is the evolutionary programming algorithm with an ensemble of memories to address the dynamic optimization. The Memory-based EP proposed a dynamic strategy parameter to make the algorithm more suitable for the dynamic optimization. The algorithm enhanced the diversity of the population by using an ensemble of external memories to be used as the initial population for a new environmental change. When the memories cannot help and the population loses its diversity, the population is reinitialized except the best one.

The CPSO is a variant of PSO that employs a hierarchical clustering technique and a fast local search procedure to deal

with the optimization problems. CPSO modified the learning strategy in standard PSO for the global search to cover as many promising local optima as possible and speed the local search. Then, CPSO used an adaptive multipopulation strategy by using a single-linkage hierarchical clustering technique to generate a suitable number of subswarms and track multiple peaks. The hierarchical clustering can be considered better than the traditional multipopulation method in the sense that it helps in producing the proper number of subpopulations [38].

The DASA is then differential ant-stigmergy algorithm. It is an ACO-based algorithm integrated with a differential graph representation. The DASA used a pheromone mechanism as a way of communication between ants. The pheromone mechanism is an automatic and dynamic means with negative and positive feedbacks imitating the short-term and long-term memories [60].

The MDE-LiGO is a modified variant of DE that utilized locality-induced genetic operators. The MDE-LiGO maintained the diversity by extending the traditional mutation and crossover operators of DE to locality-based operators to retain the local traits of the best parents in the offspring, thus helping in preserving the diversity of the population. The locality-induced operations guide the newly generated solutions to cover promising local regions throughout the search space. Then, the MDE-LiGO performed Fuzzy *C*-Means (FCM) clustering method to divide the population into distinct local regions. The best solutions from each cluster are retained and the others are reassigned (i.e., reinitialized) in the search space for a new environmental change to cover new potential regions. FCM can be considered better than the hard clustering because it is especially suitable for the overlapped datasets that are common in the optimization problems that have many basins of attractions [11].

3. The PLBA and Other BA Variants for Dynamic Optimization

In the PLBA and other BA-based algorithms used in the comparisons, to deal with the dynamic changes, the solutions of the last iteration before the detection of the change are reevaluated and used as a good start for the next change. In addition, the parameters that are related to shrinking procedure, *ngh* in Shrinking-based BA and γ_2 in the PLBA, are reinitialized each time a change is detected. In the case of dimensional change, in addition to the reevaluation of the last solutions and the reinitialization of the shrinking parameters, the solutions are resized. Therefore, it can be easily observed that the PLBA and other BA-based algorithms are used without any modification except for the reevaluations of the solutions when the change is detected and the reinitialization of the shrinking parameters.

4. Experimental Setup and Results

4.1. Experimental Setup

4.1.1. Benchmark Functions. The performance of the PLBA is evaluated in the dynamic environments using 6 dynamic multimodal benchmark problems. These problems were generated by the GDBG system that was proposed by Li and Yang [33] and provided for CEC'2009 competition on evolutionary computation in dynamic environments. According to Li et al. [34], the control parameters of the GDBG system can undergo 7 change types: T_1 (small step change), T_2 (large step change), T_3 (random change), T_4 (chaotic change), T_5 (recurrent change), T_6 (recurrent change with noise), and T_7 (random change with dimensional change). Detailed information regarding the framework of these changes can be found in the description given by Li et al. [34, 35].

Table 1: Summary of the dynamic benchmark functions employed in the experiments.

Function number	Search space	Function name
F_1	$X\,[-5,5]^d$	Dynamic rotation peak with 10 Dynamic rotation peak with 50
F_2	$X\,[-5,5]^d$	Composition of Sphere function
F_3	$X\,[-5,5]^d$	Composition of Rastrigin function
F_4	$X\,[-5,5]^d$	Composition of Griewank function
F_5	$X\,[-5,5]^d$	Composition of Ackley function
F_6	$X\,[-5,5]^d$	Hybrid composition function

The summary of the employed benchmarks is given in Table 1. The first benchmark problem F_1 is the dynamic rotation peak function with 10 and 50 peaks. Thus, two instances of F_1 are used. The other five test problems (F_2–F_6) are dynamic composition benchmark functions. Each problem instance undergoes the seven change types (T_1–T_7); thus, a total of 49 test cases are examined. The most important parameters of the test problems were set as follows: dimension (d) = 10 in the case of unchanged dimension and $d \in [5, 15]$ in the case of dimensional change, the search space $(X) \in [-5,5]^d$, the change frequency (*frequency* or Max_FES/change) = 10,000 $\times$ d, and the number of changes per run (num_change/run) = 60. The remaining parameters can be found in the description given by Li et al. [34].

4.1.2. Performance Evaluation. The performance of PLBA in the dynamic environments is evaluated on 49 test cases that result from testing seven change types with each problem instance out of the seven problem instances where the first problem has two instances, as can be seen in Table 1. For each test case, the average best (Avg_best), the average mean (Avg_mean), and the standard deviation (STD) of the absolute error in the function value over 20 runs are calculated as follows:

$$\begin{aligned}
\text{Avg_best} &= \sum_{i=1}^{RN} \min_{j=1}^{\text{num_change}} \frac{E_{i,j}^{\text{last}}(t)}{RN},\\
\text{Avg_mean} &= \sum_{i=1}^{RN} \sum_{j=1}^{\text{num_change}} \frac{E_{i,j}^{\text{last}}(t)}{(RN \times \text{num_change})},\\
\text{STD} &= \sqrt{\frac{\sum_{i=1}^{RN} \sum_{j=1}^{\text{num_change}} \left(E_{i,j}^{\text{last}}(t) - \text{Avg_mean}\right)^2}{RN \times \text{num_change}}},
\end{aligned} \tag{1}$$

Table 2: The parameters used with different values for different dynamic problems in the BA variants.

Function	Basic BA	Shrinking-based BA		PLBA					
	ngh	ngh_{init}	sf	P	γ_1	γ_{2_init}	γ_3	t	sf
F_1	0.001	5	0.990	15	1	0.1	1	1	0.990
F_2	0.001	5	0.980	15	0.1	1	1	5	0.970
F_3	1	3	0.999	19	0.1	1	1	5	0.990
F_4	0.1	3	0.990	19	0.1	1	1	5	0.980
F_5	0.1	3	0.990	19	0.1	1	1	5	0.980
F_6	0.009	3	0.990	19	0.1	1	1	5	0.980

where $E_{i,j}^{\text{last}}(t) = |f(x_{\text{best}}(t)) - f(x^*(t))|$ is the absolute error in a function value for each change after reaching the maximum number of evaluations per change (Max_FES/change). The Max_FES/change = 10,000 $*$ d. $x_{\text{best}}(t)$ and $x^*(t)$ are the best solution and global optimum, respectively, at time t. Additionally, RN is the total number of runs of the considered algorithm.

The average best and average mean of the absolute error values are used to ascertain the solution quality obtained by the PLBA on the dynamic problems. The average best gives an idea of how close the algorithm got from the global optimum during the whole environmental change type [1], whereas the average mean provides an indication of how close the algorithm was from the global optimum during the entire optimization process [1].

To evaluate the performance of the PLBA in terms of both solution quality and convergence speed, the overall performance of PLBA is measured according to Li et al. [34] as follows:

$$\text{Performance} = \sum_{k=1}^{49} \text{mark}_k, \tag{2}$$

where mark_k is the marking measurement of the performance of the algorithm on the kth test case among 49 test cases. The maximum mark values ($\text{mark}_{\max}$), as concluded from the description given by Li et al. [34], are 1.5, 2.4, 1, and 1.6 for (T_1–T_6) in F_1, (T_1–T_6) in (F_2–F_6), T_7 in F_1, and T_7 in (F_2–F_6), respectively. The corresponding percentages are 0.015, 0.024, 0.01, and 0.016, respectively. Based on these percentages, the mark_k is defined as

$$\text{mark}_k = \text{percentage}_k \times \sum_{i=1}^{RN} \sum_{j=1}^{\text{num\_change}} \frac{r_{i,j}}{(RN \times \text{num\_change})}, \tag{3}$$

where $r_{i,j}$ is defined as follows:

$$r_{i,j} = \frac{r_{i,j}^{\text{last}}}{1 + \sum_{s=1}^{S} \left(\left(1 - r_{i,j}^{s}\right)/S\right)}, \tag{4}$$

where $r_{i,j}^{\text{last}}$ is the relative value of the best solution to the global optimum for each change after reaching Max_FES and $r_{i,j}^{\text{last}} = f(x_{\text{best}}(t))/f(x^*(t))$ for maximization problems and $r_{i,j}^{\text{last}} = f(x^*(t))/f(x_{\text{best}}(t))$ for minimization problems, whereas $r_{i,j}^{s}$ is the relative value of the best solution to the global optimum at the sth sampling during one change, and S = Max_FES/change/s_f, where s_f is the sampling frequency and $s_f = 100$ according to Li et al. [34].

Table 3: Summary of the overall performance of the algorithms employed in the comparisons.

Algorithm	Overall performance
DASA [37]	65.201996
Memory-EP [36]	58.093927
CPSO [38]	57.574184
PLBA	*45.72351*
Shrinking-based BA	43.447886
dopt-aiNet [1]	38.29035
rGA [38]	38.079177
rPSO [38]	33.255316
Basic BA	13.15165
PSO [1]	0.0014

Additionally, the median performance of the relative value ($r(t)$) of the best solution ($x_{\text{best}}(t)$) to the global optimum ($x^*(t)$) is depicted on convergence graphs for each problem for total runs with termination after Total_FES over num_change changes.

4.1.3. Parameter Settings. The performance of PLBA is compared with that of Basic BA and Shrining-based BA in the dynamic environment. In addition, an additional set of experiments is conducted to compare the performance of PLBA with other variants of the current state-of-the-art population-based algorithms. The parameter settings of these algorithms can be found in their references.

In order to perform a fair comparison among the BAs, the three versions of BA are executed with the same setting for the common parameters: $n = 20$ for the number of scout bees, $m = 3$ for the number of selected sites, $e = 1$ for the number of elite sites, $nep = 4$ for the number of recruited bees for each site of the e sites, and $nsp = 1$ for the number of bees recruited for every site of the remaining ($m - e$) sites. In addition, the parameters relevant to each version are set to different values for different problems, as shown in Table 2.

4.2. Experimental Results. Because no results regarding the mark scores of MDE-LiGO were reported by Mukherjee et

Table 4: Average best error values achieved for problems F_1–F_6 by PLBA.

Problem	T_1 Avg_best	T_2 Avg_best	T_3 Avg_best	T_4 Avg_best	T_5 Avg_best	T_6 Avg_best	T_7 Avg_best
$F_1(10)$	0	0	0	0	0	0	0
$F_1(50)$	0	0	0	0	0	0	0
F_2	0	0	0	0	0	0	0
F_3	0	8.6383	$5.64881e-013$	0	8.68001	1.23564	0
F_4	$4.44089e-013$	0	$3.33955e-013$	0	0	0	$5.29354e-013$
F_5	$4.08562e-014$	$4.26326e-014$	$4.26326e-014$	$4.08562e-014$	$4.26326e-014$	$4.08562e-014$	$4.26326e-014$
F_6	0	0	0	0	0	0	0

Table 5: Mean error values achieved for problems F_1–F_6 by PLBA.

Problem	T_1		T_2		T_3		T_4		T_5		T_6		T_7	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
$F_1(10)$	8.07	13.04	13.13	15.21	**6.10**	10.71	14.93	16.93	**2.99**	5.31	14.51	19.92	9.07	10.05
$F_1(50)$	11.62	11.47	11.98	13.50	**8.92**	7.99	16.72	18.85	3.56	4.43	19.96	24.34	9.82	11.72
F_2	13.68	12.21	**77.09**	149.65	**72.69**	142.82	**28.70**	25.69	**124.44**	171.60	**12.90**	17.66	**11.89**	14.46
F_3	**59.67**	159.57	**790.01**	248.01	**709.51**	290.15	**477.03**	450.46	**724.34**	192.92	**533.58**	435.76	**549.51**	370.49
F_4	20.34	19.11	289.50	267.22	**149.17**	229.86	**16.35**	20.87	**249.67**	236.95	**24.69**	66.18	91.94	170.28
F_5	**10.45**	11.88	14.85	19.22	12.77	15.00	**24.34**	27.84	15.57	17.13	**10.57**	15.10	30.68	178.70
F_6	**9.34**	13.81	**34.62**	68.48	**48.26**	368.03	**18.37**	23.67	**110.27**	225.74	31.99	79.47	**25.05**	45.82

Table 6: Average best error values achieved for problems F_1–F_6 by Shrinking-based BA.

Problem	T_1 Avg_best	T_2 Avg_best	T_3 Avg_best	T_4 Avg_best	T_5 Avg_best	T_6 Avg_best	T_7 Avg_best
$F_1(10)$	0	0	0	0	0	0	0
$F_1(50)$	0	0	0	0	0	0	0
F_2	0	$3.73035e-013$	$4.68958e-013$	$3.97016e-013$	$1.0516e-012$	$3.89022e-013$	0
F_3	50.0557	59.6404	400.825	35.0292	764.626	73.2274	68.2266
F_4	0	0	0	0	0	0	0
F_5	$4.26326e-014$	$4.17444e-014$	$4.17444e-014$	$4.08562e-014$	$4.26326e-014$	$4.08562e-014$	$4.17444e-014$
F_6	$4.44089e-013$	$5.71987e-013$	$4.26326e-013$	0	0	0	0

Table 7: Mean error values achieved for problems F_1–F_6 by Shrinking-based BA.

Problem	T_1		T_2		T_3		T_4		T_5		T_6		T_7	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
$F_1(10)$	**4.38**	8.22	**12.01**	16.84	11.30	14.13	**11.01**	17.44	5.68	6.57	**11.81**	18.32	**7.08**	8.51
$F_1(50)$	**9.24**	9.75	**10.99**	11.89	9.48	10.60	**8.22**	12.44	**2.29**	3.82	**19.35**	21.06	**9.43**	9.11
F_2	**11.80**	14.80	88.87	163.86	97.74	166.20	31.58	31.39	180.59	190.25	19.86	34.82	23.72	53.02
F_3	655.75	150.43	912.92	196.89	890.63	100.87	794.33	305.47	901.48	60.69	914.66	298.7	793.31	254.67
F_4	**8.99**	11.23	**118.79**	211.95	150.28	228.57	29.02	30.32	257.19	229.92	27.21	87.61	**36.60**	108.87
F_5	28.65	20.95	**13.87**	18.40	**7.43**	10.34	27.58	27.47	**15.17**	15.98	11.25	14.72	**10.76**	12.62
F_6	30.18	17.64	47.39	98.33	69.65	178.07	33.25	62.76	194.69	306.43	**28.66**	66.42	40.54	110.67

Table 8: Average best error values achieved for problems F_1–F_6 by Basic BA.

Problem	T_1 Avg_best	T_2 Avg_best	T_3 Avg_best	T_4 Avg_best	T_5 Avg_best	T_6 Avg_best	T_7 Avg_best
$F_1(10)$	0.0575077	0.0458898	0.0412556	0.178853	0.0776968	0.131012	0.0468226
$F_1(50)$	0.0711283	0.0486264	0.0467527	0.114642	0.111445	0.0694867	0.0379723
F_2	0.803693	11.9098	8.75428	0.992304	22.436	1.09661	1.1218
F_3	587.682	804.557	774.819	585.857	863.274	530.679	469.399
F_4	62.5302	105.805	110.402	84.2297	111.169	81.7393	86.4602
F_5	193.573	201.513	204.459	175.131	255.796	175.39	123.798
F_6	19.0539	166.304	396.32	141.408	413.333	27.1496	24.2779

Table 9: Mean error values achieved for problems F_1–F_6 by Basic BA.

Problem	T_1		T_2		T_3		T_4		T_5		T_6		T_7	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
$F_1(10)$	6.52	9.53	21.50	18.02	24.98	14.60	29.54	22.32	33.53	19.39	38.45	27.07	30.30	17.86
$F_1(50)$	12.00	9.99	20.14	13.48	24.32	14.29	23.91	21.01	19.36	15.17	25.35	23.59	27.74	15.63
F_2	32.92	49.21	451.14	145.69	407.20	147.86	200.54	231.59	401.04	89.40	334.78	255.47	349.24	204.19
F_3	778.73	70.90	1040.93	69.30	996.42	78.99	929.32	250.12	1015.76	58.92	1000.11	260.08	912.83	144.03
F_4	121.99	48.65	511.29	179.66	471.55	171.99	236.81	195.04	455.12	123.28	318.18	245.83	332.36	193.25
F_5	297.68	51.64	366.33	63.20	355.66	101.99	302.17	54.18	409.26	123.51	311.39	101.42	355.64	278.32
F_6	359.18	324.98	869.70	125.34	824.36	94.28	695.83	258.23	833.96	88.88	772.18	253.43	763.19	209.37

Table 10: Average rankings of BA variants (Friedman) on dynamic benchmark problems F_1–F_6 in terms of the solution quality.

Algorithm	Ranking
PLBA	1.4286
Shrinking-based BA	1.5918
Basic BA	2.9796
p value	0

Table 11: Average rankings of BA variants (Friedman) on dynamic benchmark problems F_1–F_6 in terms of both solution quality and convergence speed.

Algorithm	Ranking
PLBA	1.4286
Shrinking-based BA	1.5714
Basic BA	3
p value	0

Table 12: Adjusted p values associated with BA variants (Friedman) on dynamic benchmark problems F_1–F_6 in terms of the solution quality.

Algorithm	Unadjusted p	p Holm	p Hochberg
Basic BA	0	0	0
Shrinking-based BA	0.41902	0.41902	0.41902

Table 13: Adjusted p values associated with BA variants (Friedman) on dynamic benchmark problems F_1–F_6 in terms of both solution quality and convergence speed.

Algorithm	Unadjusted p	p Holm	p Hochberg
Basic BA	0	0	0
Shrinking-based BA	0.4795	0.4795	0.4795

Table 14: Average rankings of PLBA and other algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of the solution quality.

Algorithm	Ranking
MDE-LiGO	1.5102
CPSO	3.0408
DASA	3.2857
Memory-based EP	3.7959
PLBA	*5.3878*
dopt-aiNet	5.4898
rGA	6.2041
rPSO	7.2857
p value	0

al. [11], the MDE-LiGO was not included in the comparisons of the overall performance (see Table 3) and the comparisons among the algorithms in terms of both the solution quality and convergence speed (see Tables 15 and 17), which are indicated by the mark scores. The MDE-LiGO was employed in the comparisons in terms of the solution quality only (see Tables 14 and 16) because of the availability of its average means in Mukherjee et al. [11], which give indication of the quality of the solutions. Table 3 presents a summary of the overall performance of the PLBA, Shrinking-based BA, Basic BA, dopt-aiNet, PSO, rPSO, rGA, Memory-based EP, CPSO, and DASA. In addition, the overall performance and mark scores of these algorithms on each test case are given in Tables 22–30. Tables 4–9 show the results of the average best, average mean, and standard deviations of the error values achieved for each test case by PLBA, Shrinking-based BA, and Basic BA over 20 runs. Figures 1–3 present the convergence graphs of PLBA, Shrinking-based BA, and Basic BA, respectively, on the tested problems. In these figures, relative values ($r(t)$) are presented for each change type (T_i) as $r(t) + i - 1$, where $0 \leq r(t) \leq 1$ and $i = 1, 2, \ldots, 7$.

4.2.1. Analysis of the Results of the PLBA. From the overall performance and mark scores in Table 22 and from Figure 1, it can be seen that the PLBA algorithm achieved the best performance on F_1 (the rotation peak problem), followed by F_5 (the composition Ackley problem) over all change types. On the other hand, the PLBA algorithm had the worst performance on F_3 (the composition Rastrigin). For this function, all the compared algorithms had the worst performance on this function and they could not track the optimum of this function. This is may be attributed to the difficulties that are exposed by the surface of this function in static environments [61] that make it more complicated in the case of dynamic environments. Nevertheless, PLBA performed better than some algorithms such as dopt-aiNet, CPSO, rPSO, Shrinking-based BA, and Basic BA on this function.

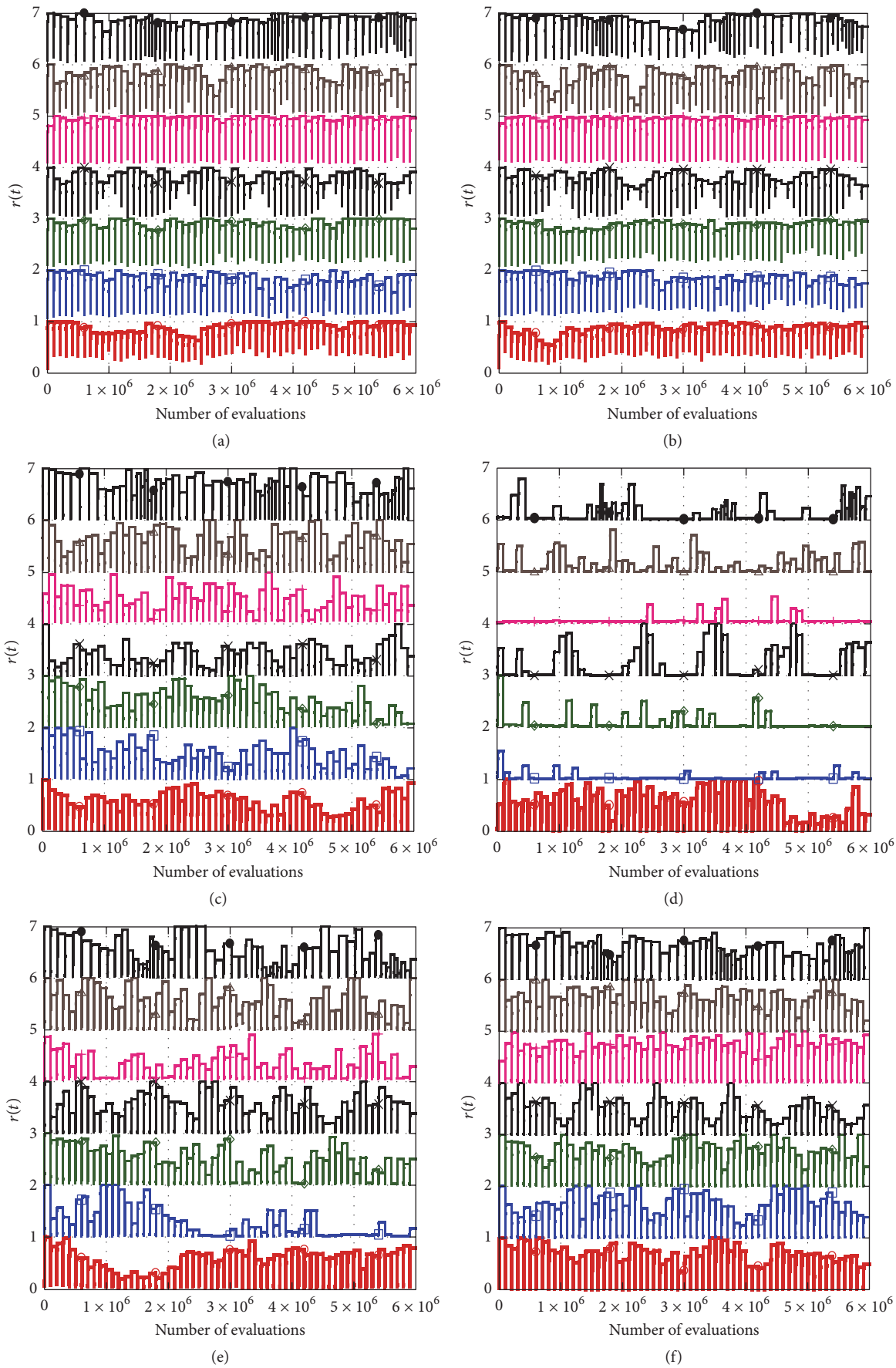
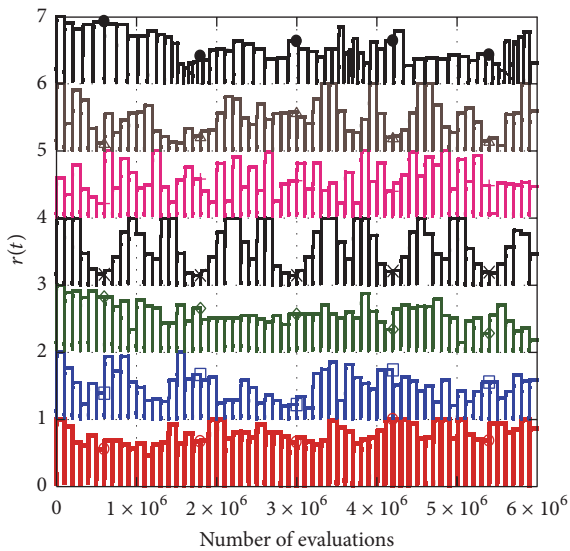


Figure 1: Continued.



(g)

Figure 1: Convergence graph of PLBA for (a) F_1 ($m = 10$), (b) F_1 ($m = 50$), (c) F_2, (d) F_3, (e) F_4, (f) F_5, and (g) F_6 where relative values ($r(t)$) are presented for each T_i as $r(t) + i - 1$, where $0 \leq r(t) \leq 1$ and $i = 1, 2, \ldots, 7$.

Table 15: Average rankings of PLBA and other algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of both solution quality and convergence speed.

Algorithm	Ranking
DASA	1.551
CPSO	2.5714
Memory-based EP	2.9796
PLBA	*4.2041*
rGA	5.1429
dopt-aiNet	5.4082
rPSO	6.1429
p value	0

Table 16: Adjusted p values associated with PLBA and other algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of the solution quality.

Algorithm	Unadjusted p	p Holm	p Hochberg
rPSO	0	0	0
rGA	0	0	0
dopt-aiNet	0	0	0
PLBA	0	0	0
Memory-based EP	0.000004	0.000012	0.000012
DASA	0.000333	0.000667	0.000667
CPSO	0.001982	0.001982	0.001982

For other problems, the performance of the PLBA algorithm is fairly well and can be considered competitive with other optimization techniques. Furthermore, it can be seen in Table 22 and Figure 1 that although the dimensional change is the most challenging scenario, the PLBA performed fairly well on this change type in all benchmark problems and better than other algorithms such as Basic BA, dopt-aiNet, and rPSO, as can be seen in Tables 24–26.

Table 17: Adjusted p values associated with PLBA and other algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of both solution quality and convergence speed.

Algorithm	Unadjusted p	p Holm	p Hochberg
rPSO	0	0	0
dopt-aiNet	0	0	0
rGA	0	0	0
PLBA	0	0	0
Memory-based EP	0.001063	0.002126	0.002126
CPSO	0.019385	0.019385	0.019385

By investigating the results in Tables 4 and 5 and in Figure 1, it can be seen that the PLBA performed well on F_1 (the rotation peak function) and F_5 (the composition Ackley function) over all change types. This confirms what has been pointed out that PLBA had the best performance on these two functions. For F_2 (the composition sphere function) and F_6 (the hybrid composition problem), the PLBA presented similar behaviors and obtained good results in change types T_1 (the small change), T_4 (the chaotic change), T_6 (the recurrent change with noise), and T_7 (the random change with changed dimension). On the other hand, PLBA struggled to find the optimum in other change types, especially the recurrent change (T_5) where the average mean of this change type was very high compared to other types.

Concerning the composition Griewank function (F_4), although PLBA was able to produce good results in some change types, it still struggled to search for the optimum in some change types, especially T_2 (large step change), T_3 (random change), and T_5 (recurrent change) where their

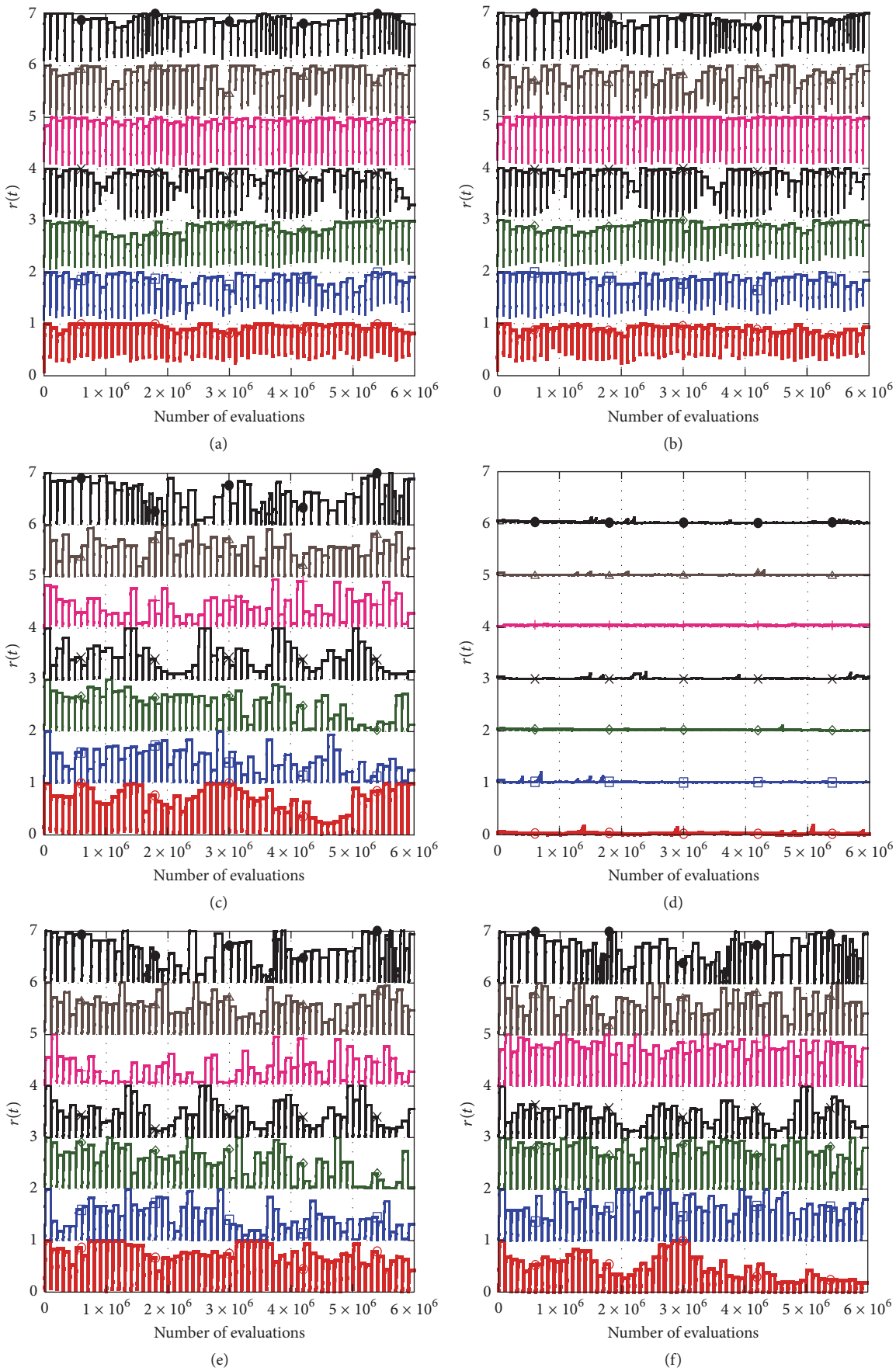
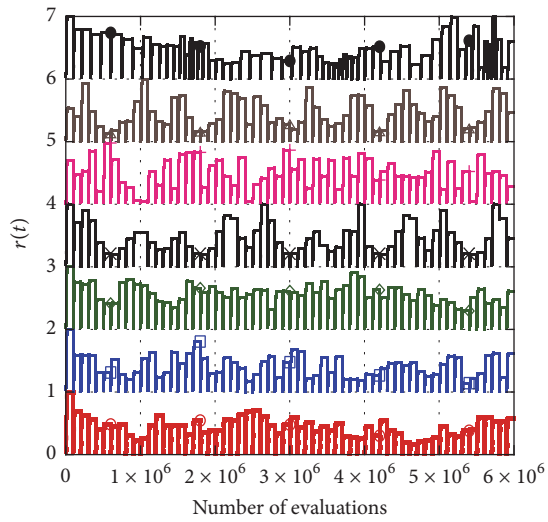


Figure 2: Continued.



(g)

Figure 2: Convergence graph of Shrinking-based BA for (a) F_1 ($m = 10$), (b) F_1 ($m = 50$), (c) F_2, (d) F_3, (e) F_4, (f) F_5, and (g) F_6 where relative values ($r(t)$) are presented for each T_i as $r(t) + i - 1$, where $0 \leq r(t) \leq 1$ and $i = 1, 2, \ldots, 7$.

average means were very high compared to other change types. Regarding the most difficult function, which is the composition Rastrigin function (F_3), although PLBA was able to reach the global optimum or stay close to it in its best cases (Avg_best results) in almost all change types, PLBA, on average, performed badly in all change types except for the small change type where its average mean was very low compared to other change types as expected, because of the simplicity of this change type.

An interesting finding is that although it is quite obvious for the problems to be solved smoothly in the case of the small step change, it is not natural for an algorithm to handle the chaotic or dimensional changes. Nevertheless, PLBA was able to handle these change types as well. It can also be observed that PLBA performed fairly well on F_6, even though it is a hybrid composition function, as can be seen in the Avg_best results in all change instances and in the Avg_mean in most change types.

4.2.2. Comparisons among the BAs. By comparing the overall performance of all BA-based algorithms, it can be clearly seen in Table 3 and Figures 1–3 that the Basic BA is much worse than the PLBA and Shrinking-based BA. On the other hand, the PLBA achieved the best performance. However, although the overall performance of PLBA is higher than that of Shrinking-based BA, the difference is not significant and the results are generally comparable, as can be seen in Figures 1 and 2 and as will be observed in the statistical analysis in Tables 12 and 13. This may be due to the dynamic change that is caused by the shrinking procedure and is suitable to the dynamic environment to track the optimum. In addition, the number of dimensions ($d = 10$) that is not very large may contribute to this comparable performance of Shrinking-based BA with PLBA. By examining the results of test cases, it can be observed in Tables 4 and 6, Tables 5 and 7, Tables 22 and 23 and in Figures 1(d) and 2(d) that PLBA achieved much better performance than Shrinking-based BA in all change types on the composition Rastrigin function (F_3), which is the most difficult problem. As mentioned before, this function exposes difficulties in the static environments, and it is expected to expose more difficulties in the dynamic environment.

The results of the average means and mark scores of test cases were analyzed statistically to test the significance of the results in terms of solution quality only and in terms of both solution quality and convergence speed, respectively. To statistically analyze the results produced by PLBA and other BA versions, the Friedman test was used. The Friedman test is the best-known statistical method for testing the performance differences between more than two algorithms [62]. Tables 10 and 11 tabulate the ranks achieved by this test for the three BA-based algorithms using the average mean and mark score results, respectively. It can be clearly seen in these tables that PLBA ranked first, followed by Shrinking-based BA in terms of solution quality only or in terms of both solution quality and convergence speed. The p values calculated using the Friedman test (0) showed a highly significant difference among the BA-based algorithms.

Subsequently, Holm and Hochberg post hoc tests were used to detect the concrete differences among the BA-based algorithms. Tables 12 and 13 depict the adjusted p values obtained by the pot hoc tests considering the PLBA as the control method using the average mean and mark score results, respectively. As can be seen in these tables, the Holm and Hochberg test strongly suggested a highly significant difference in the performance between the PLBA and the Basic BA in terms of solution quality only or in terms of both solution quality and convergence speed. On the other hand, no significant difference was found between PLBA and Shrinking-based BA. This confirmed the previous conclusion that has been done based on the overall performance.

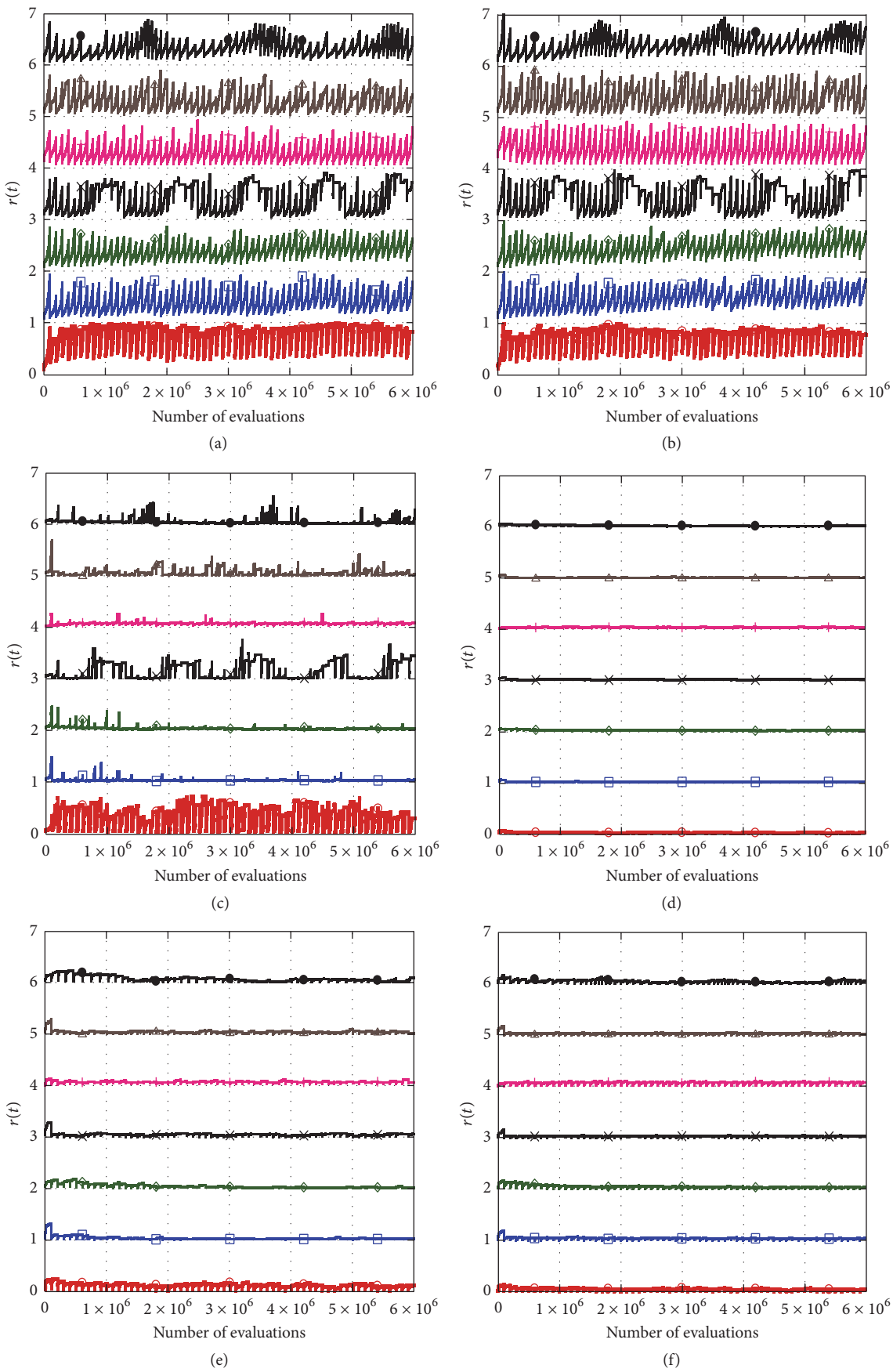
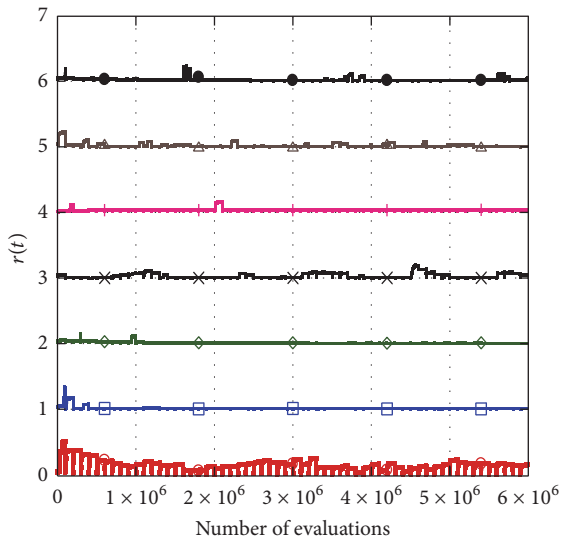


Figure 3: Continued.



(g)

Figure 3: Convergence graph of Basic BA for (a) F_1 ($m = 10$), (b) F_1 ($m = 50$), (c) F_2, (d) F_3, (e) F_4, (f) F_5, and (g) F_6 where relative values ($r(t)$) are presented for each T_i as $r(t) + i - 1$, where $0 \leq r(t) \leq 1$ and $i = 1, 2, \ldots, 7$.

4.2.3. Comparisons with Other State-of-the-Art Algorithms. From the overall performance in Table 3, it can be obviously noticed that the standard PSO without any modification had the worst performance and the DASA achieved the best performance. It can also be observed that the PLBA performed better than rPSO, rGA, and dopt-aiNet and performed worse than CPSO, Memory-based EP, and DASA in the dynamic environment. A point of interest to note is that the PLBA achieved better performance than the dopt-aiNet that was designed especially to deal with DOPs and included all the desirable features to deal with these problems such as diversity generation, diversity maintenance, memory usage, and multipopulation capability.

It is also worth noting that although CPSO outperformed PLBA, PLBA performed better than CPSO on F_3 (the composition Rastrigin). PLBA also performed better than dopt-aiNet on F_5 (the composition Ackley problem) and on F_6 (the hybrid composition function) in all change types (see Tables 22, 25, and 29).

There is an interesting finding that the Basic BA performed much better than the standard PSO, with total mark score of 13.15165 and 0.0014, respectively. Basic BA may take advantage from its better diversity by keeping a large portion of the population scouting for new promising solutions. Another interesting finding is that PLBA achieved good results although it does not have all the required features in the dynamic environments such as the usage of memory and multipopulation capabilities. However, as mentioned before in Section 1, features such as the patch concept and Levy flights help in guiding the search in different promising areas along the search space. The patch environment in the initialization part helps in spreading the solutions out along the search space. The Levy flights with a suitable search size in the initialization and global search parts enhance PLBA to maintain diversity because of the rare long jumps. In addition, at the same time, the greedy local search based on Levy flight with a suitable small search size reduces the length of the long steps of Levy flight that work together with the frequent short steps on exploiting the new regions and thus finding the new optimum.

The Friedman test was also used to test the global differences in the results obtained by PLBA and the other state-of-the-art algorithms in terms of solution quality only and in terms of both solution quality and convergence speed. Tables 14 and 15 depict the ranks calculated through the Friedman test employing the average mean and mark score results, respectively. As mentioned above in Section 4.2, the results of MDE-LiGO were not included in the comparisons of the mark scores, which give indication of both the solution quality and convergence speed. On the other hand, the MDE-LiGO was considered in the comparisons of the average means, which give indication of the solution quality only. This was because there were no results of the mark scores reported for the MDE-LiGO, as mentioned in Section 4.2. Therefore, including MDE-LiGO, it can be clearly seen that MDE-LiGO was the best performing algorithm and CPSO was the second best performing algorithm considering only the solution quality. Considering both solution quality and convergence speed, DASA was the best algorithm and CPSO was the second best algorithm, given that the MDE-LiGO was excluded for the reason explained above. The different ranks of CPSO and DASA in the two cases showed that the CPSO achieved better results than DASA in the dynamic environments but DASA was faster than CPSO. In both cases, rPSO was the worst performing algorithm. For the PLBA, it ranked the fifth in the first case and the fourth in the second case. The p values calculated using the statistic from the Friedman test (0) strongly suggested a significant difference among the algorithms considered.

Table 18: Average rankings of PLBA and other comparable algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of the solution quality.

Algorithm	Ranking
PLBA	1.8163
dopt-aiNet	2.2245
rGA	2.5306
rPSO	3.4286
p value	0

Table 19: Average rankings of PLBA and other comparable algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of both solution quality and convergence speed.

Algorithm	Ranking
PLBA	1.6327
rGA	2.449
dopt-aiNet	2.5918
rPSO	3.3265
p value	0

Subsequently, Holm and Hochberg methods were used to detect the concrete difference between the best performing algorithm and the rest of the algorithms. Table 16 shows the adjusted p values obtained by Holm and Hochberg methods considering MDE-LiGO (the best algorithm considering only the solution quality) as the control method, whereas Table 17 presents the adjusted p values considering DASA (the best algorithm considering both solution quality and convergence speed) as the control method. It can be clearly seen in Table 16 that Holm and Hochberg test strongly suggested a significant difference between MDE-LiGO and each of the rest of the algorithms in terms of solution quality at significance level $\alpha = 0.01$. Considering both solution quality and convergence speed, Holm and Hochberg test strongly suggested a significant difference between DASA and each of the remaining algorithms. It should be noted that the worst performing algorithm was actually PSO or Basic PSO, but because there is no detailed results reported for it by De França and Von Zuben [1], it was not included in the statistical comparisons.

Two additional sets of statistical comparisons were conducted among the PLBA and the algorithms that were outperformed by PLBA employing the average mean and mark score results. The aim was to check if there is a significant difference among them, in general, and between PLBA and each of these algorithms, in particular. Regarding the ranks, it can be clearly seen in Tables 18 and 19 that PLBA ranked first, as can also be seen from the previous rankings where PLBA ranked before these algorithms. The p values computed based on the ranks calculated by the Friedman test (0) suggested a highly significant difference among the algorithms. Considering only the solution quality, the Holm and Hochberg tests showed a highly significant improvement of PLBA over rPSO at a significance level $\alpha = 0.01$ and over rGA with a level of significance $\alpha = 0.05$,

Table 20: Adjusted p values associated with PLBA and other comparable algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of the solution quality.

Algorithm	Unadjusted p	p Holm	p Hochberg
rPSO	0	0	0
rGA	0.00617	0.01234	0.01234
dopt-aiNet	0.117601	0.117601	0.117601

Table 21: Adjusted p values associated with PLBA and other comparable algorithms (Friedman) on dynamic benchmark problems F_1–F_6 in terms of both solution quality and convergence speed.

Algorithm	Unadjusted p	p Holm	p Hochberg
rPSO	0	0	0
dopt-aiNet	0.000235	0.000471	0.000471
rGA	0.001749	0.001749	0.001749

whereas no significant difference was found between dopt-aiNet and PLBA, as can be seen in Table 20. On the other hand, considering both solution quality and convergence speed, PLBA showed a significant improvement over all other considered algorithms, including dopt-aiNet, with a level of significance $\alpha = 0.01$, as can be seen in Table 21.

In general, the superiority of an algorithm over other algorithms can be accounted for by its better diversity and its ability to adapt to environmental changes because of the strategies and features that characterize this algorithm to address the DOPs. For example, it can be noticed that MDE-LiGO was the best performing algorithm and rGA, rPSO, and standard PSO were the worst performing algorithms in terms of solution quality. This can be explained by the notion that the MDE-LiGO integrated features that have advantages over the features of the other algorithms such as the FCM clustering that is considered better than hard clustering in CPSO as pointed out in Section 2.4. It also retained the traits of the best individuals in offspring, thus helping in increasing the diversity of the population. On the other hand, PSO might face the difficulty of diversity loss due to the attraction of the global best particle to all particles causing premature convergence on local optima [38]. For the rGA and rPSO, they represented the simple and standard versions of GA and PSO, respectively, with only a reinitialization mechanism to maintain the diversity when an environmental change is detected.

In addition, it can be observed that PLBA achieved better results than rGA, rPSO, and standard PSO. This can be accounted for by the patch concept, Levy motion, and the shrinking parameters, in addition to the replacement of worst solutions (i.e., random immigrant scheme), where all these features can help in maintaining the diversity as explained above in this section and in Section 1. Keeping the last solutions of the previous dynamic change to be used as the initial solutions for the next change might contribute to the good performance of PLBA over the rGA, rPSO, and PSO. The PLBA also performed better than dopt-aiNet, although it was designed especially for dynamic environments. The lower performance for the dopt-aiNet, which is a modified variant

Table 22: PLBA performance and mark scores.

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.846037) \times 0.015$	$(0.780239) \times 0.015$	$(0.4332) \times 0.024$	$(0.407837) \times 0.024$	$(0.432046) \times 0.024$	$(0.530365) \times 0.024$	$(0.617454) \times 0.024$
T_2	$(0.760603) \times 0.015$	$(0.784613) \times 0.015$	$(0.390459) \times 0.024$	$(0.0252816) \times 0.024$	$(0.218513) \times 0.024$	$(0.504237) \times 0.024$	$(0.348125) \times 0.024$
T_3	$(0.863504) \times 0.015$	$(0.81134) \times 0.015$	$(0.43077) \times 0.024$	$(0.0632427) \times 0.024$	$(0.387817) \times 0.024$	$(0.534607) \times 0.024$	$(0.383495) \times 0.024$
T_4	$(0.746768) \times 0.015$	$(0.725178) \times 0.015$	$(0.310821) \times 0.024$	$(0.19893) \times 0.024$	$(0.458301) \times 0.024$	$(0.392676) \times 0.024$	$(0.443277) \times 0.024$
T_5	$(0.919279) \times 0.015$	$(0.90821) \times 0.015$	$(0.368444) \times 0.024$	$(0.0375549) \times 0.024$	$(0.237809) \times 0.024$	$(0.577608) \times 0.024$	$(0.407787) \times 0.024$
T_6	$(0.76002) \times 0.015$	$(0.697635) \times 0.015$	$(0.522247) \times 0.024$	$(0.110784) \times 0.024$	$(0.451481) \times 0.024$	$(0.516985) \times 0.024$	$(0.353072) \times 0.024$
T_7	$(0.812886) \times 0.01$	$(0.807512) \times 0.01$	$(0.576808) \times 0.016$	$(0.101501) \times 0.016$	$(0.410207) \times 0.016$	$(0.476383) \times 0.016$	$(0.353495) \times 0.016$
Mark	8.1572025	7.8683345	6.8171512	2.18711408	5.902652	8.09776	6.693296
Performance ($100 \times \sum$ Mark): 45.723510.							

Table 23: Shrinking-based BA performance and mark scores.

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.878606) \times 0.015$	$(0.800866) \times 0.015$	$(0.600197) \times 0.024$	$(0.0178491) \times 0.024$	$(0.61201) \times 0.024$	$(0.339452) \times 0.024$	$(0.29295) \times 0.024$
T_2	$(0.774722) \times 0.015$	$(0.78263) \times 0.015$	$(0.358243) \times 0.024$	$(0.013469) \times 0.024$	$(0.355886) \times 0.024$	$(0.555452) \times 0.024$	$(0.280425) \times 0.024$
T_3	$(0.762906) \times 0.015$	$(0.797389) \times 0.015$	$(0.428504) \times 0.024$	$(0.0137154) \times 0.024$	$(0.419966) \times 0.024$	$(0.654967) \times 0.024$	$(0.427704) \times 0.024$
T_4	$(0.786096) \times 0.015$	$(0.816008) \times 0.015$	$(0.364093) \times 0.024$	$(0.0128649) \times 0.024$	$(0.371271) \times 0.024$	$(0.34914) \times 0.024$	$(0.357991) \times 0.024$
T_5	$(0.845282) \times 0.015$	$(0.909383) \times 0.015$	$(0.318328) \times 0.024$	$(0.0199837) \times 0.024$	$(0.252069) \times 0.024$	$(0.602017) \times 0.024$	$(0.366453) \times 0.024$
T_6	$(0.775814) \times 0.015$	$(0.677594) \times 0.015$	$(0.469633) \times 0.024$	$(0.00810847) \times 0.024$	$(0.465058) \times 0.024$	$(0.49536) \times 0.024$	$(0.328168) \times 0.024$
T_7	$(0.814974) \times 0.01$	$(0.798108) \times 0.01$	$(0.487569) \times 0.016$	$(0.0148754) \times 0.016$	$(0.510078) \times 0.016$	$(0.551156) \times 0.016$	$(0.349243) \times 0.016$
Mark	8.050113	7.973913	6.8737056	0.230178008	6.7591488	8.0731808	5.4876472
Performance ($100 \times \sum$ Mark): 43.447886.							

Table 24: Basic BA performance and mark scores.

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.746208) \times 0.015$	$(0.689187) \times 0.015$	$(0.323402) \times 0.024$	$(0.0119347) \times 0.024$	$(0.0757679) \times 0.024$	$(0.0314036) \times 0.024$	$(0.104704) \times 0.024$
T_2	$(0.468538) \times 0.015$	$(0.512421) \times 0.015$	$(0.0369478) \times 0.024$	$(0.0086004) \times 0.024$	$(0.0235056) \times 0.024$	$(0.024771) \times 0.024$	$(0.0148151) \times 0.024$
T_3	$(0.421312) \times 0.015$	$(0.46508) \times 0.015$	$(0.0452712) \times 0.024$	$(0.0102585) \times 0.024$	$(0.0292991) \times 0.024$	$(0.0294702) \times 0.024$	$(0.014061) \times 0.024$
T_4	$(0.453126) \times 0.015$	$(0.493221) \times 0.015$	$(0.152998) \times 0.024$	$(0.00606636) \times 0.024$	$(0.031411) \times 0.024$	$(0.0178572) \times 0.024$	$(0.0235711) \times 0.024$
T_5	$(0.358608) \times 0.015$	$(0.473773) \times 0.015$	$(0.0506407) \times 0.024$	$(0.0171315) \times 0.024$	$(0.0421611) \times 0.024$	$(0.0425067) \times 0.024$	$(0.0227348) \times 0.024$
T_6	$(0.361845) \times 0.015$	$(0.454074) \times 0.015$	$(0.0697546) \times 0.024$	$(0.00569767) \times 0.024$	$(0.027215) \times 0.024$	$(0.0177132) \times 0.024$	$(0.0162067) \times 0.024$
T_7	$(0.380785) \times 0.01$	$(0.447991) \times 0.01$	$(0.079394) \times 0.016$	$(0.0102517) \times 0.016$	$(0.0403979) \times 0.016$	$(0.0303234) \times 0.016$	$(0.0208072) \times 0.016$
Mark	4.595241	5.079625	1.756665	0.159657	0.6151	0.44145	0.503914
Performance ($100 \times \sum$ Mark): 13.15165.							

Table 25: Dopt-aiNet algorithm performance and mark scores [1].

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.902411) \times 0.015$	$(0.894062) \times 0.015$	$(0.732236) \times 0.024$	$(0.0151338) \times 0.024$	$(0.66186) \times 0.024$	$(0.387405) \times 0.024$	$(0.462213) \times 0.024$
T_2	$(0.75644) \times 0.015$	$(0.79796) \times 0.015$	$(0.504531) \times 0.024$	$(0.00836012) \times 0.024$	$(0.289467) \times 0.024$	$(0.361252) \times 0.024$	$(0.137545) \times 0.024$
T_3	$(0.763523) \times 0.015$	$(0.791155) \times 0.015$	$(0.478578) \times 0.024$	$(0.00852786) \times 0.024$	$(0.387061) \times 0.024$	$(0.357802) \times 0.024$	$(0.118135) \times 0.024$
T_4	$(0.775512) \times 0.015$	$(0.806397) \times 0.015$	$(0.658821) \times 0.024$	$(0.00586033) \times 0.024$	$(0.570938) \times 0.024$	$(0.283915) \times 0.024$	$(0.362031) \times 0.024$
T_5	$(0.706345) \times 0.015$	$(0.742108) \times 0.015$	$(0.305631) \times 0.024$	$(0.018781) \times 0.024$	$(0.153231) \times 0.024$	$(0.0599851) \times 0.024$	$(0.0341991) \times 0.024$
T_6	$(0.674239) \times 0.015$	$(0.706515) \times 0.015$	$(0.503996) \times 0.024$	$(0.00516513) \times 0.024$	$(0.454639) \times 0.024$	$(0.13306) \times 0.024$	$(0.106351) \times 0.024$
T_7	$(0.769844) \times 0.01$	$(0.785618) \times 0.01$	$(0.570433) \times 0.016$	$(0.00880818) \times 0.016$	$(0.424358) \times 0.016$	$(0.233935) \times 0.016$	$(0.13733) \times 0.016$
Mark	7.637549	7.892914	8.553796	0.162481	6.720243	4.174502	3.148866
Performance ($100 \times \sum$ Mark): 38.290350.							

Table 26: rPSO algorithm performance and mark scores [38].

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.851997) \times 0.015$	$(0.826273) \times 0.015$	$(0.31644) \times 0.024$	$(0.0989168) \times 0.024$	$(0.325098) \times 0.024$	$(0.299304) \times 0.024$	$(0.282714) \times 0.024$
T_2	$(0.798678) \times 0.015$	$(0.777322) \times 0.015$	$(0.228137) \times 0.024$	$(0.0178981) \times 0.024$	$(0.185257) \times 0.024$	$(0.303377) \times 0.024$	$(0.265428) \times 0.024$
T_3	$(0.767943) \times 0.015$	$(0.729666) \times 0.015$	$(0.290895) \times 0.024$	$(0.0334057) \times 0.024$	$(0.2586) \times 0.024$	$(0.363386) \times 0.024$	$(0.289041) \times 0.024$
T_4	$(0.647215) \times 0.015$	$(0.659869) \times 0.015$	$(0.242093) \times 0.024$	$(0.049537) \times 0.024$	$(0.24865) \times 0.024$	$(0.231125) \times 0.024$	$(0.182946) \times 0.024$
T_5	$(0.850754) \times 0.015$	$(0.874163) \times 0.015$	$(0.314564) \times 0.024$	$(0.040489) \times 0.024$	$(0.239287) \times 0.024$	$(0.474711) \times 0.024$	$(0.391785) \times 0.024$
T_6	$(0.543241) \times 0.015$	$(0.504159) \times 0.015$	$(0.220579) \times 0.024$	$(0.0432205) \times 0.024$	$(0.213175) \times 0.024$	$(0.220548) \times 0.024$	$(0.180305) \times 0.024$
T_7	$(0.737354) \times 0.01$	$(0.709785) \times 0.01$	$(0.334463) \times 0.016$	$(0.0744435) \times 0.016$	$(0.287422) \times 0.016$	$(0.333918) \times 0.016$	$(0.294171) \times 0.016$
Mark	7.427096	7.266963	4.405640	0.799431	3.988036	5.076151	4.291999
Performance ($100 \times \sum$ Mark): 33.255316.							

Table 27: rGA algorithm performance and mark scores [38].

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.871751) \times 0.015$	$(0.843782) \times 0.015$	$(0.371132) \times 0.024$	$(0.262886) \times 0.024$	$(0.336349) \times 0.024$	$(0.383906) \times 0.024$	$(0.333081) \times 0.024$
T_2	$(0.807799) \times 0.015$	$(0.794346) \times 0.015$	$(0.253051) \times 0.024$	$(0.111445) \times 0.024$	$(0.21354) \times 0.024$	$(0.36591) \times 0.024$	$(0.29161) \times 0.024$
T_3	$(0.746394) \times 0.015$	$(0.729673) \times 0.015$	$(0.317967) \times 0.024$	$(0.161118) \times 0.024$	$(0.257419) \times 0.024$	$(0.405055) \times 0.024$	$(0.336495) \times 0.024$
T_4	$(0.672787) \times 0.015$	$(0.708963) \times 0.015$	$(0.326789) \times 0.024$	$(0.15134) \times 0.024$	$(0.292427) \times 0.024$	$(0.308862) \times 0.024$	$(0.260051) \times 0.024$
T_5	$(0.831102) \times 0.015$	$(0.874166) \times 0.015$	$(0.298643) \times 0.024$	$(0.0723514) \times 0.024$	$(0.187631) \times 0.024$	$(0.508448) \times 0.024$	$(0.38096) \times 0.024$
T_6	$(0.585837) \times 0.015$	$(0.524853) \times 0.015$	$(0.294985) \times 0.024$	$(0.139062) \times 0.024$	$(0.297623) \times 0.024$	$(0.332803) \times 0.024$	$(0.247792) \times 0.024$
T_7	$(0.76308) \times 0.01$	$(0.717596) \times 0.01$	$(0.395579) \times 0.016$	$(0.1558) \times 0.016$	$(0.336565) \times 0.016$	$(0.409334) \times 0.016$	$(0.396201) \times 0.016$
Mark	7.536585	7.431271	5.103087	2.404966	4.342478	5.073895	5.073895
Performance ($100 \times \sum$ Mark): 38.079177.							

Table 28: Memory-EP algorithm performance and mark scores [36].

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.85365) \times 0.015$	$(0.85275) \times 0.015$	$(0.62892) \times 0.024$	$(0.2391) \times 0.024$	$(0.6198) \times 0.024$	$(0.55021) \times 0.024$	$(0.46945) \times 0.024$
T_2	$(0.78253) \times 0.015$	$(0.74912) \times 0.015$	$(0.61966) \times 0.024$	$(0.24515) \times 0.024$	$(0.59625) \times 0.024$	$(0.52358) \times 0.024$	$(0.43583) \times 0.024$
T_3	$(0.76886) \times 0.015$	$(0.70595) \times 0.015$	$(0.66252) \times 0.024$	$(0.27313) \times 0.024$	$(0.62127) \times 0.024$	$(0.56062) \times 0.024$	$(0.46881) \times 0.024$
T_4	$(0.92328) \times 0.015$	$(0.92981) \times 0.015$	$(0.74413) \times 0.024$	$(0.33056) \times 0.024$	$(0.72009) \times 0.024$	$(0.65535) \times 0.024$	$(0.5996) \times 0.024$
T_5	$(0.82083) \times 0.015$	$(0.89972) \times 0.015$	$(0.65367) \times 0.024$	$(0.26922) \times 0.024$	$(0.65597) \times 0.024$	$(0.52499) \times 0.024$	$(0.37705) \times 0.024$
T_6	$(0.83403) \times 0.015$	$(0.9015) \times 0.015$	$(0.69926) \times 0.024$	$(0.26169) \times 0.024$	$(0.6832) \times 0.024$	$(0.56829) \times 0.024$	$(0.40828) \times 0.024$
T_7	$(0.72982) \times 0.01$	$(0.75187) \times 0.01$	$(0.63527) \times 0.016$	$(0.30215) \times 0.016$	$(0.59808) \times 0.016$	$(0.5208) \times 0.016$	$(0.43222) \times 0.016$
Mark	8.204590	8.310145	10.636016	4.368680	10.308720	8.952576	7.313200
Performance ($100 \times \sum$ Mark): 58.093927.							

Table 29: CPSO algorithm performance and mark scores [38].

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.942163) \times 0.015$	$(0.940825) \times 0.015$	$(0.727937) \times 0.024$	$(0.263052) \times 0.024$	$(0.687955) \times 0.024$	$(0.664818) \times 0.024$	$(0.556384) \times 0.024$
T_2	$(0.892462) \times 0.015$	$(0.887899) \times 0.015$	$(0.574953) \times 0.024$	$(0.0183243) \times 0.024$	$(0.470139) \times 0.024$	$(0.611798) \times 0.024$	$(0.439927) \times 0.024$
T_3	$(0.868731) \times 0.015$	$(0.837547) \times 0.015$	$(0.580325) \times 0.024$	$(0.0375368) \times 0.024$	$(0.489701) \times 0.024$	$(0.602617) \times 0.024$	$(0.431848) \times 0.024$
T_4	$(0.976683) \times 0.015$	$(0.975418) \times 0.015$	$(0.899955) \times 0.024$	$(0.276901) \times 0.024$	$(0.883281) \times 0.024$	$(0.874479) \times 0.024$	$(0.630821) \times 0.024$
T_5	$(0.889075) \times 0.015$	$(0.917639) \times 0.015$	$(0.568815) \times 0.024$	$(0.0271835) \times 0.024$	$(0.462767) \times 0.024$	$(0.608712) \times 0.024$	$(0.414599) \times 0.024$
T_6	$(0.881828) \times 0.015$	$(0.873017) \times 0.015$	$(0.643585) \times 0.024$	$(0.0345534) \times 0.024$	$(0.568689) \times 0.024$	$(0.538666) \times 0.024$	$(0.358982) \times 0.024$
T_7	$(0.85684) \times 0.01$	$(0.829573) \times 0.01$	$(0.65473) \times 0.016$	$(0.07386) \times 0.016$	$(0.572144) \times 0.016$	$(0.588575) \times 0.016$	$(0.41351) \times 0.016$
Mark	9.033253	8.978091	10.636936	1.696298	9.465507	10.304336	7.459762
Performance ($100 \times \sum$ Mark): 57.574184.							

Table 30: DASA performance and mark scores [37].

Change type	$F_1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
T_1	$(0.980667) \times 0.015$	$(0.970000) \times 0.015$	$(0.777083) \times 0.024$	$(0.588750) \times 0.024$	$(0.732917) \times 0.024$	$(0.842083) \times 0.024$	$(0.615833) \times 0.024$
T_2	$(0.904667) \times 0.015$	$(0.892667) \times 0.015$	$(0.602500) \times 0.024$	$(0.030000) \times 0.024$	$(0.513750) \times 0.024$	$(0.838333) \times 0.024$	$(0.480833) \times 0.024$
T_3	$(0.853333) \times 0.015$	$(0.827333) \times 0.015$	$(0.659583) \times 0.024$	$(0.072500) \times 0.024$	$(0.552917) \times 0.024$	$(0.845833) \times 0.024$	$(0.556250) \times 0.024$
T_4	$(0.944000) \times 0.015$	$(0.948667) \times 0.015$	$(0.7875) \times 0.024$	$(0.309167) \times 0.024$	$(0.745000) \times 0.024$	$(0.853750) \times 0.024$	$(0.557083) \times 0.024$
T_5	$(0.930667) \times 0.015$	$(0.958667) \times 0.015$	$(0.591667) \times 0.024$	$(0.092917) \times 0.024$	$(0.454583) \times 0.024$	$(0.841250) \times 0.024$	$(0.569583) \times 0.024$
T_6	$(0.903333) \times 0.015$	$(0.897333) \times 0.015$	$(0.760833) \times 0.024$	$(0.189583) \times 0.024$	$(0.707917) \times 0.024$	$(0.843333) \times 0.024$	$(0.549167) \times 0.024$
T_7	$(0.885000) \times 0.01$	$(0.832000) \times 0.01$	$(0.759375) \times 0.016$	$(0.176250) \times 0.016$	$(0.628125) \times 0.016$	$(0.841250) \times 0.016$	$(0.606250) \times 0.016$
Mark	9.160001	9.074001	11.244998	3.361001	9.902002	13.500997	8.958998
Performance ($100 \times \sum$ Mark): 65.201996.							

of another dopt-aiNet, compared with the PLBA might be due to the modifications that have been done to the original dopt-aiNet in order to reduce the computation complexity of the algorithm. The modifications included the removal of some mutation operators and the limitation of the search for the step size of the Gaussian mutation to a specific range and to a low number of function evaluations. In addition, the memory population was not used and the number of clones was reduced to one clone per cell. These modifications might contribute to reducing the computation cost, but on the other hand they might reduce the diversity of the algorithm. The nature of the algorithm and its procedure can be suitable for dynamic optimization such as DASA that has dynamic and adaptive pheromone mechanism that can act as short-term and long-term memories, as described in Section 2.4. Generally, the contribution of a specific strategy to the diversity maintenance depends on the way of its implementation and affected by other schemes integrated with it.

5. Conclusion

This paper introduced a recently proposed version of Bees Algorithm (PLBA) for solving DOPs and investigated its performance on a set of dynamic multimodal benchmark problems with environmental changes of different degrees of difficulties. In addition, the performance of PLBA was compared with other BA versions, and with other state-of-the-art algorithms found in the literature. The results have shown that PLBA could track the optimal solution and give reasonable solutions most of the time. The comparisons among BA-based algorithms showed that PLBA outperformed Shrinking-based BA and Basic BA. However, although the PLBA was better than Shrinking-based BA on the dynamic problems, the results were generally comparable. Therefore, it could be concluded that the dynamic change of the step size is a desirable characteristic of an algorithm to track the changed optimal solution. The experiments have also indicated that PLBA performed much better than the standard PSO, rPSO, rGA, and dopt-aiNet algorithms on dynamic optimization and was competitive with others on some results.

There as an interesting note that, apart from the reevaluation of the last solutions of the previous change and the reinitialization of the shrinking parameter, no modifications were needed for PLBA to deal with dynamic environments and achieve promising results. The patch environment, Levy flights, and keeping a large portion of the population scouting were good capabilities to deal with the tested DOPs. The inclusion of other features that are desirable for dealing with dynamic environments may enhance the performance of PLBA much more on the dynamic optimization.

Appendix

See Tables 22–30.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, for providing facilities and financial support under Fundamental Research Grant Schemes nos. GP-K007984, PRGS/1/2016/ICT02/UKM/02/1 entitled "Intelligent Vehicle Identity Recognition for Surveillance," DIP-2015-023 entitled "Object Descriptor via Optimized Unsupervised Learning Approaches," and FRGS/1/2016/ICT02/UKM/02/10 entitled "Commute-Time Convolution Kernels for Graph Clustering to Represent Images."

References

[1] F. O. De França and F. J. Von Zuben, "A dynamic artificial immune algorithm applied to challenging benchmarking problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 423–430, Trondheim, Norway, May 2009.

[2] C. J. Eyckelhof and M. Snoek, "Ant systems for a dynamic TSP," in *Ant Algorithms*, M. Dorigo, G. D. Caro, and M. Sampels, Eds., pp. 88–99, Springer, Berlin, Germany, 2002.

[3] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence—Introduction and Applications*, C. Blum and D. Merkle, Eds., pp. 193–217, Springer, Berlin, Germany, 2008.

[4] N. Mori and H. Kita, "Genetic algorithms for adaptation to dynamic environments-a survey," in *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society (IECON '00)*, Nagoya, Japan, 2000.

[5] R. K. Ursem, "Multinational GAs: multimodal optimization techniques in dynamic environments," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00)*, pp. 19–26, Las Vegas, Nev, USA, July 2000.

[6] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: a genetic algorithm approach," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 2031–2038, IEEE, July 1999.

[7] S. Yang, "Memory-based immigrants for genetic algorithms in dynamic environments," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 1115–1122, ACM, June 2005.

[8] A. Simões and E. Costa, "An immune system-based genetic algorithm to deal with dynamic environments: diversity and memory," in *Artificial Neural Nets and Genetic Algorithms*, pp. 168–174, Springer, Vienna, Austria, 2003.

[9] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC '05)*, vol. 3, pp. 2808–2815, September 2005.

[10] J. Brest, P. Korošec, J. Šilc, A. Zamuda, B. Bošković, and M. S. Maučec, "Differential evolution and differential ant-stigmergy on dynamic optimisation problems," *International Journal of Systems Science*, vol. 44, no. 4, pp. 663–679, 2013.

[11] R. Mukherjee, S. Debchoudhury, and S. Das, "Modified differential evolution with locality induced genetic operators for dynamic optimization," *European Journal of Operational Research*, vol. 253, no. 2, pp. 337–355, 2016.

[12] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the Congress on Evolutionary Computation*, pp. 94–100, Seoul, Korea, May 2001.

[13] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1666–1670, IEEE, Honolulu, Hawaii, USA, May 2002.

[14] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, Portland, Ore, USA, June 2004.

[15] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, pp. 489–500, Springer, Berlin, Germany, 2004.

[16] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[17] H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, M. Giacobini, Ed., pp. 637–646, Springer, Berlin, Germany, 2007.

[18] J. K. Kordestani, A. Rezvanian, and M. R. Meybodi, "An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 28, no. 1-2, pp. 137–149, 2016.

[19] W. Tfaili, J. Dréo, and P. Siarry, "Fitting of an ant colony approach to dynamic optimization through a new set of test functions," *International Journal of Computational Intelligence Research*, vol. 3, no. 3, pp. 205–218, 2007.

[20] S. Raziuddin, S. A. Sattar, R. Lakshmi, and M. Parvez, "Differential artificial bee colony for dynamic environment," in *Proceedings of the International Conference on Computer Science and Information Technology*, pp. 59–69, June 2011.

[21] S. Jiang, "Dynamic function optimization by improved artificial bee colony algorithm," *Applied Mechanics and Materials*, vol. 556–562, pp. 3562–3566, 2014.

[22] M. Kojima, H. Nakano, and A. Miyauchi, "An Artificial Bee Colony algorithm for solving dynamic optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 2398–2405, June 2013.

[23] H. Nakano, M. Kojima, and A. Miyauchi, "An artificial bee colony algorithm with a memory scheme for dynamic optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '15)*, pp. 2657–2663, IEEE, May 2015.

[24] S. K. Nseef, S. Abdullah, A. Turky, and G. Kendall, "An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems," *Knowledge-Based Systems*, vol. 104, pp. 14–23, 2016.

[25] L. N. De Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 699–704, Honolulu, Hawaii, USA, May 2002.

[26] F. O. De França, F. J. Von Zuben, and L. N. De Castro, "An artificial immune network for multimodal function optimization on dynamic environments," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 289–296, ACM, New York, NY, USA, June 2005.

[27] A. M. Turky and S. Abdullah, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Information Sciences*, vol. 272, pp. 84–95, 2014.

[28] N. Fouladgar and S. Lotfi, "A novel approach for optimization in dynamic environments based on modified cuckoo search algorithm," *Soft Computing*, vol. 20, no. 7, pp. 2889–2903, 2016.

[29] D. Yazdani, A. Sepas-Moghaddam, A. Dehban, and N. Horta, "A novel approach for optimization in dynamic environments based on modified artificial fish swarm algorithm," *International Journal of Computational Intelligence and Applications*, vol. 15, no. 2, Article ID 1650010, 2016.

[30] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[31] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, p. 1882, IEEE, Washington, DC, USA, July 1999.

[32] R. W. Morrison and K. A. De Jong, "A test problem generator for non-stationary environments," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 2047–2053, IEEE, Washington, DC, USA, July 1999.

[33] C. Li and S. Yang, "A generalized approach to construct benchmark problems for dynamic optimization," in *Simulated Evolution and Learning*, pp. 391–400, Springer, Berlin, Germany, 2008.

[34] C. Li, S. Yang, T. Nguyen et al., *Benchmark Generator for CEC 2009 Competition on Dynamic Optimization*, University of Leicester and University of Birmingham, Birmingham, UK, 2008.

[35] C. Li, S. Yang, and D. A. Pelta, *Benchmark Generator for the IEEE WCCI-2012 Competition on Evolutionary Computation for Dynamic Optimization Problems*, China University of Geosciences, Brunel University, University of Granada, 2011.

[36] E. L. Yu and P. N. Suganthan, "Evolutionary programming with ensemble of explicit memories for dynamic optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 431–438, Trondheim, Norway, May 2009.

[37] P. Korošec and J. Šilc, "The Differential ant-stigmergy algorithm applied to dynamic optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 407–414, Trondheim, Norway, May 2009.

[38] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 439–446, Trondheim, Norway, May 2009.

[39] J. Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic Publishers, Norwell, Mass, USA, 2002.

[40] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing*, vol. 15, no. 7, pp. 1427–1448, 2011.

[41] 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.

[42] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm-a novel tool for complex optimisation problems," in *Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS '06)*, Cardiff, UK, 2006.

[43] M. Castellani, Q. T. Pham, and D. T. Pham, "Dynamic optimisation by a modified bees algorithm," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 226, no. 7, pp. 956–971, 2012.

[44] D. Pham, Q. Pham, A. Ghanbarzadeh, and M. Castellani, "Dynamic optimisation of chemical engineering processes using the bees algorithm," in *Proceedings of the 17th International Federation of Automatic Control (IFAC) World Congress*, Seoul, Korea, July 2008.

[45] W. A. Hussein, S. Sahran, and S. N. H. S. Abdullah, "An improved Bees Algorithm for real parameter optimization," *International Journal of Advanced Computer Science & Applications*, vol. 6, pp. 23–39, 2015.

[46] W. A. Hussein, S. Sahran, and S. N. H. S. Abdullah, "A fast scheme for multilevel thresholding based on a modified bees algorithm," *Knowledge-Based Systems*, vol. 101, pp. 114–134, 2016.

[47] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach," in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference*, pp. 1201–1208, ACM, July 2006.

[48] A. Simões and E. Costa, "Improving prediction in evolutionary algorithms for dynamic environments," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference (GECCO '09)*, pp. 875–882, ACM, July 2009.

[49] J. Branke and D. C. Mattfeld, "Anticipation and flexibility in dynamic scheduling," *International Journal of Production Research*, vol. 43, no. 15, pp. 3103–3129, 2005.

[50] J. Eggermont and T. Lenaerts, "Dynamic optimization using evolutionary algorithms with a case-based memory," in *Proceedings of the 14th Belgium Netherlands Artificial Intelligence Conference (BNAIC '02)*, H. Blockeel and M. Denecker, Eds., 2000.

[51] J. Branke, *Evolutionary Optimization in Dynamic Environments*, vol. 3, Springer, 2012.

[52] S. Yang, "On the design of diploid genetic algorithms for problem optimization in dynamic environments," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1362–1369, IEEE, July 2006.

[53] M. Daneshyari and G. G. Yen, "Dynamic optimization using cultural based PSO," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11)*, pp. 509–516, IEEE, June 2011.

[54] J. Branke, T. Kaussler, C. Schmidt, and H. Schmeck, "A multipopulation approach to dynamic optimization problems," in *Adaptive Computing in Design and Manufacturing*, pp. 299–308, Springer, 2000.

[55] C. Li, S. Yang, and M. Yang, "An adaptivemulti-swarm optimizer for dynamic optimization problems," *Evolutionary Computation*, vol. 22, no. 4, pp. 559–594, 2014.

[56] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 590–605, 2016.

[57] T. Blackwell, "Particle swarm optimization in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 51 of *Studies in Computational Intelligence*, pp. 29–49, Springer, Berlin, Germany, 2007.

[58] M. Z. Ali, N. H. Awad, P. N. Suganthan, and R. G. Reynolds, "An adaptive multipopulation differential evolution with dynamic population reduction," *IEEE Transactions on Cybernetics*, 2016.

[59] W. A. Hussein, S. Sahran, and S. N. H. S. Abdullah, "The variants of the Bees Algorithm (BA): a survey," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 67–121, 2017.

[60] M. S. Packianather, M. Landy, and D. T. Pham, "Enhancing the speed of the bees algorithm using pheromone-based recruitment," in *Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN '09)*, pp. 789–794, Wales, UK, June 2009.

[61] W. A. Hussein, S. Sahran, and S. N. H. S. Abdullah, "Patch-Levy-based initialization algorithm for Bees Algorithm," *Applied Soft Computing Journal*, vol. 23, pp. 104–121, 2014.

[62] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.