An Adaptive Multi-population Artificial Bee Colony Algorithm for Dynamic Optimisation Problems

Shams K. Nseef[1], Salwani Abdullah[1], Ayad Turky[2] and Graham Kendall[3,4]

[1]Data Mining and Optimisation Research Group, Centre for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia.
E-mail: shams.shamosa91@gmail.com; salwani@ukm.edu.my
[2]Swinburne University of Technology, Melbourne, Victoria, Australia
E-mail: aturky@swin.edu.au
[3]University of Nottingham Malaysia Campus, Semenyih, Malaysia
[4]ASAP Research Group, University of Nottingham, Nottingham, United Kingdom
Email: Graham.Kendall@nottingham.edu.my; Graham.Kendall@nottingham.ac.uk

Abstract

Recently, interest in solving real-world problems that change over the time, so called dynamic optimisation problems (DOPs), has grown due to their practical applications. A DOP requires an optimisation algorithm that can dynamically adapt to changes and several methodologies have been integrated with population-based algorithms to address these problems. Multi-population algorithms have been widely used, but it is hard to determine the number of populations to be used for a given problem. This paper proposes an adaptive multi-population artificial bee colony (ABC) algorithm for DOPs. ABC is a simple, yet efficient, nature inspired algorithm for addressing numerical optimisation, which has been successfully used for tackling other optimisation problems. The proposed ABC algorithm has the following features. Firstly it uses multi-populations to cope with dynamic changes, and a clearing scheme to maintain the diversity and enhance the exploration process. Secondly, the number of sub-populations changes over time, to adapt to changes in the search space. The moving peaks benchmark DOP is used to verify the performance of the proposed ABC. Experimental results show that the proposed ABC is superior to the ABC on all tested instances. Compared to state of the art methodologies, our proposed ABC algorithm produces very good results.

Keywords: dynamic optimisation, artificial bee colony algorithm, adaptive multi-population method, meta-heuristics

1. Introduction

Many real-world optimisation problems have the characteristic of changing over time in terms of decision variables, constraints and the objective function [1], [2]. These problems are referred to as dynamic optimisation problems (DOPs) in the scientific literature. A DOP requires an optimisation algorithm that can dynamically adapt to the changes and track the optimum solution during the execution of the algorithm [2]. Given their practical applications and complexity, DOPs have attracted a lot of research attention. Population-based algorithms, which are a set of methodologies that utilise a population of solutions distributed over the search space, have attracted particular attention, due to their good performance [1], [2]. A key challenge in developing an optimisation algorithm for DOPs is how to maintain population diversity during the search process in order to keep track of landscape changes [3]. Several interesting diversity schemes have been developed in order to improve the search capability of population-based algorithms so that they can adapt effectively to the problem as it changes.

Prediction based methods is one of the diversity schemes which has been widely integrated with other algorithms to maintain the diversity. This methodology uses an algorithm to learn patterns from previous searches, which are then used to predict future changes. It should be noted that memory methods can be categorised as a special case of the prediction method as they store a set of solutions to be used when a problem changes [2]. The prediction method is suitable for problems with cyclic changes. Hatzakis and Wallace [4] proposed a hybrid algorithm that combines an evolutionary algorithm and a forecasting methodology for DOPs. Forecasting is used to predict the movement of the optimum based on previous movements. The results demonstrate that this method is suitable for problems which change quickly if the movement of the optimum solution is predicted correctly. Sim et al. [5] used a prediction based method to predict how the environment would change and the time of the next change. The authors utilised a Markov chain that uses the previous movement of the search in order to predict future changes and a linear regression to predict when the change will occur. The results demonstrate that the hybrid algorithm performs with prediction, than without. Branke and Mattfeld [6] proposed an anticipation-based algorithm for DOPs. This algorithm attempts to simultaneously search for a good quality solution and move the search into a different area based on the previous changes. This proposed algorithm was tested on a dynamic job-shop scheduling problem and it was shown to produce very good results compared to other algorithms. The advantage of the prediction method is that it can be effective in detecting the global optima quickly, if the predictions are accurate [2]. The main drawback with this method is that it depends on the training model and in many cases the data used during the training process does not capture real world scenarios and there is a possibility of training errors due to lack of training data [7], [2].

Memory based methodologies aim to maintain diversity. They use a memory with a fixed size to store some of promising solutions that are captured during the search process. When a change is detected, the stored solutions will be reinserted into the current population and the population will be filtered to include only the best solutions. Examples of memory based methodologies can be found in [8], [9], [10], [11]. These methodologies have worked well when the dynamic problems are periodical or cyclic. The drawback is that they have parameter sensitivities that need to be determined in advance, and most real world problems are not cyclic in nature.

Self-adaptive algorithms attempt to adaptively improve the diversification of population-based algorithms based on environmental changes. They use mechanisms to adapt the algorithm to the changes in the search space [2]. Adaptive mechanisms can improve algorithm search behaviour and also reduce the need for manual parameter tuning. The idea is to apply different operators or parameter values for different problems by adaptively changing them during the search process [7], [2]. Grefenstette [12] proposed a self-adaptive genetic algorithm for DOPs. The proposed algorithm adaptively selects different crossover/mutation operators at each generation. The author uses an agent based concept to control the selection process, and each agent represents a crossover or mutation operator. All agents are executed simultaneously and the one that generates the best solution is selected for the current instance. Promising results were achieved when compared to other algorithms. Grefenstette [12] also proposes an idea called a genetic mutation rate for the DOP. The idea is to set the value of the mutation rate based on the fitness of the population. This idea was shown to generate better results compared to the basic genetic algorithm. Ursem [13] proposed a multinational genetic algorithm for the DOP. The main parameters are encoded with the decision variables and are evolved during the solution process. The results show that this algorithm is very good for simple instances in which the velocity of the moving peaks is constant. It is also able to adapt by changing the algorithm parameters during the search. However, encoding the parameters with the solution decision variables requires specialist evolutionary operators. In addition, it is also very difficult to determine the values of the parameters [2].

Multi-population methods improve diversity by dividing the population of solutions into several sub-populations and distributing them throughout the search landscape so that they can more effectively capture the problem changes. The idea is to maintain population diversity by assigning a different sub-population to a different area, where each one is responsible for either intensifying or diversifying the search process [7], [2]. These sub-populations interact with each other via a merge and divide process when a change in the environment is detected. The multi-population method has been shown to be effective in dealing with various problem changes, whether they are cyclic or non-cyclic, and it has outperformed other methods on various problem sizes. Branke et al. [14] proposed a self-organising scouts multi-population evolutionary algorithm for the DOP. The population of solutions is divided into two groups; small and large. The small population group is responsible for tracking promising solutions found so far, while the large population group tries to find a new region of the search space that has a new peak. The proposed algorithm was tested on the moving peaks benchmark (MPB), obtaining very good results. Blackwell and Branke [15] proposed a multi-swarm optimisation algorithm for the DOP. The swarm is divided into subsets of swarms. These multi-swarms interact with each other locally, through algorithm parameters, and globally by using an anti-convergence mechanism. The anti-convergence mechanism searches for new peaks by removing the worst ones and re-initialising them into a different area in the search space. The proposed algorithm obtained very good results when tested on MPB problems. Mendes and Mohais [16] presented a multi-population differential evolution algorithm for the DOP. The population of solutions is divided into several sub-populations. Each sub-population is assigned to a different area of the search space. The experimental results show that this algorithm obtains very good results for MPB problems. Li and Yang [17] proposed a fast multi-swarm Particle Swarm Optimisation (PSO) algorithm for the DOP. The swarm population is divided into two types of swarms; parents and children. The parent swarm explores the entire search space to seek the global optima, while the child swarm is responsible for monitoring the search behaviour around the best solution obtained by the parent swarm. The position of the child swarm is dynamically updated during the process. The algorithm was tested on the MPB problems and produced good results when compared to other methods. Yang and Li [18] presented a clustering-based particle swarm optimiser for the DOP. The swarm is divided based on a hierarchical clustering method to locate and track multiple peaks. The algorithm achieved very good results when tested on the MPB. Turky and Abdullah [19] proposed a multi-population electromagnetic algorithm for DOPs. The proposed algorithm divides the population into several sub-populations to simultaneously explore and exploit the search process. The algorithm was tested on MPB problems and obtained very good results when compared to other population diversity mechanisms. The same authors [20] also presented a multi-population harmony search algorithm for the DOP. The population is divided into sub-populations. Each sub-population is responsible for either exploring or exploiting the search space. An external archive is utilised to track the best solutions found so far, which are used to replace the worst ones when a change is detected. The results show that this algorithm produces good results when compared to other methods. Sharifi et al. [21] proposed a hybrid PSO and local search algorithm for DOPs. The algorithm utilises a fuzzy social-only model to locate the peaks. The results show that this algorithm can produce very good results for MPB problems. In Li et al. [22] comprehensive experimental analysis was reported on the performance of a multi-population method with various algorithms in relation to DOPs. The authors concluded that the multi-population method is able to deal effectively with various DOPs and has the ability to maintain population diversity. It is also able to help the search in locating a new area through a divide and merge process and information exchange. The authors also highlighted several weaknesses of their method that relate to the number of sub-populations, the distribution of solutions and the reaction to problem changes.

Existing works on DOPs demonstrate that employing multi-population methods are the most effective method in maintaining population diversity. The features that make the multi-population methodologies popular are [3]: i) it divides the population into sub-populations, where the overall population diversity can be maintained since different populations can be located in different areas of the problem landscape, ii) it has the ability to search different areas simultaneously, enabling it to track the movement of the optimum, and iii) various single population-based algorithms can be integrated within multi-population methods.

Although multi-population methods have shown success when applied to DOPs, most of them use a number of sub-populations and the population diversity is maintained only through the sub-population distribution [3]. The number of sub-populations has a crucial impact on algorithm performance as it relates to the difficulty of the problem, which is not known in advance, and changes during the search. In addition, the solutions in the sub-populations may not be diverse enough as some methods are only concerned with how to divide the population into sub-populations, rather than focussing on diversification. To address these issues, this work proposes an adaptive multi-population artificial bee colony (ABC) algorithm for the DOP. The proposed ABC utilises a clearing scheme to remove redundant solutions in order to maintain diversity and enhance the exploration process. To efficiently track the landscape changes, the proposed ABC algorithm adaptively updates the number of sub-populations based on the problem change strength.

In this paper, the key objectives are:

i. To propose an artificial bee colony algorithm that utilises a multi-population and a population clearing scheme to efficiently solve the dynamic optimisation problem.
ii. To propose an adaptive multi-population algorithm that updates the number of the sub-populations based on the problem change strength.
iii. To test the performance of the proposed algorithm on dynamic optimisation problems using different scenarios and compare the results with other methodologies.

We used the moving peaks benchmark DOP with a different number of peaks to evaluate the effectiveness of the proposed ABC. Results demonstrate that the proposed ABC performs better than a basic ABC on all tested scenarios. Compared to the state of the art method, the proposed ABC produces very good results for many instances.

2. The proposed algorithm

This section presents the basic artificial bee colony algorithm, as well as our proposed adaptive multi-population algorithm.

2.1 Basic artificial bee colony algorithm

The Artificial Bee Colony (ABC) algorithm is a simple, yet efficient, nature inspired algorithm for addressing numerical optimization problems. It was proposed in [23] as a nature inspired swarm intelligence algorithm based on the observation of bee foraging behaviour. In ABC, there are a set of food sources and a set of bees. The quality of the food sources is based on the amount of nectar they contain. Bees search and collaborate with each other, seeking better food sources. To address an optimization problem using ABC, food sources represent the population of solutions for a given problem and bees are categorised into three types: scout, employee and onlooker bees. The amount of nectar corresponds to the quality (objective function) of the problem being addressed. The three types of bees work together in an iterative manner to improve the quality of the population of solutions (food sources). The pseudo-code of a basic ABC is shown in Algorithm 1 [24]. ABC first sets the main parameters, initializes the population of solutions and then evaluates them. Next, the main loop is executed in an attempt to solve the given optimisation problem by calling the employee bees, onlooker bees and scout bees until the stopping condition is satisfied.

```
Algorithm 1:  The pseudo-code of basic ABC
Step 1: Set the parameter values
Step 2: Initialize the population of solutions
Step 3: Evaluate the population of solutions
while termination condition is not met do
    Step 4: Employed Bees step
    Step 5: Onlooker Bees step
    Step 6: Scout Bees step
end while
```

The basic ABC has the following steps:

Step 1- Set ABC parameters. In this step the main parameters of ABC are initialized. These include: the maximum number of iterations (*MaxIt*) which represents the stopping condition of ABC, the number of solutions or population size (*Ps*) which represent how many solutions will be generated, the total number of bees (*Sbees*) which is set to be twice the size of *Ps*, where half of them are employee bees and the other half are onlooker bees, the limit parameter (*Lit*), which is used to determine if the solution should be replaced by a new one.

Step 2- Initialise the population of solutions. A set of solutions with size equal to *Ps* are randomly generated as follows:

$$x_{i,j} = x_{i,j}^{\min} + Rand[0,1](x_{i,j}^{\max} - x_{i,j}^{\min}) \quad (1)$$

where i is the index of the solution, j is the current decision variable, *Rand* [0,1] generates a random number between zero and one and $x_{i,j}^{\min}$ and $x_{i,j}^{\max}$ are the lower and upper bonds for the j^{th} decision variable.

Step 3- Evaluate the population of solutions. The fitness (quality) of the generated solutions are calculated using the objective function. The objective function is problem dependent. The objective function used in this work is shown in Section 3.2.

Step 4- Employed bees. Each employee bee is sent to one food source (solution). Its main role is to explore the neighbourhood of the current solution, seeking an improving solution. A neighbourhood solution, *v*, is created by modifying the i^{th} solution, *x*, as follows:

$$v_{i,j} = x_{i,j} + \Phi_{i,j}(x_{i,j} - x_{k,j}) \qquad (2)$$

where *k* is a randomly selected solution from *Ps* and *Φ* is a random number between [-1, 1]. The generated neighbourhood solution will be replaced with current solution if it has better fitness.

Step 5- Onlooker bees. Onlooker bees seek to improve the current population of solutions by exploring their neighbourhood using Equation (2), the same as the employee bee. The difference is that onlooker bees select the solutions probabilistically based on their fitness values as follows:

$$p_i = \frac{fitness_i}{\sum_{j=1}^{Ps} fitnessj} \qquad (3)$$

That is, the solution with the higher fitness has a higher chance of being selected (i.e. roulette wheel selection). Onlooker bees use a greedy selection mechanism, where the better solution in terms of fitness is selected.

Step 6- Scout bees. This step is activated if both employed and onlooker bees cannot improve the current solution for a number of consecutive iterations defined by the limit parameter, *Lit*. This indicates that the current solution is not good enough to search its neighbourhood and it should be discarded. In this case, the scout bee will generate a new solution using Equation (1) to replace the discarded one. This can help ABC to escape from a local optimum and explore a different area of the search space.

2.2 The proposed artificial bee colony algorithm

Existing works on DOPs have demonstrated that multi-population methods are state of the art, in that they outperform other methods on many scenarios. However, although multi-population methods have achieved success in solving DOPs, most of them use a fixed number of sub-populations and the population diversity is maintained through the sub-population distribution. To address these issues, this work proposes an adaptive population ABC (denoted as Multi-pop-ABC). In Multi-pop-ABC, three major modifications are added to the basic ABC. These are:

i. Multi-population method. To deal with DOP, the proposed ABC uses a multi-population method to divide the population into several sub-populations. By using a multi-population method, the solutions are scattered over the search space instead of focusing on a specific area. Thus the algorithm can generate high quality solutions and track the problem changes.

ii. Adaptive scheme. To track the landscape changes that occur during the search process, the proposed Multi-pop-ABC updates the number of sub-populations based on the strength of the problem change. That is the number of sub-populations is either decreased or increased during the search process. By using the proposed adaptive method, the number of sub-populations can be changed adaptively based on the strength of the environment changes, which helps the search track the optimum solution and also improves the diversification and exploration processes.

iii. Population clearing scheme. To ensure that the solutions are diverse enough, a population clearing scheme is called when a change is detected to delete redundant solutions and replace them with new solutions. This scheme removes redundant solutions in order to maintain diversity and enhance the exploration process.

The flowchart of the proposed Multi-pop-ABC for DOPs is shown in Figure 1. It starts by setting the parameter values. It creates the population of solutions and then evaluates them. Next, the population of solutions is divided into *m* sub-populations. Each sub-population utilises an ABC algorithm. If a change in the problem is detected, the algorithm calculates the change strength to update the sub-population size and checks the stopping condition. If the specified stopping condition (we set this as a maximum number of fitness evaluations) has been reached, the algorithm terminates and the best solution is returned. Otherwise, the algorithm merges all the sub-populations, updates the population, runs the clearing method, re-divides the population into *m* sub-populations and starts a new iteration.

The main steps are described in further detail below:

- **Step 1: Set parameters**. The main parameters of Multi-pop-ABC are initialised. The algorithm has five parameters. Four of them are the same as the basic ABC. These are: the maximum number of iterations (*MaxIt*), population size (*Ps*), number of bees (*Sbees*), and the limit parameter (*Lit*). The fifth parameter is the sub-population size (*m*), which represents the number of sub-populations (*Ps*/*m*). Initially, *m*=2 and during the search process, it is either decreased or increased.

1- **Step 2: Initialise the population of solutions**. Same as Step 2 in the basic ABC, Section 2.1.

2- **Step 3: Evaluate the population of solutions**. Same as Step 3 in the basic ABC, Section 2.1.

3- **Step 4: Divide the population.** The population of solutions is divided into *m* sub-populations (*Ps*/*m*). Each sub-population is assigned to explore a different area of the search space. These sub-populations interact with each other through merging and re-dividing every time a change in the environment is detected. Each solution in the population is randomly assigned to a sub-population. The number of sub-populations *m* is either increased or decreased based on the environment change strength. The initial value of *m* is set to two (*m*=2) and it is updated during the search.

4- **Step 5: Assign ABC to each sub-population.** Each sub-population has its own ABC algorithm. Each ABC executes all the steps presented in Section 2.1. It starts with a population of solutions and iteratively calls the following until the stopping condition is satisfied (the algorithm stops when a change in the environment is detected):

i. **Employee bees**. Same as Step 4 in the basic ABC, Section 2.1.

ii. **Onlooker bees**. Same as Step 5 in the basic ABC, Section 2.1.

iii. **Scout bees**. Same as Step 6 in the basic ABC, Section 2.1.

5- **Step 6: Check the change strength.** This step is activated when a change in the environment is detected. Its main role is to update the number of sub-populations based on the environment change strength. It first calculates the objective function of the best solution before and after the environment change as follows:

$$Cs = f(best\_before) - f(best\_after) \tag{4}$$

where Cs is the change strength, $f(best\_before)$ is the quality of the best solution before the environment change and $f(best\_after)$ is the quality of the best solution after the environment change. If the Cs is less than the defined threshold (Tv) and m is greater than 2, the number of sub-populations m is decreased as the algorithm needs to be more exploitive than explorative (m=m-1). Otherwise, m is increased by one with the aim of increasing the exploration aspect of the search (m=m+1). It should be noted that when m is an odd number, the *extra* solution is randomly assigned to one of the sub-populations.

6- **Step 7: Check the stopping condition.** This step checks the termination criterion of the search process. In this work, it is set as a maximum number of fitness evaluations in line with previous works. If the specified stopping condition is reached, the search process stops and returns the best solution. Otherwise, the algorithm performs the following processes:

i. **Population clearing scheme:** This scheme calculates the similarity between solutions in the population. The similarity is calculated by using a matching algorithm, which matches each pair of solutions in terms of phenotype. Two solutions are similar if they have the same values in all the cells of both solutions. If two or more solutions are similar, these solutions are deleted and replaced with randomly generated ones.

ii. **Population update**: All sub-populations are merged to form one population.

iii. **Re-divide the population:** The population is re-divided into *m* sub-populations and the algorithm continues by starting the process at step 1 with a new generation.

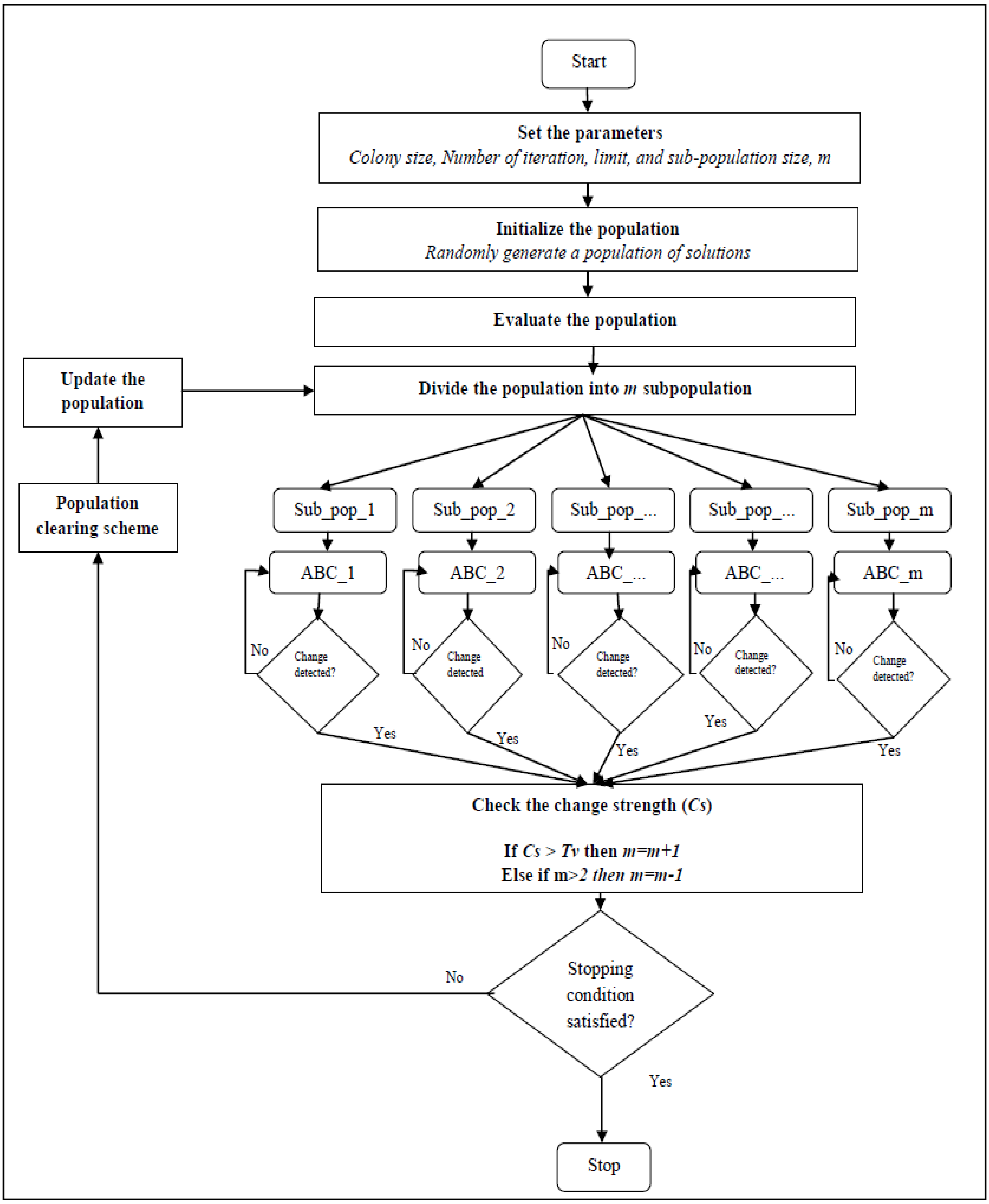

Figure 1. The proposed Multi-pop-ABC

3. Experimental Setup

This section discusses the Moving Peak Benchmark (MPB), evaluation metric and the parameter settings.

3.1 The Moving Peak Benchmark

The moving peak benchmark (MPB) is a maximization dynamic continuous optimization problem proposed by [9], [25], and has been commonly used as a testbed for the performance of optimisation algorithms. MPB consists of a set of peaks that move over the problem landscape. It takes the given solution as an input and returns the value of the highest peak. The returned value represents the quality of this solution. MPB can be mathematically expressed as follows:

$$F(\vec{x},t) = \max_{i=1,\dots,p} \left(\frac{H_i(t)}{1 + W_i(t)\sum_{j=1}^{D}(x_j(t) - X_{ij}(t))^2} \right) \quad (5)$$

where $F(x, t)$ is the quality of solution x at time t, p is the number of peaks, D is the problem dimension (number of decision variables where each variable has an upper and lower boundary (DB)), $H_i(t)$ is the height of peak i, $W_i(t)$ is the width of peak i, and X_{ij} is the j^{th} element of the location of peak i. Note that Equation (5) is a stationary optimization problem. Thus, to change it to a dynamic problem, MPB randomly shifts the position of all peaks by vector $\vec{v}_i$ of a distance s (s is also known as the shift length that determines the severity degree) as follows:

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)) \quad (6)$$

where $\vec{r}$ is a random vector, λ is the correlation between consecutive movements of a single peak that takes either "0" if the movement of peaks are completely uncorrelated or "1" if they move in the same direction. To make a fair comparison with existing algorithms, in this paper, we used $\lambda=0$ [6]. The change of height and width of a peak in a given solution can be mathematically expressed as follows:

$$H_i(t) = H_i(t-1) + height\_severity * \sigma \quad (7)$$

$$W_i(t) = W_i(t-1) + width\_severity * \sigma \quad (8)$$

where *height_severity* and *width_severity* are calculated based on the problem severity. σ is a normally distributed random number between 0 and 1. Then, the change of a solution x is given as follows:

$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t) \qquad (9)$$

The change frequency (*cf*) occurs every 5,000 fitness evaluations [9]. The parameter values of all MPBs that have been used in our experiments are shown in Table 1 [25].

Table 1 MPB parameter values

Parameters	*Description*	*Value*
p	Number of peaks	1–200
cf	Change frequency	5000
height_severity	Height severity	7.0
width_severity	Width severity	1.0
Peak shape	Peak shape	Cone
s	Shift length	1.0
D	Number of dimensions	5
λ	Correlation coefficient	0
DB	Each dimension boundaries	[0,100]
H	Peak height	[30.0,70.0]
W	Peak width	[1,12]

3.2 Evaluation Metric

To fairly compare the proposed ABC with existing algorithms, we use the same evaluation metric known as the offline error as suggested by [25]. This has also been used by other researchers. The offline error is calculated as follows:

$$off = \frac{1}{g}\sum_{i=1}^{g}\Omega_i \qquad (10)$$

where g is the number of generations and **Ω** is the best performance since the last change at i^{th} fitness evaluation.

3.3 Parameter Settings

The parameter values of our Multi-pop-ABC are set by carrying out a set of initial experiments, with the exception of the stopping condition which was set to be the same as the compared algorithms (50,000 fitness evaluations). For each parameter, we tested various values and the best values were selected. This is achieved by varying the value of one parameter while fixing others. We have selected two scenarios of MPB for the parameter tunning process: 50 peaks and 200 peaks. The proposed ABC has three parameters:

population size (*Ps*), limit (*Lit*) and the change strength threshold (*Tv*). First, we fixed *Lit* to 30, *Tv* to 0.09 and changed *Ps*. Table 2 shows the offline error of various *Ps* values for 50 and 200 peaks. The best result is highlighted in bold. Next, we fixed *Ps* to 60, *Tv* to 0.09 and changed *Lit* as shown in Table 3. Finally, we fixed *Ps* to 60, *Lit* to 30 and changed *Tv* as shown in Table 4. The parameter settings of the proposed ABC that were used across all scenarios are presented in Table 5.

Table 2 The value of *Ps* parameter

Ps value	50 peaks	200 peaks
20	0.95669	2.70215
40	0.319632	1.8935
60	0.5810	**0.34865**
80	0.576911	1.15134

Table 3 The value of *Lit* parameter

Lit value	50 peaks	200 peaks
10	1.03474	1.18977
20	1.27535	1.70215
30	1.29851	**0.24824**
40	1.841891	1.28967

Table 4 The value of *Tv* parameter

Tv value	50 peaks	200 peaks
0.03	0.95669	1.89663
0.05	0.96573	**1.08053**
0.07	0.89978	1.37518
0.09	1.23491	1.77956

Table 5 The parameter settings of the proposed ABC

#	Parameter	Value
1-	Maximum number of iterations (*MaxIt*)	50,000 fitness evaluations
2-	Population size (*Ps*)	60
3-	Limit parameter (*Lit*)	30
4-	Change strength threshold (*Tv*)	0.05

4. Results

We carried out three set of experiments. In first one, we compare the results of Multi-pop-ABC with the basic ABC. In second one, the results obtained by Multi-pop-ABC are compared with state of the art methods. In the third experiment, the results of Multi-pop-ABC on well-known test functions are compared with state of the art methods.

4.1 Results comparison of Multi-pop-ABC and the basic ABC

This section aims to verify the effectiveness of the additional components that we have added to the basic ABC. Specifically, the objective is to investigate the impact of the proposed enhancements on the performance of the basic ABC when dealing with DOPs. Four different algorithms were derived as follows:

- Multi-pop-ABC: the proposed ABC that utilises the adaptive multi-population and population clearing scheme
- Multi-pop-ABC_1: same as above but without the population clearing scheme
- Multi-pop-ABC_2: same as above but uses a fixed number of sub-populations and without the population clearing scheme. The sub-populations were fixed to be the same as [26]
- ABC: basic ABC algorithm.

The computational comparisons of Multi-pop-ABC, Multi-pop-ABC_1, Multi-pop-ABC_2 and basic ABC are presented in Table 6. The comparison is in terms of the offline error, ± standard error for each number of peaks. The best results are highlighted in bold. The results clearly show the good performance of Multi-pop-ABC when compared to Multi-pop-ABC_1, Multi-pop-ABC_2 and basic ABC. Indeed, Multi-pop-ABC outperformed Multi-pop-ABC_1, Multi-pop-ABC_2 and basic ABC on both the offline error and the standard error on all tested scenarios. The results demonstrate that the enhancements we made to the basic ABC improve the algorithmic performance.

Table 6 Results of the Multi-pop-ABC, Multi-pop-ABC_1, Multi-pop-ABC_2 and basic ABC

	Number of Peaks										
Algorithm	1	2	5	7	10	20	30	40	50	100	200
Multi-pop-ABC	**0.14 ±0.00**	**0.12 ±0.00**	**0.20 ±0.00**	**0.38 ±0.01**	**0.22 ±0.01**	**0.35 ±0.00**	**0.46 ±0.00**	**0.52 ±0.01**	**0.44 ±0.01**	**0.52 ±0.00**	**0.93 ±0.00**
Multi-pop-ABC_1	1.81 ±0.18	1.42 ±0.32	1.11 ±0.13	1.01 ±0.22	1.57 ±0.12	1.43 ±0.15	1.45 ±0.14	1.62 ±0.10	1.21 ±0.21	1.73 ±0.11	1.22 ±0.10
Multi-pop-ABC_2	1.12 ±0.18	1.21 ±0.41	1.61 ±0.10	1.65 ±0.11	1.71 ±0.15	1.11 ±0.14	1.72 ±0.19	1.42 ±0.13	1.62 ±0.18	1.41 ±0.12	1.42 ±0.12
Basic ABC	5.88 ±2.48	5.52 ±4.31	4.12 ±3.7	4.5 ±2.3	5.2 ±3.16	6.3 ±3.51	3.38 ±4.32	7.14 ±3.60	6.21 ±2.01	6.97 ±2.11	7.03 ±3.44

Note: Values in bold font indicate the best results.

To further verify the results, we conducted a comparison between Multi-pop-ABC and each method separately. We used a Wilcoxon statistical test with a confidence level of 0.05. The *p*-values of Multi-pop-ABC against Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC for each scenario is presented in Table 7. A value less than 0.05 indicates Multi-pop-ABC is superior (i.e. statistically different). As can be seen from Table 7, Multi-pop-ABC is superior to Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC on 9 out of 11 tested scenarios ($p < 0.05$). The table also shows than on two scenarios (1 peak and 2 peaks) Multi-pop-ABC is not superior to Multi-pop-ABC$_1$ and Multi-pop-ABC$_2$. This can be attributed to the fact that these two scenarios are relatively easy to solve and thus all methods produce very good solutions. The results of the statistical test also demonstrate that the proposed enhancements have a positive impact and improve the search process.

Table 7 *p*-values of the of Multi-pop-ABC against other methods

	Number of Peaks										
Multi-pop-ABC vs.	**1**	**2**	**5**	**7**	**10**	**20**	**30**	**40**	**50**	**100**	**200**
Multi-pop-ABC$_1$	0.06	0.08	0.04	0.03	0.00	0.00	0.00	0.01	0.01	0.00	0.00
Multi-pop-ABC$_2$	0.07	0.06	0.02	0.07	0.06	0.00	0.04	0.01	0. 02	0.00	0.00
Basic ABC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Note: Values less than 0.05 indicate that Multi-pop-ABC is better than the compared methods.

4.2 Comparison with state of the art methods

There are numerous methods that use different schemes to handle diversification, and which have been tested on MPB. In this section, we evaluate the performance of our algorithm by comparing it with several recently proposed algorithms taken from the scientific literature. The algorithms are:

- Multiswarms, exclusion, and anti-convergence in dynamic environments (mCPSO) [27].
- Multiswarms, exclusion, and anti-convergence in dynamic environments (mQSO) [27]
- Multiswarms, exclusion, and anti-convergence in dynamic environments (mCPSO*) [27]

- Multiswarms, exclusion, and anti-convergence in dynamic environments (mQSO*) [27].
- Competitive population evaluation in a differential evolution algorithm for dynamic environments (CDE) [28].
- Differential evolution for dynamic environments with unknown numbers of optima (DynPopDE) [29].
- Dynamic function optimization with hybridized extremal dynamics (EO + HJ) [30]
- A competitive clustering particle swarm optimizer for dynamic optimization problems (CCPSO) [31].
- A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems (CHPSO(ES-NDS)) [32].

To ensure a fair comparison, we used the same stopping condition (50,000 fitness evaluations), the same change frequency (every 5,000 fitness evaluations) and the same evaluation metric (Offline error). We also used 11 MPB instances with a different number of peaks ranging between 1 to 200 peaks.

The results of Multi-pop-ABC and the compared algorithms are presented in Table 8. The results in the table are in terms of offline error, ± standard error and computational times for each number of peaks. In the table, the symbol '-' indicates that the scenario has not been tested. We indicate in bold the best obtained results. From Table 8, it can be seen that Multi-pop-ABC is superior to the other algorithms in most of the cases in terms of offline error. In particular, Multi-pop-ABC obtained new best results for 9 out of 11 tested MPB instances. Multi-pop-ABC was inferior on only two MPB instances: 1 peak and 2 peaks. Nevertheless, the results of Multi-pop-ABC for these two scenarios are very competitive, where it obtained the second best results. In terms of the standard error, Multi-pop-ABC produced a better standard error for 6 scenarios, being similar on 5 scenarios out of the 11 tested.

Table 8 Results of Multi-pop-ABC compared to the state of the art methods

	Number of Peaks										
Algorithm	1	2	5	7	10	20	30	40	50	100	200
Multi-pop-ABC	**0.14 ±0.00** 8.11	**0.12 ±0.00** 9.10	**0.20 ±0.00** 10.20	**0.38 ±0.01** 10.63	**0.22 ±0.01** 11.12	**0.35 ±0.00** 13.75	**0.46 ±0.00** 15.13	**0.52 ±0.01** 17.17	**0.44 ±0.01** 20.23	**0.52 ±0.00** 28.64	**0.93 ±0.00** 56.48
mCPSO	4.93 ±0.17	3.36 ±0.26	2.07 ±0.08	2.11 ±0.11	2.08 ±0.07	2.64 ±0.07	2.63 ±0.08	2.67 ±0.07	2.65 ±0.06	2.49 ±0.04	2.44 ±0.04

mQSO	5.07 ±0.17	3.47 ±0.23	1.81 ±0.07	1.77 ±0.07	1.80 ±0.06	2.42 ±0.07	2.48 ±0.07	2.55 ±0.07	2.50 ±0.06	2.36 ±0.04	2.26 ±0.03
mCPSO*	4.93 ±0.17	3.36 ±0.26	2.07 ±0.11	2.11 ±0.11	2.05 ±0.07	2.95 ±0.08	3.38 ±0.11	3.69 ±0.11	3.68 ±0.11	4.07 ±0.09	3.97 ±0.08
mQSO*	5.07 ±0.17	3.47 ±0.23	1.81 ±0.07	1.77 ±0.07	1.75 ±0.06	2.74 ±0.07	3.27 ±0.11	3.60 ±0.08	3.65 ±0.11	3.93 ±0.08	3.86 ±0.07
CDE	-	-	-	-	0.92 ±0.07	-	-	-	-	-	-
DynPopDE	-	-	1.03 ±0.13	-	1.39 ±0.07	-	-	-	2.10 ±0.06	2.34 ±0.05	2.44 ±0.05
EO + HJ	7.08 ±1.99	-	-	-	0.25 ±0.10	0.39 ±0.10	0.49 ±0.09	0.56 ±0.09	0.58 ±0.09	0.66 ±0.07	-
CCPSO	0.09 ±0.00	0.09 ±0.00	0.25 ±0.01	0.53 ±0.03	0.75 ±0.06	1.21 ±0.08	1.40 ±0.07	1.47 ±0.08	1.50 ±0.09	1.76 ±0.09	-
CHPSO(ES-NDS)	0.19 ± 0.00	-	0.44 ±0.02	-	0.64 ±0.02	0.91 ±0.01	0.99 ±0.01	1.02 ±0.01	1.03 ±0.01	1.04 ±0.01	1.01 ±0.00

Note: Values in bold font indicate the best results.

To further verify the effectiveness of the proposed Multi-pop-ABC, we statistically compare it with other methods. We followed the procedure described in [33]. First, Friedman test and Iman and Davenport statistical tests with 0.05 confidence levels are carried out to detect if there is a difference between the results of Multi-pop-ABC and other methods. It should be noted that only those methods that were tested on all scenarios were considered for this test. Both the Friedman test and Iman and Davenport tests returned *p*-values (0.000009 and 0.000000009061) less than 0.05 indicating the compared results are statistically different. We next conducted a Friedman test to obtain rankings, and Holm and Hochberg post-hoc tests. The ranking value for each method obtained by a Friedman test is presented in Table 9 (the lower the better), where Multi-pop-ABC obtained the first rank followed by mQSO second rank, mCPSO third rank, mQSO* fourth rank and mCPSO* fifth rank. Consequently, Multi-pop-ABC will be the controlling method for the Holm and Hochberg post-hoc tests. The *p*-values of Holm and Hochberg tests are shown in Table 10. From the table, one can see that Multi-pop-ABC is statistically better than the compared methods on both Holm and Hochberg tests in which all the obtained *p*-values are less than 0.05.

Table 9 The average ranking of Friedman test

#	*Algorithm*	*Ranking*
1	Multi-pop-ABC	1
2	mQSO	2.6364
3	mCPSO	3.3636
4	mQSO*	3.6364
5	mCPSO*	4.3636

Table 10 The adjusted *p*-value of the compared methods

#	*Algorithm*	*Unadjusted P*	*P Holm*	*P Hochberg*

1	mCPSO*	0.000001	0.000002	0.000002
2	mQSO*	0.000092	0.000276	0.000276
3	mCPSO	0.000455	0.00091	0.00091
4	mQSO	0.015219	0.015219	0.015219

The above results reveal that, in most of the tested scenarios, the proposed Multi-pop-ABC is better than the compared methods. These results are supported by statistical tests.

We hypothesise that several key features contribute to the high performance of the proposed algorithm (Multi-pop-ABC) on the dynamic problem. These can be summarised as follows:

- Multi-population: This feature is beneficial for maintaining the diversity of solutions in the population during the search process.
- Adaptive number of sub-populations: This feature helps the algorithm in changing the solution distribution over the search landscape to get better diversification and intensification based on the problem change strength.
- Population clearing scheme: This feature helps avoid having similar solutions within the population in order to further add to the diversification.

4.3 Comparison with state-of-the-art approaches on test functions

In this section, we evaluate our proposed algorithm based on other well-known ten test functions. The tested functions are widely used by researchers [34-37]. These functions are:

$$f_1\left(\vec{x}\right)=\sum_{i=1}^{n} x_i^2 \qquad [-100, 100]^n$$

$$f_2\left(\vec{x}\right)=\sum_{i=1}^{n} |x_i| + \prod_{i}^{n} = 1|x_i| \qquad [-10, 10]^n$$

$$f_3\left(\vec{x}\right)=\sum_{i=1}^{n} \left(\sum_{i}^{i} = 1^{x_j}\right)^2 \qquad [-100, 100]^n$$

$$f_4\left(\vec{x}\right)= \max_i \{|x_i|, 1 \le i \le n\} \qquad [-100, 100]^n$$

$$f_5\left(\vec{x}\right)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right] \qquad [-30, 30]^n$$

$$f_6\left(\vec{x}\right)=\sum_{i=1}^{n} i x_i^4 + random(0,1) \qquad [-1.28, 1.28]^n$$

$$f_7\left(\vec{x}\right)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{|x_i|}\right) \qquad [-500, 500]^n$$

$$f_8\left(\vec{x}\right)=\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right) \qquad [-5.12, 5.12]^n$$

$$f_9\left(\vec{x}\right)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e \qquad [-32, 32]^n$$

$$f_{10}\left(\vec{x}\right)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \qquad [-600, 600]^n$$

For every benchmark function, respectively assume the dimension as 30, 50 and 100. The results in Tables 11, 12 and 13 demonstrate that Multi-pop-ABC performs better than the compared ABC, PS-ABC and PS-ABCII algorithms [34-36] in terms of both mean and standard deviation (SD). Note that the best results are highlighted in bold. The presented results indicate that the Multi-pop-ABC outperforms other methods over all test functions.

Table 11 Mean, the standard deviation (SD) of functions with 30 dimensions.

F	***ABC***			***PS-ABC***		***PS-ABCII***		***LWGSODE***		**CFOA**		***Multi-pop-ABC***	
	Dim	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**Mean**	**Mean**	**SD**	**Mean**	**SD**	**SD**	**SD**
f1	30	3.3955×10^{-9}	4.5376×10^{-9}	**0**	0	**0**	0	1.68×10^{-7}	1.63×10^{-7}	1×10^{-309}	-	**0**	0
f2	30	$5.1029 \times 10\text{-}^{6}$	1.8417×10^{-6}	**0**	0	**0**	0	1.10×10^{-3}	4.09×10^{-4}	1×10^{-155}	-	**0**	0
f3	30	1.2598×10^{4}	2.9192×10^{3}	7.2696×10^{3}	1.4359×10^{3}	4.0756×10^{4}	8.1760×10^{3}	-	-	-	-	$\mathbf{6.4361 \times 10^{2}}$	1.9164×10^{2}
f4	30	2.4044×10^{1}	3.3935	**0**	0	**0**	0	-	-	-	-	**0**	0
f5	30	3.2873	3.4035	1.4048	2.7168	2.8408×10^{1}	0.1154	2.60×10^{1}	2.70×10^{-2}	-	-	**1.0196**	1.9217
f6	30	1.5788×10^{-1}	3.6701×10^{-1}	1.8545×10^{-2}	5.3198×10^{-3}	5.5447×10^{-4}	1.2352×10^{-3}	-	-	1×10^{-309}	-	$\mathbf{3.2124 \times 10^{-4}}$	1.0172×10^{-3}
f7	30	-12185.9	1.4299×10^{2}	-12549.7	4.4891×10^{1}	-12088.9	1.8715×10^{2}	-	-	-	-	**-14847.9**	3.5787×10^{1}
f8	30	4.0160×10^{-1}	6.2228×10^{-1}	**0**	0	**0**	0	3.85×10^{-5}	3.75×10^{-5}	-	-	**0**	0
f9	30	2.4076×10^{-5}	1.2439×10^{-5}	8.8817×10^{-16}	0	8.8817×10^{-16}	0	2.99×10^{-4}	1.86×10^{-4}	1×10^{-308}	-	**0**	0
10	30	1.4335×10^{-3}	4.0152×10^{-3}	**0**	0	**0**	0	-	-	1×10^{-309}	-	**0**	0

Table 12 Mean, the standard deviation (SD) of functions with 50 dimension.

F	ABC			PS-ABC		PS-ABCII		Multi-pop-ABC	
	Dim	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
f1	50	1.1483×10^{-5}	1.6272×10^{-5}	**0**	0	**0**	0	**0**	0
f2	50	2.8511×10^{-3}	1.3944×10^{-3}	**0**	0	**0**	0	**0**	0
f3	50	4.6422×10^{4}	6.9821×10^{3}	3.0638×10^{3}	3.4739×10^{3}	1.2539×10^{5}	2.1047×10^{4}	$\mathbf{2.1041 \times 10^{3}}$	2.0893×10^{3}
f4	50	5.6020×10^{1}	5.1905	1.8782×10^{1}	5.7908	**0**	0	**0**	0
f5	50	3.7224×10^{1}	3.6453×10^{1}	3.1451×10^{1}	2.9224×10^{1}	4.8504×10^{1}	1.3535×10^{-1}	$\mathbf{2.2310 \times 10^{1}}$	2.1102×10^{1}
f6	50	4.2726×10^{-1}	8.2393×10^{-2}	5.7802×10^{-2}	1.6469×10^{-2}	5.4388×10^{-4}	6.4470×10^{-4}	2.1847×10^{-4}	3.2711×10^{-4}
f7	50	-19359.1	3.1097×10^{2}	-20893.4	7.9224×10^{1}	-19414.1	3.3738×10^{2}	**-26893.4**	7.8394×10^{1}
f8	50	8.1857	2.4195	**0**	0	**0**	0	**0**	0
f9	50	4.0637×10^{-2}	3.2467×10^{-2}	8.8817×10^{-16}	0	8.8817×10^{-16}	0	**0**	0
10	50	9.9977×10^{-3}	1.1718×10^{-2}	**0**	0	**0**	0	**0**	0

Table 13 Mean, the standard deviation (SD) of functions with 100 dimension.

F	ABC			PS-ABC		PS-ABCII		Multi-pop-ABC	
	Dim	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
f1	100	4.9461×10^{-3}	1.1389×10^{-2}	8.3417×10^{-47}	4.5689×10^{-46}	**0**	0	**0**	0
f2	100	2.7814×10^{-1}	4.0035×10^{-1}	**0**	0	**0**	0	**0**	0
f3	100	1.8854×10^{5}	2.1886×10^{4}	1.3544×10^{5}	1.2851×10^{4}	5.4823×10^{5}	1.0256×10^{5}	$\mathbf{1.4211 \times 10^{4}}$	1.0937×10^{4}
f4	100	8.2376×10^{1}	3.0440	7.2160×10^{1}	4.0371	**0**	0	**0**	0
f5	100	3.3118×10^{2}	3.8309×10^{2}	2.0376×10^{2}	6.7028×10^{1}	9.8590×10^{1}	1.5702×10^{-1}	$\mathbf{4.1781 \times 10^{1}}$	1.2011×10^{-1}
f6	100	1.5950	3.2657×10^{-1}	2.2021×10^{-1}	4.1119×10^{-2}	1.6151×10^{-3}	3.2646×10^{-3}	$\mathbf{1.1260 \times 10^{-3}}$	2.9615×10^{-3}
f7	100	-34413.8	5.0878×10^{2}	-39976.6	3.3634×10^{2}	-37405.7	5.5665×10^{2}	-40182.4	2.6738×10^{2}
f8	100	8.5540×10^{1}	1.1018×10^{1}	**0**	0	**0**	0	**0**	0
f9	100	3.8186	3.6198×10^{-1}	2.3270×10^{-14}	1.2259×10^{-13}	8.8817×10^{-16}	0	0	0
10	100	1.4344×10^{-1}	1.3282×10^{-1}	1.6904×10^{-3}	6.4786×10^{-3}	**0**	0	**0**	0

5. Conclusion

This paper has presented a modified artificial bee colony algorithm for dynamic optimization problems. The aims of our modifications were to enhance the capability of the algorithm to efficiently deal with DOPs. We first integrated it with a multi-population method to scatter the solution over the search process so that they can search and track the optimum solution simultaneously. An adaptive multi-population was also proposed to adaptively change the number of sub-populations based on the problem change strength. In addition, a population clearing scheme was proposed to remove redundant solutions in the population. To evaluate the performance of the proposed algorithm, experimental tests were carried out using the moving peaks benchmark DOP, with a different number of peaks. Comparisons were carried out between the proposed algorithm, the basic ABC and state of the art methods. The results demonstrated that the proposed algorithm outperforms basic ABC on all tested scenarios. It also produced better results than the state of the art methods on many scenarios, indicating that the proposed algorithm is an effective method for the DOP.

Acknowledgements

This work was supported by the Ministry of Education, Malaysia (FRGS/1/2015/ICT02/UKM/01/2) and the Universiti Kebangsaan Malaysia (DIP-2012-15).

References

1. Jin, Y. and J. Branke, *Evolutionary optimization in uncertain environments-a survey*. IEEE Transactions on Evolutionary Computation, 2005. **9**(3): p. 303-317.
2. Nguyen, T.T., S. Yang, and J. Branke, *Evolutionary dynamic optimization: A survey of the state of the art*. Swarm and Evolutionary Computation, 2012. **6**: p. 1-24.
3. Li, C., T.T. Nguyen, M. Yang, S. Yang, and S. Zeng, *Multi-population methods in unconstrained continuous dynamic environments: The challenges*. Information Sciences, 2015. **296**: p. 95-118.
4. Hatzakis, I. and D. Wallace. *Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach*. in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006. p. 1201-1208. ACM.
5. Simões, A. and E. Costa. *Improving prediction in evolutionary algorithms for dynamic environments*. in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. 2009. p. 875-882. ACM.
6. Branke, J. and D.C. Mattfeld, *Anticipation and flexibility in dynamic scheduling*. International Journal of Production Research, 2005. **43**(15): p. 3103-3129.
7. Cruz, C., J.R. González, and D.A. Pelta, *Optimization in dynamic environments: a survey on problems, methods and measures*. Soft Computing, 2011. **15**(7): p. 1427-1448.
8. Branke, J. *Memory enhanced evolutionary algorithms for changing optimization problems*. in *In Congress on Evolutionary Computation CEC99*. 1999. 3:1875-1882.
9. Branke, J., *Evolutionary optimization in dynamic environments*. Vol. 3. 2012: Springer Science & Business Media.
10. Yang, S. *On the design of diploid genetic algorithms for problem optimization in dynamic environments*. in *IEEE Congress on Evolutionary Computation CEC 2006*. pp. 1362-1369.
11. Daneshyari, M. and G.G. Yen. *Dynamic optimization using cultural based PSO*. in *IEEE Congress on Evolutionary Computation CEC 2011* pp. 509-516.
12. Grefenstette, J.J. *Evolvability in dynamic fitness landscapes: A genetic algorithm approach*. in *IEEE Congress on Evolutionary Computation CEC 1999*. Vol. 3, pp.2031 -2038 1999.
13. Ursem, R.K. *Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments*. in *In GECCO, pp. 19-26*. 2000.
14. Branke, J., T. Kaußler, C. Schmidt, and H. Schmeck, *A multi-population approach to dynamic optimization problems*. Adaptive computing in design and manufacturing,2000:, 2000: p. 299-308.
15. Blackwell, T. and J. Branke, *Multi-swarm optimization in dynamic environments*. Applications of Evolutionary Computing, 2004: p. 489-500.

16. Mendes, R. and A.S. Mohais. *DynDE: a differential evolution for dynamic optimization problems*. in *IEEE Congress on Evolutionary Computation CEC 2005* vol. 3, pp. 2808-2815.
17. Li, C. and S. Yang. *Fast multi-swarm optimization for dynamic optimization problems*. in *Fourth International Conference on Natural Computation, 2008. ICNC'08*. . vol. 7, pp. 624-628. IEEE.
18. Yang, S. and C. Li, *A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments*. IEEE Transactions on Evolutionary Computation, , 2010. **14**(6): p. 959-974.
19. Turky, A.M. and S. Abdullah, *A multi-population electromagnetic algorithm for dynamic optimisation problems*. Applied Soft Computing, 2014. **22**(1): p. 474-482.
20. Turky, A.M. and S. Abdullah, *A multi-population harmony search algorithm with external archive for dynamic optimization problems*. Information Sciences, 2014. **272**(1): p. 84-95.
21. Sharifi, A., J.K. Kordestani, M. Mahdaviani, and M.R. Meybodi, *A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems*. Applied Soft Computing, 2015. **32**(1): p. 432-448.
22. Li, C., T.T. Nguyen, M. Yang, S. Yang, and S. Zeng, *Multi-population methods in unconstrained continuous dynamic environments: The challenges*. Information Sciences, 2015. **296**(1): p. 95-118.
23. Karaboga, D., *An idea based on honey bee swarm for numerical optimization*. 2005, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.
24. Karaboga, D. and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*. Journal of global optimization, 2007. **39**(3): p. 459-471.
25. Branke, J. and H. Schmeck, *Designing evolutionary algorithms for dynamic optimization problems*, in *Advances in evolutionary computing*. 2003, Springer. p. 239-262.
26. Branke, J., T. Kaußler, C. Smidt, and H. Schmeck, *A multi-population approach to dynamic optimization problems*, in *Evolutionary Design and Manufacture*. 2000, Springer. p. 299-307.
27. Blackwell, T. and J. Branke, *Multiswarms, exclusion, and anti-convergence in dynamic environments*. IEEE Transactions on Evolutionary Computation, 2006. **10**(4): p. 459-472.
28. Du Plessis, M.C. and A.P. Engelbrecht, *Using competitive population evaluation in a differential evolution algorithm for dynamic environments*. European Journal of Operational Research, 2012. **218**(1): p. 7-20.
29. Du Plessis, M.C. and A.P. Engelbrecht, *Differential evolution for dynamic environments with unknown numbers of optima*. Journal of Global Optimization, 2013. **55**(1): p. 73-99.
30. Moser, I. and R. Chiong, *Dynamic function optimisation with hybridised extremal dynamics*. Memetic Computing, 2010. **2**(2): p. 137-148.
31. Nickabadi, A., M.M. Ebadzadeh, and R. Safabakhsh, *A competitive clustering particle swarm optimizer for dynamic optimization problems*. Swarm Intelligence, 2012. **6**(3): p. 177-206.
32. Sharifi, A., J.K. Kordestani, M. Mahdaviani, and M.R. Meybodi, *A novel hybrid adaptive collaborative approach based on particle swarm optimization and local*

search for dynamic optimization problems. Applied Soft Computing, 2015. **32**: p. 432-448.

33. García, S., A. Fernández, J. Luengo, and F. Herrera, *Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power*. Information Sciences, 2010. **180**(10): p. 2044-2064.
34. Li, G., P. Niu, Y. Ma, H. Wang, and W. Zhang, *Tuning extreme learning machine by an improved artificial bee colony to model and optimize the boiler efficiency*. Knowledge-Based Systems, 2014. **67**: p. 278-289.
35. Cui, H., J. Feng, J. Guo, and T. Wang, *A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction*. Knowledge-Based Systems, 2015. **88**: p. 195-209.
36. Mitić, M., N. Vuković, M. Petrović, and Z. Miljković, *Chaotic fruit fly optimization algorithm*. Knowledge-Based Systems, 2015. **89**: p. 446-458.
37. Yeh, W.-C., *An improved simplified swarm optimization*. Knowledge-Based Systems, 2015. **82**: p. 60-69.

An Adaptive Multi-population Artificial Bee Colony Algorithm for Dynamic Optimisation Problems

Shams K. Nseef[1], Salwani Abdullah[1], Ayad Turky[2] and Graham Kendall[3,4]

[1]Data Mining and Optimisation Research Group, Centre for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia.
E-mail: shams.shamosa91@gmail.com; salwani@ukm.edu.my
[2]Swinburne University of Technology, Melbourne, Victoria, Australia
E-mail: aturky@swin.edu.au
[3]University of Nottingham Malaysia Campus, Semenyih, Malaysia
[4]ASAP Research Group, University of Nottingham, Nottingham, United Kingdom
Email: Graham.Kendall@nottingham.edu.my; Graham.Kendall@nottingham.ac.uk

Abstract

Recently, interest in solving real-world problems that change over the time, so called dynamic optimisation problems (DOPs), has grown due to their practical applications. A DOP requires an optimisation algorithm that can dynamically adapt to changes and several methodologies have been integrated with population-based algorithms to address these problems. Multi-population algorithms have been widely used, but it is hard to determine the number of populations to be used for a given problem. This paper proposes an adaptive multi-population artificial bee colony (ABC) algorithm for DOPs. ABC is a simple, yet efficient, nature inspired algorithm for addressing numerical optimisation, which has been successfully used for tackling other optimisation problems. The proposed ABC algorithm has the following features. Firstly it uses multi-populations to cope with dynamic changes, and a clearing scheme to maintain the diversity and enhance the exploration process. Secondly, the number of sub-populations changes over time, to adapt to changes in the search space. The moving peaks benchmark DOP is used to verify the performance of the proposed ABC. Experimental results show that the proposed ABC is superior to the ABC on all tested instances. Compared to state of the art methodologies, our proposed ABC algorithm produces very good results.

Keywords: dynamic optimisation, artificial bee colony algorithm, adaptive multi-population method, meta-heuristics

1. Introduction

Many real-world optimisation problems have the characteristic of changing over time in terms of decision variables, constraints and the objective function [1], [2]. These problems are referred to as dynamic optimisation problems (DOPs) in the scientific literature. A DOP requires an optimisation algorithm that can dynamically adapt to the changes and track the optimum solution during the execution of the algorithm [2]. Given their practical applications and complexity, DOPs have attracted a lot of research attention. Population-based algorithms, which are a set of methodologies that utilise a population of solutions distributed over the search space, have attracted particular attention, due to their good performance [1], [2]. A key challenge in developing an optimisation algorithm for DOPs is how to maintain population diversity during the search process in order to keep track of landscape changes [3]. Several interesting diversity schemes have been developed in order to improve the search capability of population-based algorithms so that they can adapt effectively to the problem as it changes.

Prediction based methods is one of the diversity schemes which has been widely integrated with other algorithms to maintain the diversity. This methodology uses an algorithm to learn patterns from previous searches, which are then used to predict future changes. It should be noted that memory methods can be categorised as a special case of the prediction method as they store a set of solutions to be used when a problem changes [2]. The prediction method is suitable for problems with cyclic changes. Hatzakis and Wallace [4] proposed a hybrid algorithm that combines an evolutionary algorithm and a forecasting methodology for DOPs. Forecasting is used to predict the movement of the optimum based on previous movements. The results demonstrate that this method is suitable for problems which change quickly if the movement of the optimum solution is predicted correctly. Sim et al. [5] used a prediction based method to predict how the environment would change and the time of the next change. The authors utilised a Markov chain that uses the previous movement of the search in order to predict future changes and a linear regression to predict when the change will occur. The results demonstrate that the hybrid algorithm performs with prediction, than without. Branke and Mattfeld [6] proposed an anticipation-based algorithm for DOPs. This algorithm attempts to simultaneously search for a good quality solution and move the search into a different area based on the previous changes. This proposed algorithm was tested on a dynamic job-shop scheduling problem and it was shown to produce very good results compared to other algorithms. The advantage of the prediction method is that it can be effective in detecting the global optima quickly, if the predictions are accurate [2]. The main drawback with this method is that it depends on the training model and in many cases the data used during the training process does not capture real world scenarios and there is a possibility of training errors due to lack of training data [7], [2].

Memory based methodologies aim to maintain diversity. They use a memory with a fixed size to store some of promising solutions that are captured during the search process. When a change is detected, the stored solutions will be reinserted into the current population and the population will be filtered to include only the best solutions. Examples of memory based methodologies can be found in [8], [9], [10], [11]. These methodologies have worked well when the dynamic problems are periodical or cyclic. The drawback is that they have parameter sensitivities that need to be determined in advance, and most real world problems are not cyclic in nature.

Self-adaptive algorithms attempt to adaptively improve the diversification of population-based algorithms based on environmental changes. They use mechanisms to adapt the algorithm to the changes in the search space [2]. Adaptive mechanisms can improve algorithm search behaviour and also reduce the need for manual parameter tuning. The idea is to apply different operators or parameter values for different problems by adaptively changing them during the search process [7], [2]. Grefenstette [12] proposed a self-adaptive genetic algorithm for DOPs. The proposed algorithm adaptively selects different crossover/mutation operators at each generation. The author uses an agent based concept to control the selection process, and each agent represents a crossover or mutation operator. All agents are executed simultaneously and the one that generates the best solution is selected for the current instance. Promising results were achieved when compared to other algorithms. Grefenstette [12] also proposes an idea called a genetic mutation rate for the DOP. The idea is to set the value of the mutation rate based on the fitness of the population. This idea was shown to generate better results compared to the basic genetic algorithm. Ursem [13] proposed a multinational genetic algorithm for the DOP. The main parameters are encoded with the decision variables and are evolved during the solution process. The results show that this algorithm is very good for simple instances in which the velocity of the moving peaks is constant. It is also able to adapt by changing the algorithm parameters during the search. However, encoding the parameters with the solution decision variables requires specialist evolutionary operators. In addition, it is also very difficult to determine the values of the parameters [2].

Multi-population methods improve diversity by dividing the population of solutions into several sub-populations and distributing them throughout the search landscape so that they can more effectively capture the problem changes. The idea is to maintain population diversity by assigning a different sub-population to a different area, where each one is responsible for either intensifying or diversifying the search process [7], [2]. These sub-populations interact with each other via a merge and divide process when a change in the environment is detected. The multi-population method has been shown to be effective in dealing with various problem changes, whether they are cyclic or non-cyclic, and it has outperformed other methods on various problem sizes. Branke et al. [14] proposed a self-organising scouts multi-population evolutionary algorithm for the DOP. The population of solutions is divided into two groups; small and large. The small population group is responsible for tracking promising solutions found so far, while the large population group tries to find a new region of the search space that has a new peak. The proposed algorithm was tested on the moving peaks benchmark (MPB), obtaining very good results. Blackwell and Branke [15] proposed a multi-swarm optimisation algorithm for the DOP. The swarm is divided into subsets of swarms. These multi-swarms interact with each other locally, through algorithm parameters, and globally by using an anti-convergence mechanism. The anti-convergence mechanism searches for new peaks by removing the worst ones and re-initialising them into a different area in the search space. The proposed algorithm obtained very good results when tested on MPB problems. Mendes and Mohais [16] presented a multi-population differential evolution algorithm for the DOP. The population of solutions is divided into several sub-populations. Each sub-population is assigned to a different area of the search space. The experimental results show that this algorithm obtains very good results for MPB problems. Li and Yang [17] proposed a fast multi-swarm Particle Swarm Optimisation (PSO) algorithm for the DOP. The swarm population is divided into two types of swarms; parents and children. The parent swarm explores the entire search space to seek the global optima, while the child swarm is responsible for monitoring the search behaviour around the best solution obtained by the parent swarm. The position of the child swarm is dynamically updated during the process. The algorithm was tested on the MPB problems and produced good results when compared to other methods. Yang and Li [18] presented a clustering-based particle swarm optimiser for the DOP. The swarm is divided based on a hierarchical clustering method to locate and track multiple peaks. The algorithm achieved very good results when tested on the MPB. Turky and Abdullah [19] proposed a multi-population electromagnetic algorithm for DOPs. The proposed algorithm divides the population into several sub-populations to simultaneously explore and exploit the search process. The algorithm was tested on MPB problems and obtained very good results when compared to other population diversity mechanisms. The same authors [20] also presented a multi-population harmony search algorithm for the DOP. The population is divided into sub-populations. Each sub-population is responsible for either exploring or exploiting the search space. An external archive is utilised to track the best solutions found so far, which are used to replace the worst ones when a change is detected. The results show that this algorithm produces good results when compared to other methods. Sharifi et al. [21] proposed a hybrid PSO and local search algorithm for DOPs. The algorithm utilises a fuzzy social-only model to locate the peaks. The results show that this algorithm can produce very good results for MPB problems. In Li et al. [22] comprehensive experimental analysis was reported on the performance of a multi-population method with various algorithms in relation to DOPs. The authors concluded that the multi-population method is able to deal effectively with various DOPs and has the ability to maintain population diversity. It is also able to help the search in locating a new area through a divide and merge process and information exchange. The authors also highlighted several weaknesses of their method that relate to the number of sub-populations, the distribution of solutions and the reaction to problem changes.

Existing works on DOPs demonstrate that employing multi-population methods are the most effective method in maintaining population diversity. The features that make the multi-population methodologies popular are [3]: i) it divides the population into sub-populations, where the overall population diversity can be maintained since different populations can be located in different areas of the problem landscape, ii) it has the ability to search different areas simultaneously, enabling it to track the movement of the optimum, and iii) various single population-based algorithms can be integrated within multi-population methods.

Although multi-population methods have shown success when applied to DOPs, most of them use a number of sub-populations and the population diversity is maintained only through the sub-population distribution [3]. The number of sub-populations has a crucial impact on algorithm performance as it relates to the difficulty of the problem, which is not known in advance, and changes during the search. In addition, the solutions in the sub-populations may not be diverse enough as some methods are only concerned with how to divide the population into sub-populations, rather than focussing on diversification. To address these issues, this work proposes an adaptive multi-population artificial bee colony (ABC) algorithm for the DOP. The proposed ABC utilises a clearing scheme to remove redundant solutions in order to maintain diversity and enhance the exploration process. To efficiently track the landscape changes, the proposed ABC algorithm adaptively updates the number of sub-populations based on the problem change strength.

In this paper, the key objectives are:

i. To propose an artificial bee colony algorithm that utilises a multi-population and a population clearing scheme to efficiently solve the dynamic optimisation problem.
ii. To propose an adaptive multi-population algorithm that updates the number of the sub-populations based on the problem change strength.
iii. To test the performance of the proposed algorithm on dynamic optimisation problems using different scenarios and compare the results with other methodologies.

We used the moving peaks benchmark DOP with a different number of peaks to evaluate the effectiveness of the proposed ABC. Results demonstrate that the proposed ABC performs better than a basic ABC on all tested scenarios. Compared to the state of the art method, the proposed ABC produces very good results for many instances.

2. The proposed algorithm

This section presents the basic artificial bee colony algorithm, as well as our proposed adaptive multi-population algorithm.

2.1 Basic artificial bee colony algorithm

The Artificial Bee Colony (ABC) algorithm is a simple, yet efficient, nature inspired algorithm for addressing numerical optimization problems. It was proposed in [23] as a nature inspired swarm intelligence algorithm based on the observation of bee foraging behaviour. In ABC, there are a set of food sources and a set of bees. The quality of the food sources is based on the amount of nectar they contain. Bees search and collaborate with each other, seeking better food sources. To address an optimization problem using ABC, food sources represent the population of solutions for a given problem and bees are categorised into three types: scout, employee and onlooker bees. The amount of nectar corresponds to the quality (objective function) of the problem being addressed. The three types of bees work together in an iterative manner to improve the quality of the population of solutions (food sources). The pseudo-code of a basic ABC is shown in Algorithm 1 [24]. ABC first sets the main parameters, initializes the population of solutions and then evaluates them. Next, the main loop is executed in an attempt to solve the given optimisation problem by calling the employee bees, onlooker bees and scout bees until the stopping condition is satisfied.

Algorithm 1: The pseudo-code of basic ABC

Step 1: Set the parameter values
Step 2: Initialize the population of solutions
Step 3: Evaluate the population of solutions
while termination condition is not met do
    Step 4: Employed Bees step
    Step 5: Onlooker Bees step
    Step 6: Scout Bees step
end while

The basic ABC has the following steps:

Step 1- Set ABC parameters. In this step the main parameters of ABC are initialized. These include: the maximum number of iterations (*MaxIt*) which represents the stopping condition of ABC, the number of solutions or population size (*Ps*) which represent how many solutions will be generated, the total number of bees (*Sbees*) which is set to be twice the size of *Ps*, where half of them are employee bees and the other half are onlooker bees, the limit parameter (*Lit*), which is used to determine if the solution should be replaced by a new one.

Step 2- Initialise the population of solutions. A set of solutions with size equal to *Ps* are randomly generated as follows:

$$x_{i,j} = x_{i,j}^{\min} + Rand[0,1](x_{i,j}^{\max} - x_{i,j}^{\min}) \tag{1}$$

where i is the index of the solution, j is the current decision variable, *Rand* [0,1] generates a random number between zero and one and $x_{i,j}^{\min}$ and $x_{i,j}^{\max}$ are the lower and upper bonds for the j^{th} decision variable.

Step 3- Evaluate the population of solutions. The fitness (quality) of the generated solutions are calculated using the objective function. The objective function is problem dependent. The objective function used in this work is shown in Section 3.2.

Step 4- Employed bees. Each employee bee is sent to one food source (solution). Its main role is to explore the neighbourhood of the current solution, seeking an improving solution. A neighbourhood solution, *v*, is created by modifying the i^{th} solution, *x*, as follows:

$$v_{i,j} = x_{i,j} + \Phi_{i,j}(x_{i,j} - x_{k,j}) \qquad (2)$$

where *k* is a randomly selected solution from *Ps* and *Φ* is a random number between [-1, 1]. The generated neighbourhood solution will be replaced with current solution if it has better fitness.

Step 5- Onlooker bees. Onlooker bees seek to improve the current population of solutions by exploring their neighbourhood using Equation (2), the same as the employee bee. The difference is that onlooker bees select the solutions probabilistically based on their fitness values as follows:

$$p_i = \frac{fitness_i}{\sum_{j=1}^{Ps} fitnessj} \qquad (3)$$

That is, the solution with the higher fitness has a higher chance of being selected (i.e. roulette wheel selection). Onlooker bees use a greedy selection mechanism, where the better solution in terms of fitness is selected.

Step 6- Scout bees. This step is activated if both employed and onlooker bees cannot improve the current solution for a number of consecutive iterations defined by the limit parameter, *Lit*. This indicates that the current solution is not good enough to search its neighbourhood and it should be discarded. In this case, the scout bee will generate a new solution using Equation (1) to replace the discarded one. This can help ABC to escape from a local optimum and explore a different area of the search space.

2.2 The proposed artificial bee colony algorithm

Existing works on DOPs have demonstrated that multi-population methods are state of the art, in that they outperform other methods on many scenarios. However, although multi-population methods have achieved success in solving DOPs, most of them use a fixed number of sub-populations and the population diversity is maintained through the sub-population distribution. To address these issues, this work proposes an adaptive population ABC (denoted as Multi-pop-ABC). In Multi-pop-ABC, three major modifications are added to the basic ABC. These are:

i. Multi-population method. To deal with DOP, the proposed ABC uses a multi-population method to divide the population into several sub-populations. By using a multi-population method, the solutions are scattered over the search space instead of focusing on a specific area. Thus the algorithm can generate high quality solutions and track the problem changes.

ii. Adaptive scheme. To track the landscape changes that occur during the search process, the proposed Multi-pop-ABC updates the number of sub-populations based on the strength of the problem change. That is the number of sub-populations is either decreased or increased during the search process. By using the proposed adaptive method, the number of sub-populations can be changed adaptively based on the strength of the environment changes, which helps the search track the optimum solution and also improves the diversification and exploration processes.

iii. Population clearing scheme. To ensure that the solutions are diverse enough, a population clearing scheme is called when a change is detected to delete redundant solutions and replace them with new solutions. This scheme removes redundant solutions in order to maintain diversity and enhance the exploration process.

The flowchart of the proposed Multi-pop-ABC for DOPs is shown in Figure 1. It starts by setting the parameter values. It creates the population of solutions and then evaluates them. Next, the population of solutions is divided into m sub-populations. Each sub-population utilises an ABC algorithm. If a change in the problem is detected, the algorithm calculates the change strength to update the sub-population size and checks the stopping condition. If the specified stopping condition (we set this as a maximum number of fitness evaluations) has been reached, the algorithm terminates and the best solution is returned. Otherwise, the algorithm merges all the sub-populations, updates the population, runs the clearing method, re-divides the population into *m* sub-populations and starts a new iteration.

The main steps are described in further detail below:

- **Step 1: Set parameters**. The main parameters of Multi-pop-ABC are initialised. The algorithm has five parameters. Four of them are the same as the basic ABC. These are: the maximum number of iterations (*MaxIt*), population size (*Ps*), number of bees (*Sbees*), and the limit parameter (*Lit*). The fifth parameter is the sub-population size (*m*), which represents the number of sub-populations (*Ps*/*m*). Initially, *m*=2 and during the search process, it is either decreased or increased.

1- Step 2: Initialise the population of solutions. Same as Step 2 in the basic ABC, Section 2.1.

2- Step 3: Evaluate the population of solutions. Same as Step 3 in the basic ABC, Section 2.1.

3- Step 4: Divide the population. The population of solutions is divided into *m* sub-populations (*Ps*/*m*). Each sub-population is assigned to explore a different area of the search space. These sub-populations interact with each other through merging and re-dividing every time a change in the environment is detected. Each solution in the population is randomly assigned to a sub-population. The number of sub-populations *m* is either increased or decreased based on the environment change strength. The initial value of *m* is set to two (*m*=2) and it is updated during the search.

4- Step 5: Assign ABC to each sub-population. Each sub-population has its own ABC algorithm. Each ABC executes all the steps presented in Section 2.1. It starts with a population of solutions and iteratively calls the following until the stopping condition is satisfied (the algorithm stops when a change in the environment is detected):

i. **Employee bees**. Same as Step 4 in the basic ABC, Section 2.1.

ii. **Onlooker bees**. Same as Step 5 in the basic ABC, Section 2.1.

iii. **Scout bees**. Same as Step 6 in the basic ABC, Section 2.1.

5- **Step 6: Check the change strength.** This step is activated when a change in the environment is detected. Its main role is to update the number of sub-populations based on the environment change strength. It first calculates the objective function of the best solution before and after the environment change as follows:

$$Cs = f(best\_before) - f(best\_after) \tag{4}$$

where *Cs* is the change strength, *f*(*best_before*) is the quality of the best solution before the environment change and *f*(*best_after*) is the quality of the best solution after the environment change. If the *Cs* is less than the defined threshold (*Tv*) and *m* is greater than 2, the number of sub-populations *m* is decreased as the algorithm needs to be more exploitive than explorative (*m*=*m*-1). Otherwise, *m* is increased by one with the aim of increasing the exploration aspect of the search (*m*=*m*+1). It should be noted that when *m* is an odd number, the *extra* solution is randomly assigned to one of the sub-populations.

6- **Step 7: Check the stopping condition.** This step checks the termination criterion of the search process. In this work, it is set as a maximum number of fitness evaluations in line with previous works. If the specified stopping condition is reached, the search process stops and returns the best solution. Otherwise, the algorithm performs the following processes:

i. **Population clearing scheme:** This scheme calculates the similarity between solutions in the population. The similarity is calculated by using a matching algorithm, which matches each pair of solutions in terms of phenotype. Two solutions are similar if they have the same values in all the cells of both solutions. If two or more solutions are similar, these solutions are deleted and replaced with randomly generated ones.

ii. **Population update**: All sub-populations are merged to form one population.

iii. Re-divide the population: The population is re-divided into *m* sub-populations and the algorithm continues by starting the process at step 1 with a new generation.

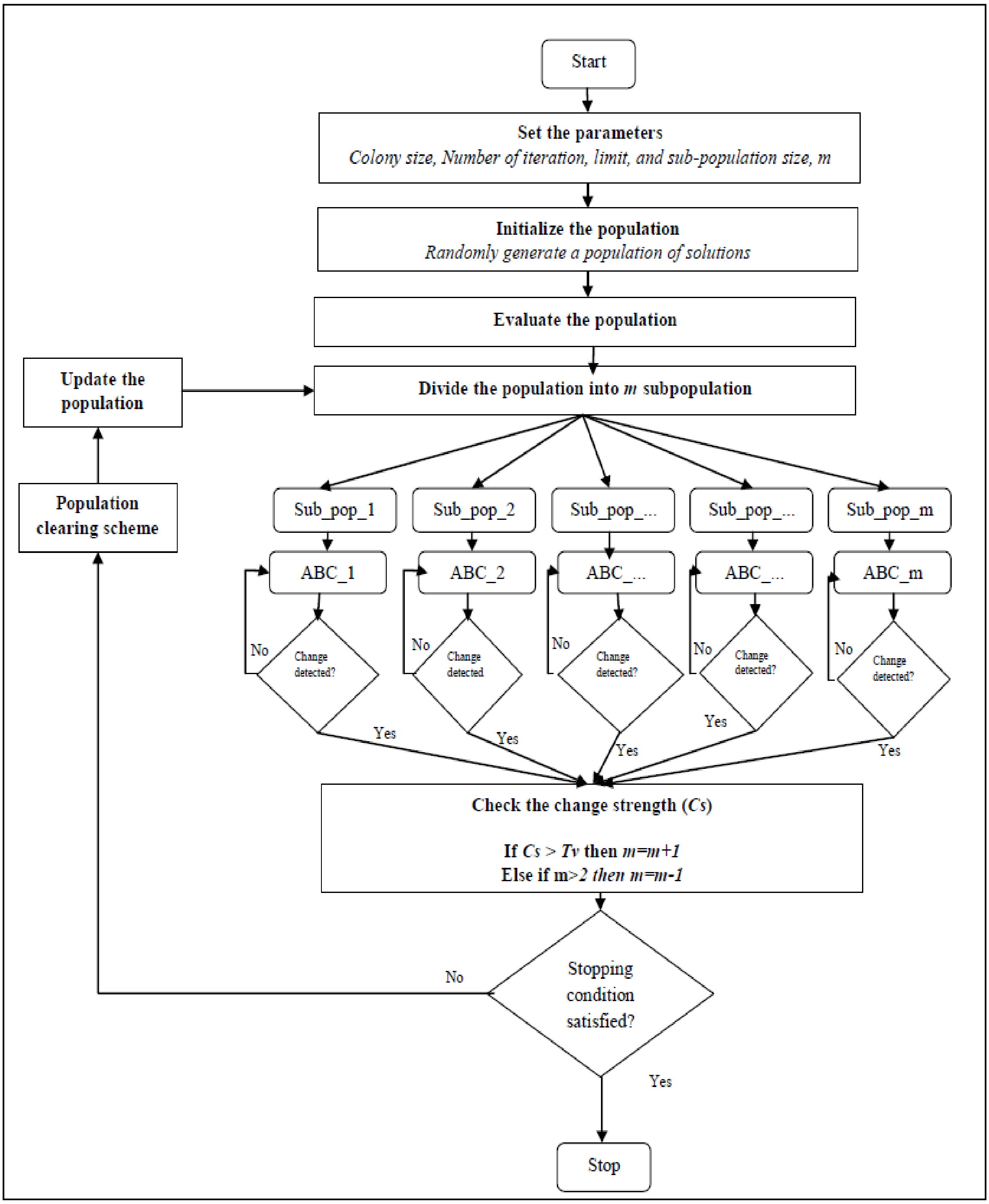

Figure 1. The proposed Multi-pop-ABC

3. Experimental Setup

This section discusses the Moving Peak Benchmark (MPB), evaluation metric and the parameter settings.

3.1 The Moving Peak Benchmark

The moving peak benchmark (MPB) is a maximization dynamic continuous optimization problem proposed by [9], [25], and has been commonly used as a testbed for the performance of optimisation algorithms. MPB consists of a set of peaks that move over the problem landscape. It takes the given solution as an input and returns the value of the highest peak. The returned value represents the quality of this solution. MPB can be mathematically expressed as follows:

$$F(\vec{x},t) = \max_{i=1,\dots,p}\left(\frac{H_i(t)}{1+W_i(t)\sum_{j=1}^{D}(x_j(t)-X_{ij}(t))^2}\right) \quad (5)$$

where $F(x, t)$ is the quality of solution x at time t, p is the number of peaks, D is the problem dimension (number of decision variables where each variable has an upper and lower boundary (DB)), $H_i\ (t)$ is the height of peak i, $W_i\ (t)$ is the width of peak i, and X_{ij} is the j^{th} element of the location of peak i. Note that Equation (5) is a stationary optimization problem. Thus, to change it to a dynamic problem, MPB randomly shifts the position of all peaks by vector $\vec{v}_i$ of a distance s (s is also known as the shift length that determines the severity degree) as follows:

$$\vec{v}_i(t) = \frac{s}{|\vec{r}+\vec{v}_i(t-1)|}((1-\lambda)\vec{r}+\lambda\vec{v}_i(t-1)) \quad (6)$$

where $\vec{r}$ is a random vector, λ is the correlation between consecutive movements of a single peak that takes either "0" if the movement of peaks are completely uncorrelated or "1" if they move in the same direction. To make a fair comparison with existing algorithms, in this paper, we used λ=0 [6]. The change of height and width of a peak in a given solution can be mathematically expressed as follows:

$$H_i(t) = H_i(t-1) + height\_severity * \sigma \quad (7)$$

$$W_i(t) = W_i(t-1) + width\_severity * \sigma \quad (8)$$

where *height_severity* and *width_severity* are calculated based on the problem severity. σ is a normally distributed random number between 0 and 1. Then, the change of a solution x is given as follows:

$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t) \quad (9)$$

The change frequency (*cf*) occurs every 5,000 fitness evaluations [9]. The parameter values of all MPBs that have been used in our experiments are shown in Table 1 [25].

Table 1 MPB parameter values

Parameters	*Description*	*Value*
p	Number of peaks	1–200
cf	Change frequency	5000
height_severity	Height severity	7.0
width_severity	Width severity	1.0
Peak shape	Peak shape	Cone
s	Shift length	1.0
D	Number of dimensions	5
λ	Correlation coefficient	0
DB	Each dimension boundaries	[0,100]
H	Peak height	[30.0,70.0]
W	Peak width	[1,12]

3.2 Evaluation Metric

To fairly compare the proposed ABC with existing algorithms, we use the same evaluation metric known as the offline error as suggested by [25]. This has also been used by other researchers. The offline error is calculated as follows:

$$off = \frac{1}{g}\sum_{i=1}^{g}\Omega_i \quad (10)$$

where g is the number of generations and Ω is the best performance since the last change at i^{th} fitness evaluation.

3.3 Parameter Settings

The parameter values of our Multi-pop-ABC are set by carrying out a set of initial experiments, with the exception of the stopping condition which was set to be the same as the compared algorithms (50,000 fitness evaluations). For each parameter, we tested various values and the best values were selected. This is achieved by varying the value of one parameter while fixing others. We have selected two scenarios of MPB for the parameter tunning process: 50 peaks and 200 peaks. The proposed ABC has three parameters:

population size (*Ps*), limit (*Lit*) and the change strength threshold (*Tv*). First, we fixed *Lit* to 30, *Tv* to 0.09 and changed *Ps*. Table 2 shows the offline error of various *Ps* values for 50 and 200 peaks. The best result is highlighted in bold. Next, we fixed *Ps* to 60, *Tv* to 0.09 and changed *Lit* as shown in Table 3. Finally, we fixed *Ps* to 60, *Lit* to 30 and changed *Tv* as shown in Table 4. The parameter settings of the proposed ABC that were used across all scenarios are presented in Table 5.

Table 2 The value of *Ps* parameter

Ps value	50 peaks	200 peaks
20	0.95669	2.70215
40	0.319632	1.8935
60	0.5810	**0.34865**
80	0.576911	1.15134

Table 3 The value of *Lit* parameter

Lit value	50 peaks	200 peaks
10	1.03474	1.18977
20	1.27535	1.70215
30	1.29851	**0.24824**
40	1.841891	1.28967

Table 4 The value of *Tv* parameter

Tv value	50 peaks	200 peaks
0.03	0.95669	1.89663
0.05	0.96573	**1.08053**
0.07	0.89978	1.37518
0.09	1.23491	1.77956

Table 5 The parameter settings of the proposed ABC

#	Parameter	Value
1-	Maximum number of iterations (*MaxIt*)	50,000 fitness evaluations
2-	Population size (*Ps*)	60
3-	Limit parameter (*Lit*)	30
4-	Change strength threshold (*Tv*)	0.05

4. Results

We carried out three set of experiments. In first one, we compare the results of Multi-pop-ABC with the basic ABC. In second one, the results obtained by Multi-pop-ABC are compared with state of the art methods. In the third experiment, the results of Multi-pop-ABC on well-known test functions are compared with state of the art methods.

4.1 Results comparison of Multi-pop-ABC and the basic ABC

This section aims to verify the effectiveness of the additional components that we have added to the basic ABC. Specifically, the objective is to investigate the impact of the proposed enhancements on the performance of the basic ABC when dealing with DOPs. Four different algorithms were derived as follows:

- Multi-pop-ABC: the proposed ABC that utilises the adaptive multi-population and population clearing scheme
- Multi-pop-ABC$_1$: same as above but without the population clearing scheme
- Multi-pop-ABC$_2$: same as above but uses a fixed number of sub-populations and without the population clearing scheme. The sub-populations were fixed to be the same as [26]
- ABC: basic ABC algorithm.

The computational comparisons of Multi-pop-ABC, Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC are presented in Table 6. The comparison is in terms of the offline error, ± standard error for each number of peaks. The best results are highlighted in bold. The results clearly show the good performance of Multi-pop-ABC when compared to Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC. Indeed, Multi-pop-ABC outperformed Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC on both the offline error and the standard error on all tested scenarios. The results demonstrate that the enhancements we made to the basic ABC improve the algorithmic performance.

Table 6 Results of the Multi-pop-ABC, Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC

	Number of Peaks										
Algorithm	1	2	5	7	10	20	30	40	50	100	200
Multi-pop-ABC	**0.14 ±0.00**	**0.12 ±0.00**	**0.20 ±0.00**	**0.38 ±0.01**	**0.22 ±0.01**	**0.35 ±0.00**	**0.46 ±0.00**	**0.52 ±0.01**	**0.44 ±0.01**	**0.52 ±0.00**	**0.93 ±0.00**
Multi-pop-ABC$_1$	1.81 ±0.18	1.42 ±0.32	1.11 ±0.13	1.01 ±0.22	1.57 ±0.12	1.43 ±0.15	1.45 ±0.14	1.62 ±0.10	1.21 ±0.21	1.73 ±0.11	1.22 ±0.10
Multi-pop-ABC$_2$	1.12 ±0.18	1.21 ±0.41	1.61 ±0.10	1.65 ±0.11	1.71 ±0.15	1.11 ±0.14	1.72 ±0.19	1.42 ±0.13	1.62 ±0.18	1.41 ±0.12	1.42 ±0.12
Basic ABC	5.88 ±2.48	5.52 ±4.31	4.12 ±3.7	4.5 ±2.3	5.2 ±3.16	6.3 ±3.51	3.38 ±4.32	7.14 ±3.60	6.21 ±2.01	6.97 ±2.11	7.03 ±3.44

Note: Values in bold font indicate the best results.

To further verify the results, we conducted a comparison between Multi-pop-ABC and each method separately. We used a Wilcoxon statistical test with a confidence level of 0.05. The *p*-values of Multi-pop-ABC against Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC for each scenario is presented in Table 7. A value less than 0.05 indicates Multi-pop-ABC is superior (i.e. statistically different). As can be seen from Table 7, Multi-pop-ABC is superior to Multi-pop-ABC$_1$, Multi-pop-ABC$_2$ and basic ABC on 9 out of 11 tested scenarios ($p < 0.05$). The table also shows than on two scenarios (1 peak and 2 peaks) Multi-pop-ABC is not superior to Multi-pop-ABC$_1$ and Multi-pop-ABC$_2$. This can be attributed to the fact that these two scenarios are relatively easy to solve and thus all methods produce very good solutions. The results of the statistical test also demonstrate that the proposed enhancements have a positive impact and improve the search process.

Table 7 *p*-values of the of Multi-pop-ABC against other methods

	Number of Peaks										
Multi-pop-ABC vs.	**1**	**2**	**5**	**7**	**10**	**20**	**30**	**40**	**50**	**100**	**200**
Multi-pop-ABC$_1$	0.06	0.08	0.04	0.03	0.00	0.00	0.00	0.01	0.01	0.00	0.00
Multi-pop-ABC$_2$	0.07	0.06	0.02	0.07	0.06	0.00	0.04	0.01	0. 02	0.00	0.00
Basic ABC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Note: Values less than 0.05 indicate that Multi-pop-ABC is better than the compared methods.

4.2 Comparison with state of the art methods

There are numerous methods that use different schemes to handle diversification, and which have been tested on MPB. In this section, we evaluate the performance of our algorithm by comparing it with several recently proposed algorithms taken from the scientific literature. The algorithms are:

- Multiswarms, exclusion, and anti-convergence in dynamic environments (mCPSO) [27].
- Multiswarms, exclusion, and anti-convergence in dynamic environments (mQSO) [27]
- Multiswarms, exclusion, and anti-convergence in dynamic environments (mCPSO*) [27]

- Multiswarms, exclusion, and anti-convergence in dynamic environments (mQSO*) [27].
- Competitive population evaluation in a differential evolution algorithm for dynamic environments (CDE) [28].
- Differential evolution for dynamic environments with unknown numbers of optima (DynPopDE) [29].
- Dynamic function optimization with hybridized extremal dynamics (EO + HJ) [30]
- A competitive clustering particle swarm optimizer for dynamic optimization problems (CCPSO) [31].
- A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems (CHPSO(ES-NDS)) [32].

To ensure a fair comparison, we used the same stopping condition (50,000 fitness evaluations), the same change frequency (every 5,000 fitness evaluations) and the same evaluation metric (Offline error). We also used 11 MPB instances with a different number of peaks ranging between 1 to 200 peaks.

The results of Multi-pop-ABC and the compared algorithms are presented in Table 8. The results in the table are in terms of offline error, ± standard error and computational times for each number of peaks. In the table, the symbol '-' indicates that the scenario has not been tested. We indicate in bold the best obtained results. From Table 8, it can be seen that Multi-pop-ABC is superior to the other algorithms in most of the cases in terms of offline error. In particular, Multi-pop-ABC obtained new best results for 9 out of 11 tested MPB instances. Multi-pop-ABC was inferior on only two MPB instances: 1 peak and 2 peaks. Nevertheless, the results of Multi-pop-ABC for these two scenarios are very competitive, where it obtained the second best results. In terms of the standard error, Multi-pop-ABC produced a better standard error for 6 scenarios, being similar on 5 scenarios out of the 11 tested.

Table 8 Results of Multi-pop-ABC compared to the state of the art methods

	Number of Peaks										
Algorithm	1	2	5	7	10	20	30	40	50	100	200
Multi-pop-ABC	**0.14 ±0.00** 8.11	**0.12 ±0.00** 9.10	**0.20 ±0.00** 10.20	**0.38 ±0.01** 10.63	**0.22 ±0.01** 11.12	**0.35 ±0.00** 13.75	**0.46 ±0.00** 15.13	**0.52 ±0.01** 17.17	**0.44 ±0.01** 20.23	**0.52 ±0.00** 28.64	**0.93 ±0.00** 56.48
mCPSO	4.93 ±0.17	3.36 ±0.26	2.07 ±0.08	2.11 ±0.11	2.08 ±0.07	2.64 ±0.07	2.63 ±0.08	2.67 ±0.07	2.65 ±0.06	2.49 ±0.04	2.44 ±0.04

mQSO	5.07 ±0.17	3.47 ±0.23	1.81 ±0.07	1.77 ±0.07	1.80 ±0.06	2.42 ±0.07	2.48 ±0.07	2.55 ±0.07	2.50 ±0.06	2.36 ±0.04	2.26 ±0.03
mCPSO*	4.93 ±0.17	3.36 ±0.26	2.07 ±0.11	2.11 ±0.11	2.05 ±0.07	2.95 ±0.08	3.38 ±0.11	3.69 ±0.11	3.68 ±0.11	4.07 ±0.09	3.97 ±0.08
mQSO*	5.07 ±0.17	3.47 ±0.23	1.81 ±0.07	1.77 ±0.07	1.75 ±0.06	2.74 ±0.07	3.27 ±0.11	3.60 ±0.08	3.65 ±0.11	3.93 ±0.08	3.86 ±0.07
CDE	-	-	-	-	0.92 ±0.07	-	-	-	-	-	-
DynPopDE	-	-	1.03 ±0.13	-	1.39 ±0.07	-	-	-	2.10 ±0.06	2.34 ±0.05	2.44 ±0.05
EO + HJ	7.08 ±1.99	-	-	-	0.25 ±0.10	0.39 ±0.10	0.49 ±0.09	0.56 ±0.09	0.58 ±0.09	0.66 ±0.07	-
CCPSO	0.09 ±0.00	0.09 ±0.00	0.25 ±0.01	0.53 ±0.03	0.75 ±0.06	1.21 ±0.08	1.40 ±0.07	1.47 ±0.08	1.50 ±0.09	1.76 ±0.09	-
CHPSO(ES-NDS)	0.19 ± 0.00	-	0.44 ±0.02	-	0.64 ±0.02	0.91 ±0.01	0.99 ±0.01	1.02 ±0.01	1.03 ±0.01	1.04 ±0.01	1.01 ±0.00

Note: Values in bold font indicate the best results.

To further verify the effectiveness of the proposed Multi-pop-ABC, we statistically compare it with other methods. We followed the procedure described in [33]. First, Friedman test and Iman and Davenport statistical tests with 0.05 confidence levels are carried out to detect if there is a difference between the results of Multi-pop-ABC and other methods. It should be noted that only those methods that were tested on all scenarios were considered for this test. Both the Friedman test and Iman and Davenport tests returned *p*-values (0.000009 and 0.000000009061) less than 0.05 indicating the compared results are statistically different. We next conducted a Friedman test to obtain rankings, and Holm and Hochberg post-hoc tests. The ranking value for each method obtained by a Friedman test is presented in Table 9 (the lower the better), where Multi-pop-ABC obtained the first rank followed by mQSO second rank, mCPSO third rank, mQSO* fourth rank and mCPSO* fifth rank. Consequently, Multi-pop-ABC will be the controlling method for the Holm and Hochberg post-hoc tests. The *p*-values of Holm and Hochberg tests are shown in Table 10. From the table, one can see that Multi-pop-ABC is statistically better than the compared methods on both Holm and Hochberg tests in which all the obtained *p*-values are less than 0.05.

Table 9 The average ranking of Friedman test

#	*Algorithm*	*Ranking*
1	Multi-pop-ABC	1
2	mQSO	2.6364
3	mCPSO	3.3636
4	mQSO*	3.6364
5	mCPSO*	4.3636

Table 10 The adjusted *p*-value of the compared methods

#	*Algorithm*	*Unadjusted P*	*P Holm*	*P Hochberg*

1	mCPSO*	0.000001	0.000002	0.000002
2	mQSO*	0.000092	0.000276	0.000276
3	mCPSO	0.000455	0.00091	0.00091
4	mQSO	0.015219	0.015219	0.015219

The above results reveal that, in most of the tested scenarios, the proposed Multi-pop-ABC is better than the compared methods. These results are supported by statistical tests.

We hypothesise that several key features contribute to the high performance of the proposed algorithm (Multi-pop-ABC) on the dynamic problem. These can be summarised as follows:

- Multi-population: This feature is beneficial for maintaining the diversity of solutions in the population during the search process.
- Adaptive number of sub-populations: This feature helps the algorithm in changing the solution distribution over the search landscape to get better diversification and intensification based on the problem change strength.
- Population clearing scheme: This feature helps avoid having similar solutions within the population in order to further add to the diversification.

4.3 Comparison with state-of-the-art approaches on test functions

In this section, we evaluate our proposed algorithm based on other well-known ten test functions. The tested functions are widely used by researchers [34-37]. These functions are:

$$f_1\left(\vec{x}\right)=\sum_{i=1}^{n} x_i^2 \qquad [-100, 100]^n$$

$$f_2\left(\vec{x}\right)=\sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \qquad [-10, 10]^n$$

$$f_3\left(\vec{x}\right)=\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2 \qquad [-100, 100]^n$$

$$f_4\left(\vec{x}\right)= \max_i \{|x_i|, 1 \le i \le n\} \qquad [-100, 100]^n$$

$$f_5\left(\vec{x}\right)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right] \qquad [-30, 30]^n$$

$$f_6\left(\vec{x}\right)=\sum_{i=1}^{n} i x_i^4 + random(0,1) \qquad [-1.28, 1.28]^n$$

$$f_7\left(\vec{x}\right)=\sum_{i=1}^{n}-x_i \sin\left(\sqrt{|x_i|}\right) \qquad [-500, 500]^n$$

$$f_8\left(\vec{x}\right)=\sum_{i=1}^{n}\left(x_i^2-10\cos\left(2\pi x_i\right)+10\right) \qquad [-5.12, 5.12]^n$$

$$f_9\left(\vec{x}\right)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e \qquad [-32, 32]^n$$

$$f_{10}\left(\vec{x}\right)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \qquad [-600, 600]^n$$

For every benchmark function, respectively assume the dimension as 30, 50 and 100. The results in Tables 11, 12 and 13 demonstrate that Multi-pop-ABC performs better than the compared ABC, PS-ABC and PS-ABCII algorithms [34-36] in terms of both mean and standard deviation (SD). Note that the best results are highlighted in bold. The presented results indicate that the Multi-pop-ABC outperforms other methods over all test functions.

Table 11 Mean, the standard deviation (SD) of functions with 30 dimensions.

F	ABC			PS-ABC		PS-ABCII		LWGSODE		CFOA		Multi-pop-ABC	
	Dim	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**Mean**	**Mean**	**SD**	**Mean**	**SD**	**SD**	**SD**
f1	30	3.3955×10^{-9}	4.5376×10^{-9}	**0**	0	**0**	0	1.68×10^{-7}	1.63×10^{-7}	1×10^{-309}	-	**0**	0
f2	30	5.1029 x 10-[6]	1.8417×10^{-6}	**0**	0	**0**	0	1.10×10^{-3}	4.09×10^{-4}	1×10^{-155}	-	**0**	0
f3	30	1.2598×10^{4}	2.9192×10^{3}	7.2696×10^{3}	1.4359×10^{3}	4.0756×10^{4}	8.1760×10^{3}	-	-	-	-	**6.4361×10^{2}**	1.9164×10^{2}
f4	30	2.4044×10^{1}	3.3935	**0**	0	**0**	0	-	-	-	-	**0**	0
f5	30	3.2873	3.4035	1.4048	2.7168	2.8408×10^{1}	0.1154	2.60×10^{1}	2.70×10^{-2}	-	-	**1.0196**	1.9217
f6	30	1.5788×10^{-1}	3.6701×10^{-1}	1.8545×10^{-2}	5.3198×10^{-3}	5.5447×10^{-4}	1.2352×10^{-3}	-	-	1×10^{-309}	-	**3.2124×10^{-4}**	1.0172×10^{-3}
f7	30	-12185.9	1.4299×10^{2}	-12549.7	4.4891×10^{1}	-12088.9	1.8715×10^{2}	-	-	-	-	**-14847.9**	3.5787×10^{1}
f8	30	4.0160×10^{-1}	6.2228×10^{-1}	**0**	0	**0**	0	3.85×10^{-5}	3.75×10^{-5}	-	-	**0**	0
f9	30	2.4076×10^{-5}	1.2439×10^{-5}	8.8817×10^{-16}	0	8.8817×10^{-16}	0	2.99×10^{-4}	1.86×10^{-4}	1×10^{-308}	-	**0**	0
10	30	1.4335×10^{-3}	4.0152×10^{-3}	**0**	0	**0**	0	-	-	1×10^{-309}	-	**0**	0

Table 12 Mean, the standard deviation (SD) of functions with 50 dimension.

F	*ABC*			*PS-ABC*		*PS-ABCII*		*Multi-pop-ABC*	
	Dim	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
f1	50	1.1483×10^{-5}	1.6272×10^{-5}	**0**	0	**0**	0	**0**	0
f2	50	2.8511×10^{-3}	1.3944×10^{-3}	**0**	0	**0**	0	**0**	0
f3	50	4.6422×10^{4}	6.9821×10^{3}	3.0638×10^{3}	3.4739×10^{3}	1.2539×10^{5}	2.1047×10^{4}	$\mathbf{2.1041 \times 10^{3}}$	2.0893×10^{3}
f4	50	5.6020×10^{1}	5.1905	1.8782×10^{1}	5.7908	**0**	0	**0**	0
f5	50	3.7224×10^{1}	3.6453×10^{1}	3.1451×10^{1}	2.9224×10^{1}	4.8504×10^{1}	1.3535×10^{-1}	$\mathbf{2.2310 \times 10^{1}}$	2.1102×10^{1}
f6	50	4.2726×10^{-1}	8.2393×10^{-2}	5.7802×10^{-2}	1.6469×10^{-2}	5.4388×10^{-4}	6.4470×10^{-4}	2.1847×10^{-4}	3.2711×10^{-4}
f7	50	-19359.1	3.1097×10^{2}	-20893.4	7.9224×10^{1}	-19414.1	3.3738×10^{2}	**-26893.4**	7.8394×10^{1}
f8	50	8.1857	2.4195	**0**	0	**0**	0	**0**	0
f9	50	4.0637×10^{-2}	3.2467×10^{-2}	8.8817×10^{-16}	0	8.8817×10^{-16}	0	**0**	0
10	50	9.9977×10^{-3}	1.1718×10^{-2}	**0**	0	**0**	0	**0**	0

Table 13 Mean, the standard deviation (SD) of functions with 100 dimension.

F	*ABC*			*PS-ABC*		*PS-ABCII*		*Multi-pop-ABC*	
	Dim	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
f1	100	4.9461×10^{-3}	1.1389×10^{-2}	8.3417×10^{-47}	4.5689×10^{-46}	**0**	0	**0**	0
f2	100	2.7814×10^{-1}	4.0035×10^{-1}	**0**	0	**0**	0	**0**	0
f3	100	1.8854×10^{5}	2.1886×10^{4}	1.3544×10^{5}	1.2851×10^{4}	5.4823×10^{5}	1.0256×10^{5}	$\mathbf{1.4211 \times 10^{4}}$	1.0937×10^{4}
f4	100	8.2376×10^{1}	3.0440	7.2160×10^{1}	4.0371	**0**	0	**0**	0
f5	100	3.3118×10^{2}	3.8309×10^{2}	2.0376×10^{2}	6.7028×10^{1}	9.8590×10^{1}	1.5702×10^{-1}	$\mathbf{4.1781 \times 10^{1}}$	1.2011×10^{-1}
f6	100	1.5950	3.2657×10^{-1}	2.2021×10^{-1}	4.1119×10^{-2}	1.6151×10^{-3}	3.2646×10^{-3}	$\mathbf{1.1260 \times 10^{-3}}$	2.9615×10^{-3}
f7	100	-34413.8	5.0878×10^{2}	-39976.6	3.3634×10^{2}	-37405.7	5.5665×10^{2}	-40182.4	2.6738×10^{2}
f8	100	8.5540×10^{1}	1.1018×10^{1}	**0**	0	**0**	0	**0**	0
f9	100	3.8186	3.6198×10^{-1}	2.3270×10^{-14}	1.2259×10^{-13}	8.8817×10^{-16}	0	0	0
10	100	1.4344×10^{-1}	1.3282×10^{-1}	1.6904×10^{-3}	6.4786×10^{-3}	**0**	0	**0**	0

5. Conclusion

This paper has presented a modified artificial bee colony algorithm for dynamic optimization problems. The aims of our modifications were to enhance the capability of the algorithm to efficiently deal with DOPs. We first integrated it with a multi-population method to scatter the solution over the search process so that they can search and track the optimum solution simultaneously. An adaptive multi-population was also proposed to adaptively change the number of sub-populations based on the problem change strength. In addition, a population clearing scheme was proposed to remove redundant solutions in the population. To evaluate the performance of the proposed algorithm, experimental tests were carried out using the moving peaks benchmark DOP, with a different number of peaks. Comparisons were carried out between the proposed algorithm, the basic ABC and state of the art methods. The results demonstrated that the proposed algorithm outperforms basic ABC on all tested scenarios. It also produced better results than the state of the art methods on many scenarios, indicating that the proposed algorithm is an effective method for the DOP.

Acknowledgements

This work was supported by the Ministry of Education, Malaysia (FRGS/1/2015/ICT02/UKM/01/2) and the Universiti Kebangsaan Malaysia (DIP-2012-15).

References

1. Jin, Y. and J. Branke, *Evolutionary optimization in uncertain environments-a survey*. IEEE Transactions on Evolutionary Computation, 2005. **9**(3): p. 303-317.
2. Nguyen, T.T., S. Yang, and J. Branke, *Evolutionary dynamic optimization: A survey of the state of the art*. Swarm and Evolutionary Computation, 2012. **6**: p. 1-24.
3. Li, C., T.T. Nguyen, M. Yang, S. Yang, and S. Zeng, *Multi-population methods in unconstrained continuous dynamic environments: The challenges*. Information Sciences, 2015. **296**: p. 95-118.
4. Hatzakis, I. and D. Wallace. *Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach*. in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006. p. 1201-1208. ACM.
5. Simões, A. and E. Costa. *Improving prediction in evolutionary algorithms for dynamic environments*. in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. 2009. p. 875-882. ACM.
6. Branke, J. and D.C. Mattfeld, *Anticipation and flexibility in dynamic scheduling*. International Journal of Production Research, 2005. **43**(15): p. 3103-3129.
7. Cruz, C., J.R. González, and D.A. Pelta, *Optimization in dynamic environments: a survey on problems, methods and measures*. Soft Computing, 2011. **15**(7): p. 1427-1448.
8. Branke, J. *Memory enhanced evolutionary algorithms for changing optimization problems*. in *In Congress on Evolutionary Computation CEC99*. 1999. 3:1875-1882.
9. Branke, J., *Evolutionary optimization in dynamic environments*. Vol. 3. 2012: Springer Science & Business Media.
10. Yang, S. *On the design of diploid genetic algorithms for problem optimization in dynamic environments*. in *IEEE Congress on Evolutionary Computation CEC 2006*. pp. 1362-1369.
11. Daneshyari, M. and G.G. Yen. *Dynamic optimization using cultural based PSO*. in *IEEE Congress on Evolutionary Computation CEC 2011* pp. 509-516.
12. Grefenstette, J.J. *Evolvability in dynamic fitness landscapes: A genetic algorithm approach*. in *IEEE Congress on Evolutionary Computation CEC 1999*. Vol. 3, pp.2031 -2038 1999.
13. Ursem, R.K. *Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments*. in *In GECCO, pp. 19-26*. 2000.
14. Branke, J., T. Kaußler, C. Schmidt, and H. Schmeck, *A multi-population approach to dynamic optimization problems*. Adaptive computing in design and manufacturing,2000:, 2000: p. 299-308.
15. Blackwell, T. and J. Branke, *Multi-swarm optimization in dynamic environments*. Applications of Evolutionary Computing, 2004: p. 489-500.

16. Mendes, R. and A.S. Mohais. *DynDE: a differential evolution for dynamic optimization problems*. in *IEEE Congress on Evolutionary Computation CEC 2005* vol. 3, pp. 2808-2815.
17. Li, C. and S. Yang. *Fast multi-swarm optimization for dynamic optimization problems*. in *Fourth International Conference on Natural Computation, 2008. ICNC'08*. . vol. 7, pp. 624-628. IEEE.
18. Yang, S. and C. Li, *A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments*. IEEE Transactions on Evolutionary Computation, , 2010. **14**(6): p. 959-974.
19. Turky, A.M. and S. Abdullah, *A multi-population electromagnetic algorithm for dynamic optimisation problems*. Applied Soft Computing, 2014. **22**(1): p. 474-482.
20. Turky, A.M. and S. Abdullah, *A multi-population harmony search algorithm with external archive for dynamic optimization problems*. Information Sciences, 2014. **272**(1): p. 84-95.
21. Sharifi, A., J.K. Kordestani, M. Mahdaviani, and M.R. Meybodi, *A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems*. Applied Soft Computing, 2015. **32**(1): p. 432-448.
22. Li, C., T.T. Nguyen, M. Yang, S. Yang, and S. Zeng, *Multi-population methods in unconstrained continuous dynamic environments: The challenges*. Information Sciences, 2015. **296**(1): p. 95-118.
23. Karaboga, D., *An idea based on honey bee swarm for numerical optimization*. 2005, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.
24. Karaboga, D. and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*. Journal of global optimization, 2007. **39**(3): p. 459-471.
25. Branke, J. and H. Schmeck, *Designing evolutionary algorithms for dynamic optimization problems*, in *Advances in evolutionary computing*. 2003, Springer. p. 239-262.
26. Branke, J., T. Kaußler, C. Smidt, and H. Schmeck, *A multi-population approach to dynamic optimization problems*, in *Evolutionary Design and Manufacture*. 2000, Springer. p. 299-307.
27. Blackwell, T. and J. Branke, *Multiswarms, exclusion, and anti-convergence in dynamic environments*. IEEE Transactions on Evolutionary Computation, 2006. **10**(4): p. 459-472.
28. Du Plessis, M.C. and A.P. Engelbrecht, *Using competitive population evaluation in a differential evolution algorithm for dynamic environments*. European Journal of Operational Research, 2012. **218**(1): p. 7-20.
29. Du Plessis, M.C. and A.P. Engelbrecht, *Differential evolution for dynamic environments with unknown numbers of optima*. Journal of Global Optimization, 2013. **55**(1): p. 73-99.
30. Moser, I. and R. Chiong, *Dynamic function optimisation with hybridised extremal dynamics*. Memetic Computing, 2010. **2**(2): p. 137-148.
31. Nickabadi, A., M.M. Ebadzadeh, and R. Safabakhsh, *A competitive clustering particle swarm optimizer for dynamic optimization problems*. Swarm Intelligence, 2012. **6**(3): p. 177-206.
32. Sharifi, A., J.K. Kordestani, M. Mahdaviani, and M.R. Meybodi, *A novel hybrid adaptive collaborative approach based on particle swarm optimization and local*

search for dynamic optimization problems. Applied Soft Computing, 2015. **32**: p. 432-448.

33. García, S., A. Fernández, J. Luengo, and F. Herrera, *Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power*. Information Sciences, 2010. **180**(10): p. 2044-2064.
34. Li, G., P. Niu, Y. Ma, H. Wang, and W. Zhang, *Tuning extreme learning machine by an improved artificial bee colony to model and optimize the boiler efficiency*. Knowledge-Based Systems, 2014. **67**: p. 278-289.
35. Cui, H., J. Feng, J. Guo, and T. Wang, *A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction*. Knowledge-Based Systems, 2015. **88**: p. 195-209.
36. Mitić, M., N. Vuković, M. Petrović, and Z. Miljković, *Chaotic fruit fly optimization algorithm*. Knowledge-Based Systems, 2015. **89**: p. 446-458.
37. Yeh, W.-C., *An improved simplified swarm optimization*. Knowledge-Based Systems, 2015. **82**: p. 60-69.