An Improved Bees Algorithm for Real Parameter Optimization

Wasim A. Hussein[a,*], Shahnorbanun Sahran[b], Siti Norul Huda Sheikh Abdullah[c]
Pattern Recognition Research Group,
Center of Artificial Intelligence Technology,
Faculty of Information Systems and Technology,
Universiti Kebangsaan Malaysia,
43650 Bandar Baru Bangi, Malaysia

***Abstract*—The Bees Algorithm (BA) is a bee swarm-based search algorithm inspired by the foraging behavior of a swarm of honeybees. BA can be divided into four parts: the parameter tuning part, the initialization part, the local search part, and the global search part. Recently, BA based on Patch-Levy-based Initialization Algorithm (PLIA-BA) has been proposed. However, the initial stage remains an initial step, and its improvement is not enough for more challenging problem classes with different properties. The local and global search capabilities are also required to be enhanced to improve the quality of final solution and the convergence speed of PLIA-BA on such problems. Consequently, in this paper, a new local search algorithm has been adopted based on the Levy looping flights. Moreover, the mechanism of the global search has been enhanced to be closer to nature and based on the patch-Levy model adopted in the initialization algorithm (PLIA). The improvements in local and global search parts are incorporated into PLIA-BA to advise a new version of BA that is called Patch-Levy-based Bees Algorithm (PLBA). We investigate the performance of the proposed PLBA on a set of challenging benchmark functions. The results of the experiments indicate that PLBA significantly outperforms the other BA variants, including PLIA-BA and can produce comparable results with other state-of-the-art algorithms.**

Keywords*—Bees algorithm; Population initialization; Local search; Global search; Levy flight; Patch environment

I. Introduction

Most population-based metaheuristic algorithms, especially in recent years, are inspired by the collective intelligent behaviors of swarms of animals and insects such as fish, birds, bacteria, ants, termites, wasps, and fireflies. The biological studies showed that a swarm of such animals has impressive abilities to achieve fascinating, complex collective behaviors despite the simple behavior of each [1, 2]. It was found that the explanation of this amazing observation is the feature of self-organization that social animals have [2]. Self-organization can be considered as an organization without organizer in which no guidance from external or internal controller is needed [1]. Instead, decentralized control mechanisms are required for these social beings to update their activities by themselves based on some limited and local information [1]. These intelligent collective behaviors and the incredible capabilities of social animals to solve their daily life problems fascinated researchers to model their behaviors to solve real-world optimization problems. Then the model can be used as a base to develop artificial versions, either by tuning the model parameters using values outside the biological range or by assuming additional non-biological characteristics in the model design [2]. As a result, swarm intelligence in nature has been transferred from biological systems to artificial systems. Thus a new field called Swarm Intelligence (SI) was emerged under the field of Artificial Intelligence (AI), particularly under the Computational Intelligence (CI) field. Therefore, algorithms such as Ant Colony Optimization (ACO) [3], Particle Swarm Optimization (PSO) [4], Bacterial Foraging Optimization (BFO) [5], and the Firefly Algorithm (FA) [6] have been developed.

Among the social living beings that present interesting behavior and features are honeybees. The honeybees are very interesting creatures that exhibit several surprising intelligent behaviors when they behave as swarms of honeybees. Over the past decade, the collective intelligent behaviors of swarms of bees have been attracting the attention of researchers seeking to develop intelligent search algorithms. Examples of algorithms inspired by the behavior of bees include the Honey Bee Mating Optimization (HBMO) [7], the Artificial Bee Colony (ABC) algorithm [8], and Bee Colony Optimization (BCO) [9] algorithms.

One of the most recent bee-based algorithms is the Bees Algorithm (BA). BA is a population-based search algorithm proposed by Pham et al. [10] and inspired by the foraging behavior of swarms of honeybees searching for good food sources. Fundamentally, the algorithm performs a kind of exploitative local or neighborhood search combined with an exploratory global search. Both kinds of search modes implement uniform random search. In the global search, the scout bees are distributed uniformly at random to different areas of the search space to scout for potential solutions. In the local or neighborhood search, follower bees are recruited for patches found by scout bees to be more promising to exploit these patches. BA has been successfully applied to problems in many fields, such as function (continuous) optimization [10, 11], training neural networks [12], the job shop scheduling problem [11], and solving timetabling problems [13].

As a result of this, and also its simplicity and closeness to the actual behavior in nature, BA has garnered a significant amount of interest from researchers since its invention. We can divide BA into four parts or components: the parameter tuning or parameter setting part, the initialization part, the local search (exploitation) part, and the global search (exploration) part. Several studies have sought to improve BA and to enhance its performance by improving some of these parts. Some of these studies focused on the parameter tuning and setting part. The resulting improvements in these studies were gained by reducing the number of tunable parameters [12] or by developing tuning methods to tune the parameters of BA [14, 15]. Other studies focused on developing other concepts and strategies for the local (neighborhood) search part [16-18], or for both the local and global search parts [15, 19, 20].

However, limited attention has been paid to the improvement of the initialization part. In the initialization component, the foragers or searchers fly at random to initial resources. The initial location of foragers relative to the optimal resource (target) may affect the degree of optimality of other algorithm components. Therefore, recently, the initialization part of Basic BA has been paid attention and enhanced to improve the final solution quality and the convergence speed. Consequently, an initialization algorithm called the Patch-Levy-based Initialization Algorithm (PLIA) has been proposed and incorporated into Basic BA to adopt a BA version denoted by PLIA-BA [21]. However, the initial stage remains an initial step, and its improvement is not enough for more challenging problem classes with different properties. The local and global search capabilities are also required to be enhanced to improve the exploitativeness and exploration abilities of the algorithm, respectively. Thus, the quality of final solution and the convergence speed of PLIA-BA is improved on such problems. Hence, in this paper, the local and global search parts of PLIA-BA have been improved.

Most of the improvements achieved on BA were not inspired by natural bee behaviors. However, the imitation of the best characteristics in nature can lead to efficient metaheuristic algorithms [22]. Therefore, it is good to search for additional natural bee aspects that can be modeled in BA and improve its performance. There are numerous biological features in nature associated with honeybee foragers and food sources that can be beneficial if they are properly modeled and incorporated into Basic BA. Among these features that we can model are the distribution of food sources and the distribution of honeybees when they fly away from the hive foraging for food. In nature, flowers are usually distributed in patches that regenerate and are rarely completely depleted [23]. In addition, a scout honeybee flies away from the hive and moves randomly throughout the space according to Levy flight motion [24-26], which has been found to constitute the optimal search strategy [23, 24, 27]. During the harvesting season, a portion of the colony population is kept as scout bees foraging for new food sources on the global scale [10, 28]. Furthermore, in nature, it has been found that Levy looping search triggers the flight paths that is performed by honeybee foragers conducting a local search around a known food source [25]. Consequently, in this paper, we enhance PLIA-BA, and propose an improved version of BA called Patch-Levy-based Bees Algorithm (PLBA). PLBA utilizes the PLIA for initialization stage [21], a new local search algorithm that models Levy looping flights, and an enhanced global search that is improved based on the patch-Levy model adopted in PLIA. The proposed local search algorithm is called Greedy-Levy-based Local Search Algorithm (GLLSA).

Levy flights were first proposed as models of random walks in optimization algorithms by Gutowski [29], who advocated the use of Levy distributions instead of uniform and Gaussian distributions as mechanisms to generate the size of steps. The justification for this was that the frequent short steps generated by Levy distributions enable the optimization algorithm to intensify the search in regions around the current promising points. In addition, the occasional long jumps produced by Levy distributions help the optimization algorithm to escape from local optima. Levy flights were subsequently utilized as a search mechanism in many optimization algorithms, such as Levy Flights Optimization (LFO) algorithms [30], Cuckoo Search [31], the FA [32], the Bat algorithm [33], the Krill Herd (KH) algorithm [34], the ABC algorithm [35], and PLIA-BA [21]. However, to the best of the author's knowledge, this is the first time Levy looping flights are being used to conduct the local search as in nature instead of freely roaming Levy flights.

The remainder of this paper is organized as follows. Section II reviews the Bees Algorithm based on Patch-Levy-based Initialization Algorithm (PLIA-BA). Section III describes the Levy flight and the proposed algorithm. Section IV presents the results of performance evaluations and experiments obtained for the proposed BA variant (PLBA) and compares these results with those obtained using other BA variants, including PLIA-BA and other state-of-the-art algorithms. Finally, section V concludes this paper.

II. The Bees Algorithm Based on Patch-Levy-Based Initialization Algorithm (PLIA-BA)

The PLIA-BA is an improved version over Basic BA incorporating the Patch-Levy-based initialization algorithm (PLIA) proposed by Hussein et al. [21] to enhance the initialization stage of Basic BA. The PLIA can be summarized as follows:

1) Divide the search space equally into P patches or segments.

2) Evaluate the vector of areas in the hive from which scout bees will fly ($c_1, c_2, \dots, c_P$), represented by the centers of the patches (segments).

3) Divide and assign the (n) scout bees into the hive areas and evaluate the number of scout bees (nb) to be assigned in each area in the hive using the following equation:

$$nb = Int\left(\frac{n}{P}\right) \tag{1}$$

4) Evaluate the number of remaining scout bees still not assigned (nrb) to be assigned in the last area in the hive using the following equation:

$$nrb = n\%P \tag{2}$$

5) Set j = 1

Set Current area = c_j

While (Current area (c_j) is not the last area (c_P))

Distribute (*nb*) bees from the current area (c_j) inside the hive to the patches according to Levy flight distribution to constitute (*nb*) bees in the initial population

Set $j = j + 1$

Set Current area = c_j

End while

6) Distribute (nb+ nrb) bees from the last hive area to the patches according to Levy flight distribution to constitute (nb + nrb) bees in the initial population.

7) Return the constructed initial population of (n) scout bees.

III. Proposed Algorithm

In this paper, a new local search algorithm (GLLSA) that models the Levy lopping flights is developed and the global search is improved to be based on the patch-Levy model adopted in the initialization algorithm, PLIA. The initialization algorithm (PLIA) [21], the proposed local search (GLLSA), and the enhanced global search are incorporated into an enhanced version of BA called PLBA. In this section, Levy flights and the importance of the local and global search components are described and the proposed improvements in these components are presented. Then, the development of PLBA is outlined on the basis of the proposed improvements in local search and global search parts.

A. Levy flight

The existence of Levy flights as a movement pattern in biological organisms was first noted by Shlesinger and Klafter, who stated that the characteristics of Levy flights can be observed in foraging ants [36]. Levandowsky et al. subsequently introduced Levy walks as a swimming behavior in microorganisms [36]. Various empirical and theoretical studies have subsequently identified the Levy flight patterns and characteristics in the foraging of various animals and species such as the wandering albatross, reindeer, jackals, dinoflagellates, spider monkeys, sharks, bony fish, sea turtles, penguins, fruit flies (drosophila), bumblebees, and honeybees [36]. Evidence of using Levy flights as search patterns was also found in the foraging patterns of humans such as the Dobe Ju/’hoansi hunter-gatherers [37].

Levy flights are random walks that are named after Paul Levy, a French mathematician [29]. Levy flights consist of independent, randomly oriented steps with lengths *l*, drawn at random from an inverse power-law distribution with heavy and long tail, $p(l) \sim l^{-\mu}$ where $1 < \mu \le 3$. Levy flights are scale-free since they do not have any characteristic scale because of the divergent variance of $p(l)$, and they present the same fractal patterns regardless of the range over which they are viewed. 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.

Levy flights can be categorized into two categories: freely roaming Levy flights and Levy looping flights [25, 27]. The freely roaming flights consist of sequences of Levy steps, whereas the Levy looping flights comprise loops of the Levy steps. The freely roaming Levy flights constitute the optimal search strategy for the foragers searching for sparsely and randomly distributed patches. On the other hand, Levy looping search strategy is adopted by many foragers for exploiting a promising patch or area. In the freely roaming flights, the forager continues the search from the last location, whereas in the looping flights, the foragers restarts the search from the original promising location that represents the center of a potential patch until the target is found. This strategy aims at searching the local area of the potential patch more intensively. However, if no progress has been achieved with the time, the probability of finding the target in the vicinity of that location decreases, thus the original site is abandoned and freely roaming flights are adopted.

It can be easily observed from the definition of Levy flights that these flights constitute a series of displacements and orientations. Therefore, two steps are required to mimic Levy movements in nature and to implement these flights. The first step generates a random direction to mimic the random choice of direction by drawing it from a uniform random distribution. In the proposed algorithm, the direction is drawn from a uniform distribution between -1 and 1. The second step generates the step length that obeys a Levy distribution.

B. Local and Global Search Capabilities

The local search and global search components are two of the main components of the metaheuristic algorithms. These two components are equivalent to two important characteristics, which are the intensification and diversification characteristics, respectively [22]. In the intensification, the current promising areas are exploited and the best solution is selected. Whereas, in diversification, the search space is explored on the global scale. Therefore, the enhancement of the local and global search mechanisms of PLIA-BA can lead to significant improvement in the overall performance of PLIA-BA in terms of the solution quality, convergence speed, and success rate. In addition, as mentioned earlier, the imitation of the best characteristics in nature can lead to efficient metaheuristic algorithms [22]. Therefore, in this paper, the local search part of PLIA-BA is enhanced by proposing a new local search algorithm called Greedy Levy-based Local Search Algorithm (GLLSA) that models Levy looping search for the exploitation of patches. The global search part is also improved by employing the patch-Levy model adopted in the initialization stage.

1) Local Search

In this important stage, the areas (patches) of promising resources are exploited by recruiting other bees to these areas and conducting local search. Since there is a high chance to find the optimal solution near the good solutions, the exploitation step is considered an important step.

In Basic BA and PLIA-BA, the best sites (m) are selected for the local search. Then, the local search part is performed by first determining the size of the patch in which the search will be conducted and then the recruit bees are distributed uniformly at random to food sources inside the determined patch. However, in nature, it has been found that the honeybee foragers conducting local search around a known food source adopt Levy looping flights [25]. In the Levy looping search, the honeybee flies from the site whose local area is searched. If the target is found the foraging stops, otherwise the forager returns back to the same original site and randomly chooses a direction and length before foraging again. If no progress has been achieved with the time, the probability of finding the target in the vicinity of that location decreases, thus the original site is abandoned and freely roaming flights are adopted.

As a result of this natural behavior of bees conducting local search and since the imitation of the best characteristics in nature can lead to efficient algorithms, this paper proposes a new local search algorithm that very closely mimics the actual behavior of bees in nature. This local search algorithm is called the Greedy Levy-based Local Search Algorithm (GLLSA). In GLLSA, a recruit bee restarts the search from the current best site in a patch based on Levy flight distribution for a predetermined time until a better site is found. Once a better site is found, it becomes the current best site. The aim of this strategy is to search the area where the optimal solution is expected to be more intensively. In the proposed GLLSA algorithm, each recruit bee of the remaining recruit bees starts foraging from the last resource found to be the best source by previous recruit bees. This limitation is not found in nature but it is assumed to increase the property of being greedy of the BA and to increase the possibility of finding the optimal solution.

2) The Proposed Local Search Algorithm: Greedy Levy-based Local Search Algorithm (GLLSA)

Based on the concepts above, the Greedy Levy-based Local Search Algorithm (GLLSA) is proposed to conduct the local (neighborhood) search stage in the PLIA-BA. The pseudo-code of the algorithm is presented in Fig. 1.

Before conducting the local search, as in Basic BA and PLIA-BA, the fittest *m* sites of the fittest *m* bees are determined as the promising resources which require exploitation. Among these *m* sites, *e* sites need more exploitation by recruiting for them more bees than the remaining (*m* - *e*) sites. As in Basic BA and PLIA-BA, the number of recruited bees for the *e* sites is determined by the parameter *nep*. For the remaining (*m* - *e*) sites, the number of recruited bees is identified by the parameter *nsp*. After this selection process of sites for the local search, the local search algorithm starts.

As can be seen from the pseudo-code of the algorithm in Fig. 1, for each site (patch) of the *m* selected sites (patches), a number of bees are recruited (*RecruitBee*). *RecruitBee* can be either *nep* or *nsp* as mentioned before. Each site of the *m* sites represents the current best site in its patch. Every recruited bee achieves searching and foraging for better solutions or sites for some time (*t*) that can be found empirically by a trial and error and is usually simply set to *RecruitBee* or to one. During this time, the first recruit bee restarts searching from the same original best site as long as there is no better site has been found. If the recruit bee finds a better solution while it is searching for the predetermined time, this recruit bee stops searching, the site found is considered the current best site, and the next recruit bee starts the search. The foraging of the next recruit bee is continued by the same way from the last current best site. The same process is repeated for the remaining recruit bees. Finally, each original exploited site of the m sites is replaced with the last best solution found by the bees recruited for that site.

Each recruit bee conducts searching in a patch according to the following equation illustrated in the schematic diagram in Fig. 2:

$$b_i = s_{bestcur} + (2r-1) \times Levy(\gamma_2), \quad i = 1, 2, \ldots, nep \, (\text{or } nsp) \tag{3}$$

where b_i is the position of the i^{th} recruit bee, $s_{bestcur}$ is the current best site in the patch, $(2r-1)$ gives the direction of fly drawn from a uniform random distribution between -1 and 1, i.e. $(2r-1) \in \text{uniform}(-1,1)$, $r \in \text{uniform}(0,1)$ and $Levy(\gamma_2)$ represents the step length generated randomly from a Levy flight distribution with a search size or scale parameter γ_2.

This search size or scale is shrunk at each iteration of the algorithm after the local search, thus the step size of the bee generated from Levy distribution is decreased from time to time while foraging inside the neighborhood of the potential solution. The aim is to decrease the length of the long steps [30], thus, increasing the intensification and exploitativeness capability of the proposed PLBA and searching the region around the promising solution comprehensively. Additionally, the decrease of the long jumps can prevent the recruit bees from going beyond the regions of promising sites that may result from the dependency behavior of GLLSA where the foraging of each recruit bee depends on the foraging results of the previous one.

C. Global Search

In nature, during the harvesting season, the bee colony keeps a portion of the bee population as scout bees foraging for new food sources [10, 28]. These scout bees fly away from the hive and move throughout a patchily distributed environment at random according to Levy flight pattern [23, 26]. In the Basic BA, PLIA-BA and the proposed PLBA, this portion of scout bees is represented by *n* - *m* scout bees, thus a number (*n* - *m*) of scout bees is distributed to perform the global search.

In Basic BA and PLIA-BA, the global search is conducted by distributing this proportion of scout bees uniformly at random into the search space in the same way as in the initialization stage of Basic BA. On the other hand, the global search in the proposed PLBA is enhanced by imitating the natural behavior, which was the Levy motion in a patchily distributed environment. In the global search of the proposed PLBA, 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, these bees start to scout according to the Levy flight with search size or scale (γ_3) as in the initial step.

D. The Proposed Patch-Levy-based Bees Algorithm (PLBA)

Having presented the initialization algorithm (PLIA) in [21], the proposed local search algorithm (GLLSA) in Section 2), and the enhancement on global search in Section C, the main steps of the proposed Patch-Levy-based Bees Algorithm (PLBA) can be summarized as follows:

1. Initialize the population using the initialization algorithm (PLIA) [21].
2. Evaluate the fitness of the population.
3. While (stopping criterion not met)
   // Form new population
4. Select (m) sites for neighborhood search.
5. Conduct Local (neighborhood search) according to the proposed local search algorithm (GLLSA).
6. Redistribute the remaining bees (n - m) from their previous areas inside the hive to the patches according to Levy flight distribution to scout again and evaluate their fitness.
7. End while.

In the proposed PLBA, it is assumed that the number of patches in the initialization and global search parts are the same and represented by the same centers. However, in the local search, each selected site out of the m sites represents a patch center as in Basic BA and PLIA-BA.

IV. Experimental Setup and Results

A. Experimental Setup

1) Benchmark Functions

Most of the standard benchmark functions have some properties that could be exploited by some general-purpose metaheuristic algorithms to achieve good results [38]. Among these properties are the symmetric dimensions of the global optimum (i.e., the dimensions have the same values), and the location of the global optimum at origin, at the center of the search range, or on the bounds. For instance, if the optimization problem has symmetric dimensions, an optimization algorithm with a neighborhood operator that copies the value of one dimension to other dimensions can converge quickly to the global optimum. Such properties are considered problems in the standard test functions that should be avoided in order to test a novel global optimization algorithm [38].

As a result, to evaluate the proposed PLBA and compare it with other BA variants and with other state-of-the-art population-based algorithms, we employ a set of 25 challenging scalable benchmark functions with different characteristics and complexities provided for CEC'2005 session on real-parameter optimization [39]. These benchmark functions include functions with different properties such as unimodal, multimodal, shifted, rotated and unrotated, global minimum on the bounds and global minimum not on the bounds, with and without noise, separable and non-separable, hybrid composition functions, and so on. Experiments were carried out on the 10 and 30-dimensional versions of these challenging problems. Detailed information about these function optimization problems can be found in [39].

2) Performance Evaluation

In the evaluation of the algorithms on a set of test problems, especially the multimodal problems, some algorithms may have a small probability of success on a test function but converge fast. On the other hand, other algorithms may have a larger probability of success but converge slower [40]. Hence, it is good to evaluate the performance of an algorithm in terms of both convergence speed and success rate. One way to do this is to use the Success Performance (SP) measure [40]. SP is the expected number of function evaluations to achieve a certain success level (accuracy level) on a specific function. Lower value of SP for an algorithm on a problem means the algorithm is faster in solving that problem. SP can be defined as follows [39, 40]:

$$\mathrm{SP} = \frac{mean\left(\mathrm{FES}_{successful}\right)}{p_{success}}, \qquad (4)$$

where SP is the success performance of an algorithm on a specific function, $mean\left(\mathrm{FES}_{successful}\right)$ is the mean number of function evaluations for the successful runs, and $p_{success}$ is the probability of success of the algorithm on the function, and $p_{success} = \frac{RN_{successful}}{RN_{All}}$, where $RN_{successful}$ is the number of successful runs, and RN_{All} is the total number of runs performed. Each run of an algorithm on a function was deemed to be successful when the error value of that function is less than or equal to the acceptance (accuracy) level specified for that function.

From (4), it can be seen that $\mathrm{SP} = mean\left(\mathrm{FES}_{successful}\right)$ in the case of SR = 100% where $p_{success}$ = 1. Therefore, to compare the convergence speed of a set of algorithms with success rate (SR) of 100%, the mean number of evaluations can be used. However, when the success rates of the algorithms are of different values, the success rates should be taken into account in addition to the mean number of evaluations only in the successful runs in order to evaluate the convergence speed. Therefore, we use three performance metrics to evaluate the performance of the proposed PLBA algorithm; namely, the mean function error value, the success performance (SP), and the success rate (SR).

The mean function error value gives indication of the quality of the final solutions obtained by the proposed PLBA algorithm, whereas the success performance is employed to ascertain the convergence speed of this algorithm. The function error value is calculated as follows: $E = \left|f\left(x_{\mathrm{best}}\right) - f\left(x^{*}\right)\right|$, where x_{best} is the best solution and x^{*} is the global optimum. Success rate (SR) is one of the performance criteria used in the literature for evaluating the reliability of algorithms. It can be calculated as follows: $\mathrm{SR} = p_{success} \times 100$. In addition, convergence graphs for each function, in the case of d = 30, that plot the function error value against the number of function evaluations are used. These graphs show the median performance of the total runs. Further, the extra parameters relevant to the proposed PLBA are analyzed.

In this evaluation of the performance of PLBA, the 25 functions are employed using dimensions of d = 10, and d = 30 with maximum number of function evaluations (Max_FES) of 100,000, and 300,000 evaluations, respectively. The PLBA is run 25 times for each problem. Each run of the PLBA is terminated when the number of evaluations reaches the Max_FES or if the function error value is 10-8 or less. Each run of an algorithm is determined to be a successful when the error in the function value (*E*) is less than or equal to the accuracy level prescribed in [39] as follows: 1×10^{-6} for functions $f_1 - f_5$, 1×10^{-2} for functions $f_6 - f_{16}$, and 1×10^{-1} for functions $f_{17} - f_{25}$.

3) Parameter Settings

We compare the performance of PLBA with that of Basic BA, Shrinking-based BA, Standard BA, and PLIA-BA. Shrinking-based BA is an improved version of BA that utilizes a neighborhood shrinking procedure [41] and Standard BA is an improved variant that employs both shrinking and site abandonment procedures [42]. In addition, we conduct an additional set of experiments to compare the performance of PLBA with other population-based metaheuristic algorithms. These algorithms include Restart CMA-ES [43], DMS-PSO [44], SPC-PNX [45], DE [46], SaDE [47], ABC [48], and modified ABC (MABC) [48]. The Restart CMA-ES is the covariance matrix adaptation evolution strategy in which the population size is increased for each restart. DMS-PSO is a dynamic multi-swarm particle swarm optimization algorithm in which the swarms are regrouped frequently and the Quasi-Newton method is employed to improve the local search capability. SPC-PNX is a steady-state real parameter genetic algorithm with parent centric normal crossover. DE is the classical differential evolution algorithm. SaDE is a self-adaptive version of the differential evolution algorithm where the learning strategy and some control parameters are self-adapted by the previous learning experience. ABC is the classical artificial bee colony algorithm. Modified ABC (MABC) is a modified version of ABC in which two parameters were added. The first parameter was to determine the number of parameters to be mutated and perturbed and the second one was to specify the magnitude of this perturbation. The parameter settings of these algorithms can be found in their references.

In order to perform a fair comparison among the BAs, we execute the five versions of BA 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 (See Appendix A: Tables A. 1 and A. 2). These parameters are closely related to the problem under study. It should be noted that it is good on these benchmarks to set small values for the Levy search size of local search (γ_2) to support the exploitation capability of the good regions and large values for the Levy search size of the global search (γ_3) to maintain the diversity of the population.

B. Experimental Results

The mean function error values achieved by PLBA after Max_FES for the 25 10-dimensional and 30-dimensional test problems are presented in Tables I and III, respectively. The success rate (SR) and success performance (SP) achieved by PLBA for the 25 functions in the case of d = 10 and d = 30 are tabulated in Tables II and IV.

From the results in Tables I and II, it could be observed that in the case of 10-dimensional problems, PLBA could find the global optimum for problems f_1, f_2, f_4, f_5, f_9, f_{12}, and f_{15} with success rate 100%, 100%, 100%, 100%, 100%, 8%, and 100%, respectively. In the case of 30-dimensional version of problems, PLBA could find the global optimum for problems f_1, f_2, f_7, f_9, and f_{15}, with success rate 100%, 96%, 68%, 100%, and 52%, respectively, as can be seen in Tables III and IV.

However, for problems $f_{16} - f_{25}$, PLBA, as other optimization algorithms as can be seen in Section E, could not find the global optimum for both 10 and 30-dimensional versions in all 25 runs owing to the high multimodality of these problems, the randomly located global optima and the huge number of randomly located deep local optima [38, 47].

C. Analysis of the Results of the Proposed PLBA

Among the unimodal functions $f_1 - f_5$, PLBA was able to solve the shifted sphere function (f_1), the Schwefel problem 1.2 with (f_2) and without (f_4) noise in fitness, and the Schwefel problem 2.6 with global optimum on bounds (f_5) in the case of the 10-dimensional problems in all 25 runs. Only the shifted rotated high conditioned elliptic function (f_3) was the problem that PLBA failed to optimize within the Max_FES. In the case of the 30-dimensional version, PLBA was able to optimize f_1 over all runs, and to solve f_2 most the time with success rate 96%. However, PLBA failed to solve the 30-dimensional version of f_4. The first three unimodal functions were of different number of conditions that make them of different complexities where f_3 is more difficult than f_2, and f_2 is more difficult than f_1. From the results, it could be observed that the high number of conditions of f_3 deteriorate the performance of the PLBA. On the other hand, although the noise disturbs the search in the optimization process, the noise had no significant effect on the performance of PLBA in the case of 10-dimensional problems. This can be confirmed by comparing the results of f_{16} (without noise) and f_{17} (f_{16} with noise) where the results of these two functions were almost the same. The same case with the best performing algorithm (Restart CMA-ES) where the noise affected its performance in the 30 dimensions case, whereas the noise had no significant effect on its performance in the case of *d* = 10, as can be seen in [43].

For the basic functions in the multimodal problem class, PLBA could find the global optimal solution of f_9, which was the shifted Rastrigin function in all 25 runs for both the 10 and 30-dimensional versions. It could be concluded from these results and from the results of PLIA-BA on the standard Rastrigin (f_6) [21] that shifting the global optimum of the Rastrigin function does not affect the performance of PLBA, which is an improved version of PLIA-BA. However, rotation seemed to pose difficulties to PLBA as can be inferred from the rotated version (f_{10}) of the function f_9. Additionally, PLBA was able to locate the global optimum of the shifted rotated Griewank function, f_7 for 30-dimensional version with success rate 68%. On the other hand, PLBA failed to solve this problem in the 10-dimensional case. This can be explained by that the increase in the maximum allowable number of function evaluations by factor of 3 (3.00E+05) gives PLBA the time to solve this function most the time despite the increase in the number of dimensions. This can also be observed with other algorithms on f_7 and other problems such as DE [46] on f_7, SaDE [47] on f_4 and f_7, DMS-PSO [44] on f_7, SPC-PNX [45] on f_6 and f_7, Restart CMA-ES [43] on f_3 and f_{15}, Shrinking-based BA on f_1, and Standard BA on f_1, f_2, and f_6 where the results of these algorithms on the stated functions in the 30-dimensional versions are better than those in the 10-dimensional versions. Furthermore, PLBA could optimize f_{12} with success rate 8% for the 10-dimensional problem.

However, the two multimodal expanded functions f_{13}, and f_{14}, and the eleven multimodal composition functions form the biggest challenge for the population-based metaheuristic algorithms as can be seen from the results. Nevertheless, PLBA was able to successfully optimize the hybrid composition problem f_{15} in all runs within the Max_FES in the 10-dimensional case and half the time (success rate 52%) in the case of 30-dimensional version.

An interesting note that all algorithms failed to solve the shifted rotated Ackley function with the global optimum on bounds, even though most of the algorithms such as the improved BAs [21] succeeded to optimize the standard Ackley function with success rate of 100%, as can be seen with f_9 in the benchmarks used in [21]. The different scaling employed in this advanced version of Ackley can accounted for this bad performance of the algorithms on the function [43]. The standard Ackley function has a flat area outside the search space ($[-32,32]^d$) and employing the linear transformation with 100 condition numbers to construct the challenging version (f_8) caused this flat area to be inside the search space [43]. Therefore, the search for the global optimum of this function looks like looking for a needle in a haystack [43, 44].

It should be noted that the different characteristics of the problems have a great effect on the performance of an optimization algorithm. However, the performance of an algorithm is not affected by these characteristics only, but also by the parameter tuning process that has a significant effect [45]. Therefore, fine-tuning of the parameters of PLBA, especially the extra parameters, on a specific problem may result in a good performance of the algorithm on that problem.

D. Comparisons among the BAs

By comparing the results of BA-based algorithms in Tables I - IV, it can be clearly seen that the performance of PLBA is much better than the other BA variants. In the case of 10-dimensional problems, all other BA variants failed to reach the success level within the predetermined Max_FES on all the test problems. However, in the case of 30-dimensional problems, Standard BA and Shrinking-based BA were able to optimize f_1 with success rate 100%, and f_2 with success rate 96% and 64%, respectively.

The convergence graphs of all BA-based algorithms including the proposed PLBA on some tested problems for d = 30 were presented in Figs. 3 and 4 to compare PLBA against other BA variants in terms of the convergence speed. It can be seen from these figures, that PLBA converges faster than other BA versions on most of the tested problems. Even though the PLBA was stuck in local optima, it could escape from that local optima after a number of evaluations. This can be accounted for by that the patch concept helps in spreading the solutions along the search space and creating diversity, then the frequent short steps of Levy flights cause rapid convergence and the rarely occurred long jumps help to increase the diversity and escape from local optima.

The good effect of controlling the combination of patch number and Levy search size can also be observed by investigating the convergence graphs of PLIA-BA where PLIA-BA performed better than Basic BA on almost all functions and equally or better than Shrinking-based BA or Standard BA on some functions. From Figs. 3 and 4, it can also be concluded that dynamic change in the step size of the local search was advantageous for BA in the Standard BA and Shrinking-based BA. However, the fast decrease in the step size might lead to premature convergence on the problems other than f_1, and f_2. The dynamic change in the search size or the scale of the Levy flight in the local search (γ_2) was also beneficial for the PLBA in many cases in controlling the length of the long steps [30]. In this case, there are still frequent short steps and rare long jumps and only the length of these steps are changed and reduced, especially the long ones. Thus, the exploitativeness of the PLBA in the local search area is increased with the possibility of escaping from local optima without making aggressive long jumps that can lead the search outside the good area.

We conducted statistical comparisons between PLBA and other BA variants in terms of the solution quality using the Friedman test. We conducted two sets of comparisons using d = 10 and d = 30 dimensions to evaluate the results and to check the behavior of the BA-based algorithms. Tables V and VI show the ranks achieved by Friedman test for both sets of comparisons. It can be clearly seen in these tables that PLBA ranked first in both sets of comparisons. Therefore, PLBA was the best performing algorithm in the 10 and 30 dimensions cases, whereas the worst one was Standard BA, followed by Basic BA in the case of d = 10, and Basic BA in the case of 30 dimensions. The *p*-values calculated using the statistic from the Friedman test were 0 and 0.000071 in the 10 and 30-dimensinal cases, respectively, as shown in Tables V and VI. These *p*-values suggested a highly significant difference among the performance of the BA-based algorithms considered.

Subsequently, we used two post hoc tests (Holm and Hochberg tests) [49] to compare PLBA with the rest of the BA-based algorithms. Tables VII and VIII show the adjusted *p*-

values obtained by the post hoc tests considering PLBA as the control method in the 10 and 30-dimensional cases, respectively. In the case of *d* = 10, PLBA showed a significant improvement over the Basic BA, Standard BA, and PLIA-BA at a level of significance α = 0.01, whereas no significant difference was found between Shrinking-based BA and PLBA. On the other hand, in the 30-dimensional case, PLBA showed a significant improvement over all other BA-based algorithms with a level of significance α = 0.01.

Although the Shrinking-based BA did not succeed in solving any function out of the 25 functions in the case of 10 dimensions, its results were generally good and no significant difference was detected between it and PLBA. It can also be observed that PLIA-BA outperformed Basic BA in both 10 and 30 dimensions cases and it also outperformed Standard BA in the case of *d* = 10. However, in both cases (*d* = 10 and *d* = 30) PLBA outperformed PLIA-BA and PLIA-BA could not successfully solve any problem. Thus, it can be concluded that although the improvement of the initialization algorithm of Basic BA can enhance the performance of Basic BA, it is not enough to solve more challenging problem classes.

E. Comparisons with Other State-of-the-art Algorithms

First, it should be pointed out that no results regarding the SR, and SP were tabulated in Tables IX and X for conventional ABC and MABC because their results have not been reported in [48]. From these tables, it can be observed that, all algorithms employed in these comparisons, including PLBA could locate the global optimum of f_1, and f_2 over all 25 trials in the 10-dimensional versions. Whereas in the 30-dimensional version, all algorithms were able to find the global optimum of f_1 in all runs, and of f_7 with different success rates. The most successful algorithm on this function f_7 was Restart CMA-ES with 100% success rate, followed by DMS-PSO, DE, SaDE, PLBA, and SPC-PNX with success rate 96%, 88%, 80%, 68%, and 64%, respectively. In addition, all algorithms failed to find the global optimal solution of the problems f_8, f_{13}, f_{14}, and $f_{16} - f_{25}$ because of the high multimodality of these problems [47].

An interesting finding that only PLBA was able to successfully solve the hybrid composition problem f_{15} in all 25 runs in the 10-dimensional version, as can be seen in Table IX. On the other hand, other algorithms either could not solve the problem in some runs such as DE, SaDE, and DMS-PSO or did not succeed in any run such as SPC-PNX and Restart CMA-ES. In addition, only PLBA was the algorithm that could find the global optimum for the 30-dimensional version of f_{15} where it achieved a success rate of 52%, as can be seen in Table X.

In general, it could be concluded from the comparisons that the shifted sphere function, f_1 is the simplest function in the test suite and the shifted rotated Ackley function (f_8), expanded functions (f_{13}, and f_{14}), and the composition functions ($f_{15} - f_{25}$) pose the greatest difficulty for the population-based optimization algorithms. In addition, from the comparisons between the 10 and 30-dimensional versions, it could be inferred that increasing the Max_FES and adjusting some parameters can significantly improve the results for an optimization algorithm for specific problems.

Regarding the convergence speed on the functions with non-zero success rate achieved by PLBA, the SP was used where the algorithm with smaller value is considered faster. It can be seen in Table IX that PLBA was faster than some algorithms and slower than others on f_1, f_5, f_9, and f_{15} in the case of *d* = 10. In this case, SaDE was the fastest on f_9 and f_{15}, followed by PLBA. On the other hand, PLBA was the slowest algorithm on f_2, f_4, and f_{12}. In the case of *d* = 30, as can be seen in Table X, PLBA was the fastest on f_{15} and after SaDE on f_9. For the other functions (f_1, f_2, and f_7) PLBA converged faster than some algorithms and slower than others.

To statistically analyze the results obtained by PLBA and other state-of-the-art metaheuristic algorithms, we also employed the Friedman test. The results of all 25 10-dimensional problems were reported for all algorithms considered in the comparisons. On the other hand, in the case of the 30-dimensional version, the results of SaDE [47], and DMS-PSO [44] for $f_{16} - f_{25}$, and $f_{20} - f_{25}$, respectively, were not reported in the literature. This means that some data are missing for some algorithms. In addition, there have not been results reported for the conventional ABC in the case of using d = 30 [48].

Therefore, we performed two sets of statistical comparisons. The first set was among all algorithms using the 10-dimensional version of all 25 benchmarks, whereas the second one was among all algorithms excluding ABC on the 30-dimensional version of the first 15 benchmarks. The ranks calculated through the Friedman test for the algorithms considered are tabulated in Tables XI and XII for both sets of comparisons. It can be clearly seen in these tables that Restart CMA-ES was the best performing algorithm in both cases of *d* = 10 and *d* = 30. Whereas the worst performing algorithm was SPC-PNX in the case of employing 10 dimensions, and was the MABC in the case of using 30 dimensions. For the proposed PLBA algorithm, it ranked sixth in the comparisons considering the 10-dimensional versions and ranked fourth in the comparisons employing the 30-dimensional versions. The *p*-values calculated using the statistic from the Friedman test were 0.010674 and 0.001348 in the 10 and 30-dimensinal cases, respectively, as shown in Tables XI and XII. These *p*-values suggested a significant difference among the performance of the algorithms considered.

Subsequently, we used Holm and Hochberg tests to test the specific difference between Restart CMA-ES and the rest of the algorithms. Tables XIII and XIV show the adjusted *p*-values obtained by Holm and Hochberg methods in the case of employing the 10 and 30–dimensional benchmarks, respectively, considering the Restart CMA-ES as the control method. The Holm and Hochberg methods suggested a significant difference in the performance between the Restart CMA-ES in one side and SPC-PNX, ABC, and PLBA in the other side in the case of using *d* = 10. It can be clearly seen in Table XIII that Restart CMA-ES demonstrated a highly significant improvement over SPC-PNX with a significance level of α = 0.01, and a significant improvement over ABC, and PLBA at a significance level of α = 0.1. However, in the case of the compression set on the 30-dimensional benchmarks, it can be clearly seen in Table XIV that Restart CMA-ES showed a significant improvement over SPC-PNX, DE, and MABC at a level of significance of α = 0.05. On the other hand, no significant difference was suggested by Holm and Hochberg between Restart CMA-ES and PLBA in the case of 30-dimensional benchmark problems. Thus, it can be concluded that PLBA perform well even if the number of dimensions increases.

V. Conclusion

Despite the importance of the initialization part, the initial stage remains an initial step and its improvement is not enough for more challenging problem classes with different properties. Thus, the local and global search capabilities were also enhanced to improve the quality of final solution and the convergence speed of PLIA-BA on such problems. In this paper, a new local search algorithm, GLLSA, which is based on the Levy looping flights, has been adopted. Moreover, the mechanism of the global search has been enhanced to be closer to nature and based on the patch-Levy model adopted in PLIA. Consequently, a new version of BA called PLBA, which utilizes the initialization algorithm (PLIA), local search algorithm (GLLSA), and the enhanced global search has been proposed

We investigated the performance of the proposed PLBA on a set of challenging benchmark functions that are free of the properties of the standard functions (e.g., symmetry) that can be exploited by the optimization methods. We compared the proposed PLBA with other state-of-the-art algorithms available in the literature. Additionally, we conducted comparisons between the BA variants.

The comparisons with BA-based algorithms have shown that PLBA significantly outperformed the other BA versions: Basic BA, Shrinking-based BA, Standard BA and PLIA-BA. The results validated what has been stated before that the improvement of the initialization stage is not enough for all problem types and the improvement of local and global search capabilities is required as well. The experiments have also indicated that PLBA was able to produce comparable results with other state-of-the-art algorithms on the 10 and 30-dimensional challenging problems employed in the comparisons.

The problems employed in this work were static problems. Future work will focus on evaluating and validating the performance of the proposed PLBA on a set of recently proposed dynamic optimization problems.

Acknowledgment

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 Scheme No. AP-2102-019 entitled "Automated Medical Imaging Diagnostic Based on Four Critical Diseases: Brain, Breast, Prostate and Lung Cancer".

References

[1] S. Garnier, J. Gautrais, G. Theraulaz, The biological principles of swarm intelligence, Swarm Intelligence, 1 (2007) 3-31.

[2] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm intelligence: from natural to artificial systems, Oxford university press, 1999.

[3] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theoretical computer science, 344 (2005) 243-278.

[4] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence, 1 (2007) 33-57.

[5] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, in: IEEE Control Systems 2002, pp. 52-67.

[6] X.-S. Yang, Firefly algorithms for multimodal optimization, in: O. Watanabe, T. Zeugmann (Eds.) Stochastic algorithms: foundations and applications, Springer, Berlin, Heidelberg, 2009, pp. 169-178.

[7] H.A. Abbass, MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach, in: Proceedings of the 2001 Congress on Evolutionary Computation, IEEE, Seoul, 2001, pp. 207-214.

[8] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of global optimization, 39 (2007) 459-471.

[9] D. Teodorović, M. Dell'Orco, Bee colony optimization–a cooperative learning approach to complex transportation problems, in: Advanced OR and AI Methods in Transportation: Proceedings of 10th Meeting of EURO Working Group on Transportation Poznan, Poland, 2005, pp. 51-60.

[10] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, 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 2006), Elsevier Science Ltd, Cardiff, UK, 2006, pp. 454-459.

[11] B. Yuce, D. Pham, M. Packianather, E. Mastrocinque, An enhancement to the Bees Algorithm with slope angle computation and Hill Climbing Algorithm and its applications on scheduling and continuous-type optimisation problem, Production & Manufacturing Research, 3 (2015) 3-19.

[12] D. Pham, H.A. Darwish, Using the bees algorithm with Kalman filtering to train an artificial neural network for pattern classification, Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 224 (2010) 885-892.

[13] S. Abdullah, M. Alzaqebah, A hybrid self-adaptive bees algorithm for examination timetabling problems, Applied Soft Computing, 13 (2013) 3608-3620.

[14] S. Otri, Improving the bees algorithm for complex optimisation problems, in, Cardiff University, 2011.

[15] Q. Pham, D. Pham, M. Castellani, A modified bees algorithm and a statistics-based method for tuning its parameters, Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 226 (2012) 287-301.

[16] M. Packianather, M. Landy, D. Pham, Enhancing the speed of the Bees Algorithm using Pheromone-based Recruitment, in: 7th IEEE International Conference on Industrial Informatics (INDIN 2009) IEEE, Cardiff, Wales, 2009, pp. 789-794.

[17] S.A. Ahmad, D.T. Pham, K.W. Ng, M.C. Ang, TRIZ-inspired Asymmetrical Search Neighborhood in the Bees Algorithm, in: Sixth Asia Modelling Symposium (AMS), IEEE, Bali, 2012, pp. 29-33.

[18] B. Yuce, M.S. Packianather, E. Mastrocinque, D.T. Pham, A. Lambiase, Honey Bees Inspired Optimization Method: The Bees Algorithm, Insects, 4 (2013) 646-662.

[19] A. Ghanbarzadeh, Bees Algorithm: a novel optimisation tool, in, Cardiff University, 2007.

[20] N. Shatnawi, S. Sahran, M. Faidzul, A Memory-based Bees Algorithm: An Enhancement, Journal of Applied Sciences, 13 (2013) 497-502.

[21] W.A. Hussein, S. Sahran, S.N.H. Sheikh Abdullah, Patch-Levy-based initialization algorithm for Bees Algorithm, Applied Soft Computing, 23 (2014) 104-121.

[22] X.-S. Yang, Review of meta-heuristics and generalised evolutionary walk algorithm, International Journal of Bio-Inspired Computation, 3 (2011) 77-84.

[24] A. Reynolds, Cooperative random Lévy flight searches and the flight patterns of honeybees, Physics letters A, 354 (2006) 384-388.

[25] A.M. Reynolds, A.D. Smith, D.R. Reynolds, N.L. Carreck, J.L. Osborne, Honeybees perform optimal scale-free searching flights when attempting to locate a food source, Journal of Experimental Biology, 210 (2007) 3763-3770.

[26] P. Bailis, R. Nagpal, J. Werfel, Positional communication and private information in honeybee foraging models, in: Proceedings of the 7th international conference on Swarm Intelligence, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 263-274.

[27] A.M. Reynolds, A.D. Smith, R. Menzel, U. Greggers, D.R. Reynolds, J.R. Riley, Displaced honey bees perform optimal scale-free search flights, Ecology, 88 (2007) 1955-1961.

[28] T.D. Seeley, The wisdom of the hive: the social physiology of honey bee colonies, Harvard University Press, Cambridge, Massachusetts, 1995.

[29] M. Gutowski, Lévy flights as an underlying mechanism for global optimization algorithms, arXiv preprint math-ph/0106003, (2001).

[30] T. Tran, T.T. Nguyen, H.L. Nguyen, Global optimization using L'evy flights, in: Proceedings of the 3rd National Symposium on Research, Development and Application of Information and Communication Technology (ICT.rda), Hanoi, Vietnam, 2004.

[31] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), IEEE, Coimbatore, 2009, pp. 210-214.

[32] X.-S. Yang, Firefly algorithm, Levy flights and global optimization, in: M. Bramer, R. Ellis, M. Petridis (Eds.) Research and Development in Intelligent Systems XXVI, Springer, London, 2010, pp. 209-218.

[33] J. Xie, Y. Zhou, H. Chen, A Novel Bat Algorithm Based on Differential Operator and Lévy Flights Trajectory, Computational intelligence and neuroscience, 2013 (2013) 13.

[34] G. Wang, L. Guo, A.H. Gandomi, L. Cao, A.H. Alavi, H. Duan, J. Li, Lévy-Flight Krill Herd Algorithm, Mathematical Problems in Engineering, 2013 (2013) 14.

[35] H. Sharma, J.C. Bansal, K. Arya, Opposition based lévy flight artificial bee colony, Memetic Computing, 5 (2013) 213-227.

[36] G. Viswanathan, E. Raposo, M. Da Luz, Lévy flights and superdiffusion in the context of biological encounters and random searches, Physics of Life Reviews, 5 (2008) 133-150.

[37] C.T. Brown, L.S. Liebovitch, R. Glendon, Lévy flights in Dobe Ju/'hoansi foraging patterns, Human Ecology, 35 (2007) 129-138.

[23] G. Viswanathan, S.V. Buldyrev, S. Havlin, M. Da Luz, E. Raposo, H.E. Stanley, Optimizing the success of random searches, Nature, 401 (1999) 911-914.

[38] J. Liang, P. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proceedings of the 2005 IEEE on Swarm Intelligence Symposium (SIS 2005) IEEE, 2005, pp. 68-75.

[39] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, in, Nanyang Technological University, Singapore and KanGAL, 2005.

[40] A. Auger, N. Hansen, Performance evaluation of an advanced local search evolutionary algorithm, in: The 2005 IEEE Congress on Evolutionary Computation IEEE, 2005, pp. 1777-1784.

[41] D. Pham, E. Koç, Design of a two-dimensional recursive filter using the bees algorithm, International Journal of Automation and Computing, 7 (2010) 399-402.

[42] D. Pham, M. Castellani, Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms, Soft Computing, 18 (2014) 871-903.

[43] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: The 2005 IEEE Congress on Evolutionary Computation IEEE, 2005, pp. 1769-1776.

[44] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: The 2005 IEEE Congress on Evolutionary Computation, Ieee, Edinburgh, Scotland, 2005, pp. 522-528.

[45] P.J. Ballester, J. Stephenson, J.N. Carter, K. Gallagher, Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX, in: The 2005 IEEE Congress on Evolutionary Computation, IEEE, Edinburgh, Scotland, 2005, pp. 498-505.

[46] J. Ronkkonen, S. Kukkonen, K.V. Price, Real-parameter optimization with differential evolution, in: The 2005 IEEE Congress on Evolutionary Computation, IEEE, Edinburgh, Scotland, 2005, pp. 506-513.

[47] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: The 2005 IEEE Congress on Evolutionary Computation, IEEE, 2005, pp. 1785-1791.

[48] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Information Sciences, 192 (2012) 120-142.

[49] J. Derrac, S. García, D. Molina, 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, 1 (2011) 3-18.

TABLE I. MEAN ERROR VALUES ACHIEVED FOR PROBLEMS $f_1 - f_{25}$ (d = 10) BY BAS

Problem	PLBA		PLIA-BA		Standard BA		Shrinking-based BA		Basic BA	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
f_1	**0.00E+00**	**0.00E+00**	4.78E-03	1.05E-03	5.40E-02	1.03E-01	1.43E-04	3.30E-05	4.77E-03	9.54E-04
f_2	**3.98E-13**	**2.83E-13**	5.52E-03	1.74E-03	5.28E-03	1.63E-02	1.93E-04	6.66E-05	5.60E-03	1.46E-03
f_3	**8.87E+03**	**1.74E+04**	1.53E+05	1.16E+05	8.63E+04	5.08E+04	1.37E+05	1.30E+05	1.21E+05	1.15E+05
f_4	**2.38E-09**	**3.23E-09**	6.84E-01	1.81E-01	5.63E-02	2.04E-01	7.31E-03	1.77E-03	6.95E-01	1.63E-01
f_5	**6.08E-11**	**7.48E-11**	1.56E+01	4.98E+00	6.65E+01	2.43E+02	6.51E+01	2.38E+02	1.65E+01	3.34E+00
f_6	**1.25E+01**	**2.19E+01**	1.66E+02	4.44E+02	1.44E+03	2.65E+03	1.76E+01	5.01E+01	6.16E+01	1.06E+02
f_7	**2.13E-01**	**8.42E-02**	6.65E-01	1.13E-01	7.94E-01	1.72E-01	7.57E-01	1.28E-01	7.64E-01	1.26E-01
f_8	**2.00E+01**	4.80E-02	**2.00E+01**	**2.31E-02**	2.04E+01	6.06E-02	**2.00E+01**	2.41E-02	**2.00E+01**	3.85E-02
f_9	**0.00E+00**	**0.00E+00**	2.80E+01	7.66E+00	3.08E+01	1.62E+01	2.69E+01	9.19E+00	2.75E+01	9.45E+00
f_{10}	2.16E+01	1.04E+01	3.64E+01	7.96E+00	3.48E+01	2.05E+01	**1.94E+01**	**8.72E+00**	3.64E+01	4.67E+00
f_{11}	4.23E+00	1.38E+00	5.33E+00	1.17E+00	9.19E+00	6.78E-01	**1.95E+00**	**9.19E-01**	6.14E+00	7.24E-01
f_{12}	**1.40E+01**	**4.21E+01**	4.55E+02	1.30E+03	5.27E+03	6.78E+03	6.15E+02	2.14E+03	1.86E+02	4.79E+02
f_{13}	**1.46E-01**	**1.05E-01**	1.89E+00	7.36E-01	5.81E+00	4.07E+00	1.29E+00	4.13E-01	3.94E+00	9.29E-01
f_{14}	**3.13E+00**	**3.88E-01**	3.20E+00	2.29E-01	3.62E+00	2.22E-01	3.48E+00	2.27E-01	3.39E+00	2.85E-01
f_{15}	**7.96E-07**	**3.26E-07**	2.56E+02	7.60E+01	3.36E+02	9.30E+01	2.03E+02	6.61E+01	2.53E+02	7.43E+01
f_{16}	1.46E+02	2.25E+01	1.72E+02	1.17E+01	1.62E+02	2.83E+01	**1.28E+02**	**1.32E+01**	1.83E+02	1.34E+01
f_{17}	1.49E+02	1.90E+01	1.94E+02	3.51E+01	1.90E+02	2.92E+01	**1.42E+02**	**1.64E+01**	1.95E+02	3.91E+01
f_{18}	4.77E+02	1.45E+02	7.10E+02	1.69E+02	**4.34E+02**	**2.00E+02**	5.80E+02	1.42E+02	6.15E+02	1.04E+02
f_{19}	**4.27E+02**	**1.69E+02**	6.87E+02	1.69E+02	4.91E+02	2.06E+02	5.82E+02	1.41E+02	6.11E+02	1.07E+02
f_{20}	**4.17E+02**	**1.33E+02**	6.77E+02	1.54E+02	4.24E+02	1.95E+02	5.82E+02	1.41E+02	6.11E+02	1.07E+02
f_{21}	**5.75E+02**	**3.13E+01**	7.48E+02	1.28E+02	8.71E+02	3.02E+02	8.08E+02	3.51E+02	7.76E+02	1.86E+02
f_{22}	7.80E+02	3.91E+01	7.79E+02	6.83E+01	7.82E+02	6.99E+00	**7.32E+02**	**1.31E+02**	7.84E+02	4.45E+01
f_{23}	**6.14E+02**	**1.51E+02**	7.81E+02	1.92E+02	6.40E+02	6.68E+01	7.08E+02	2.16E+02	7.12E+02	1.73E+02
f_{24}	2.89E+02	2.32E+02	2.30E+02	1.40E+02	2.43E+02	7.59E+01	**2.00E+02**	**1.64E-02**	2.02E+02	4.06E-01
f_{25}	4.12E+02	1.67E+00	3.99E+02	2.55E+01	4.11E+02	8.09E-01	**3.95E+02**	**4.34E+01**	4.08E+02	9.29E+00

TABLE II. SUCCESS RATE (SR%) AND SUCCESS PERFORMANCE (SP) ACHIEVED FOR PROBLEMS $f_1 - f_{25}$ (d = 10) BY BAS (SP NOT CALCULATED WHEN SR = 0% AND [-] SIGN WAS PUT INSTEAD)

Problem	PLBA		PLIA-BA		Standard BA		Shrinking-based BA		Basic BA	
	SR%	SP	SR%	SP	SR%	SP	SR%	SP	SR%	SP
f_1	**100**	**8.2294E+03**	0	-	0	-	0	-	0	-
f_2	**100**	**4.8090E+04**	0	-	0	-	0	-	0	-
f_3	0	-	0	-	0	-	0	-	0	-
f_4	**100**	**7.1668E+04**	0	-	0	-	0	-	0	-
f_5	**100**	**7.7768E+04**	0	-	0	-	0	-	0	-
$f_6 - f_8$	0	-	0	-	0	-	0	-	0	-
f_9	**100**	**1.8948E+04**	0	-	0	-	0	-	0	-
f_{10}, f_{11}	0	-	0	-	0	-	0	-	0	-
f_{12}	**8**	**1.0497E+06**	0	-	0	-	0	-	0	-
f_{13}, f_{14}	0	-	0	-	0	-	0	-	0	-
f_{15}	**100**	**4.1000E+04**	0	-	0	-	0	-	0	-
$f_{16} - f_{25}$	0	-	0	-	0	-	0	-	0	-

TABLE III. MEAN ERROR VALUES ACHIEVED FOR PROBLEMS $f_1 - f_{25}$ (d = 30) BY BAS

Problem	PLBA		PLIA-BA		Standard BA		Shrinking-based BA		Basic BA	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
f_1	**8.41E-14**	**2.90E-14**	4.96E-02	5.02E-03	2.48E-10	5.02E-11	5.44E-11	6.01E-12	5.00E-02	5.19E-03
f_2	**1.87E-08**	**7.56E-08**	1.81E-01	5.01E-02	1.98E-07	6.76E-07	4.39E+01	2.18E+02	1.81E-01	3.62E-02
f_3	**2.94E+05**	**7.68E+04**	8.16E+05	3.49E+05	2.38E+06	8.06E+05	1.23E+06	4.08E+05	7.80E+05	3.19E+05
f_4	2.87E+03	1.53E+03	5.09E+02	3.32E+02	2.54E+04	8.55E+03	**2.57E+02**	**2.29E+02**	1.50E+04	6.89E+03
f_5	7.24E+03	1.70E+03	4.24E+03	8.03E+02	8.76E+03	2.35E+03	1.10E+04	1.92E+03	**3.93E+03**	**9.76E+02**
f_6	**7.59E+01**	**8.26E+01**	4.40E+02	3.22E+02	6.84E+02	1.34E+03	2.72E+03	4.31E+03	3.36E+02	4.36E+02
f_7	**1.14E-02**	**1.03E-02**	1.12E+00	1.73E-02	8.63E-01	1.88E-01	9.38E-01	3.35E-02	1.12E+00	1.46E-02
f_8	2.01E+01	2.97E-02	**2.00E+01**	**1.92E-02**	2.10E+01	5.85E-02	2.01E+01	4.45E-02	2.01E+01	2.53E-02
f_9	**9.34E-10**	**9.99E-10**	1.93E+02	2.74E+01	1.54E+02	3.51E+01	2.20E+02	3.57E+01	2.25E+02	3.98E+01
f_{10}	**1.29E+02**	**3.08E+01**	2.47E+02	1.81E+01	1.69E+02	7.08E+01	1.56E+02	4.20E+01	2.37E+02	1.73E+01
f_{11}	2.52E+01	3.21E+00	2.65E+01	2.96E+00	3.90E+01	1.40E+00	**1.29E+01**	**3.49E+00**	3.16E+01	1.64E+00
f_{12}	**1.69E+03**	**1.16E+03**	1.73E+04	1.56E+04	6.60E+04	1.63E+05	2.00E+04	1.80E+04	1.16E+04	1.10E+04
f_{13}	**5.59E-01**	**3.02E-01**	1.60E+01	1.92E+00	6.35E+02	2.94E+03	5.33E+00	1.46E+00	2.10E+01	2.19E+00
f_{14}	**1.25E+01**	**3.35E-01**	1.27E+01	2.30E-01	1.30E+01	2.61E-01	1.31E+01	3.80E-01	1.31E+01	2.06E-01
f_{15}	**1.29E+02**	**1.64E+02**	5.76E+02	6.58E+01	4.09E+02	2.77E+01	4.20E+02	3.90E+01	5.78E+02	5.71E+01

f_{16}	1.80E+02	4.44E+01	2.77E+02	3.24E+01	**1.26E+02**	**3.28E+01**	1.91E+02	6.96E+01	2.85E+02	4.10E+01
f_{17}	2.15E+02	8.80E+01	4.60E+02	7.44E+01	**1.80E+02**	**5.43E+01**	2.41E+02	7.30E+01	4.68E+02	7.17E+01
f_{18}	9.11E+02	5.21E+00	9.15E+02	5.63E+00	**9.10E+02**	**2.92E+00**	9.12E+02	1.60E+00	9.16E+02	1.36E+00
f_{19}	**9.09E+02**	2.36E+00	9.16E+02	1.36E+00	**9.09E+02**	**2.32E+00**	9.12E+02	1.99E+00	9.16E+02	1.73E+00
f_{20}	**9.09E+02**	2.50E+00	9.16E+02	1.36E+00	**9.09E+02**	**2.22E+00**	9.12E+02	1.99E+00	9.16E+02	1.73E+00
f_{21}	1.02E+03	2.03E+02	7.18E+02	2.05E+01	**5.91E+02**	**2.02E+02**	7.66E+02	3.14E+02	7.32E+02	1.38E+01
f_{22}	**9.09E+02**	**2.82E+01**	9.44E+02	2.38E+01	9.48E+02	1.43E+01	9.86E+02	5.61E+01	9.41E+02	1.68E+01
f_{23}	1.11E+03	3.05E+00	7.46E+02	2.58E+01	7.03E+02	2.88E+01	**5.57E+02**	**1.25E+01**	7.38E+02	1.71E+01
f_{24}	9.36E+02	1.53E+02	1.36E+03	2.16E+01	2.03E+02	1.57E+01	**2.00E+02**	**0.00E+00**	1.35E+03	2.52E+01
f_{25}	2.13E+02	2.46E+00	2.12E+02	4.93E-01	2.12E+02	8.48E-01	**2.11E+02**	**3.96E-01**	2.13E+02	5.42E-01

TABLE IV. Success Rate (SR%) and Success Performance (SP) Achieved for Problems $f_1 - f_{25}$ (d = 30) by Bas (SP Not Calculated when SR = 0% and [-] Sign was Put Instead)

Problem	PLBA		PLIA-BA		Standard BA		Shrinking-based BA		Basic BA	
	SR%	SP	SR%	SP	SR%	SP	SR%	SP	SR%	SP
f_1	**100**	**7.7945E+04**	0	-	100	2.0265E+05	100	1.8600E+05	0	-
f_2	**96**	2.5518E+05	0	-	96	**2.5216E+05**	64	3.3130E+05	0	-
$f_3 - f_6$	0	-	0	-	0	-	0	-	0	-
f_7	**68**	**1.9660E+05**	0	-	0	-	0	-	0	-
f_8	0	-	0	-	0	-	0	-	0	-
f_9	**100**	**1.1871E+05**	0	-	0	-	0	-	0	-
$f_{10} - f_{14}$	0	-	0	-	0	-	0	-	0	-
f_{15}	**52**	**1.3942E+05**	0	-	0	-	0	-	0	-
$f_{16} - f_{25}$	0	-	0	-	0	-	0	-	0	-

TABLE V. Average Rankings of BA Variants (Friedman) on 10-Dimenstional Problems $f_1 - f_{25}$

Algorithm	Ranking
PLBA	1.66
Shrinking-based BA	2.34
PLIA-BA	3.52
Basic BA	3.64
Standard BA	3.84
p-value	0

TABLE VI. Average Rankings of BA Variants (Friedman) on 30-Dimenstional Problems $f_1 - f_{25}$

Algorithm	Ranking
PLBA	1.76
Standard BA	2.94
Shrinking-based BA	3.04
PLIA-BA	3.42
Basic BA	3.84
p-value	0.000071

TABLE VII. Adjusted p-Values Associated with BA Variants (Friedman) on 10-Dimenstional Problems $f_1 - f_{25}$

Algorithm	Unadjusted *p*	*p Holm*	*p Hochberg*
Standard BA	0.000001	0.000004	0.000004
Basic BA	0.00001	0.000029	0.000029
PLIA-BA	0.000032	0.000064	0.000064
Shrinking-based BA	0.128379	0.128379	0.128379

TABLE VIII. Adjusted p-Values Associated with BA Variants (Friedman) on 30-Dimenstional Problems $f_1 - f_{25}$

Algorithm	Unadjusted *p*	*p Holm*	*p Hochberg*
Basic BA	0.000003	0.000013	0.000013
PLIA-BA	0.000206	0.000617	0.000617
Shrinking-based BA	0.004208	0.008415	0.008326
Standard BA	0.008326	0.008415	0.008326

TABLE IX. SUCCESS RATE (SR%) AND SUCCESS PERFORMANCE (SP) ACHIEVED FOR PROBLEMS $f_1 - f_{25}$ (d = 10) BY PLBA AND OTHER STATE-OF-THE-ART ALGORITHMS (SP NOT CALCULATED WHEN SR = 0% AND [-] SIGN WAS PUT INSTEAD)

Problem	PLBA		DE [46]		SaDE [47]		DMS-PSO [44]		SPC-PNX [45]		Restart CMA-ES [43]	
	SR%	SP	SR%	SP	SR%	SP	SR%	SP	SR%	SP	SR%	SP
f_1	**100**	8.2294E+03	**100**	2.9410E+04	**100**	1.0126E+04	**100**	1.1912E+04	**100**	6.7252E+03	**100**	**1.6100E+03**
f_2	**100**	4.8090E+04	**100**	4.6309E+04	**100**	1.0237E+04	**100**	1.2052E+04	**100**	3.1012E+04	**100**	**2.3800E+03**
f_3	0	-	80	1.1502E+05	64	5.2306E+04	**100**	1.2480E+04	0	-	**100**	**6.5000E+03**
f_4	**100**	7.1668E+04	**100**	5.2372E+04	96	4.5601E+04	0	-	**100**	3.0714E+04	**100**	**2.9000E+03**
f_5	**100**	7.7768E+04	**100**	4.0746E+04	0	-	80	1.1336E+05	**100**	4.0259E+04	**100**	**5.8500E+03**
f_6	0	-	96	4.7398E+04	**100**	4.8777E+04	**100**	5.4677E+04	0	-	**100**	**1.0800E+04**
f_7	0	-	8	1.2000E+06	24	1.7197E+05	16	5.8672E+05	4	1.8033E+06	**100**	**4.6700E+03**
f_8	0	-	0	-	0	-	0	-	0	-	0	-
f_9	**100**	1.8948E+04	44	1.7681E+05	**100**	**1.7048E+04**	**100**	3.4612E+04	0	-	76	7.5700E+04
f_{10}	0	-	0	-	0	-	0	-	0	-	**92**	**6.5000E+04**
f_{11}	0	-	**48**	**1.8852E+05**	0	-	0	-	4	1.0943E+06	24	2.6300E+05
f_{12}	8	1.0497E+06	76	7.1904E+04	**100**	**3.1933E+04**	76	5.4443E+04	0	-	88	3.2700E+04
f_{13}, f_{14}	0	-	0	-	0	-	0	-	0	-	0	-
f_{15}	**100**	4.1000E+04	4	2.4600E+06	92	**3.3165E+04**	88	5.6563E+04	0	-	0	-
$f_{16} - f_{25}$	0	-	0	-	0	-	0	-	0	-	0	-

TABLE X. SUCCESS RATE (SR%) AND SUCCESS PERFORMANCE (SP) ACHIEVED FOR PROBLEMS $f_1 - f_{25}$ (d = 30) BY PLBA AND OTHER STATE-OF-THE-ART ALGORITHMS (SP NOT CALCULATED WHEN SR = 0% AND [-] SIGN WAS PUT INSTEAD)

Problem	PLBA		DE [46]		SaDE [47]		DMS-PSO [44]		SPC-PNX [45]		Restart CMA-ES [43]	
	SR%	SP	SR%	SP	SR%	SP	SR%	SP	SR%	SP	SR%	SP
f_1	**100**	7.7945E+04	**100**	1.3855E+05	**100**	2.0234E+04	**100**	5.0263E+03	**100**	3.0326E+04	**100**	**4.5000E+03**
f_2	96	2.5518E+05	0	-	96	1.4883E+05	**100**	1.2552E+05	88	3.1536E+05	**100**	**1.3000E+04**
f_3	0	-	0	-	0	-	84	3.4100E+05	0	-	**100**	**4.2700E+04**
f_4	0	-	0	-	52	5.3816E+05	0	-	**76**	3.6334E+05	40	**5.9000E+04**
f_5	0	-	0	-	0	-	0	-	0	-	**100**	**6.5900E+04**
f_6	0	-	0	-	0	-	98	3.2781E+05	4	5.2053E+06	**100**	**6.0000E+04**
f_7	68	1.9660E+05	88	1.9952E+05	80	1.3477E+05	96	5.9577E+04	64	3.7063E+05	**100**	**6.1100E+03**
f_8	0	-	0	-	0	-	0	-	0	-	0	-
f_9	**100**	1.1871E+05	0	-	**100**	**9.8934E+04**	0	-	0	-	36	7.9000E+05
f_{10}	0	-	0	-	0	-	0	-	0	-	**12**	**2.4200E+06**
f_{11}	0	-	0	-	0	-	0	-	0	-	**4**	**4.9800E+06**
f_{12}	0	-	0	-	0	-	16	1.5108E+06	0	-	**32**	**2.2500E+05**
f_{13}, f_{14}	0	-		-	0	-	0	-	0	-	0	-
f_{15}	**52**	**1.3942E+05**		-	0	-	0	-	0	-	0	-
$f_{16} - f_{25}$	0	-		-	0	-	0	-	0	-	0	-

TABLE XI. AVERAGE RANKINGS OF PLBA AND OTHER ALGORITHMS (FRIEDMAN) ON 10-DIMENSTIONAL PROBLEMS $f_1 - f_{25}$

Algorithm	Ranking
Restart CMA-ES	3.4
MABC	3.56
DMS-PSO	4.28
SaDE	4.28
DE	4.6
PLBA	**5.02**
ABC	5.12
SPC-PNX	5.74
p-value	0.010674

TABLE XII. AVERAGE RANKINGS OF PLBA AND OTHER ALGORITHMS (FRIEDMAN) ON 30-DIMENSTIONAL PROBLEMS $f_1 - f_{15}$

Algorithm	Ranking
Restart CMA-ES	2.8
SaDE	2.9667
DMS-PSO	3.1667
PLBA	**3.8333**
DE	4.9
SPC-PNX	5.1333
MABC	5.2
p-value	0.001348

TABLE XIII. ADJUSTED *p*-VALUES ASSOCIATED WITH PLBA AND OTHER ALGORITHMS (FRIEDMAN) ON 10-DIMENSTIONAL PROBLEMS $f_1 - f_{25}$

Algorithm	Unadjusted *p*	*p Holm*	*p Hochberg*
SPC-PNX	0.000731	0.00512	0.00512
ABC	0.013043	0.078255	0.078255
PLBA	**0.019373**	**0.096867**	**0.096867**
DE	0.083265	0.333058	0.333058
SaDE	0.204024	0.612072	0.408048
DMS-PSO	0.204024	0.612072	0.408048
MABC	0.817361	0.817361	0.817361

TABLE XIV. ADJUSTED *p*-VALUES ASSOCIATED WITH PLBA AND OTHER ALGORITHMS (FRIEDMAN) ON 30-DIMENSIONAL PROBLEMS $f_1 - f_{15}$

Algorithm	Unadjusted *p*	*p Holm*	*p Hochberg*
MABC	0.002346	0.014075	0.014075
SPC-PNX	0.003096	0.01548	0.01548
DE	0.007762	0.031049	0.031049
PLBA	**0.1902**	**0.570599**	**0.570599**
DMS-PSO	0.64205	0.832662	0.832662
SaDE	0.832662	0.832662	0.832662

```
For $CurSite = 1 : m$
    Set $CurBestSite = Sites[CurSite]$;
    While (not all $RecruitBee$ recruited)
        Set Time = $t$;
        While(Time != 0)
            Distribute the current recruit bee from the current best site to search the
            neighborhood area of the current site according to Levy flight distribution (3) with search size or scale $\gamma_2$ .
            If($NewSite$ is better than $CurBestSite$)
                $CurBestSite = NewSite$ ;
                Break;
            End If
            Time = Time – 1;
        End While
    End While
    $Sites[CurSite] = CurBestSite$;
End For
Shrink the Levy search size ( $\gamma_2$ ) by a shrinking factor ( $sf$ )
```

Fig. 1. Pseudo-code of the proposed Greedy Levy-based Local Search Algorithm (GLLSA)

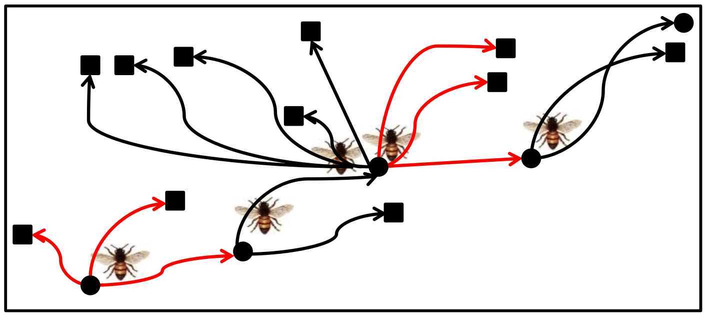

Fig. 2. A schematic diagram of recruit bees foraging and exploiting a patch in the proposed local search algorithm (● for better site than the current best site and ■ for worse sites)

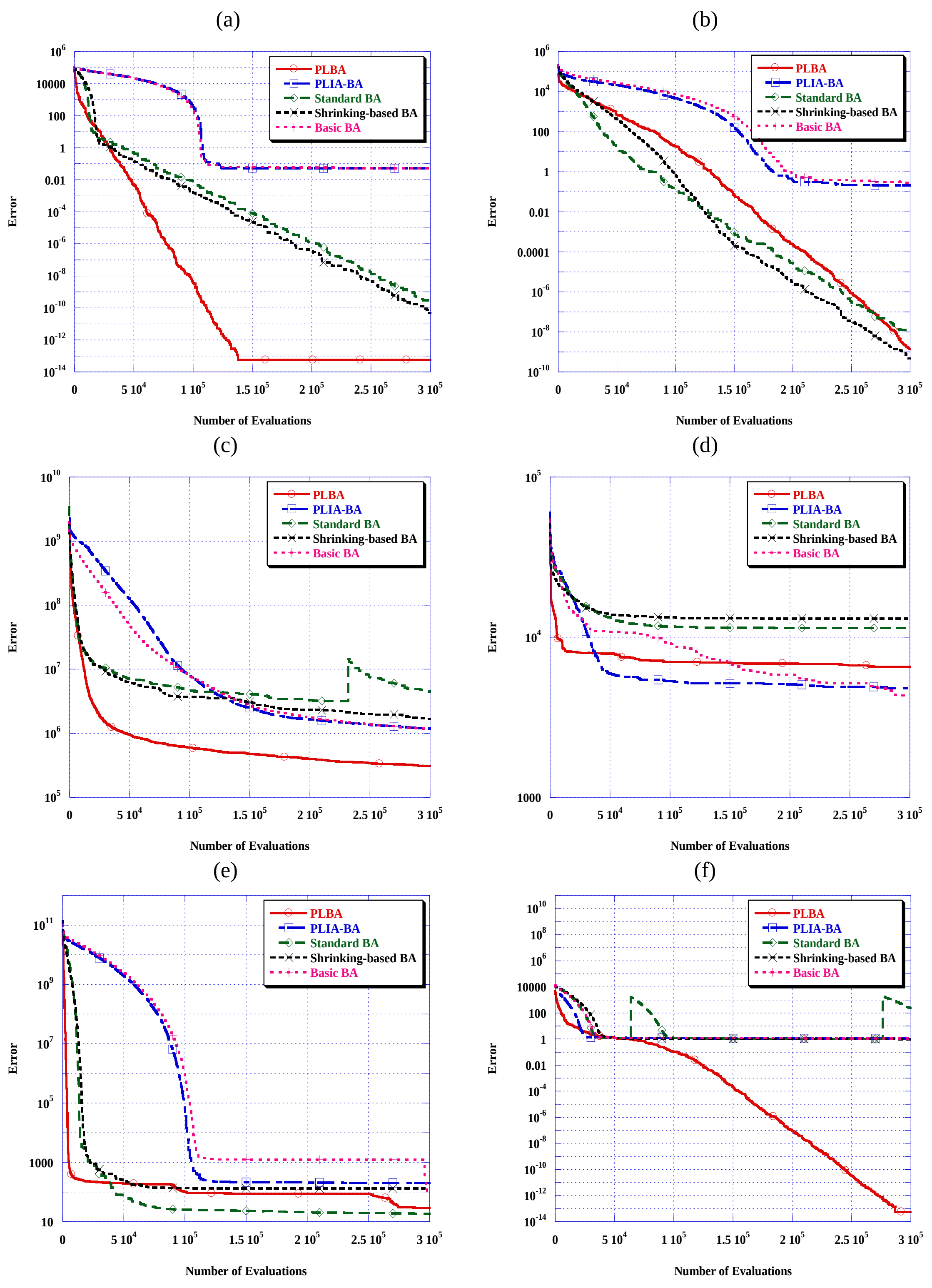

Fig. 3. Convergence behavior of the BA variants on functions: (a) f_1, (b) f_2, (c) f_3, (d) f_5, (e) f_6, and (f) f_7

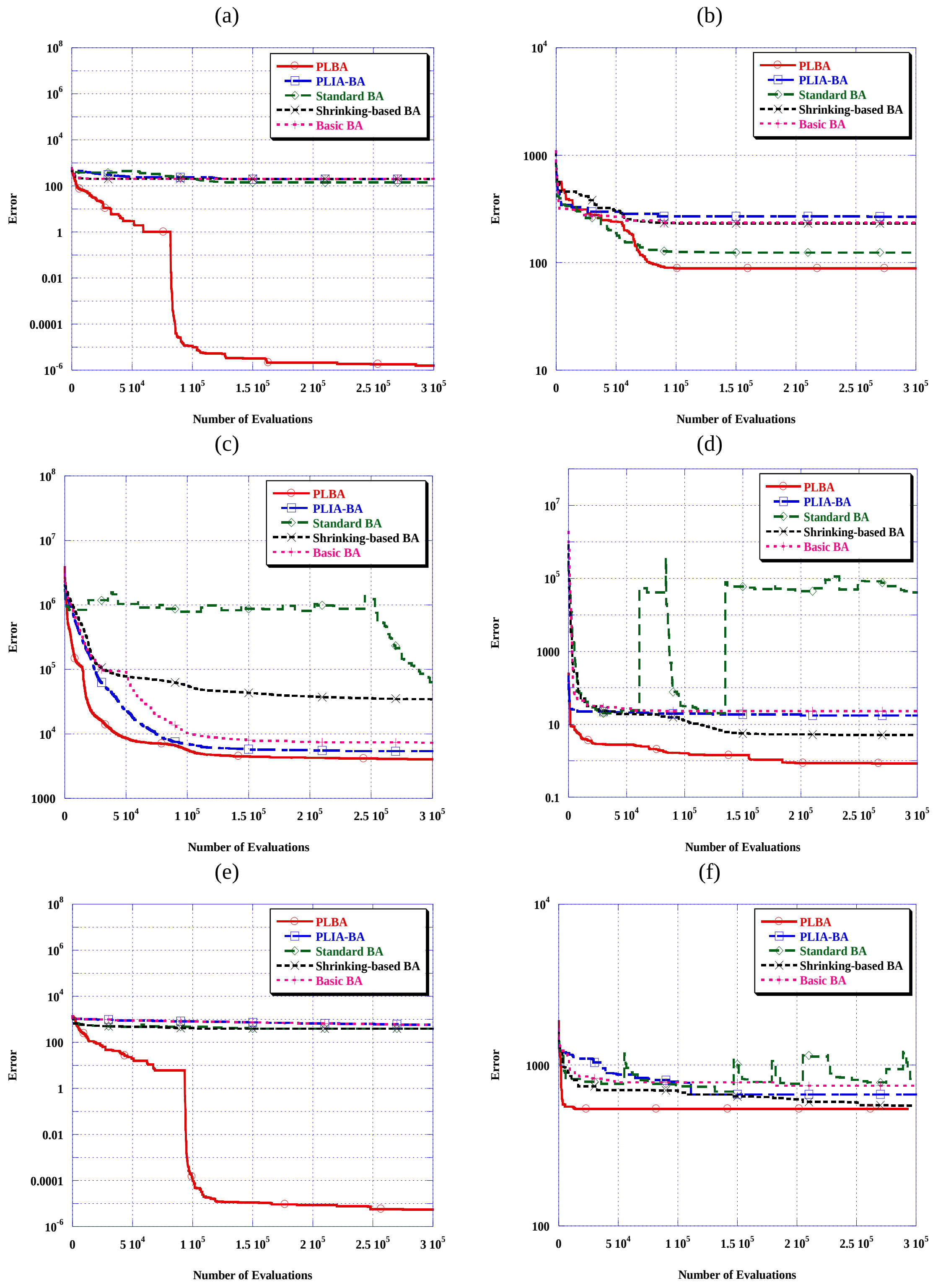

Fig. 4. Convergence behavior of the BA variants on functions: (a) f_9, (b) f_{10}, (c) f_{12}, (d) f_{13}, (e) f_{15}, and (f) f_{23}

APPENDIX A

Table A. 1 The parameters used with different values for different problems in Basic BA, Shrinking-based BA, and Standard BA

Function	Basic BA	Shrinking-based BA		Standard BA		
	ngh	ngh_{init}	*sf*	ngh_{init}	*sf*	*stlim*
f_1	0.1	1	0.999	1	0.999	700
f_2	0.1	1	0.999	1	0.999	700
f_3	0.1	1	0.9999	1	0.800	700
f_4	1	5	0.999	5	0.999	700
f_5	1	1	0.9991	1	0.9991	700
f_6	0.1	1	0.999	1	0.999	700
f_7	5	5	0.9999	5	0.9999	700
f_8	1E-3	0.001	0.999	1E-3	0.999	700
f_9	0.1	0.1	0.999	1	0.999	700
f_{10}	1	1	0.999	1	0.999	700
f_{11}	0.1	0.1	0.999	1	0.999	700
f_{12}	0.01	0.01	0.9999	1	0.999	700
f_{13}	0.1	1	0.999	1	0.999	700
f_{14}	0.1	0.1	0.999	1	0.999	700
f_{15}	1E-3	3	0.999	3	0.999	700
f_{16}	1	1	0.999	3	0.999	700
f_{17}	0.01	1	0.999	3	0.999	700
f_{18}	1	1	0.99991	3	0.999	700
f_{19}	1	1	0.99991	3	0.999	700
f_{20}	1	1	0.99991	3	0.999	700
f_{21}	1	1	0.990	1	0.990	700
f_{22}	1	1	0.999	3	0.999	700
f_{23}	1	1	0.9999	3	0.9999	700
f_{24}	0.1	1	0.999	3	0.999	700
f_{25}	1	1	0.9999	3	0.9999	700

Table A. 2 The parameters used with different values for different problems in PLIA-BA, and PLBA

Function	PLIA-BA			PLBA					
	ngh	*P*	γ_1	*P*	γ_1	γ_{2_init}	γ_3	*t*	*sf*
f_1	0.1	1	1E-7	19	3	1E-2	1E-2	5	0.985
f_2	0.1	1	1E-7	1	1	1	1	10	0.990
f_3	0.1	1	1E-7	1	1	1E-3	1	10	1
f_4	1	1	1E-7	5	3	2	1	60	0.960
f_5	1	1	1E-7	1	1	2	1	50	0.950
f_6	0.1	1	1E-7	1	1	1	1	20	0.990
f_7	5	20	1E-7	10	1	4	1	30	0.990
f_8	1E-4	1	1E-7	1	1	1E-5	1	50	1
f_9	1E-3	1	1E-7	10	4	7E-2	1E-7	45	0.960
f_{10}	1	1	1E-7	1	1	1	1	50	0.980
f_{11}	0.01	19	1	1	1	1	1	20	0.980
f_{12}	0.01	1	1E-7	17	1	1E-4	3	60	1
f_{13}	0.1	19	1E-7	12	1E-3	1E-4	1E-3	40	1
f_{14}	5	1	1E-7	1	1	1E-2	1	50	1
f_{15}	1E-3	1	1E-7	1	1E-7	4E-5	1E-2	30	1
f_{16}	1	1	1E-7	1	1	1	1	30	0.990
f_{17}	0.01	1	1E-7	1	1	1	1	50	0.990
f_{18}	1	1	1	19	1	2	1	20	1
f_{19}	1	1	3	19	3	2	3	15	0.990
f_{20}	1	1	3	19	1	1	1	20	0.990
f_{21}	1	1	1	19	3	1	3	15	1
f_{22}	1	19	1	1	1	1	1	15	0.990
f_{23}	1	1	1	1	1	1	1	20	0.999
f_{24}	0.1	1	1	1	5	1	5	40	0.980
f_{25}	1	2	1	1	5	2	5	40	0.980