Improved Spiral Dynamics and Artificial Bee Colony Algorithms with Application to Engineering Problems

By

Mohd Ruzaini Bin Hashim

Thesis submitted to the University of Sheffield for the degree of
Doctor of philosophy

Department of Automatic Control and Systems Engineering
The University of Sheffield
United Kingdom

April 2018

Abstract

The main focus of this research is to develop improved version of spiral dynamic algorithm (SDA) and srtificial bee colony algorithm (ABC) for solving various kinds of optimization problems. There are eight new algorithms developed in this research based on ABC and SDA. The first modification is on the initial distributions based on chaotic maps trajectory, random and opposition based learning. This adaptive initial distribution is used in all proposed algorithms. Second modification is to use spiral radius and chaotic rotational angle in modified SDA. Furher modifications include proposition of three adaptive ABC algorithms based on the modification of step size using exponential, linear and combination of both linear and exponential functions. Investigations have shown that SDA is a fast and simple algorithm but is subject to getting trapped in local optima and it lacks diversity in the search, while ABC is able to get accurate output at the expense of high computational time and slow convergence. Thus, the research further embarks on hybridisation of SDA and ABC, taking advantage of capabilities of both algorithms and three hybrid algorithms are thus proposed. The performances of the proposed algorithms are evaluated and assessed in single objective, multi-objective type and in practical optimization problems. Statistical and significant tests are used in the evaluations. Furthermore, comparative assessments of the performances of the proposed algorithms are carried out with their predecessor algorithms. The results show that the proposed algorithms outperform their predecessor algorithms with high accuracy and fast convergence speed.

Acknowledgements

All praise to Allah S.W.T, All mighty god. There is no god but Allah and Muhammad is his messenger. Thank you for giving me strength, ideas and good health. With His will, I have managed to complete my thesis and end my PHD journey successfully. Thank you Allah.

Many Thanks to my supervisor Professor Dr M. Osman Tokhi for his encouragement, guidance and support from beginning until my end of studies. You are the best supervisor I had and I will always remember your kindness. To my supervisor Professor Dr Mahdi Mahfouf, thank you for your support and guidance.

I would like to express my love and many thank you to my wife Ruzanna Mohamad Yunus and my children Aisyah Ramdhani, Asyraf Ramadhan and Airis Melissa, without your love, support and understanding I would not be able to complete this great journey. I also want to say thank you to Dr Hyriel Kasdirin, and Mrs Norafizah Abbas and all group members who supported me when I needed helps and gave lots of ideas and input to me during my study in ACSE. You are the best friends I have.

Many thanks also to my sponsor Ministry of higher education (MOHE) and Universiti Teknikal Malaysia Melaka for giving me a chance and opportunity to pursue my PHD study at University of Sheffield.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

This chapter presents an overview of the research is conducted. Background to the research is initially discussed to highlight the research issues and the problem statements. Then, a briefly description of bio-inspired algorithms is given along with the research objectives and the research contributions.

1.2 Research background

Optimization is a technique of finding the best solution to a problem under given circumstances [1]. A practical example is to find the best possible route for a salesman in order to have maximum sales with minimum cost. Thus, the main goal of the optimisation is to minimise the cost (effort) and to get maximum profit (benefit).

Mathematically, strategies are devised so that finding optimum solution is tailored to searching for either maximum or minimum point in the search space. These mathematically defined landscapes are categorised into multimodal and unimodal types. A unimodal landscape is characterised by having only one optimum (minimum or maximum) point. A multimodal landscape, on the other hand, has more than one optimum point, among which one is expected to be the global optimum. Thus, an optimisation strategy is formulated to search for and find the global optimum point in the landscape, based on fitness evaluation of the search point, referred to as the objective function, formulated as:-

$$\text{Minimize or maximize } f(x) = [f_1(x), f_2(x), \dots\dots, f_m(x)] \text{ , where } m \geq 1$$

$$\text{Subject to}$$

$$g_k(x) \geq 0, \quad k = 1,2, \dots., K$$

$$h_{kj}(x) \geq 0, \quad kj = 1,2, \dots., L$$

where, x are parameter variables in a parameter space. g_k is function of inequality constraints, h_{kj} is function of equality constraints, K is number of inequality constraints, and

L is number of equality constraints. If m =1 the formulation becomes single-objective problem and for m greater than 1 the formulation becomes a multi-objective problem and it is called multi-objectives problems [2][3].

Early developments of optimisation algorithms are mainly gradient based [4]. Newton' method [5] and line search [6] are examples of gradient based algorithms. The advantage of these algorithms is that they are fast in computational time because of their simple structure, which make them suitable to handle continuous problems, convex and simple nonlinearity problems. However due to their search strategy, it is difficult to find the global optimum in a multimodal problem and in dealing with real world problems. These algorithms easily get trapped in local optima and hence do not achieve accurate solution [7]. Thus, in such situations, an approach to achieve optimum solution is desired.

In recent years, optimisation techniques based on the adaptation of natural phenomena and biological behaviour referred to as evolutionary algorithms have become popular among researchers. These algorithms are gradient free and do not need differentiation operation to perform search. Thus, they can be used in many optimisation problems and are able to give good quality of solution in each iteration. Nature-inspired and bio-inspired optimization algorithms can be categorized as metaheuristic type. A metaheuristic algorithm is a method of solving general problems [8] using the structure of heuristic optimisation strategies and it may result in efficient way to perform the search. In a metaheuristic algorithm, the search of unknown optimal point in the search space (feasible area) is done in random manner and certain parameters of the algorithm need to be set in initial stages by the user in order for the algorithm to work properly. The most important factor that gives the metaheuristic optimization merit in solving problems is a balance strategy between exploration and exploitation. Exploration in the optimisation context is the phase for search of new solution in the unexplored areas within a feasible region. It can increase the chance to find good solution (not guaranteed). While exploitation is the process to focus the search in small area within which a good solution has been found in the exploration phase. Too much exploration can increase the convergence speed but it may result in the algorithm getting trapped at local optimum and may not improve the solution. Searching a solution based on a previously discovered solution during exploitation phase can increase the chance of finding a much better solution. However, extensive exploitation will result in slower convergence toward the global optimum. In reality, there is no guarantee that the balanced strategies between exploration and exploitation can produce a good solution and fast convergence speed because there will be trade-off between both [9][10].

Evolutionary algorithms (EAs) have been widely used by many researchers for obtaining optimal solutions to various problems of single objective problem and multi objective types. It also has good potential in solving real problems [11][12]. In this research, EAs are classified into two categories, namely bio-inspired and nature inspired algorithms. Bio-inspired algorithms are based on the behaviour of living organisms such as bacteria, animals, birds, insects, etc. while a nature-inspired algorithm is based on natural phenomena such as spiral pattern [13]. The advantage of EA is that it has the ability to adapt and organize itself during exploration and exploitation [14][15][16]. EA also uses randomisation and deterministic method during exploration. This kind of approach enables the algorithm to reduce the probability of getting trapped at local optima [17][18].

Examples of biological inspired algorithm are :- Genetic algorithms (GAs) are inspired from the process of human genetic evolution [19][20], artificial bee colony algorithm (ABC) is based on the behaviour of real honey bee foraging the flower nectar, particle swarm optimization (PSO) is based on the communication behaviour of bird of fish flocking [21],[22]. Bat algorithm (BA) is based on the echolocation behaviour of bats [23]. bacteria foraging algorithm (BFA) are inspired from Escherichia coli bacteria foraging pattern [24]. Dorigo proposed ant colony optimization (ACO) algorithms based on the movement of ants [25] [26], Yang [27] developed firefly algorithm (FA) based on the light pattern produced by swarm of firefly. Examples of nature-inspired algorithms are lightning search algorithm (LSA) based on natural phenomenon of lightning [28], galaxy-based search algorithm [29], harmony search algorithm based on musical phenomena [30][31].

Real world applications are in general complex and solving such problems using optimisation strategies is challenging. Moreover, there is no guarantee that the solution generated by the algorithm can be the global optimum. The high number of dimensions and nonlinearity of the problem can reduce the chance to find the best possible solution. Real world applications in which optimisation algorithms have been used include optimisation of fuzzy control of wheel chair lifting and balancing [32], optimization of PID controller for automatic voltage regulation [33] and optimization of design parameters of a composite structure [34]. Further applications include stock price prediction [35] and optimisation of drug during therapy [36]. A review of the literature has revealed the importance of optimisation algorithms in solving real world problems. However, it is not possible to solve all types of problems with the developed algorithms, and hence further research is needed to develop enhanced and powerful strategies for solving a range of single and multi-objective optimisation problems. Thus, the research presented in this thesis has embarked on development of improved algorithms based on artificial bee colony (ABC) optimisation and spiral dynamic optimisation algorithm (SDA).

1.3 Artificial bee colony optimisation algorithm

The ABC algorithm was initially proposed by Karoboga [37]. The algorithm is inspired by the bee colony and their behaviour during food foraging. Because of the ABC structure, it has good strategies during exploration, is able to memorise important information, has multi character and is able to perform local search efficiently [16]. Detailed description of ABC is given in chapter 2.

1.4 Spiral dynamic optimisation algorithm

Spiral dynamic algorithm (SDA) is based on the spiral pattern such as in tornados, galaxy pattern, nautilus shell [38]. SDA has a simple structure with only 2 control parameters. This simple structure and dynamic step size enables SDA to execute the search in short time. However, in high dimension problems, SDA easy trapped at local optimum. SDA is described in detail in chapter 2.

1.5 Research aim and objectives

The main target of the research is to develop improved versions of SDA and ABC optimisation algorithms and to test and validate their performances in several benchmark functions and applications. The SDA may be improved in terms of accuracy by taking advantage of its fast computational time and fast convergence but it easily gets trapped at local optima when dealing with various types of problems. The ABC, on the other hand, is able to give accurate result in solving complex problems, but suffers from slow convergence and slow computational speed. Based on the advantages of both algorithms, there is great potential to combine SDA and ABC either in integrative or collaborative manner, to lead to significant improvement for both algorithms in terms of fitness accuracy and convergence speed. For assessment of robustness and accuracy, the proposed algorithms will be subjected to tests in various optimisation problems. Firstly, the proposed algorithms will go through extensive testing using well known single and multi-objective benchmark functions such as sphere function or Ackley function. These benchmark functions have various types of fitness landscape from unimodal to multimodal. The algorithms will also be tested using 30 benchmark functions created during 2014 Congress on Evolutionary Computation (CEC2014). Moreover, the proposed algorithms are further tested in several practical applications.

The objectives of the research are as follows:-

1) To propose modified versions of ABC and SDA algorithms for better performance in terms of convergence speed and global optimum value in comparison to their classical predecessor algorithms.
2) To investigate the performances of the proposed ABC and SDA with different types of numerical benchmark functions, and carry out a comparative assessment of their performances with those of their predecessor algorithms.
3) To test and investigate the performances of the proposed algorithms in a set of practical applications, and to assess the results in comparison with performances of their predecessor algorithms.

1.6 Research contributions and publications

The contributions of the research include the following:-

Contribution 1: Modified spiral dynamic algorithms

To ensure good quality of solutions is placed in the search space during initial distribution of the population, the combination of chaotic, random and opposition-based learning distribution has been proposed. This novel approach enables to produce initial solution near to the optimal value and leads the algorithm to perform the search relatively faster. By changed the fix value of spiral radius and spiral angle with chaotic maps pattern it will lead this modified algorithm to perform better than the original SDA.

Contribution 2: Modified artificial bee colony algorithms

The search strategies of original ABC have been modified to enhance the exploration capabilities of onlooker bee and employed bee. The bee movement in search area in the onlooker and employed stages has been modified to let the bees move to the food source in exponential manner and in a linear trajectory motion.

Contribution 3: Hybrid spiral dynamic and artificial bee colony algorithms

Three proposed hybrid version of algorithms is created based on the combination of ABC and SDA. The first hybrid algorithm incorporates the reset strategy of ABC in scout bee stages into SDA and is able to reset the non-performing point in SDA. While in the second proposed hybrid version, the ABC and SDA algorithms are placed in sequential order. The ABC will execute first to perform the exploration and then SDA will take over to perform the exploitation. In third and last proposed hybrid version, adopts the ABC search strategy into SDA where it will perform local search around each spiral point

All of the proposed algorithms has been tested using several benchmark functions and the results have been compared with SDA and ABC. The algorithm has also been ranked using Kruskal-Willis test and subjected to the reliability of performance test.

The following publications arising from this research have been produced to date and further publications are in progress.

1. Hashim, M. R., Hyreil A. K, Tokhi, M. O. (2017). *Optimal Tuning of PD controller using Modified Artificial Bee Colony Algorithm*. Journal of Telecommunication, Electronic and Computer Engineering (JTEC) - accepted

2. Hashim, M. R., Tokhi, M. O. (2016). *Greedy Spiral Dynamic Algorithm with application to controller design*. IEEE Conference on Systems, Process and Control (ICSPC 2016), Melaka, Malaysia, 16-17 December 2016 10.1109/SPC.2016.7920698

3. Hashim, M. R., Tokhi, M. O. (2016). *Enhanced Chaotic Spiral Dynamic Algorithm with Application to Controller Design*. IEEE 6th International Conference on Power and Energy (PECON 2016). Melaka, Malaysia .18-29 November 2016

4. Hashim, M. R., Tokhi, M. O. (2016). *Chaotic Spiral Dynamic Algorithm*. 19th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, *London, UK, 12 –14 September 2016, pp. 551-558.*

5. Hashim, M. R., Tokhi, M. O. (2016). *Control of a Single Link Flexible Manipulator System Using Enhanced Chaotic Spiral Dynamic Algorithm*. Poster session presented at the ACSE PGR Symposium 2016. Department of Automatic Control and Systems Engineering, the University of Sheffield, United Kingdom.
6. Hashim, M. R., Hyreil A. K., Tokhi, M. O. (2015). *Control of a Single Link Flexible Manipulator System Using Simple Modified Artificial Bee Colony Optimisation Algorithm*. Poster session presented at the ACSE PGR Symposium 2015. Department of Automatic Control and Systems Engineering, the University of Sheffield, United Kingdom.

1.7 Organisation of the thesis

The thesis has been organised as follows:

Chapter 1:- This chapter presents overall introduction of research background, problem statement, aims and objective, methodology of the research, contribution and list of publication.

Chapter 2:- This chapter contains description and overview of original artificial bee colony optimisation algorithm and spiral dynamic algorithm. The development of ABC and SDA is also discussed in this chapter.

Chapter 3:- This chapter introduces the development of the proposed algorithms based on ABC and SDA.

Chapter 4:- This chapter presents the testing of the original and proposed algorithms using various benchmark test functions. The problems used in the tests include unconstrained single objective and constrained single objective benchmark functions. Performances of the algorithm are validated with statistical analysis.

Chapter 5:- This chapter presents the testing of the original and proposed algorithms in multi objective problems. Performances of the algorithms are validated with several performance metrics.

Chapter 6:- This chapter presents testing of the algorithms on several engineering applications. The applications considered include tuning of PD fuzzy controller of a single link manipulator system and several practical applications.

Chapter 7:- This chapter summarises the works carried out and makes suggested recommendations for further work.

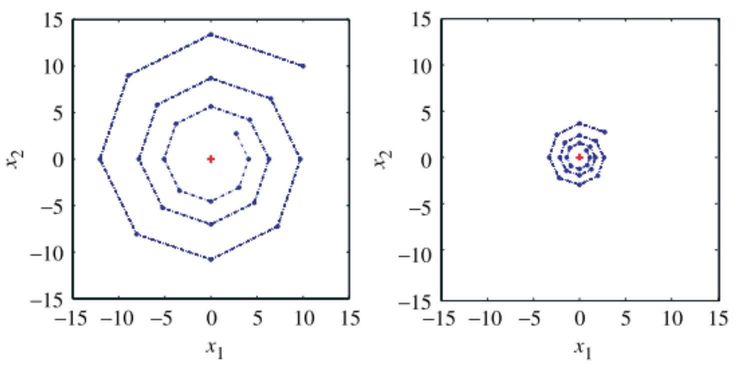
Chapter 2
Spiral Dynamic Algorithm and Artificial Bee Colony Optimisation Algorithm: An Overview

2.1 Introduction

Optimisation algorithms have been used in many areas for solving various types of problems. This chapter provides a description of spiral dynamic algorithm (SDA) and artificial bee colony algorithm (ABC). The SDA is based on natural evolution while ABC is based on the behaviour of honeybee foraging the honey. Recent modifications, variations and applications of SDA and ABC are also highlighted.

2.2 The spiral dynamic algorithm

In 2011, Spiral dynamic algorithm was introduced by Tamura and Yasuda [39][38] inspired by the natural spiral patterns in nature such as shape of DNA molecule, spiral pattern of tornado, hurricanes and galaxy. They started with two dimensional problems based on the model created from logarithmic spiral. SDA has a simple structure, relatively low computational time and is easy to program. The algorithm has diversification in early stage of the search where the aim in this stage is to perform global exploration in the search area in order to find better solution. Following the exploration stage the algorithm will perform the search for better possible solution around the good solution found during early search phase. This intensive search toward the best possible solution is called intensification. In SDA, by using logarithmic spiral as a model, the diversification and intensification strategies can be fulfilled in naturally because the spiral movements exponentially converge to the centre of the spiral [40]. Adaptation of logarithmic spiral in terms of exploration and exploitation can be shown in Figure 2.1 where it clearly shows that the diversification happens in early stage and the small step size becomes smaller in the later stage during which intensification occurs. The SDA search begins from initial point to next point in anti-clock wise direction and continues toward the centre of spiral at the inner layer. In this process, the step size will reduce gradually as the search points approaches the centre of spiral. This will allow the SDA to reach the global optimum for different types of problems of uni-modal and multi modal nature. The SDA also converges faster because in each iteration it will carry the best fitness and this fitness gives guidance to the spiral search to move toward the optimal point in that iteration.



(a) Early spiral (25 points) *(b) after Spiral (25 points)*

Figure 2.1 :- Spiral Trajectory (adopted from [38]*)*

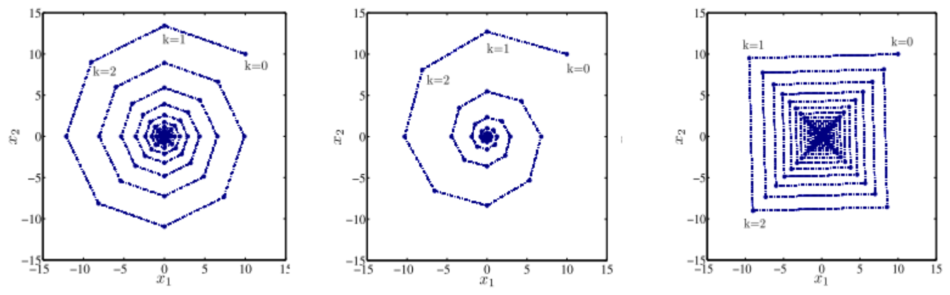
The mathematical model for n-dimension of SDA is defined in equation (2.1)

$$x_i(k+1) = S_n(r,\theta)x_i(k) - [S_n(r,\theta) - I_n]x^*\,, i = 1,2,\dots\dots, m. \tag{2.1}$$

where θ is rotational angle between 0 and 2π, I_n is identity matrix , x^* is centre of spiral, k is iteration number, r is spiral radius ranging between 0 and 1 while, $S_n(r,\theta) = rR^{(n)}(\theta_{1,2},\theta_{1,3},\dots\dots\theta_{n,n-1})$, $R^{(n)}$ is composition rotation matrix, where $R^{(n)}(\theta_{1,2},\theta_{1,3},\dots\dots\theta_{n,n-1})$ is rotation $n \times n$ matrix. The general mathematical n dimensional spiral model using $R^{(n)}(\theta_{1,2},\theta_{1,3},\dots\dots\theta_{n,n-1})$ can be expressed as:-

$$R^{(n)}(\theta_{1,2},\theta_{1,3},\dots\dots\theta_{n,n-1}) = \prod_{i=1}^{n-1}\left(\prod_{j=1}^{i} R^{(n)}_{n-i,n+1-j}\left(\theta_{n-i,n+1-j}\right)\right) \tag{2.2}$$

Figure 2.2 shows that the values of r and θ can change the behaviour of spiral model.



Case (1) , r=0.95, $\theta = {}^{\pi}/_{4}$ Case (2) , r=0.90, $\theta = {}^{\pi}/_{4}$ Case (3) , r=0.95, $\theta = {}^{\pi}/_{2}$

Figure 2.2 : Spiral model behaviour (adopted from [38]*)*

Generally, the SDA performance is based on the parameters r and θ. It is possible that the search point may settle at a local optimum and increasing the number iteration may not help the algorithm to converge to global optimum or better point. A further issue that arises is that in case of high dimension problems the computational time increases as sizes of the matric in the spiral model become bigger. Algorithm 2.1 shows the pseudo code of the spiral dynamic algorithm.

Algorithm 2.1 : Spiral dynamic algorithm

```
1:  Objective function of $f(x_d)$, set search point,$m$ , spiral radius;$r$, spiral angle, $\theta$,
    upper and lower boundary of search space, maximum iteration $MCN$
2:  **Initialisation:**
3:  Set initial points $x_i(0) \in R^n, i = 1,2,\dots,m$ in the search space at random
4:  Choose the centre of spiral $x^*$ as $x^* = x_{ig}(0)$, $ig =$
5:  **while** iteration,$k \leq$ maximum iteration, $MCN$ **do**
6:     **for all** $i$ to $m$ **do**
7:        Updating $x_i$ ; $x_i(k+1) = S_n(r,\theta)x_i(k) - [S_n(r,\theta) - I_n]x^*$ , $i =$
    $1,2,\dots\dots,m$.
8:     **End for**
9:     Updating $x^*$; $x^* = x_{i_g}(k+1), i_g = \arg min_i f\big(x_i(k+1)\big),\ \ i = 1,2,\dots,m$
10:    $k = k + 1$
11: **end while**
```

2.3 Variations of Spiral Dynamic Algorithm

Approaches adopted by researchers on development of improved optimisation algorithms have included either by modification to search method or by combining two or more algorithms. The literature is, however, limited in the work on modification or improvement of SDA.

Tamura and Yasuda [38] have evaluated the performance of SDA in comparison to the SDA, PSO algorithm and DE algorithm in 4 types of benchmark functions with different spiral angles. They have shown, after 100 trials with different sizes of dimensions, the SDA outperformed DE and PSO with the spiral angle $\pi/_2$.

SDA has been applied to solving various real problems. It has simple structure and because of this, the computational time is relatively short. However, it has been noted that the algorithm can easily get trapped in local optima when exposed to solving high dimension problems [41][42][43]. There has been an attempt to resolve this issue. For example, Nasir et al. [44] proposed an adaptive formulation by varying the size of radius and displacement of spiral model in order to resolve this issue.

In the original SDA, the rotational angle and rotational radius remained unchanged throughout the searching process. Nasir et al. [45] proposed to vary the values of rotational angle and radius according linear, quadratic and exponential function. They also used fuzzy logic method as a non-mathematical approach to find a good point in the search space by establishing relationship between fitness values and radius of spiral. The algorithm has been tested using several unimodal and also multimodal benchmark functions with different value of dimensions. In a further work Nasir et al. [44], introduced modified rotational radius and also rotational angle to replace the standard parameter. The modified angle and radius may change to different value in each iteration depending on the deviation of fitness value and global fitness value. This approach has been used to optimise parameters of an autoregressive model of a flexible manipulator system. The results show that the proposed algorithm outperformed the original SDA.

Many researchers have adopted a strategy of hybridisation of an original algorithm with other algorithms to increase their performance. For example Nasir et al. [46] have hybridised SDA and BFA. This algorithm can increase the exploration capability by letting bacteria swim and tumble in spiral pattern during exploration. The authors have used this algorithm to optimise PID parameters for hub angle of flexible manipulator system and the results show that the algorithm can tune the controller very well. However, the authors report that algorithm performs well in solving low dimension problems.

With the adaptive methods in [45][44] , the values of radius and rotational angle are not constant but vary in each iteration and thus result in improved performance. Thus based on the idea of varying the step size or radius/angle of spiral, Nasir et al. [47] proposed a hybrid scheme by combining SDA and BFA. They proposed two different strategies based on how bacteria swim. In the first strategy, the bacteria swim in spiral manner toward optimum value while in the second strategy a random approach is used. In random approach, the bacteria swim freely in current position, and this can increase the chance of finding optimum value but in spiral swimming type, the direction of search are predetermined by spiral direction. The results of the test using non-parametric Freidman and Wilcoxon on eight benchmark functions, show that the spiral swimming type perform better than random swimming type.

Nasir et al. [41] incorporated an elimination dispersal method of BFA in SDA to increase the exploration capability of the algorithm. By increasing the potential to find more good solutions in the search space, the problem of settling in local optimum can be reduced. Various benchmark function tests and analyses have been used to evaluate the strategy. The authors have also proposed hybrid strategy between SDA and BFA by making bacteria to swim in spiral pattern [42]. They used the method to optimize a single link manipulator system controller. They developed two types of hybrid strategy by incorporating SDA into BFA where the integration is in a sequential manner. The execution is done by execute the SDA or BFA first then the other will follow. They have reported that this strategy can overcome the high dimension problem facing by the algorithm in [46]. The first proposed hybrid algorithm can give better fitness accuracy but the computational time is the major drawback in this strategy because the time taken to execute both SDA and BFA stages is longer compared with original SDA. The algorithm has been developed by placing spiral search pattern during exploration in the chemotaxis phase. During the search, the bacteria are directed in spiral manner toward the best possible solution in every iteration. For the 2nd hybrid scheme reported in [42], the proposed algorithm is an improved version of the 1st scheme. According to the author, the improvement of computational time can be achieved by replacing dispersal, elimination and reproduction in BFA chemotaxis stage with spiral model. Thus, in the first stage, bacteria chemotaxis is invoked to perform the exploration and in the next stage the exploitation occurs based on the spiral movement.

To enhance the performance of SDA Nasir and Tokhi [41] focused on search strategy of the final solution in each iteration by introducing an additional step called elimination and dispersal adopted from BFA. The main structure of this algorithm is the same as the original SDA but at the end of the iteration, the elimination and dispersal is triggered. By doing so, the problem of getting trapped at local optimum is eliminated. The results of comparative tests of the algorithm with original SDA and BFA on four benchmark functions and the result show that the strategy can increase the performance of SDA.

Tsai et al. [48] have reported a modified version of SDA, referred to as distributed spiral optimisation (dSO) for data mining. The introduced an oscillation stage adopted from genetic algorithm mutation operator to reduce the possibility of SDA getting trapped at local optima. They tested the algorithm in solving clustering problems, and the results show that dSO is able to solve clustering problems.

Parallel manipulators are known to have high accuracy and able to cater bigger loads in comparison to serial manipulators, but dealing with singularity is a matter of concern. To solve this issue, Ashok [49] used SDA and PSO, where he aimed to tried to get better slider position of the system and the results show that SDA outperform PSO.

There have also been attempts to use the SDA algorithm in solving multi objective IIR filter design problems with the objective to minimize the time delay and to get matching frequency response [50]. From the results, show that SDA was able to optimise the digital filter and match the desired specification setting.

2.4 Artificial bee colony optimization algorithms

The artificial bee colony algorithm was introduced by Karaboga [37] based on the behaviour of real honey bee in finding the flower nectar and their knowledge sharing about the quality of nectar among them. Karaboga also developed ABC based on developed model of honey bee colony by [51] . To find the optimum point, the algorithm moves the possible solutions into the search space as inspired by the movement of real honey bee foraging the nectar and based on the information about the nectar shared by other bees [52]. In principle, the task in the bee hive is performed by designated individual bees; the employed bee, onlooker bee and scout bee [53]. They have different tasks to maximize the amount of nectar located in the hive by performing the self-organization and efficient of labour delegation for each bee. Robustness, easy to use and flexibility are the main advantages of ABC [52]. The ABC is requires only three control parameters, namely bee size, maximum cycle number and limit trigger [54][55].

This section will explain the basic concept of real honey bee foraging the nectar and theoretical concept of classical ABC algorithm including the process step and pseudo-code. This section further presents recent developments in ABC including recent modification, and hybridisation.

2.4.1 Classical artificial bee colony algorithm

In classical ABC algorithm, the quality of the food source (flower nectar) is represented by the value of fitness and is be based on its location/position. Basically nectar from flower is the main food source for Honey Bee. The quality or the best food depends on many factors such as the location, distance between hive and food source, the nectar richness and also the level of difficulties during foraging. The food source is a possible solution vector of the problem where the position of the food source will be in a D dimensional space or search space. The computational agent in ABC algorithm can be divided into three groups: onlooker bees, employed bees and scout bees. The employed bees are those that have a specific food source that they are presently exploiting or foraging. They also bring in together the information about the distance and direction of the food source and also the quality of the food source. The onlooker bee waits in the dance floor inside the hive for the information brought by employed bee and will choose the food source based on that information. The information exchange between employed bee and onlooker bee regarding the quality of the food sources will take place in the dance floor inside the hive. The pattern of the dance will show the quality level of the nectar. The name of this dance is waggle dance. The higher quality of nectar can attract more onlooker bees to go to the corresponding food source. The role of scout bee is to perform random search to find a new food source. In the ABC algorithm, the number of employed bees is equal to the number of onlooker bees, and that is equal to the half of food source size. The steps involved in searching for the best possible solution in ABC can be described as follows [34][56] [57].

The initial population is randomly generated with an n -dimensional vector and distributed in the search space by scout bees and this stage can be referred to as exploration phase. The x_{ij} is the i th food source for the n –dimensional vector in the population. The distribution are represented as

$$x_{ij} = ub_j + rand(0,1)(ub_j - lb_j) \qquad (2.3)$$

where food source, $i = 1,2,\dots SN,\ j = 1,2,\dots.,n.$ ub_j and lb_j are the upper and lower boundaries of search space for the dimension j and SN is size of population. The fitness of population is evaluated and each of the food source generated during this stage is randomly assign to employed bees.

The next step is exploitation phase by employed bees and onlooker bees. This phase is divided into two stages; employed bee stage and onlooker bee stage. In employed bee stage, after scout bee has performed exploration, the employed bees start to collect nectar and go back to the hive. The employed bees will dance in the dance floor to share the information about food source with onlooker bees. In this process, employed bees will produce modification to the solution (position) based on the visual information of the neighbour solution. The employed bees will check the quality of food source (fitness) and the modified solution. If the quality of food source is better compared to the previous one, the employed bees will remember that food source location and forget the previous one. But if the modified solution is not improved, the employed bees will keep the location of previous one in their memory and forget the current solution. These stages can be described as a comparison between two food sources visually by employed bees and they will create new position in their memory based on the fitness value. The modified position of food source in this stage is given by

$$v_{ij} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj}) \qquad (2.4)$$

where v_{ij} is new position of solution, x_{ij} is previous solution and x_{kj} is neighbour solution. with i & $k = 1,2, \dots SN,\ j = 1,2, \dots.,n.$ The neighbour position of solution x_k is randomly chosen and it must not be the same with position of solution x_i. The function of $rand(-1,1)$ in the equation is to control the position of neighbour food source within x_{ij}. The perturbation on the position x_{ij} will become smaller as the difference x_{ij} and x_{kj} getting smaller. Thus, the step size reduces when the search approaches the best solution in the search space.

After all of the employed bees have completed their task, they will share the information gathered during foraging with onlooker bees that wait in the dance floor in the hive. The onlooker bees will evaluate the information about the food source and will choose the best solution based on the highest recruiting probability (fitness value), p_i . The higher quality of nectar can attract more onlooker bees to go to the corresponding food source. The selection of probability of the solution in the original ABC is based on roulette wheel selection scheme. Fitness and recruiting probability can be calculated as

$$Fitness(i) = \begin{cases} \dfrac{1}{1+f(\bar{S}_i)}, & f(\bar{S}_i) \geq 0 \\ 1 + abs[f(\bar{S}_i)], & f(\bar{S}_i) < 0 \end{cases} \tag{2.5}$$

$$probability(i) = \frac{Fitness(i)}{\sum_{i=1}^{FS/2} Fitness(i)} \tag{2.6}$$

Based on 'roulette-wheel' selection method, the onlooker bee will follow one employed bee to the selected food source based on the value of recruiting probability. The onlooker bees will choose the highest value of probability because it is likely to have more chance to be optimized. This new selected food source position can be calculated and expressed using equation (2.4).

Abandonment of non-improvement food source will happen if the fitness value cannot be improved anymore by any food source position within the predetermined limit setting. In this stage the bee assigned at deserted food source position will change its role to scout bee. The scout bee will find new food source and replace the abandoned food source with the new one using

$$x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right) \tag{2.7}$$

The search process for the optimal value by employed and onlooker bees continue repeating, and the repetition will stop when the Maximum cycle number (MCN) has been reached. Algorithm 2.2 shows the pseudo code the main process of ABC [57].

Algorithm 2.2 : ABC algorithm

```
1:   Objective function of $f(x_d)$, Number of Food (NF), Number of Bee, Limit value,
     upper and lower boundary of search space, maximum iteration $MCN$
2:   **Initialization:**
3:   Set initial points $x_{ij} = LB_j + rand * (UB_j - LB_j)$ in the search space at random
4:   Evaluate the population fitness
5:   Iteration = 1
6:   while iteration, $k \leq$ maximum iteration, $MCN$ do
7:     for all $i = 1$ to $SN$ do
8:       Produced new food source using $v_{ij} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})$
9:       Evaluate the fitness of new food source $v_{ij}$
10:      Compare fitness between $x_{ij}$ and $v_{ij}$ and choose the better one. (greedy
         selection)
11:      If the solution is not improved $trial_i = trial_i + 1$
12:    End for
```

```
13:    Calculate the probability, $probability(i)$ using equation (2.6)
14:  end while
15   $t = 0, i = 1$
16:  repeat
17:      If random < $probability(i)$ then
18:          Produced new food source using $v_{ij} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})$
19:          Evaluate the fitness of new food source $v_{ij}$
20:          Compare fitness between $x_{ij}$ and $v_{ij}$ and choose the better one. (greedy
             selection)
21:          If the solution is not improved $trial_i = trial_i + 1$, otherwise $trial_i = 0$
22:          $t = t + 1$
23:      End if
24:  Until $(t = SN)$
25:  If $trial_i > limit$ then
26:      Replace abundant food source with a new food source using
                 $x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right)$
27:  End if
28:  Keep the best solution
29:  Iteration = iteration +1
30:  Until (iteration = MCN)
```

2.4.2 Comparison and variation techniques in artificial bee colony algorithm

Initially, ABC was designed to solve numerical problems [22]. The early stages of ABC, have included performance evaluations using different types of well-known benchmark functions. Comparisons with several evolutionary algorithms have also has been carried out. Performance of ABC have been compared with GA,PSO and PS-EA in optimizing multivariable functions [58]. Karaboga and Basturk [59] have used ABC to solve multi-dimensional problems and have assessed its performance in comparison with DE,EA and PSO. To demonstrate the robustness of ABC in dealing with different types of problems it has been tested with a large set of benchmark functions together with GA,DE,EA and PSO algorithms [60]. Krishnanand [61] compared performances of five types of optimization algorithms including ABC with five standard benchmark functions. The algorithms considered were ABC, PSO, IWO, GA and AI. Mala [62] compared the performances of ABC with ACO (Ant Colony Optimization) and the results showed that the ABC performed much better than ACO. Karaboga and Akay [63] investigated the effect of problem dimension on the performance of ABC,harmony search (HS), and Bee Algorithm (BA) in unimodal and multimodal benchmark functions.

To extend the performance evaluation of ABC, Karaboga and Basturk [64] were tested the ABC in solving constrained problems. The performances of ABC were tested in 13 different constrained benchmark functions and were compared with DE and PSO algorithms. The results show that ABC was able to handle the constrained problems. Mezura-Montes and colleagues [65][66] introduced an elitist ABC algorithms to tackle constrained real parameter problems. The potential of ABC in solving single objective problems in continuous search spaces has attracted many researchers to look for approaches to improve its convergence speed, processing time and to increase its accuracy. To improve ABC performance, some modifications have been proposed.

A new search mechanism based on fixed point theorem of contractive mapping in Banach spaces was proposed by Quan [67]. The method has shown improvement in convergence rate of ABC in solving the multimodal problems. The problem of premature convergence and easily getting trapped at local optimum was reported by Lee and Cai [68]. To solve this issue, they proposed a new diversity strategy that can give the balance between exploration and exploitation. Tsai et al. [69] have reported that the exploitation ability of ABC can be enhanced by incorporating universal gravitation into the movement of onlooker bees. Their proposed algorithm is referred to as interactive ABC (IABC). In the classical ABC, the movement of onlooker bee is limited to the selected nectar and random nectar. By using universal the gravitation concept, the relationship between the employed bees and onlooker bees can improve the exploitation capability of ABC. They have tested the performance of IABC in three well know benchmark functions and have reported increase in exploitation capability of the algorithm. Other researcher found that the ABC is good in exploration phases but perform poorly during exploitation [70] and have proposed modification to the search equation in order to improve the exploitation. By using the best solution value into searching process, Yi and He [71] proposed an enhanced version of ABC called novel ABC (NABC). They reported improved convergence speed in experiments using several test functions. As the roulette wheel selection may lead to premature convergence, Bao and Zing [72] replaced the roulette selection with three (3) different selection methods namely disruptive, tournament and rank selection. They reported that using those selection strategies lead to better search diversity and avoid premature convergence.

Another approach of modification that can be used to make ABC algorithm more accurate and fast, is to combine the classical ABC or modified ABC with other types of optimization algorithm. This combination can be called as a hybrid-ABC. One of the hybrid-ABC was introduced by Kang et al. [73] where they combined a Neldar and Mead method in structural inverse analysis. The approach can solve problems with high dependency of variables; combining the ABC with neldar and mead algorithm will help the ABC to adapt to complex surface topologies. Hybrid ABC - GA has been used to find the optimal PI tuning in a motor drive [74]. Hybrid artificial bee colony algorithm with bacterial foraging optimization algorithm has been presented by Zhong et al. [75], where they applied this technique in the exploitation stage of onlooker bee and employed bees and the results have shown significant improvement in term of convergence speed. Tsai et al. [76] have proposed a hybrid strategy combining cat swarm optimization (CSO) and ABC and have tested the performance of algorithm was tested using 5 benchmark functions. Li and Chan [77] have proposed a hybrid strategy between ABC and Recursive least square estimator (RLSE) in order to tune and train the complex neural fuzzy system design by them. El-Abd [78] used ABC to improve the personal best of PSO and this hybrid version was tested in CEC05 benchmark functions. Neelima and Murthy [79] have replaced the onlooker bee phase with random walk function in BAT algorithm. They reported that approach will enhance the exploration. Li et al. [80] incorporated variable neighbourhood search strategy into ABC and the result achieved enhanced performance in comparison to ABC in terms of convergence speed. Other hybrid implementations in ABC are shown in Table 2.1

Table 2.1:- ABC hybrid implementation

Algorithm	**Reference**
ABC + Quantum Evolutionary algorithm	[81] [82]
ABC + Genetic Algorithm	[83] [84]
ABC + greedy heuristic + local search	[85]
ABC + Greedy Randomized Adaptive Search Procedure (GARPS)	[86]
ABC + Greedy Sub tour Crossover	[87]
ABC + Disruptive selection	[88]
ABC + PSO	[89][78]
ABC + Hooke Jeeves pattern search	[90]
ABC + Cat Swarm (CSO)	[76]
ABC + Recursive least square estimator (RLSE)	[77]
ABC + BAT	[79]
ABC + variable Neighbourhood search	[80]

2.5 Applications

The ABC algorithm has been used in many applications. One of the application is controllers design. Karaboga and Akay [53] have designed a PID controller and have compared the design outcome with those of bee and harmony search algorithms. Shayeghi et al. [91] used ABC to obtain the best PID parameters for solving the load frequency control problems. A hybrid algorithm of ABC has been used in image restoration that was corrupted by noise [77]. Table 2.2 further application using ABC optimization algorithm in different fields.

Table 2.2: A list of applications of ABC algorithm

Area Domain	Application	References
Control engineering	Optimal Tuning of PID Controllers	[92], [53],[93], [94]
Electronic /communication engineering	Cluster Based Wireless Sensor Network Routings	[95]
	Radio signal	[96]
	Filter design	[97]
Civil engineering	Truss structure design Composite structure design	[98] [34]
Neural networks	Training Feed-Forward Neural Networks	[56]
Electrical engineering	Power System Stabilizer Design for a Turbo-Generator in a Single-Machine Power System	[91]
Others field	Solving Traveling Salesman Problem	[99]
	Image processing	[77] [100]
	Data Mining	[79]
	Health	[101]
	Transportation and logistic	[102]

2.6 Summary

In this chapter, a review of ABC and SDA has been presented with detailed description of the realisation process of both algorithms. A comparative assessment of variants of the algorithm as featured in the literature has been highlighted along with modifications and improvements as reported by researchers. In the next chapter, the new approaches will be proposed in order to improve the performance of ABC and SDA.

Chapter 3
The SDA and ABC based Algorithms

3.1 Introduction

This chapter presents the details of the proposed optimization algorithms developed based on SDA and ABC. The focus of the development of proposed algorithms is to improve the performance of SDA and ABC in convergence speed and accuracy. There are eight new algorithms have been developed based on ABC and SDA. The first modification is on the initial distributions based on chaotic maps trajectory, random and opposition based learning (OBL). This adaptive initial distribution is used in all the proposed algorithms. An adaptive SDA algorithm is proposed using chaotic spiral radius and chaotic rotational angle. Three adaptive ABC algorithms are proposed based on the modification of the step size. The step sizes in the adaptive ABC algorithms are based on the exponential, linear and combination of both linear and exponential functions. Three hybrid versions of SDA and ABC are proposed. Good feature of both algorithms are used. Figure 3.1 shows the proposed set of algorithms and their linkages.

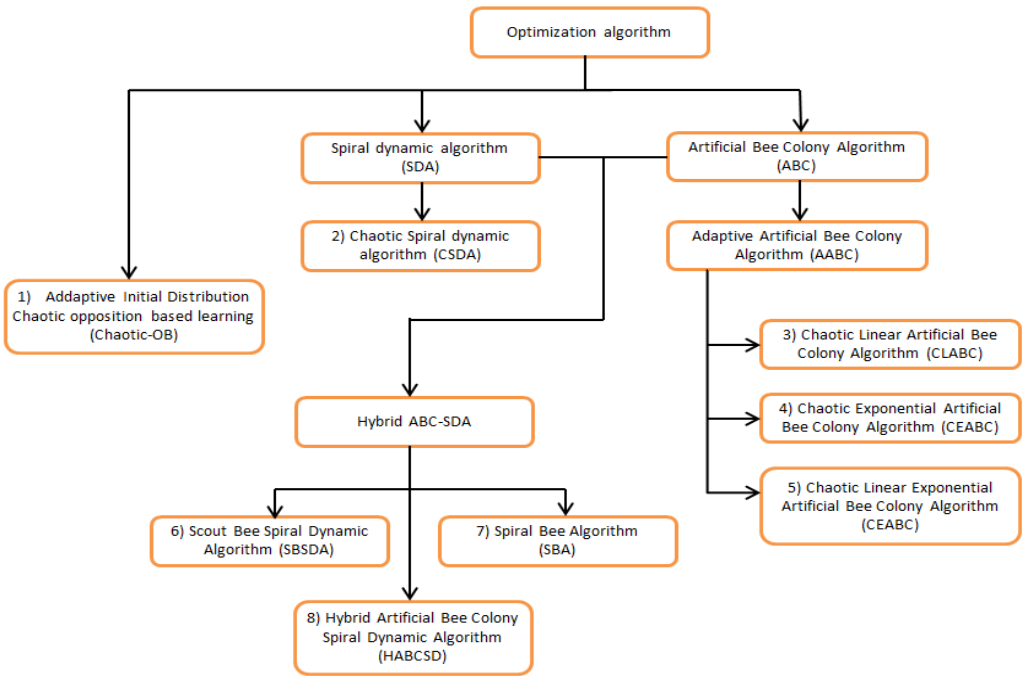

Optimization algorithm
Spiral dynamic algorithm (SDA)
Artificial Bee Colony Algorithm (ABC)
2) Chaotic Spiral dynamic algorithm (CSDA)
Adaptive Artificial Bee Colony Algorithm (AABC)
1) Addaptive Initial Distribution Chaotic opposition based learning (Chaotic-OB)
3) Chaotic Linear Artificial Bee Colony Algorithm (CLABC)
4) Chaotic Exponential Artificial Bee Colony Algorithm (CEABC)
5) Chaotic Linear Exponential Artificial Bee Colony Algorithm (CEABC)
Hybrid ABC-SDA
6) Scout Bee Spiral Dynamic Algorithm (SBSDA)
7) Spiral Bee Algorithm (SBA)
8) Hybrid Artificial Bee Colony Spiral Dynamic Algorithm (HABCSD)

Figure 3.1:- The proposed algorithms

3.2 Chaotic opposition based learning initialization

Population distribution during initialization is crucial to ensure the convergence speed and the quality of the final solution. Most of the evolutionary algorithms using random initial distribution to generate the initial population [103]. Without any prior knowledge, the initial population is randomly distributed in the allowable search space. But in some cases, the initial solutions distribution can be in opposite position of the optimal point or far from the optimal point [104]. In such a case, the computational time will increase [105]. The opposition based learning (OBL) approach proposed by H.R. Tizhoosh [106] are used here to enhance the quality of initial distribution of population and it may increase the quality of initial distribution toward the optimum point. The opposition concept can be seen in many areas such as opposition element in physics, in language, party politics, probability, religion and philosophy [107]. The balance between entities can come from the opposition. For example, the boiling water is hot and the temperature of the boiling water can be reduced using cold water. To learn something new, many start to learn at random. Known algorithms such as GA and PSO begin at random. But when dealing with the complex problems the random distribution guess can be distributed near to the optimal point so as to increase the convergence speed but it can also be far from it [108]. If the good solutions are at the opposite locations, the searching time will increase and the algorithm can potentially get trapped at local optimum point. There are no guarantees, that the random distribution can give a good initial solution. OBL has previously been used for improvement of simulated annealing algorithm, ant colony optimization, differential evolution and PSO[109][110]. Shahryar [111] has confirmed that taking random and their opposites values, in the absence of prior information, can accelerate the searching process to the optimal solution. Shahryar also compared the performance of opposition and randomness in differential evolution. And showed that the opposition based learning was able to enhance the capability of the algorithm. The OBL distribution is described as follows:

Let definition – let x in an interval $[a, b](x \in [a, b])$ to be a real number. While the opposite value of x can be denoted by $\breve{x}$, see Figure 3.2 and can be defined as follows:

$$\breve{x} = a + b - x \tag{3.1}$$

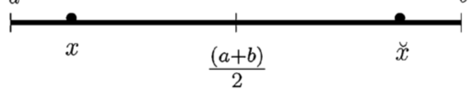


Figure 3.2: - Illustration of a point and its opposite point [108]

In a D-dimensional space the OBL can be defined by applying equation (3.1) in each dimension. Let $P(x_1, x_2, \ldots.., x_D,)$ to be a point inside D-dimensional search space where $x_1, x_2, \ldots.., x_D$ are real numbers and$x_i \in [a_i, b_i]$. The opposite point of x_i can be defined by their opposite positions $\breve{x_1}, \breve{x_2}, \ldots\ldots, \breve{x_D}$ where $\breve{x}_i = a_i + b_i - x_i$

The initialization of population based on opposition-based solution, P_{ij} is generated by

$$P_{ij} = x_{max,j} + x_{min,j} - Q_{ij} \tag{3.2}$$

where $i \in \{1,2, \ldots.., N\}$, $j \in \{1,2, \ldots.., D\}$, N is number of solutions and D is dimension size. While $x_{max,j}$ and $x_{min,j}$ are the minimum and maximum boundaries of the search space. Q_{ij} is the random solution and P_{ij} is the oposite solution of the random solution.

To increase the chance of getting a good solution during initial distribution, the combination of OBL and chaotic maps is proposed. By replacing the random initialization with chaotic distribution, the better initial population can be produced and the convergence speed can be increased [112][113]. The relations are used to generate chaotic initial population distribution.

$$C_{ij} = x_{min,j} + ch_k(x_{max,j} - x_{min,j}) \tag{3.3}$$

$$ch_{k+1} = \mu ch_k(1 - ch_k) \tag{3.4}$$

where x_0 is the initial population, ch_k is chaotic sequence, x_{min} is minimum boundary and x_{max} is maximum boundary, and ch_k is random generator. The logistic maps are used in this initial distribution.

In this initialization approach, three types of distribution are used. The first distribution is random type, Q_{ij}, followed by chaotic distribution, C_{ij} and finally, the opposition based, P_{ij}. All these distribution are executed and their fitness evaluated, following which the solutions are rank based on their fitness evaluation. Three sets of solution are thus produced in this method, where 2/3 un-performing solutions are discarded and only the top rank (1/3) of solutions are used as an initial distribution. The method is referred to as chaotic opposition based (chaotic-OB) distribution. The computation steps of this method can be seen in Algorithm 3.1.

Algorithm 3.1 : Chaotic opposition based learning distribution (Chaotic-OB)

```
 1: Set number of search point, $m$ where $m \geq 2$
 2: Set $i = 1, j = 1$ and $k = 1$
 3: Randomly set starting point of chaotic $ch_0 \in (0,1)$
 4: **for all** $k$ to $D$ **do**
 5:     $ch_{1,k+1} = \mu ch_{1,k}(1 - ch_{1,k})$
 6:     $k = k + 1$
 7: **end for**
 8: **While** $(i < m)$ **do**
 9:     **While** $(j < D)$ **do**
10:         $C_{ij} = x_{min,j} + ch_{1,j}(x_{max,j} - x_{min,j})$
11:         $j = j + 1$
12:     **end while**
13: $i = i + 1$
14: **end while**
15: Set $i = 1, j = 1$ and $k = 1$
16: **While** $(i < m)$ **do**
17:     **While** $(j < D)$ **do**
18:         $Q_{ij} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j})$
19:         $j = j + 1$
20:     **end while**
21: $i = i + 1$
22: **end while**
23:     **While** $(j < D)$ **do**
24:         $P_{ij} = x_{max,j} + x_{min,j} - Q_{ij}$
25:         $j = j + 1$
26:     **end while**
27: $i = i + 1$
28: **end while**
29: Choose the fittest solutions from $\{P(m) \cup Q(m) \cup C(m)\}$ as initial distribution.
```

A comparison between random distribution and Chaotic-OB distribution in the search space was carried out by testing these methods on multimodal two dimensional Griewank benchmark function in the range [-600,600] (Figure 3.3). The result shown in Figure 3.4 demonstrates that the new initialization method is able to produce a good initial solutions. As an example, In Figure 3.4(h), the distribution of solutions in the search space by chaotic-OB distribution were near to the optimal point, while by using random distribution in Figure 3.4(g) , the solution were scattered everywhere in search space. The Chaotic-OB distribution’s intention is to ensure that good quality of initial solutions are produced. The method enables the algorithm to reach the optimal point faster compared using used random distribution.

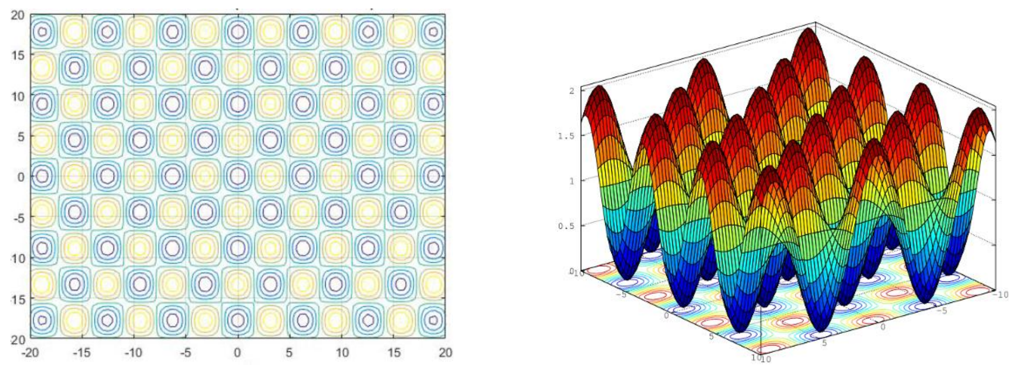
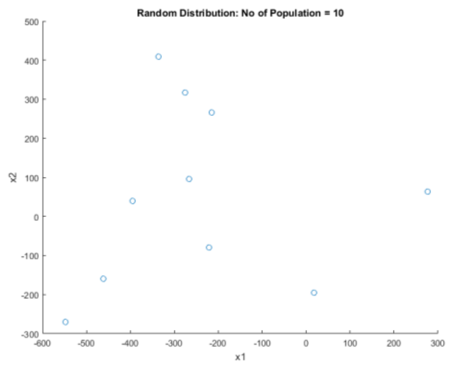
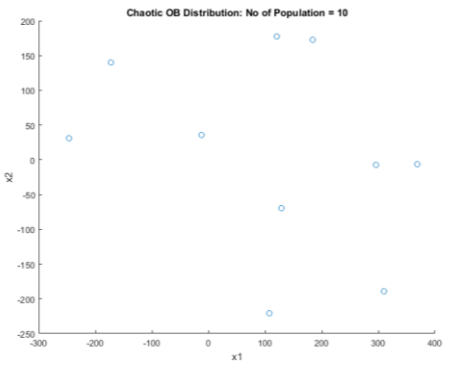


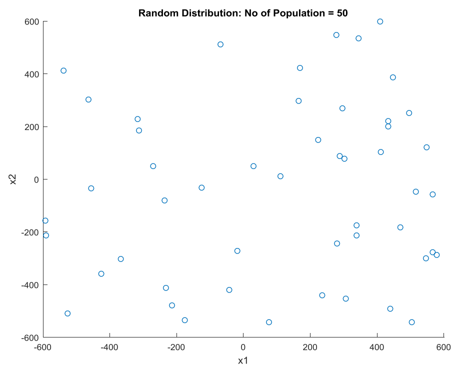
Figure 3.3 Griewank benchmark function (adapted from matlab works website)



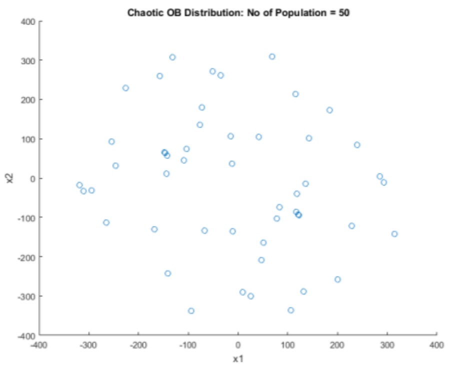
(a)



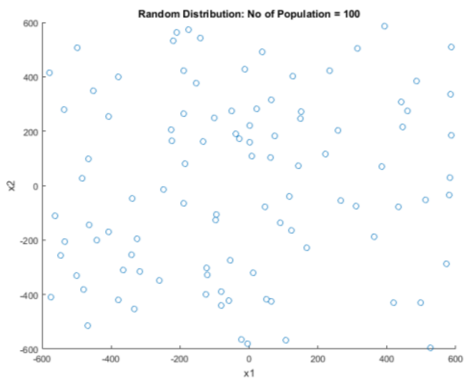
(b)



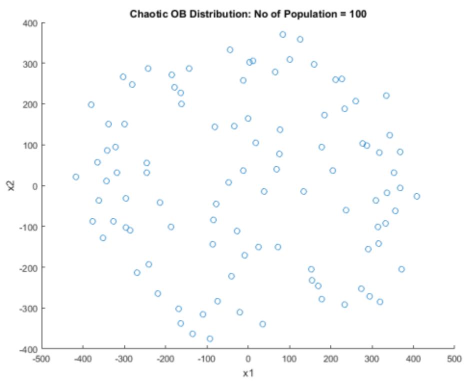
(c)



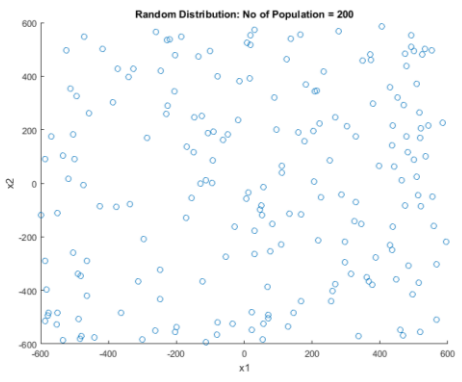
(d)

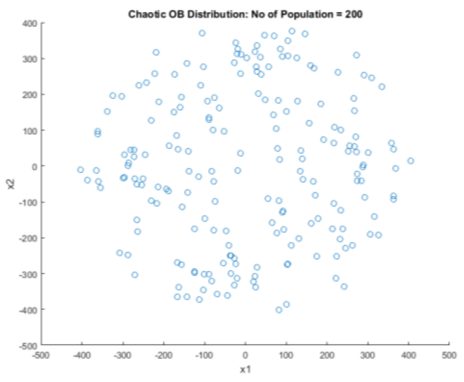


(e)



(f)





(g)

(h)

Figure 3.4: The initial distribution in search space using random and Chaotic-OB distribution

3.3 Improved spiral dynamic algorithms

3.3.1 Chaotic spiral dynamic algorithm

In evolutionary optimization algorithms, exploration to find the best possible solution is crucial , where during the exploration the algorithm should have the ability to perform the search around good solutions in the search space and must have good exploitation ability to converge to the best possible solution [114]. Thus, the exploration and exploitation must be balanced to produce accurate output. Due to the stochastic nature of the EA, there is no clear boundary between the exploration and exploitation. This drawback lead the algorithm to easily get trapped in local optima [115]. Many researchers have attempted to devise strategies for exploration and exploitation to ensure the performance of EA is enhanced. One of the methods that have been used to improve the performance of EA is to use chaos theory [116][117]. Chaos theory is based on the principle of chaotic dynamic system [118]. In the chaotic dynamic system, the initial conditions are highly sensitive where small changes in initial condition can give radically different results [119][120]. Although the chaotic map looks like it behaves randomly, but it has deterministic criteria and by determining the pattern of the map, it can provide chaotic behaviour [121].

Saremi et al. [122] integrated 10 types of chaotic maps into GA to improve the capability of selection, emigration and also the mutation probability. By using tinker bell chaos map into absorption coefficient and random parameter, Coelho [123] improved firefly algorithm to get good result in tuning PID controller. Coelho [124] replaced random sequences in PSO with Henon maps to diversify the population and it preventing the algorithm to get trapped at local optima.

As described before in the previous chapter, SDA facing problem of easily get trapped at local optimum because the spiral trajectory moves with constant step size. By using chaotic map, oscillatory movement can be inserted into the SDA to give some randomisation behaviour in distribution of population [125]. The chaotic spiral dynamic algorithms are shown in this section that use chaotic map in initial population distribution and during next point movement phase.

In classical SDA, the radius of spiral is constant through the whole searching process. In this case, the potential for the algorithm to trap at local optimum is high. To improve the performance of SDA and also to overcome the problem of algorithm getting trapped at the local optima during the search process, the chaotic maps are introduced into the SDA. Two novel approaches by using chaotic maps are presented in this section. The first algorithm is devised by replacing the rotational radius, r with the chaotic sequence maps and the second algorithm is devised by replacing the angle, θ with chaotic sequence maps. By adding chaotic sequence in SDA, the SDA step size will become more dynamic and this will help the algorithm escape from trap at local optima. The mathematical formulation of chaotic spiral dynamic algorithm (CSDA) is given as

$$x_i(k+1) = S_n(r_{chao},\theta_{chao})x_i(k) - [S_n(r_{chao},\theta_{chao}) - I_n]x^*\ , i = 1,2,\dots\dots,m. \qquad (3.5)$$

where r_{chao} is given by chaotic maps with scale value from 0 to 1. θ_{chao} is chaotic rotational angle with angle between 0 to 2π, I_n is identity matrix , x^* centre of spiral, k is number of iterations, m is number of points, $S_n(r,\theta) = rR^{(n)}(\theta_{1,2},\theta_{1,3},\dots\dots\theta_{n,n-1})$, $R^{(n)}$ is composition rotation matrix, where $R^{(n)}(\theta_{1,2},\theta_{1,3},\dots\dots\theta_{n,n-1})$ is rotation $n \times n$ matrix and m is a spiral point. A pseudo-code of CSDA is shown as follows

Algorithm 3.2 :- Chaotic spiral dynamic algorithm

```
1:  Objective function of $f(x_d)$
2:  **Initialisation:**
3:    Set upper and lower boundary of search space
4:    Set maximum iteration, $MCN$
5:    Set number of search point, $m$ where $m \geq 2$
```

```
 6:   Set chaotic spiral radius, $r_{chao}$  where $r_{chao}$ is chaotic singer map [-1,1]
 7:   Set spiral angle, $\theta$ where $0 \le \theta \le 2\pi$
 8:   Set initial points $x_i(0) \in R^n, i = 1,2,\dots,m$ in the search space using Chaotic-OB
 9:   Choose the centre of spiral $x^*$ as $x^* = x_{i_g}(0), i_g = \arg min_i\, f(x_i(0)),\ i = 1,2,\dots,m$
10: while iteration,$k \le MCN$ do
11:     for all $i$ to $m$ do
          Updating $x_i$ ;
12: $x_i(k+1) = S_n(r_{chao}, \theta_{chao}) x_i(k) - [S_n(r_{chao}, \theta_{chao}) - I_n] x^*, i = 1,2,\dots\dots,m.$
13:     End for
14:     Updating $x^*$; $x^* = x_{i_g}(k+1), i_g = \arg min_i f(x_i(k+1)),\ i = 1,2,\dots,m$
15:     $k = k + 1$
16: end while
```

3.3.2 Statistical test for selection of suitable chaotic maps for CSDA algorithm

There are many types of chaotic maps available to use in CSDA. But the question is, which one is most suitable to lead the CSDA to be faster and give accurate results. The statistical significant test is performed in order to choose the best chaotic maps. In this section, on-parametric test Kruskal-Wills is used for statistical significant analysis with various types of chaotic maps. Table 3.1 shows the list of chaotic maps that will be used in this statistical significant test. Kruskal-Willis test enables to evaluate the significant data difference between more than 2 groups. The methods are created based on one way anova and Wilcoxon rank sum test. All samples are assumed to be from the same group of continuous distribution and that they are mutually independent. All the data will be rank by Kruskal-Willis test. The main hypothesis for this test is that the algorithms perform the same and their median is the same. If Kruskal-Willis test shows that at least one algorithm is different from others, the hypothesis is rejected at 95% confidence interval. These tests are run based on 95% confidence interval and the simulations are run 30 times for each algorithm. If there are significant improvements compared to the other CSDA variations the p-value will be less than 0.05. In this test, 10 chaotic maps are used in CSDA (based on the work by Saremi et al. [122]) and each is in 6 benchmark functions (Table 3.2) in 10, 30, 50 and 70 dimensions. The selection of number of dimensions for this experiment is based on the work of Karaboga [57]. It is sufficient to evaluate the performance of algorithms. As most benchmark functions constitute minimisation problems, the lowest rank in Kruskal-Willis test show that algorithms are most likely able to perform better compared to others.

Table 3.1:- Chaotic Maps mathematical expression

Chaotic map	Mathematical expression	Range
Chebyshev	$x_{i+1} = cos\left(icos^{-1}(x_i)\right)$	[-1,1]
logistic	$x_{i+1} = ax_i(1 - x_i), a = 4$	[0,1]
Iterative	$x_{i+1} = sin\left(\frac{ax}{x_i}\right), a = 0.7$	[-1,1]
Gauss	$x_{i+1} = \begin{cases} 1 & x_i = 0 \\ \frac{1}{mod(x_i, 1)} & otherwise \end{cases}$	[0,1]
Tent	$x_{i+1} = \begin{cases} \frac{x_i}{0.7}, & x_i < 0.7 \\ \frac{10}{3}(1 - x_i), & x_i \geq 0.7 \end{cases}$	[0,1]
Sine	$x_{i+1} = \frac{a}{4} sin(\pi x_i)$	[0,1]
Circle	$x_{i+1} = mod\left(x_i + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_k), 1\right), a = 0.5\ and\ b = 0.2$	[0,1]
Piecewise	$x_{i+1} = \begin{cases} \frac{x_i}{P}, & 0 \leq x_i < P \\ \frac{x_i - P}{0.5 - P}, & P \leq x_i < 0.5 \\ \frac{1 - P - x_i}{0.5 - P}, & 0.5 \leq x_i < 1 - P \\ \frac{1 - x_i}{P}, & 1 - P \leq x_i < 1 \end{cases}, P = 0.4$	[0,1]
Singer	$x_{i+1} = \mu\left(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4\right), \mu = 1.07$	[0,1]
sinusoidal	$x_{i+1} = ax_i^2 \sin(\pi x_i), a = 2.3$	[0,1]

Table 3.3 to Table 3.6 show the numerical results in all tested dimensions for CSDA and original SDA algorithm. The results show that the CSDA outperformed SDA in all cases. The standard deviation was also improved with CSDA, which implies that CSDA was able to give a stable results. The results also show that good output was achieved with four of the chaotic maps. There were sine map, circle map, singer map and sinusoidal map. The main reason for ability of these 4 chaotic maps to improve CSDA is the initial starting point and the distribution pattern. Table 3.7 shows the statistical significant test for CSDA using Kruskal-Willis non parametric test, where the lowest rank shows that the algorithm has performed better. In this test, the CSDA using singer map was in lowest rank, which implies that the singer map enabled the CSDA to reach the lowest optimal point in most tested benchmark

functions. Thus, the singer maps will be used in CSDA in all subsequent experiments and tests. Mathematical expressions and associated ranges of the maps used in this work are shown in Table 3.2.

Table 3.2:- Benchmark functions

Function	**Mathematical formulations**	**Range**
Sphere	$\sum_{i=1}^{n} {x_i}^2$	[-5.12,5.12]
Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}\right)-\exp\left(\frac{1}{n}\sqrt{\sum_{i=1}^{n} cos(2\pi x_i)}\right)+20+e$	[-32,32]
Rosenbrock	$\sum_{i=1}^{d-1}\left[100\left(x_{i+1}-x_i^2\right)^2\right]+(x_i-1)^2$	[-5,10]
Griewank	$\frac{1}{4000}\sum_{i=1}^{n} {x_i}^2-\prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600,600]
Rastringin	$10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$	[-5.12,5.12]
Schwefel	$4.18.9829d-\sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500,500]

Note:-global minimum for all benchmark functions, $f(x_i^*)=0$

Table 3.3 :- CSDA performance result for benchmark function in dimension D=10

Benchmark function	D=10	SDA	Chaotic SDA Logistic map	Chaotic SDA Gauss map	Chaotic SDA Chebyshev map	Chaotic SDA Iterative map	Chaotic SDA Tent map	Chaotic SDA Sine map	Chaotic SDA Circle map	Chaotic SDA Piecewise map	Chaotic SDA singer map	Chaotic SDA sinusoidal map
Sphere f_1	avg	17.61	5.68	13.95	2.49	5.20	28.19	2.87	6.79	2.72	**2.13**	11.54
	Std	18.98	9.33	5.06	2.17	3.50	8.32	2.32	6.16	1.25	1.49	15.26
	worse	48.66	50.28	22.14	7.91	12.86	53.15	8.62	21.95	5.43	4.87	53.47
	best	0.24	0.11	5.57	0.15	0.53	14.77	0.66	1.80	0.78	0.12	1.84
Ackley f_2	avg	14.83	12.71	15.34	13.52	13.48	16.80	13.35	**12.51**	13.22	12.65	14.78
	Std	4.84	2.82	1.82	3.30	2.80	1.35	3.62	2.05	2.35	2.65	3.89
	worse	19.33	17.11	17.73	19.00	17.45	18.72	17.05	15.07	16.46	15.59	19.12
	best	4.17	5.25	11.30	5.73	7.21	13.59	5.65	8.55	8.70	7.30	8.19
Rosenbrock f_3	avg	4,036.05	375.87	10,444.82	1,433.71	3,552.26	38,835.41	**29.80**	218.25	783.13	1,078.98	5,370.24
	Std	15,540.13	1,735.20	9,938.11	4,182.86	10,644.49	25,993.61	58.76	340.77	1,709.33	2,389.27	10,427.89
	worse	84,573.99	9,543.10	35,221.80	22,059.53	46,117.43	100,427.05	188.94	1,043.58	5,554.99	7,735.80	30,686.90
	best	0.05	0.00	288.02	0.00	1.44	54.90	0.07	0.03	0.66	0.30	16.87
Grienwank f_4	avg	91.97	8.44	26.41	19.64	25.91	90.95	16.38	22.10	28.39	12.66	**8.92**
	Std	60.04	8.44	26.41	35.64	24.42	32.04	12.69	9.36	29.76	9.94	9.15
	worse	195.10	32.59	116.67	141.78	117.33	146.59	43.03	37.92	102.94	31.87	31.26
	best	1.38	1.20	14.18	0.72	5.50	40.03	3.61	9.75	6.39	0.85	1.11
Rastringin f_5	avg	81.05	57.15	63.71	55.71	57.00	81.54	61.76	62.07	57.36	**49.17**	64.91
	Std	26.90	19.86	12.64	19.65	23.15	16.61	18.78	13.43	25.45	19.42	16.78
	worse	129.19	107.72	89.97	102.62	115.48	110.62	95.85	85.83	109.09	84.99	82.57
	best	37.97	20.88	42.13	23.88	23.65	45.03	33.25	42.22	28.96	26.25	25.89
Schwefel's f_6	avg	-2,305.48	-2,326.99	-2,044.90	-2,140.64	-2,384.50	-1,522.58	-2,287.57	-2,337.71	-2,430.52	**-2,581.93**	-2,151.29
	Std	991.02	883.56	590.88	767.58	747.01	464.87	688.94	776.42	888.54	607.87	831.92
	worse	-1,143.63	-1,187.29	-1,020.88	-1,123.72	-1,062.84	- 906.48	-1,274.23	-1,039.15	-1,339.54	-1,483.62	6.27
	best	-3,755.55	-3,755.34	- 2,872.44	-4,189.83	-3,613.95	- 2,641.14	-3,183.06	-3,333.36	-3,538.41	-3,261.99	-2,882.49

Table 3.4:- CSDA performance result for benchmark function in dimension D=30

Benchmark function	D=30	SDA	Chaotic SDA Logistic map	Chaotic SDA Gauss map	Chaotic SDA Chebyshev map	Chaotic SDA Iterative map	Chaotic SDA Tent map	Chaotic SDA Sine map	Chaotic SDA Circle map	Chaotic SDA Piecewise map	Chaotic SDA singer map	Chaotic SDA sinusoidal map
Sphere f_1	avg	1.50E+02	4.94E+01	1.07E+02	**4.98E+01**	6.87E+01	1.47E+02	4.94E+01	6.70E+01	4.73E+01	4.85E+01	1.36E+02
	Std	5.17E+01	2.68E+01	1.86E+01	4.28E+01	3.39E+01	2.13E+01	1.39E+01	2.10E+01	1.90E+01	1.72E+01	5.38E+01
	worse	2.04E+02	1.75E+02	1.34E+02	1.81E+02	1.83E+02	1.79E+02	7.43E+01	1.05E+02	7.46E+01	8.21E+01	1.93E+02
	best	2.72E+01	2.68E+01	6.20E+01	1.78E+01	3.33E+01	1.12E+02	3.23E+01	3.95E+01	2.21E+01	2.73E+01	4.25E+01
Ackley f_2	avg	1.89E+01	1.78E+01	1.87E+01	1.76E+01	1.80E+01	1.92E+01	1.81E+01	1.82E+01	1.74E+01	**1.74E+01**	1.88E+01
	Std	1.18E+00	1.25E+00	5.07E-01	1.49E+00	6.27E-01	4.76E-01	4.62E-01	7.33E-01	1.42E+00	5.87E-01	8.64E-01
	worse	2.01E+01	1.99E+01	1.95E+01	1.99E+01	1.91E+01	2.01E+01	1.88E+01	1.91E+01	1.97E+01	1.84E+01	2.00E+01
	best	1.63E+01	1.42E+01	1.76E+01	1.31E+01	1.63E+01	1.80E+01	1.71E+01	1.67E+01	1.50E+01	1.64E+01	1.74E+01
Rosenbrock f_3	avg	8.00E+04	3.86E+04	5.12E+05	**2.27E+04**	1.14E+05	9.32E+05	5.24E+04	2.06E+05	1.07E+05	5.98E+04	1.97E+05
	Std	9.76E+04	3.40E+04	2.81E+05	2.34E+04	1.29E+05	2.60E+05	7.41E+04	1.19E+05	1.00E+05	4.40E+04	1.40E+05
	worse	5.13E+05	1.25E+05	1.15E+06	8.54E+04	6.65E+05	1.45E+06	2.42E+05	5.11E+05	3.07E+05	1.25E+05	5.40E+05
	best	3.18E+03	1.19E+03	1.11E+05	1.31E+02	4.33E+03	4.36E+05	1.40E+03	1.14E+05	7.85E+03	5.03E+03	6.68E+04
Grienwank f_4	avg	5.11E+02	1.04E+02	6.17E+01	1.96E+02	2.41E+02	4.88E+02	2.10E+02	3.09E+02	1.81E+02	2.11E+02	**1.50E+02**
	Std	1.67E+02	1.04E+02	6.17E+01	1.49E+02	1.16E+02	7.27E+01	1.63E+02	1.79E+02	4.53E+01	1.55E+02	5.67E+01
	worse	6.91E+02	6.78E+02	5.23E+02	6.40E+02	6.75E+02	6.56E+02	6.51E+02	6.49E+02	2.77E+02	6.35E+02	2.81E+02
	best	1.40E+02	6.91E+01	2.65E+02	4.82E+01	1.47E+02	3.18E+02	1.03E+02	1.34E+02	1.26E+02	9.08E+01	9.15E+01
Rastringin f_5	avg	3.05E+02	2.53E+02	3.14E+02	2.54E+02	2.66E+02	3.77E+02	**2.25E+02**	2.49E+02	2.61E+02	2.40E+02	2.37E+02
	Std	9.93E+01	5.04E+01	3.87E+01	6.35E+01	4.87E+01	3.24E+01	5.34E+01	3.64E+01	2.02E+01	7.09E+01	6.77E+01
	worse	4.76E+02	4.11E+02	3.91E+02	4.20E+02	3.65E+02	4.37E+02	3.64E+02	3.26E+02	3.03E+02	4.33E+02	3.58E+02
	best	1.71E+02	1.76E+02	2.47E+02	1.66E+02	1.82E+02	3.10E+02	1.74E+02	2.04E+02	2.40E+02	1.83E+02	1.42E+02
Schwefel's f_6	avg	-3.92E+03	-4.99E+03	-3.91E+03	-5.53E+03	-6.03E+03	-2.74E+03	-5.37E+03	-5.37E+03	-5.95E+03	**-7.28E+03**	-4.72E+03
	Std	2.45E+03	2.45E+03	1.71E+03	2.81E+03	2.34E+03	9.16E+02	2.24E+03	2.77E+03	2.06E+03	2.19E+03	2.89E+03
	worse	-1.53E+03	-1.93E+03	-1.43E+03	-1.82E+03	-2.14E+03	-1.35E+03	-2.49E+03	-1.68E+03	-2.58E+03	-3.24E+03	-1.58E+03
	best	-9.65E+03	-9.15E+03	-7.25E+03	-9.19E+03	-9.45E+03	-4.69E+03	-8.61E+03	-8.23E+03	-8.23E+03	-9.33E+03	-8.84E+03

Table 3.5:- CSDA performance result for benchmark function in dimension D=50

Benchmark function	D=50	SDA	Chaotic SDA Logistic map	Chaotic SDA Gauss map	Chaotic SDA Chebyshev map	Chaotic SDA Iterative map	Chaotic SDA Tent map	Chaotic SDA Sine map	Chaotic SDA Circle map	Chaotic SDA Piecewise map	Chaotic SDA singer map	Chaotic SDA sinusoidal map
Sphere f_1	avg	2.71E+02	1.25E+02	2.40E+02	1.12E+02	1.46E+02	2.80E+02	**8.39E+01**	1.74E+02	1.39E+02	1.27E+02	2.79E+02
	Std	9.65E+01	7.23E+01	3.49E+01	7.47E+01	4.87E+01	2.79E+01	2.00E+01	6.14E+01	6.73E+01	7.90E+01	8.80E+01
	worse	3.77E+02	3.66E+02	3.34E+02	3.34E+02	3.42E+02	3.24E+02	1.18E+02	3.41E+02	3.14E+02	3.43E+02	3.85E+02
	best	9.71E+01	5.99E+01	1.75E+02	4.79E+01	9.06E+01	2.14E+02	5.09E+01	1.32E+02	6.26E+01	5.96E+01	1.28E+02
Ackley f_2	avg	1.95E+01	1.87E+01	1.94E+01	1.83E+01	1.89E+01	1.97E+01	1.84E+01	1.88E+01	**1.84E+01**	1.84E+01	1.92E+01
	Std	8.64E-01	5.77E-01	2.51E-01	6.12E-01	5.22E-01	2.55E-01	4.23E-01	3.60E-01	2.49E-01	7.09E-01	8.14E-01
	worse	2.02E+01	1.99E+01	1.98E+01	1.98E+01	2.02E+01	2.01E+01	1.91E+01	1.94E+01	1.88E+01	2.00E+01	2.02E+01
	best	1.74E+01	1.77E+01	1.88E+01	1.72E+01	1.76E+01	1.90E+01	1.74E+01	1.81E+01	1.79E+01	1.75E+01	1.81E+01
Rosenbrock f_3	avg	3.00E+05	1.63E+05	1.57E+06	**9.46E+04**	3.90E+05	2.08E+06	1.51E+05	5.04E+05	2.84E+05	1.25E+05	7.87E+05
	Std	1.62E+05	8.56E+04	4.60E+05	6.99E+04	1.60E+05	4.46E+05	1.23E+05	2.38E+05	1.36E+05	1.03E+05	2.84E+05
	worse	6.86E+05	4.33E+05	2.32E+06	3.17E+05	6.96E+05	3.29E+06	4.43E+05	8.97E+05	5.71E+05	3.82E+05	1.12E+06
	best	7.23E+04	1.51E+04	7.78E+05	4.45E+03	1.03E+05	1.07E+06	3.28E+04	1.48E+05	1.04E+05	2.88E+04	2.27E+05
Grienwank f_4	avg	1.05E+03	1.55E+02	1.11E+02	**3.57E+02**	5.19E+02	9.69E+02	3.62E+02	5.73E+02	6.14E+02	3.47E+02	3.40E+02
	Std	2.12E+02	1.55E+02	1.11E+02	2.46E+02	1.38E+02	9.60E+01	9.26E+01	1.92E+02	3.46E+02	6.33E+01	5.99E+01
	worse	1.24E+03	1.13E+03	9.90E+02	1.11E+03	1.10E+03	1.14E+03	5.59E+02	1.11E+03	1.16E+03	4.83E+02	4.19E+02
	best	4.66E+02	2.56E+02	5.80E+02	1.44E+02	3.68E+02	7.60E+02	2.45E+02	4.64E+02	3.59E+02	2.54E+02	2.65E+02
Rastringin f_5	avg	6.35E+02	4.29E+02	6.06E+02	4.65E+02	5.14E+02	6.92E+02	4.81E+02	5.29E+02	4.77E+02	4.41E+02	**4.14E+02**
	Std	1.58E+02	6.53E+01	6.00E+01	1.40E+02	9.86E+01	5.65E+01	1.51E+02	1.04E+02	1.16E+02	8.72E+01	6.28E+01
	worse	8.18E+02	5.75E+02	7.07E+02	7.91E+02	7.71E+02	7.99E+02	7.56E+02	7.92E+02	7.58E+02	6.01E+02	4.97E+02
	best	3.71E+02	3.33E+02	4.56E+02	3.00E+02	3.48E+02	5.56E+02	2.91E+02	4.10E+02	3.19E+02	3.36E+02	2.94E+02
Schwefel's f_6	avg	-7.72E+03	-1.04E+04	-5.72E+03	-7.84E+03	-9.34E+03	-3.78E+03	**-1.07E+04**	-1.00E+04	-8.95E+03	-1.03E+04	-9.39E+03
	Std	4.78E+03	4.01E+03	2.12E+03	4.78E+03	4.33E+03	8.74E+02	4.08E+03	2.45E+03	3.97E+03	4.73E+03	4.31E+03
	worse	-2.21E+03	-2.67E+03	-2.52E+03	-2.58E+03	-2.45E+03	-2.38E+03	-3.13E+03	-3.81E+03	-2.82E+03	-2.94E+03	-2.71E+03
	best	-1.58E+04	-1.43E+04	-8.82E+03	-1.51E+04	-1.41E+04	-5.82E+03	8.39E+01	-1.28E+04	-1.23E+04	-1.42E+04	-1.28E+04

Table 3.6:- CSDA performance result for benchmark function in dimension D=70

Benchmark function	D=70	SDA	Chaotic SDA Logistic map	Chaotic SDA Gauss map	Chaotic SDA Chebyshev map	Chaotic SDA Iterative map	Chaotic SDA Tent map	Chaotic SDA Sine map	Chaotic SDA Circle map	Chaotic SDA Piecewise map	Chaotic SDA singer map	Chaotic SDA sinusoidal map
Sphere f_1	avg	3.85E+02	1.73E+02	3.70E+02	**1.79E+02**	2.63E+02	4.21E+02	1.72E+02	2.69E+02	2.44E+02	2.00E+02	4.02E+02
	Std	1.37E+02	2.73E+01	2.93E+01	1.02E+02	8.93E+01	3.33E+01	4.75E+01	6.32E+01	9.60E+01	9.88E+01	9.06E+01
	worse	5.28E+02	2.35E+02	4.39E+02	4.82E+02	4.92E+02	4.77E+02	2.75E+02	4.13E+02	5.14E+02	4.74E+02	4.92E+02
	best	1.52E+02	1.30E+02	3.14E+02	8.59E+01	1.69E+02	3.40E+02	1.15E+02	1.93E+02	1.98E+02	1.33E+02	2.79E+02
Ackley f_2	avg	1.98E+01	1.88E+01	1.96E+01	1.89E+01	1.92E+01	1.99E+01	1.90E+01	1.91E+01	1.92E+01	1.86E+01	1.99E+01
	Std	6.49E-01	4.67E-01	2.36E-01	4.78E-01	4.32E-01	2.21E-01	3.56E-01	3.44E-01	4.42E-01	3.66E-01	3.50E-01
	worse	2.04E+01	2.02E+01	2.00E+01	2.03E+01	2.02E+01	2.03E+01	1.96E+01	1.95E+01	2.03E+01	1.90E+01	2.03E+01
	best	1.84E+01	1.82E+01	1.90E+01	1.81E+01	1.85E+01	1.94E+01	1.85E+01	1.83E+01	1.87E+01	**1.79E+01**	1.92E+01
Rosenbrock f_3	avg	7.76E+05	3.47E+05	2.83E+06	2.99E+05	1.02E+06	4.09E+06	4.12E+05	1.13E+06	7.76E+05	3.96E+05	1.75E+06
	Std	2.44E+05	1.52E+05	5.72E+05	2.37E+05	4.08E+05	7.96E+05	1.84E+05	4.19E+05	3.03E+05	3.12E+05	4.50E+05
	worse	1.23E+06	7.62E+05	4.13E+06	8.72E+05	2.06E+06	5.99E+06	7.31E+05	1.70E+06	1.44E+06	1.03E+06	2.46E+06
	best	3.09E+05	9.96E+04	1.75E+06	**3.28E+04**	4.15E+05	2.85E+06	1.75E+05	5.11E+05	4.97E+05	9.25E+04	9.29E+05
Grienwank f_4	avg	1.30E+03	1.81E+02	1.25E+03	5.97E+02	9.18E+02	1.47E+03	6.17E+02	8.48E+02	7.47E+02	5.43E+02	6.93E+02
	Std	4.37E+02	1.81E+02	1.38E+02	3.01E+02	2.91E+02	1.06E+02	1.05E+02	1.10E+02	1.06E+02	7.20E+01	3.28E+02
	worse	1.80E+03	1.50E+03	1.63E+03	1.68E+03	1.75E+03	1.73E+03	7.44E+02	1.08E+03	9.25E+02	6.54E+02	1.60E+03
	best	5.76E+02	4.55E+02	1.03E+03	3.37E+02	6.16E+02	1.20E+03	4.61E+02	6.99E+02	6.09E+02	**4.36E+02**	4.50E+02
Rastringin f_5	avg	1.00E+03	7.04E+02	9.22E+02	6.80E+02	7.57E+02	1.00E+03	7.21E+02	7.87E+02	7.21E+02	7.53E+02	6.55E+02
	Std	1.73E+02	1.55E+02	7.01E+01	1.55E+02	1.02E+02	5.65E+01	1.63E+02	1.17E+02	1.56E+02	1.57E+02	6.26E+01
	worse	1.18E+03	1.11E+03	1.13E+03	1.12E+03	9.72E+02	1.11E+03	1.11E+03	1.06E+03	1.12E+03	1.10E+03	7.78E+02
	best	6.21E+02	5.34E+02	8.03E+02	5.20E+02	5.79E+02	9.10E+02	5.65E+02	6.65E+02	5.87E+02	6.23E+02	**5.76E+02**
Schwefel's f_6	avg	-1.04E+04	-1.31E+04	-7.80E+03	-1.41E+04	-1.09E+04	-5.02E+03	-1.56E+04	-1.23E+04	-1.05E+04	-1.39E+04	-1.44E+04
	Std	6.07E+03	5.56E+03	2.36E+03	5.65E+03	5.78E+03	2.12E+03	4.06E+03	3.19E+03	4.66E+03	5.71E+03	5.57E+03
	worse	-2.45E+03	-3.53E+03	-3.27E+03	-3.14E+03	-2.92E+03	-1.78E+03	-4.42E+03	-4.12E+03	-4.51E+03	-2.99E+03	-3.60E+03
	best	-1.91E+04	-1.94E+04	-1.11E+04	-2.08E+04	-1.77E+04	-9.11E+03	**-1.88E+04**	-1.52E+04	-1.57E+04	-1.93E+04	-1.78E+04

Table 3.7:- CSDA Statistical significant test

Benchmark function	D	SDA	Chaotic SDA Logistic map	Chaotic SDA Gauss map	Chaotic SDA Chebyshev map	Chaotic SDA Iterative map	Chaotic SDA Tent map	Chaotic SDA Sine map	Chaotic SDA Circle map	Chaotic SDA Piecewise map	Chaotic SDA singer map	Chaotic SDA sinusoidal map	P-value
Sphere f_1	10	120.70 (8)	86.60 (5)	164.20 (10)	61.23 (2)	104.60 (6)	200.73 (11)	69.50 (3)	113.20 (7)	72.30 (4)	**56.50 (1)**	130.80 (9)	7.41E-19
	30	179.97 (10)	63.50 (3)	148.43 (8)	**53.97 (1)**	101.67 (6)	181.30 (11)	69.10 (5)	102.00 (7)	63.20 (2)	65.80 (4)	169.90 (9)	7.70E-24
	50	171.13 (9)	68.70 (3)	151.87 (8)	53.47 (2)	99.40 (6)	174.77 (10)	**31.20 (1)**	124.90 (7)	89.80 (5)	73.00 (4)	179.60 (11)	1.51E-22
	70	166.03 (9)	51.60 (3)	153.37 (8)	**48.30 (1)**	115.60 (6)	178.60 (11)	49.90 (2)	118.60 (7)	104.60 (5)	65.00 (4)	177.90 (10)	1.88E-24
Ackley f_2	10	142.27 (10)	80.07 (3)	136.57 (9)	100.03 (6)	96.23 (5)	174.10 (11)	100.70 (7)	**70.10 (1)**	86.90 (4)	77.60 (2)	133.40 (8)	3.53E-8
	30	158.87 (10)	85.07 (3)	140.73 (8)	85.43 (5)	85.40 (4)	175.93 (11)	92.10 (6)	105.50 (7)	70.80 (2)	**49.50 (1)**	144.30 (9)	**1.22E-12**
	50	162.20 (10)	85.70 (5)	152.77 (9)	58.13 (2)	105.60 (7)	182.00 (11)	65.00 (3)	96.30 (6)	**56.80 (1)**	69.50 (4)	129.70 (8)	1.54E-17
	70	164.80 (9)	58.17 (2)	146.17 (8)	66.53 (3)	105.13 (7)	179.30 (10)	79.50 (4)	96.60 (5)	97.90 (6)	**40.90 (1)**	181.30 (11)	2.65E-20
Rosenbrock f_3	10	102.03 (7)	69.20 (2)	175.37 (10)	78.57 (3)	102.87 (8)	205.93 (11)	**55.20 (1)**	80.10 (4)	90.10 (5)	93.50 (6)	135.70 (9)	4.42E-21
	30	90.73 (5)	64.43 (3)	183.73 (10)	**44.47 (1)**	107.23 (7)	211.37 (11)	64.30 (2)	151.90 (9)	103.50 (6)	85.40 (4)	145.50 (8)	5.80E-30
	50	100.67 (6)	63.97 (4)	191.20 (10)	**35.43 (1)**	121.77 (7)	208.90 (11)	53.30 (3)	134.30 (8)	99.00 (5)	46.50 (2)	157.60 (9)	1.31E-34
	70	105.70 (6)	47.70 (2)	187.20 (10)	**38.93 (1)**	124.70 (7)	212.73 (11)	57.80 (4)	131.80 (8)	102.90 (5)	52.10 (3)	161.00 (9)	4.17E-36
Grienwank f_4	10	165.00 (10)	59.98 (2)	151.63 (9)	66.77 (3)	108.63 (7)	189.93 (11)	86.40 (5)	113.80 (8)	107.80 (6)	70.50 (4)	**52.15 (1)**	3.31E-20
	30	184.10 (11)	60.10 (2)	153.83 (9)	66.33 (3)	104.93 (7)	181.13 (10)	76.00 (5)	125.90 (8)	74.80 (4)	79.30 (6)	**49.20 (1)**	6.30E-23
	50	199.97 (11)	63.83 (5)	153.87 (9)	**43.00 (1)**	110.07 (6)	179.67 (10)	58.50 (4)	123.20 (8)	117.80 (7)	53.70 (3)	52.10 (2)	2.59E-30
	70	169.57 (10)	64.90 (4)	161.27 (9)	43.67 (2)	126.90 (8)	189.50 (11)	61.40 (3)	121.10 (7)	99.90 (6)	**36.70 (1)**	70.00 (5)	3.35E-27
Rastringin f_5	10	154.73 (10)	93.73 (5)	115.67 (8)	88.87 (2)	92.60 (4)	170.33 (11)	108.80 (6)	111.20 (7)	91.90 (3)	**71.30 (1)**	125.50 (9)	5.22E-07
	30	125.77 (9)	89.30 (5)	153.27 (10)	89.50 (6)	102.57 (7)	197.77 (11)	**54.80 (1)**	85.20 (4)	103.30 (8)	66.30 (2)	72.40 (3)	2.11E-14
	50	157.60 (10)	61.60 (2)	155.53 (9)	76.67 (4)	109.00 (7)	186.60 (11)	86.40 (5)	117.90 (8)	87.30 (6)	68.80 (3)	**55.10 (1)**	7.84E-18
	70	178.33 (10)	72.90 (3)	155.20 (9)	62.27 (2)	97.17 (7)	180.23 (11)	78.80 (4)	111.10 (8)	80.80 (5)	93.00 (6)	**54.50 (1)**	5.33E-20
Schwefel's f_6	10	106.73 (7)	103.47 (6)	126.90 (10)	115.90 (9)	98.03 (3)	171.80 (11)	102.60 (4)	102.80 (5)	92.70 (2)	**79.10 (1)**	110.80 (8)	6.13E-04
	30	139.80 (10)	107.70 (6.5)	138.63 (9)	95.90 (5)	84.37 (3)	170.60 (11)	92.30 (4)	107.70 (6.5)	83.00 (2)	**45.30 (1)**	117.20 (8)	8.62E-08
	50	124.53 (9)	80.77 (3)	147.23 (10)	114.73 (8)	96.67 (5)	169.43 (11)	**74.60 (1)**	96.40 (4)	110.50 (7)	75.40 (2)	99.50 (6)	1.15E-06
	70	122.87 (7)	87.73 (5)	148.87 (10)	72.70 (3)	124.13 (8)	180.03 (11)	**57.00 (1)**	114.70 (6)	124.40 (9)	83.90 (4)	67.50 (2)	5.20E-11
Average		145.59	73.78	153.90	69.20	105.22	185.53	71.88	110.68	92.17	66.61	115.53	
Rank average		8.88 (9)	3.60 (4)	9.13 (10)	3.17 (2)	6.17 (6)	10.83 (11)	3.50 (3)	6.48 (7)	4.79 (5)	**2.92 (1)**	6.54 (8)	

3.4 Improved artificial bee colony optimization algorithms

3.4.1 Adaptive artificial bee colony optimization algorithm

The ABC was created based on the behaviour of honey bee finding the flower nectar. So far, the classical ABC gives a good results and is an efficient optimization technique compared with other optimization algorithms. The ABC is good during exploration in finding good possible solution but its performance during exploitation is poor affecting its convergence speed [126]. For an optimisation algorithm to perform efficiently, it is crucial to have a balance between exploration and exploitation. Exploration is the ability to find the global optimum and exploitation is the process to apply the known information to find for better possible solutions. Some modifications on the foraging movement behaviour in finding the nectar have been proposed in recent years such as [127]–[133]. These modifications have focused on balancing between exploration and exploitation and the results have shown that the modified ABC has performed better than the original ABC.

In the adaptive ABC (AABC) proposed here, instead of using random step size, dynamic step size based on the fitness value of current search location in the searching area is used. The approach leads to a higher convergence speed and better accuracy since the movement of bees to the next possible solution is guided by their fitness value. If the value of fitness is large, the corresponding location in the search space is far from the optimum solution. Small fitness, on the contratary, indicates that the solution arrived by the is good and near to the optimum point. Small step size will give more opportunity to perform the search within the current position while larger step size will allow the bees to search away from the current position and the make bee move faster to the optimum point. By varying the search radius within specified range $[a_1, a_2]$ and using combination of exponential and linear trajectory good variation of step size can be realised through the iteration; a_1 is set to the lower range (> 0) and a_2 is set to the high range (≤ 1).

In order to further improve the ABC algorithm an acceleration constant, ∂ , is introduced into the algorithm along with the best solution so far, x_j and the chaotic maps. The ∂ is adaptively implemented from dynamic step size and the concept is based on the work of Nasir [134] with some modifications as described below. The search in AABC is formulated as follows

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj})\partial + \propto_{chao} (x_j - x_{kj})\partial \tag{3.6}$$

where v_{ij} is the new possible solution depending on the previous solution x_{ij}. x_j is the best so far solution in jth parameter with the ability to fine tune the step size and to ensure the algorithm does not get trapped at local optimum. x_{kj} is neighbour solution, $\emptyset_{ij}$ is random value in [-1,1] and $\propto_{chao}$ is chaotic map ranging from -1 to 1; this chaotic map is used to give sense of randomness to the step size. ∂ is non-negative and controls the value of step size.

In the next subsection, three modified versions of ABC algorithms are presented. These are Chaotic linear adaptive artificial bee colony (CLABC) algorithm, Chaotic exponential adaptive artificial bee colony (CEABC) algorithm and Chaotic linear exponential adaptive artificial bee colony (CLEABC) algorithms. The modifications provided empowers the algorithms with dynamic and efficient exploration and exploitation capability. Beside the introduction of best solution so far and chaotic maps in equation (3.6) , two types of acceleration constant namely chaotic exponential and chaotic liner acceleration are introduced. The acceleration constant can alter the movement direction of the bee.

3.4.1.1 Chaotic linear adaptive artificial bee colony algorithm

In the CLABC algorithm, the fitness difference between the best fitness so far and the best fitness in current iteration is formulated in linear form;

$$\partial_L = \Delta_a \div \left(1 + {}^{b_1}\!/_{b_2(|fitness(x_{ij}) - x_j|)}\right) + a_2 \tag{3.7}$$

where ∂_L is linear acceleration constant, Δ_a is a tunable maximum bee step size with the value must be choose within [0,-1]. a_2 is maximum radius in particular itterations with the value must be choose within [0,-1]. b_1 is positive constant to be tuned heuristically, b_2 is a positive scalling factor . x_j is the best so far solution in jth parameter while $fitness(x_{ij})$ is finess value in particular food location.Tuneable parameters b_1 *and* b_2 have the role to ensure that ∂_L does not grow too big or not too small. By applying ∂_L into ABC, the step size for next location is more effective and dynamic. The role of the best fitness in current cycle, x_j is to ensure those bees are flying toward the best position in a particular cycle. When $|fitness(x_{ij}) - x_j|$ is small the $\frac{b_1}{b_2(|fitness(x_{ij}) - x_j|)}$ will become big, the bee will search near to the maximum step size. And when the $|fitness(x_{ij}) - x_j|$ is big, the $\frac{b_2}{(|fitness(x_{ij}) - x_j|)}$ will become small, the bee will search for possible solution near to the minimum step size. This movement of search will vary linearly and depending on the fitness value of current iteration and also in particular food location. The update relation for CLABC is thus given as

$$v_{ijL} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj})\partial_L + \propto_{chao} (x_j - x_{kj})\partial_L \qquad (3.8)$$

3.4.1.2 Chaotic exponential adaptive artificial bee colony algorithm

In the CEABC algorithm, the step size is modified to ensure the searching movements for the best solution are more dynamic and efficient. Instead of using the concept of random movement, the exponential constant, ∂_E is been introduced. ∂_E enables to control the size of step size exponentially. The exponential characteristic can lead the searching toward to the optimum point. The CEABC uses an exponential function for updating the new location of bee as follows

$$\partial_E = \Delta_a \div \left(1 + {}^{b_1}\!/_{b_2 exp\left(g_1(|fitness(x_{ij}) - x_j|)\right)}\right) + a_2 \qquad (3.9)$$

where ∂_E is constant exponential acceleration, $b_1\, and\, b_2$ are constant to be tuned heuristically and b_2 acts as scaling factor. Δ_a is a tunable maximum bee step size with value within [0,-1]. a_2 is maximum radius in particular itterations with the value must be choose within [0,-1]. The positive scaling factor g_1 will control the value of $exp(g_1|fitness(x_{ij}) - x_j|)$ not to grow big when the fitness deviation is large and b_2 is used to scale down the $\frac{b_1}{(b_2 exp|g_1 fitness(x_{ij}) - x_j|)}$ to the become very small and this will lead the bee go to the nearest position of the optimum point. The other processes follow the original ABC. The CLABC is mathematically formulated as

$$v_{ijE} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj})\partial_E + \propto_{chao} (x_j - x_{kj})\partial_E \qquad (3.10)$$

3.4.1.3 Chaotic linear exponential adaptive artificial bee colony algorithm

In the CLEABC algorithm, a linear acceleration constant, ∂_L and exponential acceleration constant, ∂_E are used together. The search strategies of bees are been modified to enhance the exploration capabilities of onlooker bee and employed bees. The bee movement in the search area in onlooker bee and employed bee stages are modified to let the bees move to the food sources exponentially and then adjust their position linearly. The update equation of CLEABC is given as

$$v_{ijLE} = x_{ij} + rand(-1,1)\left(x_{ij} - x_{kj}\right)\partial_E + \propto_{chao}\left(x_j - x_{kj}\right)\partial_L \qquad (3.11)$$

where v_{ijLE} is the new possible solution depending on the previous solution x_{ij}. x_j is the best so far solution in jth parameter with the ability to fine tune the step size and to ensure the algorithm does not get trapped at local optimum. x_{kj} is neighbour solution, $\emptyset_{ij}$ is random value in [-1,1], and $\propto_{chao}$ is chaotic map ranging from -1 to 1; this chaotic map is used to give sense of randomness to the step size. The non-negative acceleration constant, ∂ controls the value of step size.

3.4.2 Computational steps of adaptive artificial bee colony algorithm

The main computation steps of adaptive ABC are similar to those of the original ABC discussed in chapter 2 with some changes. The first change is the formulation of new position of solution in onlooker bee and employed bee stages and the second chance is the initial distribution using chaotic and opposition based learning method as described earlier. As mention earlier, three types of formulation are used in the AABC for provision of new positions of solutions leading to the CLABC, CEABC and CLEABC algorithms. The computation steps of the AABC algorithm are given in the pseudo code below

Algorithm 3.3: Adaptive Artificial Bee Colony Algorithm (AABC)

```
1:  Objective function of $f(x_d)$, Dimension, D, Number of Food, NF, Number of Bee,
    Limit value, upper and lower boundary of search space, maximum iteration $MCN$
2:  **Initialization:**
3:  Set initial Distribution of population,  in the search space using Chaotic-OB
4:  Evaluate the population fitness
5:  Iteration = 1
6:  **while** iteration, $k \leq$ maximum iteration, $MCN$ **do**
7:      **for all** $i = 1$ to $SN$ **do**
```

```
8:          Produced new food source
                $v_{ijL} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})\partial_L + \propto_{chao} (x_j - x_{kj})\partial_L$
                $v_{ijE} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})\partial_E + \propto_{chao} (x_j - x_{kj})\partial_E$
                $v_{ijLE} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})\partial_E + \propto_{chao} (x_j - x_{kj})\partial_L$
9:          Evaluate the fitness of new food source $v_{ij}$
10:         Compare fitness between $x_{ij}$ and $v_{ij}$ and choose the better one. (greedy
            selection)
11:         If the solution is not improved $trial_i = trial_i + 1$
12:      **End for**
13:    Calculate the probability, $probability(i)$ using equation (2.6)
14:   **end while**
15    $\boldsymbol{t = 0, i = 1}$
16:   **repeat**
17:       **If random** < $probability(i)$ **then**
18:           Produced new food source using
                  $v_{ijL} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})\partial_L + \propto_{chao} (x_j - x_{kj})\partial_L$
                  $v_{ijE} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})\partial_E + \propto_{chao} (x_j - x_{kj})\partial_E$
                  $v_{ijLE} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj})\partial_E + \propto_{chao} (x_j - x_{kj})\partial_L$
19:           Evaluate the fitness of new food source $v_{ij}$
20:           Compare fitness between $x_{ij}$ and $v_{ij}$ and choose the better one. (greedy
              selection)
21:           If the solution is not improved  $trial_i = trial_i + 1$, otherwise $trial_i = 0$
22:           $t = t + 1$
23:       **End if**
24:   **Until** $(t = SN)$
25:   **If** $trial_i > limit$ **then**
26:         Replace abundant food source with a new food source using
                  $x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j)$
27:   **End if**
28:   Keep the best solution
29:   Iteration = iteration +1
30:   **Until** (iteration = MCN)
```

3.5 Hybridisation of spiral dynamic algorithm and artificial bee colony algorithm

Combining more than two algorithms is an alternative way to improve the performance of the algorithms. Such strategies use the best features in each algorithm to create new algorithms. The expectation in a hybrid strategy is that the algorithm should be more accurate and able to outperform the original algorithms in solving problems in applications [135]–[146]. The proposed new hybrid algorithms using SDA and ABC are introduced in this section.

These combinations of algorithms are developed based on the explanations in introduction of ABC and SDA in the previous chapters. The ABC is a good algorithm but as its structure consists of multiple search agents, during exploration and exploitation, it struggles in case of high dimension problems. The time taken to run the algorithm can be considered long because of this issue. However, ABC is able to give good and accurate final value. SDA, on the other hand, can easily get trapped at local optimum because of fix trajectory (lack of randomness) but in one hand, the execution time of SDA is relatively short. Taking the advantages of both algorithms, three types hybrid algorithms are proposed. The first algorithm is to improve the SDA by embedding scout bee into SDA. The second algorithm is to use spiral trajectory from SDA in ABC and the third hybrid algorithm in this research is combination of both algorithms in series where SDA will perform exploration while in later stage ABC will perform the exploitation.

3.5.1 Scout bee spiral dynamic algorithm

In the proposed Scout bee spiral dynamic algorithm (SBSDA), a scout bee phase is introduced to ensure that the algorithm does not get trapped at local optima and its performance is improved. The scout bee phase is adopted from ABC algorithm. Although a new phase is introduced, the main body of SDA is still maintained with Chaotic-OB initial distribution.

In the SBSDA, the scout bee will act as a reset button when SDA is trapped in local optimum point after iteration count. The reset mechanism is are trigger by predetermined value of limit. The limit counter will increase by 1 when the solution is not improved in each iteration. When the limit value is reached the non-improved solutions will regenerate to the new solutions by random distribution. When the new solutions are randomly reproduced, the centre of spiral and the best solution from the past iteration together with new solutions are used to lead the spiral movement to better optimum points in next iteration. The limit will reset back to 0 if the solution is improved. The mathematical formulation to reproduce the new solution is defined as:-

$$x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right) \tag{3.12}$$

where j is a dimensional vector, x_i is a solution, $i = 1,2 \dots, n$, x_{max} and x_{min} is the boundaries of the search space. The computation steps of SSDA are shown in pseudo code below

Algorithm 3.4: Scout Bee Spiral Dynamic Algorithm

```
 1:  Objective function of $f(x_d)$
 2:  Initialisation:
 3:    Set upper and lower boundary of search space
 4:    Set maximum iteration, $MCN$
 5:    Set number of search point, $m$ where $m \geq 2$
 7:    Set spiral angle, $\theta$ where $0 \leq \theta \leq 2\pi$
 8:    Set initial points $x_i(0) \in R^n, i = 1,2,\dots,m$ in the search space using Chaotic-OB
 9:    Choose the centre of spiral $x^*$ as $x^* = x_{i_g}(0), i_g = \arg min_i\, f(x_i(0)),\ i = 1,2,\dots,m$
10:  while iteration, $k \leq MCN$ do
11:      for all $i$ to $m$ do
12:          Updating $x_i$ ;
             $x_i(k+1) = S_n(r,\theta)x_i(k) - [S_n(r,\theta) - I_n]x^*, i = 1,2,\dots\dots,m.$        ( 1. 1 )
13:      End for
14:      Updating $x^*$; $x^* = x_{i_g}(k+1), i_g = \arg min_i f(x_i(k+1)),\ i = 1,2,\dots,m$
25:  If $trial_i > limit$ then
26:  Replace non improved point with a new point distribution using
             $x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j)$
27:  End if

15:      $k = k + 1$
16:  end while
```

3.5.2 Spiral bee algorithm

The spiral bee algorithm (SBA) is created from combination of chaotic map, searching technique of ABC and searching movement/direction of SDA. Firstly, the initial distribution of food source is made using Chaotic-OB distribution. Chaotic-OB distribution is used to ensure the initial population is from good solutions. In step 2, an employed bee moves to the new food source in spiral pattern using logistic chaotic map as a spiral radius, r_{chao}. The mathematical formulation for new food source, $x(k+1)$ is given as

$$x_i(k+1) = S_n(r_{chao},\theta_{chao})x_i(k) - [S_n(r_{chao},\theta_{chao}) - I_n]x^*, i = 1,2,\dots\dots,m. \quad (3.13)$$

where r_{chao} is given by chaotic maps with scale value from 0 to 1. θ_{chao} is chaotic rotational angle with angle between 0 to 2π, I_n is identity matrix , x^* is centre of spiral, k is number of iterations, m is number of points, $S_n(r,\theta) = rR^{(n)}(\theta_{1,2},\theta_{1,3},\ldots\ldots\theta_{n,n-1})$, $R^{(n)}$ is composition rotation matrix, where $R^{(n)}(\theta_{1,2},\theta_{1,3},\ldots\ldots\theta_{n,n-1})$ is rotation $n \times n$ matrix. To enhance the exploitation capability in local region, a new step was introduced by creating local search around the $x_i(k+1)$ point in random step size. By doing this, the searching can be focused in small area where the possibility of finding good solution will increase. The search for the new point $v_i(k+1)$ within this area is produced by

$$v_i(k+1) = x_i(k+1) + \emptyset(x_i(k+1) - x_{ki}) \tag{3.14}$$

where $v_i(k+1)$ represents new position of solution is, $x_i(k+1)$ is previous solution and x_{ki} is neighbour solution, and i & $k = 1,2,\ldots SN$,. The neighbour position of solution x_k is randomly chosen and it must not be the same with position of solution x_i. The function of random value, $\emptyset$ in the equation is to control the position of neighbour food source within $x_i(k+1)$. The difference between values of $x_i(k+1)$ and x_{ki} getting smaller will create smaller perturbation on position$x_i(k+1)$. Thus, the step size reduces as the search approaches the best solution in the search space.

After producing $v(k+1)$, The greedy selection is applied between solution $v(k+1)$ and $x_i(k+1)$. The Fitness of the two solutions is evaluated and the best value will be decided based on their fitness. If the $x_i(k+1)$ is better compared to $v(k+1)$, it will chosen as center of spiral, otherwise it will keep the previous location will be kept. This process will repeat until the termination criteria criterion is satisfied. The steps of SBA algorithm are presented in pseudo code below.

Algorithm 3.5: Spiral bee algorithm

```
1:  Objective function of $f(x_d)$, Dimension, D, Number of Food, NF, Number of Bee,
    Limit value, upper and lower boundary of search space, maximum iteration $MCN$
2:  **Initialization:**
3:  Set initial Distribution of population,  in the search space using Chaotic-OB
4:  Evaluate the population fitness
5:  Iteration = 1
6:  **while** iteration, $k \leq$ maximum iteration, $MCN$ **do**
7:      **for all** $i = 1$ to $SN$ **do**
8:          Produced new food source in spiral trajectory
```

```
        $x_i(k+1) = S_n(r_{chao}, \theta_{chao}) x_i(k) - [S_n(r_{chao}, \theta_{chao}) - I_n] x^* , i = 1,2, \dots \dots , m.$
9:          Evaluate the fitness of new food source $x_i$
10          Move the bee in random near $x_i$
11              $v_i(k+1) = x_i(k+1) + \emptyset(x_i(k+1) - x_{ki})$
12:         Evaluate the fitness of new food source $v_i$
13:         Compare fitness between $x_i(k+1)$ , $v_i(k+1)$ and $x_i$ , then choose the better
            one. (greedy selection)
14:         If the solution is not improved $trial_i = trial_i + 1$
15:      **End for**
16:    Calculate the probability, $probability(i)$ using equation (2.6)
17:  **end while**
18:  $\boldsymbol{t = 0, i = 1}$
19:  **repeat**
20:       **If random** < $probability(i)$ **then**
              Produced new food source using
21:               $x_i(k+1) = S_n(r_{chao}, \theta_{chao}) x_i(k) - [S_n(r_{chao}, \theta_{chao}) - I_n] x^* , i = 1,2, \dots \dots , m.$
22:           Evaluate the fitness of new food source $x_i(k+1)$
23:           Move the bee in random near $x_i$
24:               $v_i(k+1) = x_i(k+1) + \emptyset(x_i(k+1) - x_{ki})$
25:           Evaluate the fitness of new food source $v_i(k+1)$
26:           Compare fitness between $x_i(k+1)$ , $v_i(k+1)$ and $x_i$ , then choose the
              better one. (greedy selection)
27:           If the solution is not improved $trial_i = trial_i + 1$, otherwise $trial_i = 0$
28:           $t = t + 1$
29:       **End if**
30:  **Until** $(t = SN)$
31:  **If** $trial_i > limit$ **then**
          Replace abundant food source with a new food source using
32:               $x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right)$
33:  **End if**
34:  Keep the best solution
35:  Iteration = iteration +1
36:  **Until** (iteration = MCN)
```

3.5.3 Hybrid artificial bee colony spiral dynamic algorithm

The hybrid artificial bee colony spiral dynamic (HABCSD) algorithm is a series type hybrid strategy where the ABC and SDA parts of the algorithm are in a sequential order. In this algorithm, the first part includes chaotic exponential adaptive artificial bee colony algorithm (CEABC) and the second part is a chaotic spiral dynamic algorithm (CSDA). Combination of these two algorithms can reduce the risk of algorithm to get trapped at local optimum and may speed up the algorithm to converge.

The algorithm used in the first part of HABCSD is CEABC. In this stage, the initial distribution are done randomly. This method of distribution will allow the solutions to spread more widely and cover large area in the search space. The search for the next solution in onlooker bee and employed bee stage is then performed in exponential way together with chaotic maps pattern. The exponential step size and chaotic map give chance to the bees to move dynamically and have the potential to find good food sources much better. There are no scout bees in this stage to reset the solution if the food is depleted. The scout bee phase from the original ABC is incorporated into the last stage after execution of CSDA. The best solution found is stored and be used as a center of spiral in the next part of HABCSD.

The exploitation phase takes place in the second part of HABCSD, where, the CSDA method is used. Instead of moving straight toward to the global optimum, it is better for bees to find good food sources in wide area along their way to the global point. This can be fulfilled by moving the bees in spiral pattern. The bees will fly to find the food sources in spiral pattern where chaotic maps are for radius and the values will change in each iteration. In the first movement of bees, the bees move in dynamic step size to the local optimal point. Then in a later phase of spiral bee movement, the bees are move toward to the global optimal point. Thus be said, by moving dynamically toward the optimal point, the bees are able to explore the whole area in the search space and get to a good final solution.

After exploration by CEABC and exploitation by CSDA, the best solution found is stored, and after that, the whole process repeats over and over again until the maximum number of iterations is reached. In order to reduce the chances of HABCSD algorithm getting trapped at local optimum, the scout bee phase is introduced in the last phase of HABCSD. The limit counter of scout bee will trigger by 1 in each iteration of unimproved solution. When the limit reaches a predetermined value, the solutions in particular population will change to the new solution in random.

The part of HABCSD is an exploration phase, where the movement of bees is in random directions. The second part of HABCSD is exploitation phase during which, the bees move in spiral pattern and they are guided to the global optimal point. This action enable to produce good final solutions. The steps of HABCSD algorithm are presented in pseudo code below.

Algorithm 3.6:- Hybrid artificial bee colony spiral dynamic algorithm

```
1:  Objective function of $f(x_d)$, Dimension, D, Number of Food, NF, Number of Bee,
    Limit value, upper and lower boundary of search space, maximum iteration $MCN$
2:  **Initialization:**
3:  Set initial Distribution of population,  in the search space using **algorithm 1(**initial
    Chaotic-OBL distribution)
4:  Evaluate the population fitness
5:  Iteration = 1
6:  **while** iteration, $k \leq$ maximum iteration, $MCN$ **do**
7:     **for all** $i = 1$ to $SN$ **do**
8:        Produced new food source
              $v_{ijE} = x_{ij} + rand(-1,1)\left(x_{ij} - x_{kj}\right)\partial_E + \propto_{chao}\left(x_j - x_{kj}\right)\partial_E$
9:        Evaluate the fitness of new food source $v_{ij}$
10:       Compare fitness between $x_{ij}$ and $v_{ij}$ and choose the better one. (greedy
          selection)
11:       If the solution is not improved $trial_i = trial_i + 1$
12:    **End for**
13:  Calculate the probability,  $probability(i)$ using equation (2.6)
14: **end while**
15  $\boldsymbol{t = 0, i = 1}$
16: **repeat**
17:     **If random** < $probability(i)$ **then**
18:        Produced new food source using
              $v_{ijE} = x_{ij} + rand(-1,1)\left(x_{ij} - x_{kj}\right)\partial_E + \propto_{chao}\left(x_j - x_{kj}\right)\partial_E$
19:        Evaluate the fitness of new food source $v_{ij}$
20:        Compare fitness between $x_{ij}$ and $v_{ij}$ and choose the better one. (greedy
           selection)
21:        If the solution is not improved  $trial_i = trial_i + 1$, otherwise $trial_i = 0$
22:        $t = t + 1$
23:     **End if**
24: **Until** $(t = SN)$
25:   Set spiral angle, $\theta$ where $0 \leq \theta \leq 2\pi$
26:   Choose the centre of spiral $x^*$ as $x^* = x_{i_g}(0), i_g = \arg min_i\, f\left(x_i(0)\right),\ i =$
    $1,2,\dots,m$
27: **while** iteration,$k \leq\ MCN$ **do**
28:    **for all** $i$ to $m$ **do**
          Updating $x_i$ ;
29: $x_i(k+1) = S_n(r_{chao}, \theta_{chao})x_i(k) - [S_n(r_{chao}, \theta_{chao}) - I_n]x^*, i$
              $= 1,2, \dots \dots, m.$
30:    **End for**
31:    Updating $x^*$; $x^* = x_{i_g}(k+1), i_g = \arg min_i f\left(x_i(k+1)\right),\ i = 1,2,\dots,m$
```

```
32:     $k = k + 1$
33: **If** $trial_i > limit$ **then**
34:         Replace abundant food source with a new food source using
            $x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right)$
35: **End if**
36: Keep the best solution
37: Iteration = iteration +1
38: **Until** (iteration = MCN)
```

3.6 Summary

In this chapter, the proposed algorithms based on the SDA and ABC have been described. In the first algorithm, the initial distribution modified and the novel Chaotic-OB has been developed and used as an initial distribution for all proposed algorithms. The chaotic maps are then used in CSDA to give dynamic movement and to help SDA escape from trapped at local optima. The introduction of three adaptive ABC with different step sizes enable to improve the accuracy of ABC. In the three hybrid versions of ABC and SDA, the use of good features from both algorithms enable to help the proposed hybrid algorithms move toward the optimum point more efficiently. The hybrid strategies also give balance between exploitation and exploration and thus leading the algorithm to achieve better performance. In the next chapter, the proposed algorithms will be tested with single objective of unconstrained and constrained optimization problems. The performances of the proposed algorithms will also be assessed in comparison to those of original ABC and SDA.

Chapter 4
Single Objective Unconstrained and Constrained Problems

4.1 Introduction

This chapter present the performance analyses of the proposed algorithms in unconstrained and constrained single objective optimisation problems. In unconstrained single objective problems, the algorithms are tested in ten standard benchmark functions and in sixteen CEC2014 benchmark functions. While in the constrained problems, the algorithms are tested in ten (10) CEC 2006 constrained benchmark functions and in five engineering design problems. All the test benches have varieties of problem landscapes, dimensions and complexity levels. Non-parametric test and parametric test are used to evaluate the performances of the algorithms in comparison to the original SDA and ABC.

The experiments are carried out using personal computer with CPU Intel ® core i7-3.40Ghz, 256G solid state drive and 8 GByte memory. The algorithm are coded using MATLAB R2015b as a programming and simulation software platform.

4.2 Single objective problems

In single objective optimisation problems, the optimum point is either maximum or minimum point. An example is by arranging the raw material stock, the staff and the marketing is to efficiently maximize the profit. In such examples the objective function, $f(x)$, is often defined as minimisation or maximisation cost function [147]. According to [8] the local optimum is an optimum point in subset of x and global optimum is an optimum point for all the domain x. Figure 4.1 shows the location of minimum and maximum point for local and global optimum.

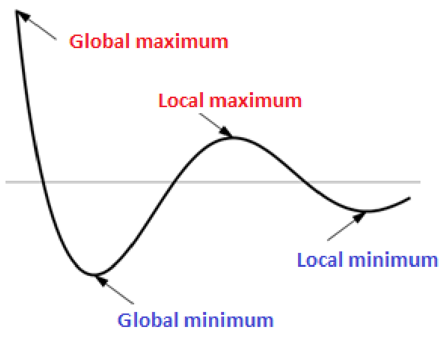


Figure 4.1:- Diagram of maximum and minimum point

The unconstrained optimisation problem can be represented as:-

$$optimise\ f(x), x = (x_1, x_2, \ldots\ldots, x_n)$$

$$\text{where } x^L \leq x_i \leq x^U \quad , \ i = 1,2,\ldots,n$$

The n number of variables x located within the search space are bounded by the lower bound, x^L and upper bound, x^U.

The all proposed algorithms are tested in single objective unconstrained optimisation problems with 10, 30 and 50 dimensions. There are 10 standard benchmark functions and 16 CEC2014 benchmark functions used to evaluate the performance of the proposed algorithms. These standard benchmark functions are from Simon [148] and have widely been used by researchers to test their algorithms in comparison to other algorithms [148]–[151]. They are also suitable to evaluate the proposed algorithms because they have different types of characteristics and fitness landscape surfaces with different levels of complexity. Table 4.1 shows the standard benchmark functions used in the experiments where most of the optimal points, f^* are located at zero.

Table 4.1: Standard benchmark functions [149].

No	Benchmark function	Type	Properties	$f^* = f(x_i^*)$
$f_1(x)$	Sphere	Unimodal	Continuous, convex has no local minima except for the global one	0
$f_2(x)$	Ackley	Multimodal	Many local minimum, continous, scalable and non-separable.	0
$f_3(x)$	Rosenbrock	Unimodal	The global minimum lies in a narrow, parabolic valley	0
$f_4(x)$	Griewank	Multimodal	Many local minimum and it regularly distributed	0
$f_5(x)$	Rastrigin	Multimodal	Many local minimum and it regularly distributed	0
$f_6(x)$	Schwefel	Multimodal	Many local minima	0
$f_7(x)$	Levy	Multimodal	Many local minima	0
$f_8(x)$	Sum Squares Function	Unimodal	No local minimum, one global minimum, continuous, convex	0
$f_9(x)$	Dixon-Price Function	Unimodal	Continuous, non-separable	0
$f_{10}(x)$	Rotated Hyper-Ellipsoid Function	Unimodal	Continuous, convex	0

Another set of benchmark function used in the tests are based on IEEE congress on evolutionary computation (CEC 2014). [152]. These comprise 16 functions with various properties and different fitness landscapes. These benchmark functions are used to test the performance of the proposed algorithms, as they possess features that are more advanced with complicated landscapes to solve. The functions further have different optimal points to challenge the algorithms. Table 4.2 shows the list of CEC 2014 benchmark functions together with their properties and their optimal points, f^*.

Table 4.2: Summary of the CEC'14 test functions [152]

Function	Name of Benchmark function	Properties	$f^* = f(x_i^*)$
$f_{11}(x)$	Rotated High Conditioned Elliptic Function	Unimodal, non-separable, quadratic ill-conditioned	100
$f_{12}(x)$	Rotated Bent Cigar Function	Unimodal, non-separable, smooth but narrow ridge	200
$f_{13}(x)$	Rotated Discus Function	Unimodal, non-separable, with one sensitive direction	300
$f_{14}(x)$	Shifted and Rotated Rosenbrock's Function	Multimodal, non-separable, having a very narrow valley from local optimum to global optimum	400
$f_{15}(x)$	Shifted and Rotated Ackley's Function	Multimodal, non-separable	500
$f_{16}(x)$	Shifted and Rotated Weierstrass Function	Multimodal, non-separable, continuous but differentiable only on a set of points	600
$f_{17}(x)$	Shifted and Rotated Griewank's Function	Multimodal, non-separable, rotated Shifted	700
$f_{18}(x)$	Shifted Rastrigin's Function	Multimodal, separable, local optima's number is huge	800
$f_{19}(x)$	Shifted and Rotated Rastrigin's Function	Multimodal, non-separable, local optima's number is huge	900
$f_{20}(x)$	Shifted Schwefel's Function	Multimodal, separable, local optima's number is huge and second better local optimum is far from the global optimum	1000
$f_{21}(x)$	Shifted and Rotated Schwefel's Function	Multimodal, non-separable, Local optima's number is huge and second better local optimum is far from the global optimum	1100
$f_{22}(x)$	Shifted and Rotated Katsuura Function	Multimodal, non-separable, Continuous everywhere yet differentiable nowhere	1200
$f_{23}(x)$	Shifted and Rotated HappyCat Function	Multimodal, non-separable	1300
$f_{24}(x)$	Shifted and Rotated HGBat Function	Multimodal, non-separable	1400
$f_{25}(x)$	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	Multimodal, non-separable	1500
$f_{26}(x)$	Shifted and Rotated Expanded Scaffer's F6 Function	Multimodal, non-separable	1600

4.2.1 Performance measurement

The performances of the proposed algorithms are tested and evaluated with the original algorithms, ABC and SDA in solving the single objective optimisation problems. To ensure the proposed algorithms are good in solving those problems, reliability tests of the algorithms are carried out using the method reported by Kasdirin [153] where the success criterion is set at 10e-4 . This objective of the test is to assess the capability of the proposed algorithms to reach predetermined threshold point (in this test the threshold is 10e-4). If the algorithm is able converge to or beyond the threshold point, that run can be said as a successful run and the number of fitness evaluation and time taken to reach that threshold point is recorded for further evaluations. But if in that run the algorithm fails to converge to the threshold point, that run will be indicated as unsuccessful or labelled as 'US'. By calculating the percentage of success rate (SR) the reliability of the algorithm to reach the fitness value within predefined number of fitness evaluations (NFEs) can be evaluated. The SR is given as

$$SR = \frac{n_{SR}}{n_{run}} x100\%$$

where n_{SR} is the number of successful runs and n_{run} is the number of runs. The average success rate $Average_{SR}$ is given as:-

$$Average_{SR} = \frac{\sum_{i=1}^{n_{fun}} SR_i}{n_{fun}}, \quad i = 1,2, \ldots \ldots, n_{fun}$$

where n_{fun} is total functions used.

Beside using reliability test to check the performance of algorithms, the non-parametric and parametric tests are also used to show the performance of the proposed algorithms verses the original algorithms. In the parametric test, to be considered as a good algorithm, the mean value should be small to show that the algorithm is able to converge to the optimal value. The standard deviation shows how far the optimal values deviate from average. A small value of deviation indicates that the algorithm is more stable and more accurate. Non-parametric test is used to check the significant improvement by the proposed algorithms in comparison to the original SDA and ABC.

In this research, the non-parametric Kruskal-Wallis test is used. The Kruskal Wallis test are enable to compare or to evaluate the significant difference from more than two groups. In the test, the samples from all groups must be mutually independent and must be in the same population distribution group. Kruskal-Wallis test is one way variance test where the hypothesis is that, all the groups perform the same and in the same median. The hypothesis is rejected when the p-value is less than 0.05 confident interval, where at least one of the groups is significantly different compared to others. In the experiments, the smallest rank value indicates that the algorithm perform better to reach the global optimum.

Nine algorithms are evaluated, two of which are the original SDA and ABC. Table 4.3 shows the initial parameters for the tests. These initial parameters are used in all tests for fair performance comparison of the algorithms. To evaluate the robustness and stability of the of the proposed algorithms, each algorithm is tested in 30 independent runs for each problem. The numerical results consist of the mean, standard deviation and the best result obtained in each test. The 30 independent runs is commonly used by researchers in the field to evaluate the performances of their algorithms and it provides opportunity for researchers to perform fair comparison with other algorithms with same parameter settings [148]–[151] .

Table 4.3:- Initial Parameter for single objective tests

	Algorithms								
Parameters	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
r	-	0.95	0.95	-	-	-	0.95	0.95	0.95
θ	-	45^0	45^0	-	-	-	45^0	45^0	45^0
m	-	30	30	-	-		30	30	30
Limit	100	-	-	100	100	100	100	100	100
NF	30	-	-	30	30	30	30	30	30
itter	1600	1600	1600	1600	1600	1600	1600	1600	1600
$x(0)_{chao}$	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
Δ_a	-	-	-	1	1	1	-	-	-
a_2	-	-	-	0.5	0.5	0.5	-	-	-
b_1	-	-	-	0.5	0.5	0.5	-	-	-
b_2	-	-	-	0.8	0.8	0.8	-	-	-

4.2.2 Standard benchmark functions tests

In this section, the performances of the proposed algorithms and the original algorithms are assessed with ten standard benchmark functions. The statistical results of the proposed algorithms based on 30 independent runs with different dimensions (10, 30, 50 and 70) are shown in Tables 4.4 – 4.7, where the smallest average optimum value indicates that the algorithm is able to produce a solution near to the global optimum point. The standard deviations are also shown for demonstrating the robustness of the algorithms, where the lowest standard deviation means the optimal solutions generated is more consistent and stable. The best and worse optimal value are also recorded to show the performance of algorithm in the search space. The best average results are highlighted in bold font. In Table 4.4, the statistical results with 10 dimension are presented. It is noted that the original SDA with 10 dimension could not reach near to the optimal point in all benchmark functions as it apparently got trapped at local optimum. The results shown in Tables 4.5, 4.6 and 4.7 indicate similar issues for SDA in dimension 30, 50 and 70. The original ABC, on other hand, was able to converge to the near optimal point in all benchmark functions for all dimensions. The proposed algorithms, as noted outperformed the original SDA in all benchmark functions.

The featured integration of ABC and SDA has enabled HABCSDA to outperform all algorithms in at least 5 benchmark functions for all tested dimensions by converging nearest to the optimal point. It is also noted that HABCSDA achieved a good result in f_3 with dimension 10. However, in dimensions 30, 50 and 70, ABC performed better. Table 4.4 shows that the modified versions of ABC (CEABC, CLABC and CLEABC) performed better than the original ABC, while in the modified version of SDA, CSDA performed better than SDA in all benchmark functions in all tested dimensions.

Table 4.4: -Results of benchmark function with 10 dimension

		ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
f_1	Avg	9.53E-17	1.70E+01	5.23E+00	6.92E-17	3.51E-25	**2.47E-29**	4.80E+00	1.94E-16	3.67E-29
	SD	3.60E-17	1.81E+01	4.94E+00	1.48E-17	1.11E-24	7.75E-29	5.47E+00	8.70E-17	1.16E-28
	worse	1.91E-16	4.98E+01	1.44E+01	8.31E-17	3.51E-24	2.45E-28	1.82E+01	3.09E-16	3.67E-28
	best	5.84E-17	2.68E-01	1.24E+00	3.48E-17	8.98E-46	2.50E-50	5.18E-01	8.39E-17	2.39E-55
f_2	Avg	8.35E-15	1.92E+01	1.58E+01	5.51E-15	**8.88E-16**	**8.88E-16**	1.64E+01	2.72E-14	**8.88E-16**
	SD	1.07E-15	1.06E+00	1.72E+00	1.72E-15	0.00E+00	0.00E+00	3.62E+00	1.10E-14	0.00E+00
	worse	1.15E-14	2.03E+01	1.82E+01	7.99E-15	8.88E-16	8.88E-16	2.00E+01	4.35E-14	8.88E-16
	best	7.99E-15	1.71E+01	1.22E+01	4.44E-15	8.88E-16	8.88E-16	9.42E+00	1.51E-14	8.88E-16
f_3	Avg	3.73E-02	5.11E+03	9.28E+02	4.35E-02	6.11E+00	6.27E+00	2.13E+03	5.72E-02	**2.54E-02**
	SD	3.42E-02	9.81E+03	2.81E+03	3.71E-02	1.07E+00	1.61E+00	6.62E+03	4.61E-02	2.81E-02
	worse	1.27E-01	3.15E+04	8.93E+03	1.44E-01	7.06E+00	7.84E+00	2.10E+04	1.46E-01	7.73E-02
	best	5.75E-03	8.74E-01	2.23E-01	1.71E-02	4.55E+00	3.38E+00	1.16E+00	1.49E-02	9.74E-04
f_4	Avg	1.48E-03	6.32E+01	3.07E+01	3.44E-16	**0.00E+00**	**0.00E+00**	1.06E+01	1.96E-03	**0.00E+00**
	SD	2.97E-03	7.20E+01	2.07E+01	1.05E-15	0.00E+00	0.00E+00	7.18E+00	3.69E-03	0.00E+00
	worse	7.43E-03	1.97E+02	6.88E+01	3.33E-15	0.00E+00	0.00E+00	2.75E+01	1.01E-02	0.00E+00
	best	1.11E-16	2.35E+00	9.39E+00	0.00E+00	0.00E+00	0.00E+00	1.69E+00	1.11E-16	0.00E+00
f_5	Avg	**0.00E+00**	9.19E+01	4.73E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.04E+01	1.28E-14	**0.00E+00**
	SD	0.00E+00	2.91E+01	2.77E+01	0.00E+00	0.00E+00	0.00E+00	1.65E+01	1.56E-14	0.00E+00
	worse	0.00E+00	1.25E+02	1.13E+02	0.00E+00	0.00E+00	0.00E+00	8.06E+01	4.26E-14	0.00E+00
	best	0.00E+00	3.41E+01	2.06E+01	0.00E+00	0.00E+00	0.00E+00	3.22E+01	0.00E+00	0.00E+00
f_6	Avg	1.27E-04	2.57E+03	1.58E+03	1.27E-04	1.27E-04	3.26E+02	1.21E+03	**-1.9E+173**	1.27E-04
	SD	4.17E-13	8.04E+02	9.24E+02	0.00E+00	4.39E-13	1.02E+02	4.04E+02	-1.9E+173	4.39E-13
	worse	1.27E-04	3.07E+03	2.85E+03	1.27E-04	1.27E-04	4.57E+02	1.93E+03	-1.48E+06	1.27E-04
	best	1.27E-04	4.34E+02	6.51E+02	1.27E-04	1.27E-04	1.20E+02	7.90E+02	-1.9E+174	1.27E-04
f_7	Avg	**8.20E-17**	7.43E+00	6.12E+00	1.04E-16	9.55E-17	1.44E-16	4.98E+00	1.76E-16	1.12E-16
	SD	9.69E-18	5.90E+00	3.95E+00	4.06E-17	1.29E-17	6.76E-17	2.79E+00	8.09E-17	5.21E-17
	worse	9.54E-17	2.12E+01	1.34E+01	2.12E-16	1.10E-16	2.61E-16	1.06E+01	3.07E-16	2.21E-16
	best	6.61E-17	9.09E-01	1.53E+00	6.37E-17	7.04E-17	4.48E-17	1.46E+00	4.29E-17	5.86E-17
f_8	Avg	7.61E-17	1.40E+02	6.96E+01	5.08E-17	2.20E-189	2.47E-17	1.21E+02	1.80E-16	**0.00E+00**
	SD	1.53E-17	2.20E+02	4.22E+01	1.56E-17	0.00E+00	1.06E-17	1.31E+02	9.92E-17	0.00E+00
	worse	1.05E-16	7.58E+02	1.45E+02	8.17E-17	2.20E-188	4.45E-17	4.06E+02	3.27E-16	0.00E+00
	best	5.27E-17	3.07E+01	6.13E+00	2.59E-17	0.00E+00	1.30E-17	3.12E+00	6.95E-17	0.00E+00
f_9	Avg	**2.76E-09**	4.01E+02	8.33E+02	3.16E-06	6.67E-01	1.25E-05	1.12E+03	5.42E-05	1.35E-05
	SD	2.84E-09	7.32E+02	1.81E+03	3.70E-06	1.53E-16	1.35E-05	1.69E+03	5.37E-05	1.18E-05
	worse	8.26E-09	1.78E+03	5.60E+03	1.16E-05	6.67E-01	4.08E-05	4.70E+03	1.58E-04	3.65E-05
	best	2.00E-11	1.12E-06	1.71E-07	9.83E-08	6.67E-01	1.27E-06	3.95E-03	4.45E-06	4.26E-07
f_{10}	Avg	7.67E-17	1.95E+04	6.55E+03	4.46E-17	**0.00E+00**	2.44E-17	6.46E+03	1.83E-16	**0.00E+00**
	SD	2.24E-17	1.58E+04	5.18E+03	1.20E-17	0.00E+00	9.67E-18	5.21E+03	8.07E-17	0.00E+00
	worse	1.02E-16	4.53E+04	1.91E+04	6.86E-17	0.00E+00	4.33E-17	1.80E+04	2.73E-16	0.00E+00
	best	2.62E-17	1.10E+03	1.61E+03	2.67E-17	0.00E+00	1.35E-17	1.11E+03	8.36E-17	0.00E+00

Table 4.5: -Results of benchmark function with 30 dimension

		ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
f_1	Avg	8.07E-16	1.58E+02	3.64E+01	5.64E-16	**1.07E-48**	6.80E-25	5.26E+01	2.67E-11	4.26E-34
	SD	1.51E-16	4.95E+01	2.12E+01	9.18E-17	3.39E-48	2.14E-24	2.02E+01	3.84E-11	1.35E-33
	worse	1.13E-15	2.06E+02	6.28E+01	7.11E-16	1.07E-47	6.77E-24	1.05E+02	1.31E-10	4.26E-33
	best	6.38E-16	5.30E+01	8.08E+00	4.64E-16	2.67E-170	7.45E-147	3.38E+01	1.30E-12	8.36E-180
f_2	Avg	2.52E-10	2.00E+01	1.94E+01	2.99E-13	**8.88E-16**	**8.88E-16**	1.93E+01	7.46E-05	**8.88E-16**
	SD	1.10E-10	5.98E-01	6.38E-01	1.10E-13	0.00E+00	0.00E+00	7.19E-01	3.84E-05	0.00E+00
	worse	5.06E-10	2.04E+01	2.05E+01	5.44E-13	8.88E-16	8.88E-16	2.03E+01	1.56E-04	8.88E-16
	best	8.21E-11	1.84E+01	1.86E+01	1.75E-13	8.88E-16	8.88E-16	1.77E+01	3.00E-05	8.88E-16
f_3	Avg	**5.93E-02**	6.99E+04	7.92E+04	5.41E-01	2.63E+01	2.75E+01	2.11E+04	1.10E+00	8.80E-01
	SD	8.45E-02	6.88E+04	8.08E+04	7.55E-01	1.24E+00	5.37E-01	2.15E+04	1.56E+00	6.80E-01
	worse	2.84E-01	2.52E+05	2.18E+05	2.48E+00	2.75E+01	2.80E+01	5.24E+04	5.29E+00	2.11E+00
	best	2.61E-03	1.21E+04	4.88E+03	2.92E-02	2.35E+01	2.61E+01	9.94E+01	4.62E-02	1.55E-01
f_4	Avg	9.99E-04	5.34E+02	2.06E+02	2.57E-12	**0.00E+00**	**0.00E+00**	1.83E+02	1.37E-03	**0.00E+00**
	SD	3.00E-03	1.86E+02	1.39E+02	8.10E-12	0.00E+00	0.00E+00	9.19E+01	3.91E-03	0.00E+00
	worse	9.99E-03	6.88E+02	5.67E+02	2.56E-11	0.00E+00	0.00E+00	3.66E+02	1.25E-02	0.00E+00
	best	2.22E-16	1.90E+02	7.63E+01	1.11E-16	0.00E+00	0.00E+00	1.47E+01	2.90E-09	0.00E+00
f_5	Avg	1.93E-13	3.28E+02	2.51E+02	3.07E-13	**0.00E+00**	**0.00E+00**	2.47E+02	4.69E-05	**0.00E+00**
	SD	1.20E-13	1.03E+02	8.50E+01	7.34E-13	0.00E+00	0.00E+00	4.65E+01	1.26E-04	0.00E+00
	worse	4.55E-13	4.61E+02	4.62E+02	2.39E-12	0.00E+00	0.00E+00	3.31E+02	4.03E-04	0.00E+00
	best	5.68E-14	1.95E+02	1.61E+02	0.00E+00	0.00E+00	0.00E+00	1.90E+02	2.31E-07	0.00E+00
f_6	Avg	1.19E+01	8.72E+03	7.60E+03	3.82E-04	1.35E+02	1.69E+03	5.15E+03	**-1.90E+33**	1.72E+02
	SD	3.55E+01	2.70E+03	2.63E+03	2.85E-07	9.34E+01	1.45E+02	2.17E+03	6.02E+33	1.40E+02
	worse	1.18E+02	1.09E+04	1.04E+04	3.83E-04	2.59E+02	1.87E+03	9.74E+03	-2.97E+07	4.75E+02
	best	3.82E-04	4.50E+03	3.50E+03	3.82E-04	3.82E-04	1.42E+03	3.30E+03	-1.90E+34	4.47E-04
f_7	Avg	**8.55E-16**	3.67E+01	3.02E+01	1.08E-12	3.78E-11	5.06E-13	3.40E+01	4.29E-11	1.22E-10
	SD	1.41E-16	8.70E+00	7.74E+00	2.44E-12	2.51E-11	4.56E-13	1.79E+01	2.60E-11	1.47E-10
	worse	1.17E-15	4.82E+01	4.61E+01	7.96E-12	7.29E-11	1.23E-12	8.20E+01	8.26E-11	5.16E-10
	best	7.28E-16	2.14E+01	2.14E+01	5.01E-14	3.78E-12	6.30E-14	1.98E+01	1.10E-11	1.98E-11
f_8	Avg	7.48E-16	7.14E+03	2.72E+03	4.35E-16	**0.00E+00**	2.61E-16	1.91E+03	1.51E-09	**0.00E+00**
	SD	1.17E-16	3.89E+03	2.28E+03	1.49E-16	0.00E+00	6.88E-17	7.23E+02	2.35E-09	0.00E+00
	worse	9.92E-16	1.10E+04	8.94E+03	6.26E-16	0.00E+00	3.28E-16	3.32E+03	7.82E-09	0.00E+00
	best	5.50E-16	1.54E+03	1.08E+03	1.31E-16	0.00E+00	9.36E-17	8.06E+02	1.95E-10	0.00E+00
f_9	Avg	**1.02E-03**	8.91E+04	9.06E+03	2.22E-02	6.67E-01	2.07E-02	1.57E+04	4.97E-02	4.47E-02
	SD	7.00E-04	6.14E+04	6.60E+03	2.38E-02	4.20E-07	2.15E-02	1.53E+04	5.06E-02	5.59E-02
	worse	2.44E-03	2.27E+05	2.02E+04	8.41E-02	6.67E-01	8.04E-02	4.05E+04	1.79E-01	1.91E-01
	best	2.79E-04	2.55E+04	4.75E+01	2.45E-03	6.67E-01	8.29E-03	3.79E+03	1.18E-02	5.27E-03
f_{10}	Avg	7.90E-16	2.84E+05	1.28E+05	4.99E-16	1.26E-95	2.95E-16	1.51E+05	3.54E-08	**0.00E+00**
	SD	1.11E-16	1.74E+05	1.01E+05	1.19E-16	3.99E-95	8.65E-17	3.94E+04	3.24E-08	0.00E+00
	worse	9.81E-16	5.34E+05	3.90E+05	7.36E-16	1.26E-94	4.46E-16	2.26E+05	9.66E-08	0.00E+00
	best	6.64E-16	8.35E+04	5.19E+04	3.11E-16	0.00E+00	1.58E-16	9.90E+04	1.57E-09	0.00E+00

Table 4.5 shows the results of tests of seven proposed algorithms and the two original algorithms for 30 dimensional benchmark functions and Table 4.6 and Table 4.7 show the results in 50 and 70 dimensions respectively. As noted in those tables, all algorithms struggled to reach the optimal point as the problem dimension increased from 10 to 70. It is also noted that the proposed algorithms were able to deal with the problems effectively in multimodal problems $(f_2, f_4, f_5, f_6, f_7)$. In unimodal benchmark problems $(f_1, f_3, f_5, f_6, f_7)$, the proposed algorithms CEABC,CLEABC and HABCSDA were able to achieve more accurate results, but the ABC outperformed other algorithms in unimodal function f_3, in 30,70 dimensional problems and f_3, f_9 in 50 dimensional dimension problems. In case of the hybrid algorithms, it is clearly noted that HABCSDA performed better than others. HABCSDA was able to handle the high dimensional problems effectively as shown in

Table 4.7. Adaptation of scout bee into SDA has enhanced the search capability of SBSDA to ensure it does not get trapped at local optima.

The statistical performance measurements show that the CSDA and SBSDA were able to avoid local optimum problems and outperform the original SDA. Adaptive ABC based on the modified step size outperformed the original ABC in most of the problems. The hybrid version HABCSDA was able to converge near to the optimal effectively in multimodal problems and had high accuracy in dealing with unimodal problems.

Figure 4.2 and Figure 4.3 show the convergence of algorithms in 10 and 50 dimensional problems. These convergence plots are based on 30 independent runs. As noted in Figure 4.2 ABC, CLABC, CEABC, SBA, CLEABC, HABCSDA converged faster in the first 1000 iterations. It is also noted that CLEABC and CEABC jumped straight to the minimum optimal point for all functions in less than 100 iteration. HABCSDA performed very well in all functions for 10 dimensional problems. The HABCSDA converged slowly in early stage of iteration, but in the exploitation stage, HABCSDA reached the optimal point faster than others. Although CSDA and SBSDA are developed to resolve the problem of SDA getting trapped at local optima, they only improved slightly better than SDA.

Most of the proposed algorithms performed well in low dimensional problems. To further evaluate the performance of the algorithms they were tested in 70 dimensional problems as shown in Figure 4.3. As noted, HABCSDA converged to the optimal point in most of the benchmark functions except in function f_8. The exponential trajectory helped CEABC and CLEABC to converge faster to the minimum point except in functions f_1, f_8. In function f_1, only HABCSDA was able to go further toward the optimal point. Consistent trapping at local optimum happened for SDA and SBSDA. It appears that the scout bee was not able to lead SDA to solve the given problems efficiently.

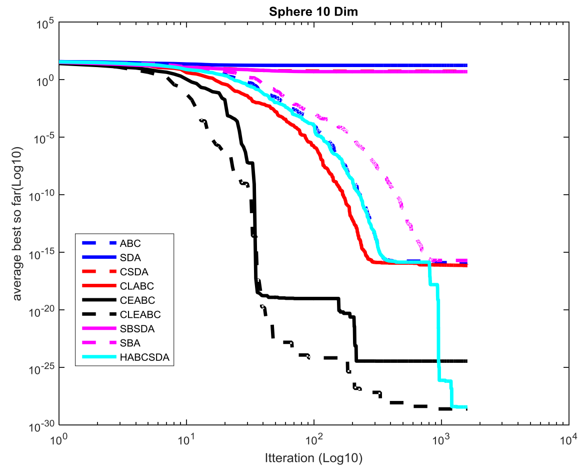
The statistical and convergence analyses show that the proposed algorithms outperformed the original algorithms. Combination of SDA and ABC has enable HABCSDA to converge to the optimal point faster than other algorithms with more accurate results. The proposed ABC variants have performed well in all problems.

Table 4.6: -Results of benchmark function with 50 dimension

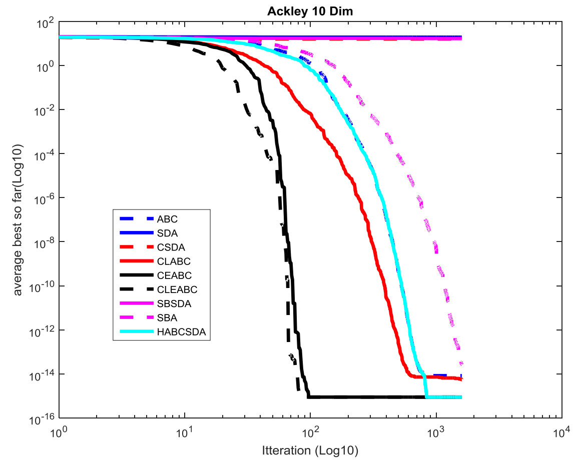
		ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
f_1	Avg	4.40E-13	2.91E+02	1.11E+02	1.23E-15	3.37E-21	2.92E-20	1.24E+02	2.14E-07	**5.15E-267**
	SD	4.83E-13	8.72E+01	2.06E+01	2.63E-16	1.07E-20	8.32E-20	6.12E+01	9.78E-08	0.00E+00
	worse	1.61E-12	3.63E+02	1.40E+02	1.62E-15	3.37E-20	2.65E-19	2.86E+02	4.13E-07	5.15E-266
	best	6.47E-14	1.04E+02	8.15E+01	8.07E-16	0.00E+00	6.38E-43	7.02E+01	1.00E-07	0.00E+00
f_2	Avg	2.66E-05	2.02E+01	1.98E+01	3.23E-08	**8.88E-16**	2.31E-15	1.98E+01	1.78E-02	**8.88E-16**
	SD	1.41E-05	4.07E-01	3.90E-01	2.51E-08	0.00E+00	1.83E-15	4.47E-01	9.09E-03	0.00E+00
	worse	5.61E-05	2.06E+01	2.03E+01	9.17E-08	8.88E-16	4.44E-15	2.04E+01	3.91E-02	8.88E-16
	best	1.06E-05	1.94E+01	1.92E+01	6.47E-09	8.88E-16	8.88E-16	1.92E+01	7.09E-03	8.88E-16
f_3	Avg	**7.69E-01**	3.46E+05	1.79E+05	3.15E+01	4.70E+01	4.77E+01	2.07E+05	1.78E+01	2.78E+01
	SD	5.15E-01	1.80E+05	1.27E+05	2.53E+01	6.80E-01	3.35E-01	1.73E+05	2.59E+01	1.92E+01
	worse	1.61E+00	7.38E+05	4.65E+05	7.52E+01	4.78E+01	4.82E+01	6.48E+05	8.07E+01	4.88E+01
	best	3.76E-02	1.51E+05	6.33E+04	1.00E+00	4.59E+01	4.70E+01	5.25E+04	6.58E-01	3.68E+00
f_4	Avg	1.80E-05	8.28E+02	3.78E+02	1.86E-12	**0.00E+00**	**0.00E+00**	3.97E+02	8.61E-03	**0.00E+00**
	SD	5.38E-05	3.86E+02	8.91E+01	5.71E-12	0.00E+00	0.00E+00	9.50E+01	1.21E-02	0.00E+00
	worse	1.79E-04	1.23E+03	4.74E+02	1.81E-11	0.00E+00	0.00E+00	5.52E+02	3.32E-02	0.00E+00
	best	1.26E-10	2.95E+02	1.79E+02	5.55E-16	0.00E+00	0.00E+00	2.72E+02	1.00E-04	0.00E+00
f_5	Avg	4.98E-01	6.56E+02	4.06E+02	6.55E-12	**0.00E+00**	**0.00E+00**	4.36E+02	1.72E+00	**0.00E+00**
	SD	4.98E-01	1.51E+02	5.53E+01	3.49E-12	0.00E+00	0.00E+00	5.63E+01	1.44E+00	0.00E+00
	worse	1.00E+00	7.84E+02	4.95E+02	1.40E-11	0.00E+00	0.00E+00	5.33E+02	5.03E+00	0.00E+00
	best	7.40E-08	4.40E+02	3.07E+02	2.73E-12	0.00E+00	0.00E+00	3.62E+02	2.64E-03	0.00E+00
f_6	Avg	5.37E+02	1.74E+04	1.06E+04	3.19E+02	8.51E+02	3.92E+03	1.17E+04	**-8.39E+274**	7.92E+02
	SD	2.75E+02	2.65E+03	4.59E+03	1.45E+02	3.03E+02	1.36E+02	4.43E+03	-8.39E+274	3.90E+02
	worse	9.01E+02	1.94E+04	1.75E+04	5.21E+02	1.42E+03	4.22E+03	1.72E+04	-4.65E+05	1.36E+03
	best	1.18E+02	1.03E+04	6.50E+03	1.18E+02	3.62E+02	3.78E+03	6.60E+03	-8.39E+275	1.21E+02
f_7	Avg	**1.89E-12**	6.05E+01	5.21E+01	7.85E-08	7.44E-07	1.17E-07	5.78E+01	8.13E-07	5.07E-06
	SD	1.17E-12	8.78E+00	1.21E+01	7.32E-08	4.91E-07	1.91E-07	1.03E+01	4.94E-07	7.45E-06
	worse	4.29E-12	7.42E+01	7.62E+01	2.41E-07	1.40E-06	6.57E-07	7.93E+01	1.63E-06	2.55E-05
	best	4.69E-13	4.47E+01	2.97E+01	1.22E-08	1.48E-07	1.55E-08	4.15E+01	2.55E-07	6.66E-07
f_8	Avg	3.50E-11	2.86E+04	1.66E+04	9.69E-16	1.82E-34	8.09E-16	1.01E+04	7.42E-06	**0.00E+00**
	SD	2.50E-11	7.93E+03	1.21E+04	2.14E-16	5.75E-34	2.29E-16	4.79E+03	5.10E-06	0.00E+00
	worse	9.61E-11	3.46E+04	3.26E+04	1.21E-15	1.82E-33	1.15E-15	1.87E+04	1.54E-05	0.00E+00
	best	7.49E-12	7.27E+03	6.41E+03	6.38E-16	0.00E+00	4.73E-16	4.63E+03	9.80E-07	0.00E+00
f_9	Avg	**6.65E-02**	6.34E+05	1.43E+05	9.01E-01	6.67E-01	5.05E-01	2.31E+05	1.52E+00	6.13E-01
	SD	5.07E-02	3.47E+05	5.40E+04	5.32E-01	2.53E-07	1.82E-01	1.40E+05	1.44E+00	1.48E-01
	worse	1.92E-01	1.19E+06	2.07E+05	2.12E+00	6.67E-01	6.67E-01	5.78E+05	4.62E+00	6.75E-01
	best	1.95E-02	1.26E+05	6.88E+04	3.59E-02	6.67E-01	2.26E-01	7.26E+04	3.16E-01	2.00E-01
f_{10}	Avg	7.58E-10	1.11E+06	4.90E+05	1.51E-15	1.73E-44	9.91E-16	6.38E+05	2.82E-04	**0.00E+00**
	SD	8.68E-10	3.63E+05	1.01E+05	2.42E-16	5.48E-44	5.98E-16	2.94E+05	3.35E-04	0.00E+00
	worse	2.46E-09	1.52E+06	6.40E+05	1.76E-15	1.73E-43	2.52E-15	1.16E+06	1.06E-03	0.00E+00
	best	7.27E-11	4.77E+05	3.67E+05	9.73E-16	0.00E+00	3.98E-16	2.62E+05	9.77E-06	0.00E+00

Table 4.7: -Results of benchmark function with 70 dimension

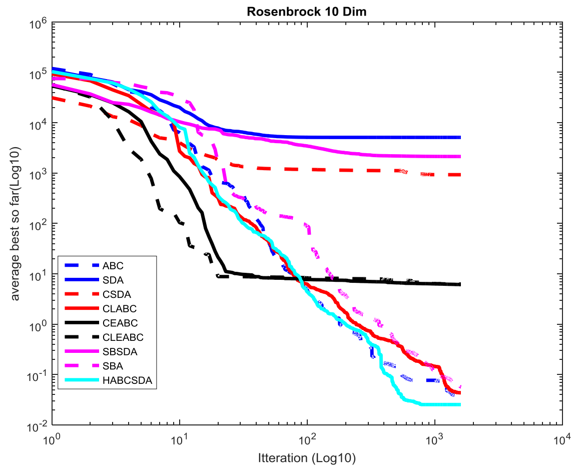
		ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
f_1	Avg	8.54E-09	3.76E+02	1.76E+02	1.79E-12	5.52E-19	1.04E-19	2.46E+02	1.49E-05	**5.08E-229**
	SD	4.76E-09	1.57E+02	3.00E+01	1.59E-12	1.36E-18	2.36E-19	9.95E+01	8.79E-06	0.00E+00
	worse	1.66E-08	5.50E+02	2.29E+02	5.82E-12	4.39E-18	7.65E-19	5.20E+02	2.84E-05	5.08E-228
	best	3.78E-09	1.92E+02	1.30E+02	5.54E-13	5.76E-98	3.19E-37	1.87E+02	3.47E-06	0.00E+00
f_2	Avg	3.96E-03	2.04E+01	2.01E+01	6.75E-06	4.09E-15	6.57E-15	2.00E+01	2.43E-01	**3.73E-15**
	SD	1.72E-03	2.67E-01	4.29E-01	2.57E-06	2.62E-15	1.83E-15	3.32E-01	2.95E-01	2.25E-15
	worse	6.60E-03	2.07E+01	2.08E+01	1.16E-05	7.99E-15	7.99E-15	2.04E+01	1.06E+00	7.99E-15
	best	1.12E-03	1.98E+01	1.96E+01	3.60E-06	8.88E-16	4.44E-15	1.95E+01	4.92E-02	8.88E-16
f_3	Avg	**3.06E+00**	7.65E+05	4.10E+05	6.42E+01	6.67E+01	6.75E+01	4.17E+05	6.20E+01	4.39E+01
	SD	3.12E+00	3.07E+05	1.87E+05	4.11E+01	1.07E+00	6.20E-01	1.50E+05	3.31E+01	2.66E+01
	worse	9.64E+00	1.22E+06	8.16E+05	1.17E+02	6.78E+01	6.84E+01	7.29E+05	1.00E+02	6.87E+01
	best	2.12E-07	3.82E+05	2.13E+05	1.98E+00	6.47E+01	6.68E+01	2.69E+05	7.06E+00	9.83E+00
f_4	Avg	2.88E-05	1.42E+03	7.33E+02	5.57E-07	**0.00E+00**	**0.00E+00**	6.65E+02	3.16E-02	**0.00E+00**
	SD	3.57E-05	4.60E+02	3.19E+02	1.76E-06	0.00E+00	0.00E+00	1.72E+02	2.70E-02	0.00E+00
	worse	1.15E-04	1.77E+03	1.62E+03	5.57E-06	0.00E+00	0.00E+00	1.09E+03	7.06E-02	0.00E+00
	best	9.80E-07	7.14E+02	5.29E+02	3.23E-11	0.00E+00	0.00E+00	4.10E+02	6.46E-04	0.00E+00
f_5	Avg	6.28E+00	9.50E+02	7.54E+02	7.47E-08	**0.00E+00**	**0.00E+00**	6.69E+02	1.00E+01	**0.00E+00**
	SD	2.54E+00	2.19E+02	1.98E+02	4.91E-08	0.00E+00	0.00E+00	7.89E+01	4.65E+00	0.00E+00
	worse	1.05E+01	1.15E+03	1.10E+03	1.57E-07	0.00E+00	0.00E+00	8.05E+02	1.90E+01	0.00E+00
	best	2.07E+00	6.46E+02	5.81E+02	2.09E-08	0.00E+00	0.00E+00	5.44E+02	4.86E+00	0.00E+00
f_6	Avg	1.66E+03	2.01E+04	1.14E+04	1.23E+03	2.19E+03	7.34E+03	9.55E+03	**-9.42E+196**	2.22E+03
	SD	2.83E+02	6.29E+03	4.50E+03	3.13E+02	4.73E+02	1.92E+02	5.45E+02	-9.42E+196	3.06E+02
	worse	2.10E+03	2.68E+04	2.39E+04	1.88E+03	2.83E+03	7.63E+03	1.07E+04	-1.79E+05	2.62E+03
	best	1.07E+03	1.32E+04	8.32E+03	7.20E+02	1.34E+03	7.10E+03	8.73E+03	-9.42E+197	1.68E+03
f_7	Avg	**2.98E-08**	1.23E+02	9.40E+01	2.61E-05	1.01E-04	1.11E-05	8.80E+01	4.89E-05	6.40E-04
	SD	4.28E-08	3.53E+01	1.95E+01	2.44E-05	6.50E-05	8.53E-06	7.91E+00	8.62E-05	1.17E-03
	worse	1.55E-07	2.13E+02	1.33E+02	6.93E-05	2.14E-04	2.53E-05	9.87E+01	2.92E-04	3.88E-03
	best	3.63E-09	7.66E+01	6.67E+01	4.01E-06	2.31E-05	1.70E-06	7.52E+01	2.94E-06	2.52E-05
f_8	Avg	1.05E-07	4.75E+04	2.66E+04	1.56E-11	5.13E-19	5.56E-13	2.09E+04	4.01E-04	**1.54E-57**
	SD	6.05E-08	2.19E+04	1.86E+04	1.33E-11	1.22E-18	7.05E-13	9.16E+03	2.97E-04	4.86E-57
	worse	2.43E-07	6.88E+04	6.34E+04	4.65E-11	3.77E-18	2.06E-12	3.83E+04	8.71E-04	1.54E-56
	best	2.47E-08	1.95E+04	1.23E+04	4.67E-12	0.00E+00	2.63E-14	1.19E+04	4.51E-05	0.00E+00
f_9	Avg	2.38E+00	1.62E+06	8.98E+05	5.11E+00	**6.67E-01**	6.99E-01	1.21E+06	9.52E+00	7.10E-01
	SD	2.52E+00	6.94E+05	3.35E+05	3.04E+00	1.02E-05	1.02E-01	3.08E+05	5.80E+00	8.97E-02
	worse	7.08E+00	3.15E+06	1.48E+06	9.82E+00	6.67E-01	9.89E-01	1.67E+06	1.74E+01	9.61E-01
	best	1.48E-01	9.59E+05	5.59E+05	1.34E+00	6.67E-01	6.67E-01	5.91E+05	1.36E+00	6.67E-01
f_{10}	Avg	6.52E-06	2.28E+06	1.40E+06	4.50E-10	3.21E-19	2.85E-11	1.31E+06	2.64E-02	**8.17E-98**
	SD	1.39E-05	6.95E+05	6.60E+05	3.16E-10	1.02E-18	2.07E-11	4.20E+05	2.71E-02	2.58E-97
	worse	4.52E-05	2.94E+06	2.72E+06	9.53E-10	3.21E-18	7.58E-11	2.05E+06	8.36E-02	8.17E-97
	best	1.14E-07	1.05E+06	8.43E+05	5.99E-11	0.00E+00	3.66E-12	8.18E+05	2.49E-03	0.00E+00



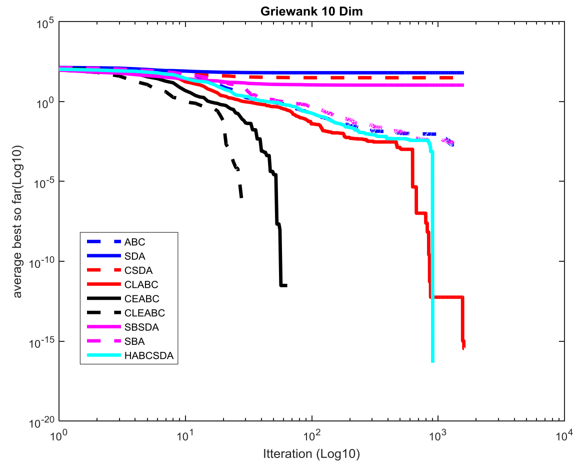
(a)



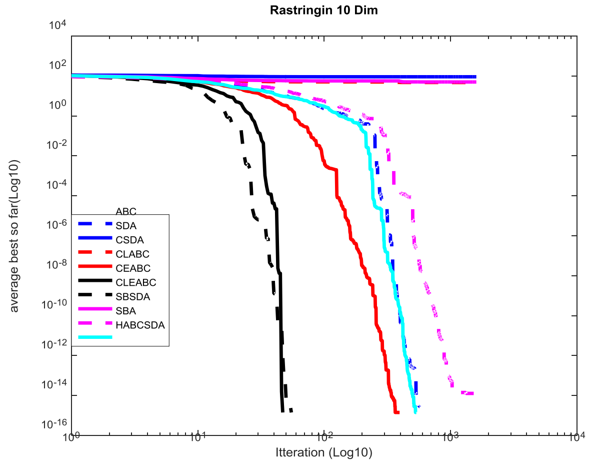
(b)



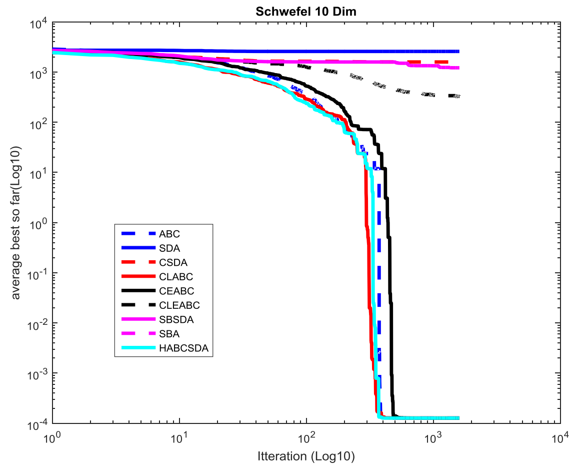
(c)



(d)

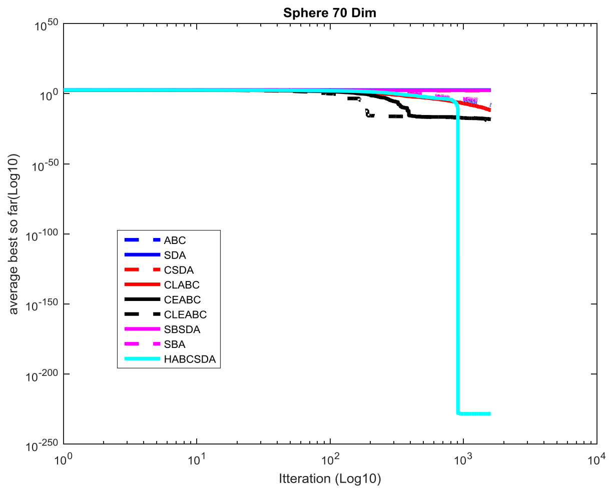


(e)

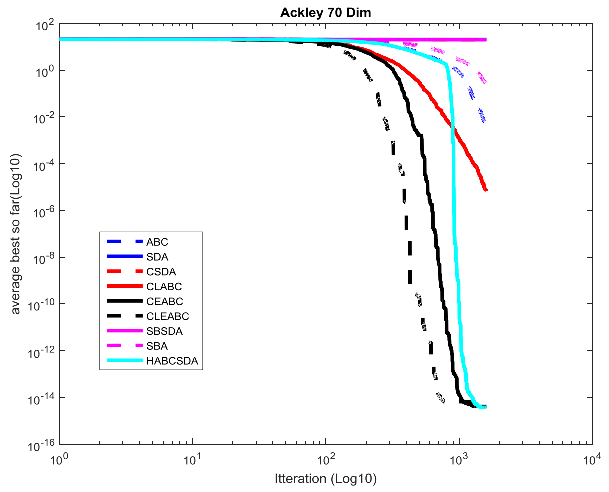


(f)

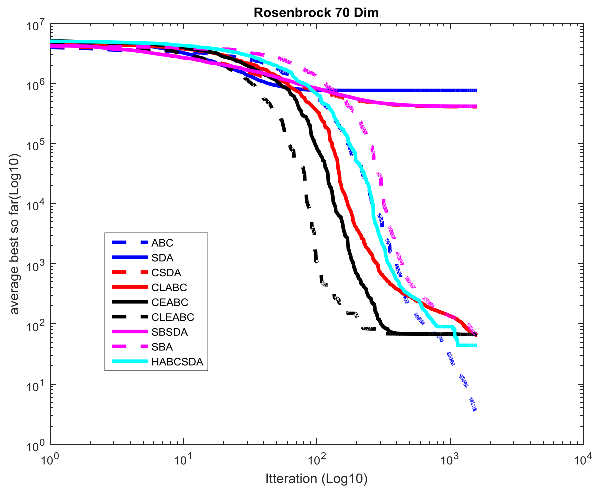
Figure 4.2:- Convergence plot with 10 dimensional standard benchmark functions , (a)Sphere, (b)Akcley, (c)Rosenbrock, (d)Griewank, (e)Rastringin, (f)Schwefel.



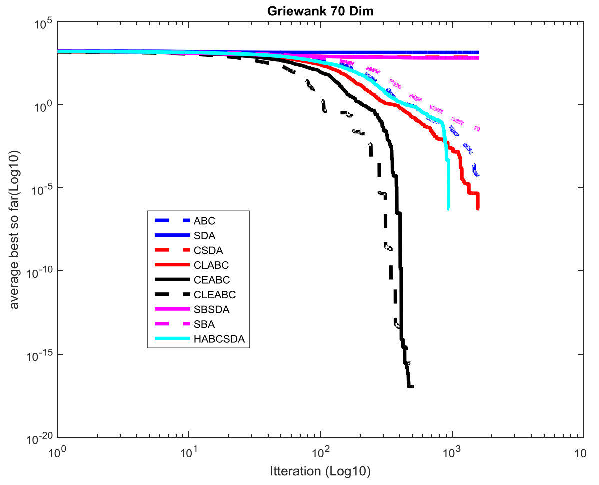
(a)



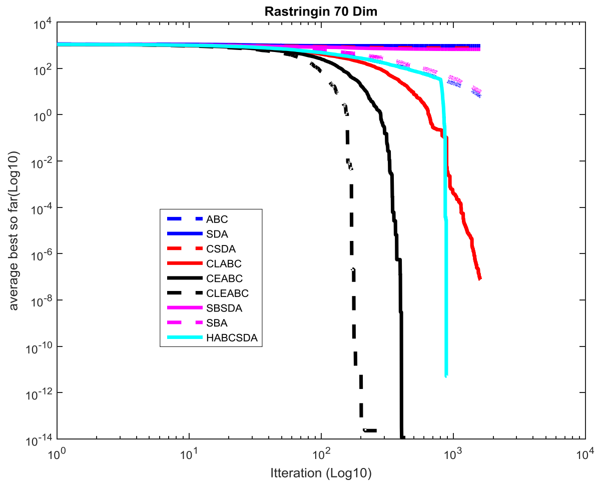
(b)



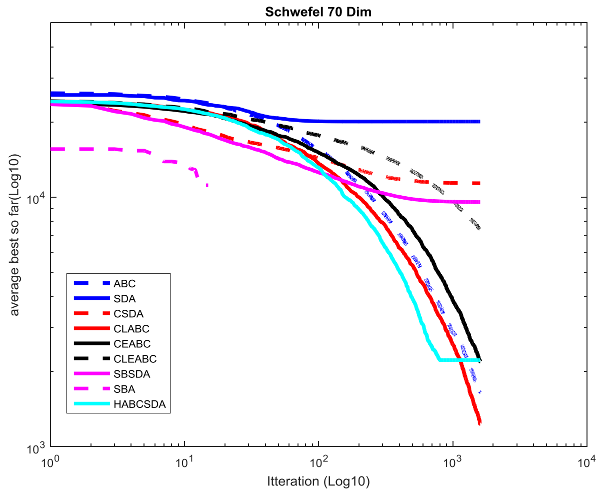
(c)



(d)



(e)



(f)

Figure 4.3:- Convergence plot with 70 dimensional standard benchmark functions , (a)Sphere, (b)Akcley, (c)Rosenbrock, (d)Griewank, (e)Rastringin, (f)Schwefel.

4.2.2.1 Reliability test

Table 4.8 and Table 4.9 show results of success rates (SRs) of algorithms for all benchmark functions with associated dimensions. As noted, SDA, CSDA and SBSDA were not able to reach the threshold marker 10e-4 mainly due to the spiral trajectory not been efficient enough to ensure these algorithms to go further to the optimal point. It is further noted the algorithms failed to score the success rate in Rosenbrock benchmark function. The algorithms did not perform well in 50 and 70 dimensional Dixon Price benchmark function. This failure is due to the complexity of the landscape and as the approach was not able to lead the algorithm to threshold marker 10e-4 in this kind of problems.

Table 4.8:- Results of success rate (SR)

F	**D**	**ABC**		**SDA**		**CSDA**		**CLABC**		**CEABC**	
		SR	**NFE**	**SR**	**NFE**	**SR**	**NFE**	**SR**	**NFE**	**SR**	**NFE**
f_1	10	1	2400	0	-	0	-	1	1560	1	630
	30	1	8280	0	-	0	-	1	5910	1	2670
	50	1	13560	0	-	0	-	1	10350	1	5040
	70	1	20460	0	-	0	-	1	15900	1	7740
f_2	10	1	6870	0	-	0	-	1	3810	1	1590
	30	1	21120	0	-	0	-	1	12540	1	5550
	50	1	37380	0	-	0	-	1	21750	1	10230
	70	0	-	0	-	0	-	1	30060	1	15750
f_3	10	0	-	0	-	0	-	0	-	0	-
	30	0	-	0	-	0	-	0	-	0	-
	50	0	-	0	-	0	-	0	-	0	-
	70	0	-	0	-	0	-	0	-	0	-
f_4	10	0	-	0	-	0	-	1	19020	1	1200
	30	1	31140	0	-	0	-	1	17070	1	4950
	50	1	34230	0	-	0	-	1	17940	1	7170
	70	1	42000	0	-	0	-	1	33600	1	10380
f_5	10	1	8100	0	-	0	-	1	3750	1	1020
	30	1	36060	0	-	0	-	1	11340	1	3570
	50	0	-	0	-	0	-	1	19290	1	6660
	70	0	-	0	-	0	-	1	28410	1	10350
f_6	10	1	11310	0	-	0	-	1	10380	1	14010
	30	0	-	0	-	0	-	1	42960	0	-
	50	0	-	0	-	0	-	0	-	0	-
	70	0	-	0	-	0	-	0	-	0	-
f_7	10	1	2730	0	-	0	-	1	3180	1	3300
	30	1	9660	0	-	0	-	1	12180	1	15720
	50	1	17310	0	-	0	-	1	22530	1	29190
	70	1	25830	0	-	0	-	1	29760	1	40740
f_8	10	1	2820	0	-	0	-	1	2010	1	1200
	30	1	9630	0	-	0	-	1	7530	1	5310
	50	1	15960	0	-	0	-	1	12660	1	9180
	70	1	23700	0	-	0	-	1	18660	1	14430
f_9	10	1	13650	0	-	0	-	1	17670	1	-
	30	1	47970	0	-	0	-	0	-	0	-
	50	0	-	0	-	0	-	0	-	0	-
	70	0	-	0	-	0	-	0	-	0	-
f_{10}	10	1	2910	0	-	0	-	1	2670	1	1530
	30	1	10890	0	-	0	-	1	9000	1	6810
	50	1	20700	0	-	0	-	1	16800	1	12690
	70	1	28590	0	-	0	-	1	23880	1	18960

Adaptive ABC, CLEABC were able to meet the success rate criterion in smallest number of fitness evaluations (NFEs) in $f_1, f_2, f_4, f_5, f_8, f_{10}$ and this implies that CLEABC was the fastest algorithm to converge and hit the 10e-4 marker. The HABCSDA was the fastest algorithm to converge in f_7, f_9. As a summary, 3 adaptive ABC algorithms achieved 100% success rate in most of the benchmark functions where the combination of exponential and linear trajectory helped CLEABC to be the faster algorithm to converge to 10e-4 point with 100% success rate, while SDA, and SDA based proposed algorithm failed in this test.

Table 4.9:- Results of success rate (SR) (cont.)

F	D	CLEABC		SBSDA		SBA		HABCSDA	
		SR	NFE	SR	NFE	SR	NFE	SR	NFE
f_1	10	1	**360**	0	-	1	3060	1	2070
	30	1	**1500**	0	-	1	10260	1	8010
	50	1	**2550**	0	-	1	18180	1	13980
	70	1	**3780**	0	-	1	25650	1	17820
f_2	10	1	**1170**	0	-	1	11910	1	6630
	30	1	**4500**	0	-	1	38010	1	21360
	50	1	**6690**	0	-	0	-	1	25440
	70	1	**9480**	0	-	0	-	1	26910
f_3	10	0	0	0	-	0	-	0	-
	30	0	0	0	-	0	-	0	-
	50	0	0	0	-	0	-	0	-
	70	0	0	0	-	0	-	0	-
f_4	10	1	**660**	0	-	0	-	1	26100
	30	1	**3750**	0	-	1	47970	1	24060
	50	1	**6720**	0	-	0	-	1	25200
	70	1	**7290**	0	-	0	-	1	27150
f_5	10	1	**780**	0	-	1	10830	1	7170
	30	1	**2280**	0	-	1	36060	1	24270
	50	1	**3780**	0	-	0	-	1	25230
	70	1	**5070**	0	-	0	-	1	25950
f_6	10	0	-	0	-	0	-	1	10560
	30	0	-	0	-	0	-	0	-
	50	0	-	0	-	0	-	0	-
	70	0	-	0	-	1	**450**	0	-
f_7	10	1	4050	0	-	1	4020	1	**2490**
	30	1	12090	0	-	0	-	1	**9030**
	50	1	21210	0	-	1	20130	1	**16440**
	70	1	28050	0	-	1	27750	1	**22740**
f_8	10	1	**750**	0	-	1	3510	1	2580
	30	1	**2940**	0	-	1	13950	1	8670
	50	1	**5670**	0	-	1	30690	1	16380
	70	1	**8610**	0	-	1	45870	1	22560
f_9	10	1	18810	0	-	1	23700	1	**12210**
	30	0	-	0	-	0	-	0	-
	50	0	-	0	-	0	-	0	-
	70	0	-	0	-	0	-	0	-
f_{10}	10	1	**960**	0	-	1	5370	1	3240
	30	1	**3480**	0	-	1	25050	1	11310
	50	1	**5940**	0	-	1	42450	1	19320
	70	1	**9450**	0	-	0	-	1	24570

4.2.2.2 Parametric test:- Statistical significant test

In this significant test, Kruskal-Wallis test is used. The results are based on 30 runs and 95% confidence interval. All the data consists of average rank and the p-value is recorded. There will be a significant improvement for an algorithm if the p-value is less than 0.05. Table 4.10 shows the outcome from the significant test using Kruskal-Wallis method for the algorithms with dimensions 10, 30, 50 and 70. As noted, all algorithms in all tested conditions have shown` significant difference among them. To see which algorithm is most significant than others, the result are ranked from smallest to largest. The smallest rank indicates the algorithm has the most significant difference than others. While the largest rank means that the algorithm is likely to have less significant difference from others. It is clear from the results that HABCSDA had the most significant difference than others with average mean rank of 44.69 and rank 1.18. Three adaptive ABC (CEABC, CLABC&CLEABC) algorithms had 2nd,3rd and 4th lowest rank. Thus, these three adaptive ABC algorithms were significantly different than original ABC. Although CSDA and SBSDA had 2nd and 3rd highest rank respectively, CSDA and SBSDA still showed significant difference than original ABC. It can be concluded that the modified version of ABC and HABCSDA performed better and showed significant improvement as compared with the original algorithms and other proposed algorithms

Table 4.10 : Result of significant test for standard benchmark functions.

F	D	ABC	SDA	CSDA	CLABC	CEABC	CLABC	SBSDA	SBA	HABCSDA	p-value
f_1	10	133.10	236.30	221.90	112.40	53.00	43.40	218.30	161.00	40.10	1.03E-48
	30	135.80	253.40	187.40	111.20	39.50	69.80	217.70	168.50	36.20	5.26E-47
	50	141.50	252.20	173.60	111.50	48.05	77.90	215.30	171.50	27.95	2.32E-44
	70	137.50	244.90	204.70	107.50	63.40	61.60	232.90	167.50	16.60	3.20E-50
f_2	10	137.45	246.50	169.70	115.55	51.50	51.50	224.30	171.50	51.50	1.45E-44
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	5.32E-67
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	4.90E-51
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	3.86E-49
f_3	10	56.90	227.90	195.50	69.20	172.10	180.80	201.20	74.60	41.30	2.29E-41
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	1.87E-45
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	6.05E-49
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	3.56E-41
f_4	10	142.10	235.10	235.40	71.60	57.50	57.50	206.00	156.80	57.50	2.61E-50
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	1.13E-51
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	1.29E-51
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	9.84E-52
f_5	10	83.00	246.80	211.10	83.00	83.00	83.00	218.60	128.00	83.00	5.40E-49
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	3.08E-50
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	5.61E-52
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	7.16E-51
f_6	10	86.00	245.90	217.10	104.00	86.00	165.80	213.20	15.50	86.00	2.43E-49
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	5.40E-44
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	3.42E-49
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	1.78E-49
f_7	10	48.20	230.30	227.00	82.10	81.50	114.20	219.20	128.00	89.00	1.08E-38
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	3.87E-49
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	2.28E-49
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	1.06E-46
f_8	10	137.60	228.20	222.20	107.00	32.00	77.90	226.10	159.50	29.00	2.77E-50
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	2.55E-51
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	3.36E-51
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	1.47E-50
f_9	10	15.50	203.00	205.40	63.80	180.50	94.40	230.90	126.50	99.50	1.49E-41
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	3.78E-48
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	3.65E-45
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	1.51E-46
f_{10}	10	132.80	238.40	219.50	106.70	30.50	79.10	218.60	163.40	30.50	1.07E-50
	30	135.34	244.28	220.28	105.45	45.50	45.50	214.59	165.66	45.50	2.75E-51
	50	135.50	242.00	218.00	105.50	39.50	57.50	216.50	165.50	39.50	1.43E-51
	70	137.50	243.70	224.50	107.50	40.75	64.60	214.30	167.50	37.15	2.47E-51
Average		126.56	241.47	216.39	102.79	52.76	66.63	216.27	157.00	44.69	
Average (rank)		4.55	8.93	7.83	3.85	1.68	2.55	7.23	5.83	1.18	

4.2.3 CEC2014 benchmark functions tests

In this section, the performances of the algorithms are tested in 16 CEC2014 benchmark problems with dimensions 2, 10, 30 and 50. Each algorithm was run 30 times and the results were recorded. The error, average, standard deviation, worse and best results of the algorithms are presented in Table 4.11 - 4.12 with dimensions 2, 10, 30 and 50. The smallest error obtained is indicated with bold font.

Table 4.11 shows the statistical results for 2 and 10 dimensional problems. It is noted that, for 2 dimensional problems, most of the proposed algorithms were able to reach near to the optimal point except the CSDA and SBSDA. The function $\boldsymbol{f_{11}}$ is unimodal problem, but as noted all algorithms struggled to reach the optimal point. The deviation from the average value was also big. In function $\boldsymbol{f_{16}}$, most of the algorithms except SDA had zero error which means that those algorithms reached the optimal point and had very high accuracy with small or no-standard deviation. Although SDA did not perform well in this function, the SDA standard deviation was small and still considered good. The adaptive ABC performed well in all multi-modal functions $\boldsymbol{f_{14}, f_{15}, f_{16}, f_{18}, f_{19}, f_{20}}$. On the other hand, HABCSDA outperformed other algorithms in all functions except in functions $\boldsymbol{f_{11}, f_{13}, f_{22}, f_{23}}$. While for 10 dimensional problems, as noted in Table 4.11, HABCSDA performed well with lowest error in all functions except in functions $\boldsymbol{f_{16}, f_{22}, f_{24}}$. Similar in 2 dimensional problems, the SDA and variants of SDA struggled to reach the optimal point with small error. On the other hand, the adaptive ABC performed well with small error.

Table 4.12 show the statistical results for 30 and 50 dimensions problem. When the dimension is increased, once against the hybrid algorithm, HABCSDA proved it able to handle high dimensional problems. The nearest competitor of HABCSDA was the original ABC**.**

Table 4.11: Numerical Results of benchmark function with 2 and 10 dimension

F	D	Stat	ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
f_{11}	2	Error	1.86E+02	2.74E+05	7.48E+02	3.17E+02	3.61E+02	2.26E+02	6.26E+02	**3.92E+01**	4.09E+01
		SD	1.75E+02	7.23E+05	1.01E+03	2.66E+02	2.86E+02	1.65E+02	9.26E+02	4.20E+01	3.85E+01
	10	Error	2.74E+05	2.77E+08	2.19E+08	2.62E+05	3.51E+05	3.38E+05	1.75E+08	1.77E+05	**1.51E+05**
		SD	2.13E+05	2.78E+08	2.13E+08	1.91E+05	3.96E+05	4.09E+05	1.65E+08	1.14E+05	5.97E+04
f_{12}	2	Error	1.61E+01	6.24E+07	5.62E+07	1.93E+01	1.28E+01	8.39E+00	5.82E+06	8.42E+01	**7.44E+00**
		SD	3.02E+01	1.54E+08	1.11E+08	2.41E+01	9.36E+00	7.07E+00	1.08E+07	6.23E+01	7.29E+00
	10	Error	2.54E+02	1.70E+10	1.64E+10	3.30E+02	1.19E+02	2.73E+02	2.02E+10	3.24E+02	**9.37E+01**
		SD	1.94E+02	3.48E+09	6.47E+09	3.85E+02	1.23E+02	2.96E+02	3.81E+09	3.28E+02	1.40E+02
f_{13}	2	Error	1.65E+01	6.05E+05	1.45E+06	7.83E+01	1.58E+01	**6.75E+00**	7.01E+03	3.81E+01	7.48E+00
		SD	2.08E+01	1.18E+06	3.13E+06	8.11E+01	2.12E+01	8.30E+00	7.80E+03	3.10E+01	8.27E+00
	10	Error	3.99E+02	1.24E+07	4.17E+06	4.35E+02	4.98E+02	4.37E+02	1.63E+06	2.73E+02	**2.72E+02**
		SD	3.90E+02	1.28E+07	7.74E+06	3.09E+02	2.20E+02	2.09E+02	2.54E+06	1.85E+02	1.82E+02
f_{14}	2	Error	**0.00E+00**	7.56E+00	3.19E+00	2.22E-07	**0.00E+00**	**0.00E+00**	2.28E+00	**0.00E+00**	**0.00E+00**
		SD	0.00E+00	1.57E+01	3.81E+00	3.53E-07	0.00E+00	3.28E-14	2.50E+00	2.68E-14	0.00E+00
	10	Error	2.17E-01	4.16E+03	3.67E+03	1.29E+00	2.83E-01	6.07E-01	3.75E+03	1.24E+00	**5.89E-02**
		SD	1.50E-01	1.36E+03	1.58E+03	1.87E+00	2.15E-01	4.49E-01	1.68E+03	9.31E-01	3.04E-02
f_{15}	2	Error	**0.00E+00**	1.73E+01	1.65E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.89E+01	**0.00E+00**	**0.00E+00**
		SD	0.00E+00	3.79E+00	3.71E+00	1.89E-14	0.00E+00	0.00E+00	2.93E+00	2.68E-14	0.00E+00
	10	Error	1.84E+01	2.02E+01	2.02E+01	1.70E+01	**1.69E+01**	1.83E+01	2.01E+01	2.01E+01	2.00E+01
		SD	4.97E+00	2.16E-01	2.77E-01	4.91E+00	6.56E+00	3.87E+00	2.07E-01	1.99E-02	5.88E-02
f_{16}	2	Error	**0.00E+00**	5.89E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
		SD	0.00E+00	1.86E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	10	Error	2.09E+00	1.31E+01	1.35E+01	2.72E+00	2.38E+00	2.42E+00	1.22E+01	3.85E+00	**2.00E+00**
		SD	5.51E-01	1.57E+00	2.09E+00	6.58E-01	7.03E-01	6.02E-01	1.46E+00	4.82E-01	5.93E-01
f_{17}	2	Error	3.57E-05	8.88E-03	8.88E-03	6.62E-04	2.99E-06	1.18E-05	6.16E-03	2.67E-05	**0.00E+00**
		SD	1.12E-04	1.58E-02	8.79E-03	1.92E-03	6.31E-06	3.26E-05	3.34E-03	4.40E-05	0.00E+00
	10	Error	2.62E-02	9.07E+01	9.69E+01	4.54E-02	1.95E-02	3.52E-02	8.54E+01	2.72E-02	**9.47E-03**
		SD	1.79E-02	1.64E+01	2.28E+01	2.62E-02	1.24E-02	2.49E-02	1.26E+01	1.19E-02	1.08E-02
f_{18}	2	Error	**0.00E+00**	1.29E+00	8.95E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.95E-02	**0.00E+00**	**0.00E+00**
		SD	0.00E+00	1.49E+00	1.19E+00	0.00E+00	0.00E+00	0.00E+00	3.15E-01	0.00E+00	0.00E+00
	10	Error	4.32E-12	1.42E+02	1.51E+02	7.01E-09	2.05E-12	2.27E-12	1.21E+02	**0.00E+00**	**0.00E+00**
		SD	1.34E-11	2.37E+01	2.68E+01	1.20E-08	4.04E-12	6.64E-12	3.38E+01	1.83E-12	3.79E-14
f_{19}	2	Error	**0.00E+00**	8.43E+00	1.10E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.85E+00	**0.00E+00**	**0.00E+00**
		SD	0.00E+00	4.25E+00	3.80E+00	0.00E+00	0.00E+00	0.00E+00	1.95E+00	0.00E+00	0.00E+00
	10	Error	9.96E+00	1.28E+02	1.37E+02	1.13E+01	1.44E+01	1.14E+01	1.27E+02	1.07E+01	**7.77E+00**
		SD	1.70E+00	3.11E+01	3.12E+01	2.21E+00	2.08E+00	2.19E+00	2.33E+01	2.64E+00	2.38E+00
f_{20}	2	Error	**0.00E+00**	8.29E+01	5.19E+01	2.68E-10	**0.00E+00**	**0.00E+00**	2.89E+01	3.54E-02	**0.00E+00**
		SD	0.00E+00	7.00E+01	4.50E+01	8.45E-10	0.00E+00	1.31E-13	2.06E+01	9.81E-02	0.00E+00
	10	Error	1.63E-01	9.94E+02	9.04E+02	2.56E-01	2.07E+00	1.98E-01	1.10E+03	3.43E+01	**1.05E-01**
		SD	4.95E-02	6.75E+02	3.07E+02	9.03E-02	1.66E+00	7.88E-02	5.39E+02	6.00E+00	8.11E-02
f_{21}	2	Error	5.29E-06	1.56E+02	1.31E+02	1.31E-04	1.64E-03	3.93E-04	1.01E+02	1.10E-01	**7.85E-08**
		SD	1.62E-05	3.93E+01	6.33E+01	3.36E-04	3.22E-03	1.24E-03	4.82E+01	1.36E-01	2.48E-07
	10	Error	2.75E+02	1.94E+03	2.13E+03	3.50E+02	2.90E+02	2.88E+02	1.95E+03	5.53E+02	**2.22E+02**
		SD	8.51E+01	6.46E+02	4.08E+02	1.21E+02	9.22E+01	7.88E+01	3.53E+02	1.02E+02	6.40E+01
f_{22}	2	Error	9.56E-02	6.90E-01	1.54E-01	**5.80E-02**	6.07E-02	7.01E-02	5.41E-01	6.29E-02	7.43E-02
		SD	4.64E-02	1.55E+00	4.87E-01	2.66E-02	3.23E-02	4.11E-02	1.14E+00	3.96E-02	3.53E-02
	10	Error	2.60E-01	1.12E+00	7.97E-01	**2.54E-01**	2.75E-01	2.66E-01	1.42E+00	2.75E-01	2.91E-01
		SD	5.75E-02	1.34E+00	4.45E-01	4.36E-02	6.00E-02	6.43E-02	6.94E-01	9.57E-02	5.91E-02
f_{23}	2	Error	3.71E-03	1.08E+00	1.08E+00	9.86E-03	**6.36E-04**	6.59E-03	7.15E-01	6.59E-03	1.14E-02
		SD	1.83E-03	2.66E-01	3.01E-01	7.97E-03	9.15E-04	4.86E-03	1.40E-02	6.04E-03	7.69E-03
	10	Error	1.61E-01	6.04E+00	6.20E+00	1.75E-01	1.73E-01	1.52E-01	5.95E+00	1.37E-01	**1.17E-01**
		SD	2.48E-02	8.98E-01	1.17E+00	3.66E-02	2.25E-02	2.56E-02	1.03E+00	1.69E-02	2.24E-02
f_{24}	2	Error	1.21E-02	1.47E+00	1.45E+00	1.52E-02	1.43E-02	1.27E-02	1.86E+00	7.89E-03	**3.81E-03**
		SD	5.27E-03	8.57E-01	1.06E+00	8.21E-03	5.28E-03	7.16E-03	9.72E-01	3.26E-03	2.23E-03
	10	Error	1.97E-01	4.13E+01	2.62E+01	2.21E-01	2.03E-01	2.05E-01	3.28E+01	**1.07E-01**	1.61E-01
		SD	2.53E-02	2.53E+01	2.10E+01	3.72E-02	2.06E-02	1.67E-02	2.39E+01	2.48E-02	1.92E-02
f_{25}	2	Error	9.28E-09	2.20E+00	5.85E-01	1.05E-05	**0.00E+00**	3.36E-06	3.50E-02	**0.00E+00**	**0.00E+00**
		SD	2.68E-08	3.23E+00	1.70E+00	2.92E-05	1.52E-13	1.03E-05	5.00E-02	2.23E-12	0.00E+00
	10	Error	1.24E+00	5.69E+04	3.28E+04	1.53E+00	1.49E+00	1.36E+00	2.88E+03	1.89E+00	**1.10E+00**
		SD	1.78E-01	8.50E+04	6.33E+04	4.64E-01	2.67E-01	2.76E-01	1.62E+03	3.76E-01	1.36E-01
f_{26}	2	Error	5.14E-04	7.33E-01	7.35E-01	1.13E-04	7.58E-05	8.94E-05	6.44E-01	5.94E-04	**0.00E+00**
		SD	1.53E-03	2.44E-01	2.15E-01	1.07E-04	1.22E-04	2.31E-04	3.04E-01	5.58E-04	1.86E-13
	10	Error	2.47E+00	4.66E+00	4.54E+00	2.62E+00	2.47E+00	2.41E+00	4.67E+00	3.14E+00	**2.35E+00**
		SD	3.25E-01	1.12E-01	2.24E-01	2.53E-01	2.46E-01	2.27E-01	6.04E-02	9.58E-02	2.66E-01

Table 4.12: Numerical Results of benchmark function with 30 and 50 dimension

F	D	Stat	ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
f_{11}	2	Error	2.06E+07	1.47E+09	1.51E+09	3.02E+07	2.10E+07	1.98E+07	1.18E+09	2.00E+07	**1.40E+07**
		SD	1.04E+07	1.58E+09	1.32E+09	8.51E+06	6.01E+06	7.56E+06	7.40E+08	6.46E+06	6.80E+06
	10	Error	3.31E+07	9.67E+09	8.58E+09	3.90E+07	2.33E+07	2.47E+07	6.48E+09	6.16E+07	**2.31E+07**
		SD	1.47E+07	3.90E+09	5.06E+09	1.34E+07	9.39E+06	6.53E+06	3.80E+09	1.46E+07	8.45E+06
f_{12}	2	Error	2.66E+03	1.58E+11	1.68E+11	3.12E+04	1.09E+04	4.95E+03	1.46E+11	6.73E+03	**7.68E+02**
		SD	3.29E+03	2.17E+10	2.28E+10	3.02E+04	8.89E+03	3.81E+03	3.19E+10	9.10E+03	1.45E+03
	10	Error	1.47E+05	3.13E+11	3.13E+11	2.90E+06	1.86E+05	1.14E+05	3.20E+11	2.46E+05	**1.45E+04**
		SD	1.36E+05	4.03E+10	3.86E+10	7.00E+06	1.28E+05	1.10E+05	3.22E+10	1.93E+05	1.15E+04
f_{13}	2	Error	1.91E+03	4.20E+06	1.16E+06	3.51E+03	2.44E+03	2.79E+03	2.67E+05	3.49E+03	**1.34E+03**
		SD	1.22E+03	1.04E+07	8.14E+05	4.09E+03	1.87E+03	2.90E+03	5.49E+04	2.40E+03	1.32E+03
	10	Error	8.66E+03	1.75E+06	6.61E+05	1.14E+04	**6.63E+03**	1.04E+04	4.62E+05	3.27E+04	8.98E+03
		SD	4.54E+03	3.97E+06	3.93E+05	4.34E+03	1.93E+03	2.81E+03	7.61E+04	1.63E+04	2.50E+03
f_{14}	2	Error	6.37E+01	5.02E+04	4.65E+04	9.31E+01	8.21E+01	8.19E+01	4.25E+04	8.89E+01	**5.94E+01**
		SD	2.79E+01	9.40E+03	1.09E+04	1.95E+01	2.55E+01	9.16E+00	1.20E+04	1.54E+01	2.85E+01
	10	Error	1.39E+02	1.22E+05	1.06E+05	2.21E+02	1.69E+02	1.51E+02	7.98E+04	1.43E+02	**1.20E+02**
		SD	2.59E+01	3.08E+04	4.58E+04	4.13E+01	3.18E+01	3.21E+01	4.00E+04	3.63E+01	1.58E+01
f_{15}	2	Error	**2.04E+01**	2.07E+01	2.07E+01	**2.04E+01**	**2.04E+01**	**2.04E+01**	2.06E+01	2.05E+01	**2.04E+01**
		SD	5.34E-02	3.38E-01	3.81E-01	3.65E-02	3.30E-02	3.96E-02	2.91E-01	4.39E-02	3.54E-02
	10	Error	**2.06E+01**	2.07E+01	2.07E+01	**2.06E+01**	**2.06E+01**	**2.06E+01**	**2.06E+01**	2.07E+01	**2.06E+01**
		SD	2.95E-02	2.58E-01	3.05E-01	2.23E-02	4.73E-02	3.43E-02	2.94E-01	4.78E-02	3.28E-02
f_{16}	2	Error	1.76E+01	4.55E+01	4.60E+01	1.87E+01	1.75E+01	1.75E+01	4.47E+01	1.96E+01	**1.67E+01**
		SD	1.56E+00	5.71E+00	4.85E+00	9.86E-01	1.79E+00	1.34E+00	4.46E+00	1.50E+00	1.84E+00
	10	Error	**3.67E+01**	7.87E+01	8.62E+01	4.05E+01	3.81E+01	3.80E+01	7.79E+01	4.52E+01	3.75E+01
		SD	8.03E-01	8.35E+00	3.10E+00	1.68E+00	2.24E+00	2.81E+00	5.42E+00	2.27E+00	1.44E+00
f_{17}	2	Error	4.44E-02	8.80E+02	5.86E+02	5.57E-01	5.07E-02	3.77E-02	6.89E+02	4.10E-02	**3.50E-03**
		SD	3.43E-02	4.14E+02	6.87E+01	1.77E-01	5.20E-02	2.37E-02	2.46E+02	2.64E-02	7.67E-03
	10	Error	4.91E-01	2.30E+03	2.59E+03	1.34E+00	7.57E-01	5.60E-01	2.14E+03	7.72E-01	**1.12E-01**
		SD	1.00E-01	8.55E+02	5.47E+02	2.22E-01	1.94E-01	1.35E-01	7.07E+02	1.22E-01	4.92E-02
f_{18}	2	Error	1.25E+00	5.31E+02	5.38E+02	6.71E+00	5.39E+00	2.55E+00	5.24E+02	1.81E+00	**3.78E-10**
		SD	6.49E-01	7.22E+01	2.72E+01	1.00E+00	9.17E-01	1.31E+00	8.22E+01	7.51E-01	8.20E-10
	10	Error	8.66E+00	1.00E+03	9.82E+02	2.60E+01	2.05E+01	1.12E+01	9.68E+02	1.00E+01	**2.84E+00**
		SD	2.35E+00	5.29E+01	4.50E+01	4.36E+00	5.12E+00	2.48E+00	5.26E+01	3.99E+00	1.43E+00
f_{19}	2	Error	1.06E+02	6.64E+02	6.78E+02	1.06E+02	1.21E+02	1.20E+02	5.77E+02	1.16E+02	**1.00E+02**
		SD	1.32E+01	7.09E+01	4.89E+01	1.18E+01	1.37E+01	1.51E+01	1.01E+02	1.30E+01	1.53E+01
	10	Error	**2.17E+02**	1.29E+03	1.28E+03	2.60E+02	2.44E+02	2.53E+02	1.22E+03	2.78E+02	2.32E+02
		SD	2.03E+01	6.44E+01	9.66E+01	2.98E+01	2.04E+01	2.30E+01	1.14E+02	1.93E+01	3.22E+01
f_{20}	2	Error	6.25E+00	5.49E+03	5.15E+03	1.63E+01	3.46E+01	1.89E+01	5.06E+03	6.00E+01	**4.62E+00**
		SD	3.42E+00	1.45E+03	1.31E+03	8.21E+00	4.80E+01	3.66E+01	9.07E+02	4.51E+01	2.19E+00
	10	Error	**4.31E+01**	9.07E+03	9.60E+03	3.78E+02	3.01E+02	1.78E+02	9.78E+03	8.96E+01	7.06E+01
		SD	3.97E+01	1.06E+03	1.87E+03	9.27E+01	1.12E+02	7.94E+01	1.53E+03	4.18E+01	4.89E+01
f_{21}	2	Error	2.71E+03	6.69E+03	6.63E+03	2.82E+03	2.82E+03	2.61E+03	6.40E+03	3.04E+03	**2.35E+03**
		SD	2.85E+02	1.52E+03	1.87E+03	1.99E+02	2.82E+02	3.65E+02	1.16E+03	4.39E+02	3.35E+02
	10	Error	5.55E+03	1.23E+04	1.24E+04	6.12E+03	5.85E+03	5.78E+03	1.18E+04	7.27E+03	**5.11E+03**
		SD	3.25E+02	2.07E+03	2.91E+03	4.38E+02	5.36E+02	4.99E+02	1.33E+03	2.26E+02	5.83E+02
f_{22}	2	Error	5.01E-01	2.03E+00	2.26E+00	5.53E-01	5.27E-01	5.30E-01	2.55E+00	7.88E-01	**4.86E-01**
		SD	6.48E-02	1.14E+00	7.48E-01	1.01E-01	4.83E-02	9.74E-02	8.37E-01	8.62E-02	6.92E-02
	10	Error	6.83E-01	2.03E+00	3.52E+00	6.99E-01	6.51E-01	6.26E-01	3.43E+00	9.89E-01	**5.66E-01**
		SD	3.24E-02	6.19E-01	1.20E+00	6.40E-02	8.27E-02	9.44E-02	9.26E-01	1.14E-01	8.86E-02
f_{23}	2	Error	2.87E-01	1.13E+01	1.18E+01	3.04E-01	3.36E-01	**2.82E-01**	1.13E+01	4.42E-01	2.86E-01
		SD	2.66E-02	1.29E+00	1.47E+00	5.29E-02	2.67E-02	2.12E-02	1.42E+00	3.69E-02	2.54E-02
	10	Error	4.56E-01	1.30E+01	1.36E+01	4.95E-01	4.86E-01	4.64E-01	1.38E+01	4.99E-01	**3.81E-01**
		SD	2.39E-02	1.13E+00	1.31E+00	4.36E-02	4.20E-02	4.67E-02	7.31E-01	2.65E-02	2.47E-02
f_{24}	2	Error	**2.08E-01**	3.18E+02	2.82E+02	2.35E-01	2.19E-01	2.23E-01	2.22E+02	2.42E-01	2.24E-01
		SD	2.32E-02	1.84E+02	1.26E+02	1.80E-02	1.36E-02	1.63E-02	9.76E+01	2.41E-02	1.97E-02
	10	Error	2.94E-01	3.30E+02	2.43E+02	3.00E-01	2.93E-01	**2.83E-01**	3.10E+02	3.66E-01	2.88E-01
		SD	1.43E-02	1.59E+02	2.60E+01	1.81E-02	2.27E-02	3.01E-02	1.24E+02	4.49E-02	2.23E-02
f_{25}	2	Error	1.62E+01	5.62E+06	1.50E+07	1.83E+01	1.58E+01	1.77E+01	4.22E+05	1.75E+01	**1.34E+01**
		SD	3.11E+00	8.67E+06	2.21E+07	3.10E+00	2.84E+00	2.02E+00	3.89E+05	2.06E+00	2.35E+00
	10	Error	3.78E+01	1.84E+08	2.53E+08	4.55E+01	4.36E+01	4.01E+01	8.71E+07	4.44E+01	**3.18E+01**
		SD	6.86E+00	1.10E+08	2.07E+08	6.09E+00	5.13E+00	5.69E+00	8.12E+07	5.22E+00	5.16E+00
f_{26}	2	Error	1.10E+01	1.40E+01	1.40E+01	1.12E+01	1.10E+01	1.12E+01	1.39E+01	1.19E+01	**1.09E+01**
		SD	3.22E-01	1.71E-01	2.61E-01	2.69E-01	2.41E-01	3.63E-01	3.66E-01	3.56E-01	3.10E-01
	10	Error	1.99E+01	2.43E+01	2.40E+01	2.00E+01	**1.97E+01**	2.01E+01	2.41E+01	2.11E+01	2.00E+01
		SD	4.77E-01	2.35E-01	3.19E-01	2.20E-01	4.14E-01	3.65E-01	2.65E-01	4.32E-01	2.93E-01

4.2.3.1 Parametric test:- Statistical significant test (CEC2014)

In this significant test, Kruskal-Wallis test is used and the results are based on 30 runs and 95% confidence interval. The data consists of average rank and the p-value is recorded. There will be significant improvement for an algorithm if the p-value is less than 0.05.

Table 4.13 shows the outcome from the significant test using Kruskal-Wallis method for the algorithms with dimensions 2, 10, 30 and 50. The algorithms were tested in 16 CEC benchmark functions used in previous test. As noted all algorithms in all tested conditions had significant difference among them. To see which algorithm is most significant than others, the result are ranked from smallest to largest. The smallest rank indicates the algorithm is the most significant than others. While the largest rank means that the algorithm is likely to have less significant difference from others. The results, as noted, clearly show that HABCSDA was the most significantly different that others with average mean rank 59.19 and rank 1.75. Algorithms CLEABC, CEABC and CLABC were with 2nd ,3rd and 5th lowest rank. Although the CSDA and SBSDA had 2nd and 3rd highest rank respectively, CSDA and SBSDA still showed significant difference than original SDA. It can be concluded that the hybrid version of ABC and SDA algorithm were capable to perform well and significantly better than the original algorithms and other proposed algorithms in CEC2014 benchmark functions problems.

Table 4.13: Result of significant test for CEC2014 benchmark functions

F	D	ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA	p-value
f_{11}	2	166.70	221.60	135.20	158.90	126.20	145.70	141.50	59.90	63.80	8.66E-18
	10	92.60	224.90	229.10	96.20	105.50	99.20	222.50	75.80	73.70	1.82E-35
	30	92.90	223.10	228.80	140.60	85.10	88.40	224.60	86.30	49.70	5.62E-39
	50	58.00	235.90	229.30	118.00	92.80	68.20	217.30	160.00	58.00	3.98E-42
f_{12}	2	100.40	221.00	236.30	91.70	62.30	74.60	219.20	145.40	68.60	1.45E-39
	10	69.34	222.97	218.10	101.52	105.86	104.72	236.52	114.76	50.41	2.35E-52
	30	115.40	226.10	235.10	157.10	61.10	92.60	215.30	87.20	29.60	8.74E-45
	50	102.10	226.90	225.10	161.80	81.10	76.60	230.50	115.00	18.40	6.71E-45
f_{13}	2	97.30	243.40	237.70	159.80	105.80	82.30	227.80	132.60	86.20	4.03E-30
	10	113.90	238.70	221.00	98.00	83.30	108.80	216.80	67.70	71.30	8.42E-37
	30	113.90	238.70	221.00	98.00	83.30	108.80	216.80	67.70	71.30	8.42E-37
	50	42.70	233.50	233.20	104.80	67.60	94.60	215.80	165.70	79.60	1.18E-42
f_{14}	2	68.00	220.40	232.10	165.50	68.00	90.50	224.00	83.00	68.00	1.27E-48
	10	73.70	232.70	219.50	125.30	64.70	113.30	224.30	143.30	22.70	4.14E-45
	30	101.30	234.20	225.50	128.00	65.00	93.20	216.80	112.10	43.40	8.99E-40
	50	114.70	239.50	229.30	156.10	76.00	89.50	213.70	75.70	43.00	1.36E-41
f_{15}	2	86.00	222.50	217.70	95.00	86.00	86.00	236.30	104.00	86.00	4.05E-48
	10	88.70	212.90	160.40	118.10	117.80	112.40	132.20	150.20	126.80	3.34E-08
	30	81.50	183.50	186.50	124.40	89.00	121.10	161.90	189.80	81.80	1.29E-14
	50	109.30	181.90	172.30	94.60	106.60	97.00	129.70	214.30	131.80	2.17E-12
f_{16}	2	134.00	147.50	134.00	134.00	134.00	134.00	134.00	134.00	134.00	2.14E-03
	10	80.00	228.50	235.10	103.10	60.50	85.40	212.90	159.80	54.20	1.98E-42
	30	83.30	227.00	227.90	113.00	80.00	71.30	221.60	143.30	52.10	4.33E-40
	50	79.00	220.90	249.70	122.50	44.20	79.30	211.90	167.50	62.50	9.45E-45

f_{17}	2	98.75	125.15	185.45	158.60	110.15	131.45	195.95	171.50	42.50	3.83E-17
	10	72.80	227.60	231.80	126.80	92.00	109.40	217.10	102.50	39.50	1.34E-39
	30	92.00	232.10	219.20	165.50	86.60	87.80	225.20	92.00	19.10	8.10E-46
	50	111.10	227.80	235.60	167.50	63.40	76.60	219.10	118.90	17.50	2.29E-47
f_{18}	2	114.50	210.05	194.90	114.50	114.50	114.50	127.55	114.50	114.50	1.02E-27
	10	95.00	227.00	238.10	163.10	76.25	84.95	211.40	100.40	23.30	3.81E-46
	30	140.30	225.20	227.30	160.70	57.50	92.60	224.00	76.40	15.50	1.83E-49
	50	141.10	238.60	224.80	161.80	65.80	88.60	219.10	78.40	19.30	1.99E-47
f_{19}	2	90.50	222.50	238.70	90.50	90.50	90.50	215.30	90.50	90.50	6.03E-51
	10	151.40	223.40	232.10	96.80	68.60	101.30	221.00	88.40	36.50	3.54E-42
	30	125.00	232.10	236.00	67.70	70.10	118.40	208.40	107.00	54.80	1.20E-39
	50	84.10	232.60	231.70	114.40	40.60	100.00	218.20	145.60	70.30	2.73E-40
f_{20}	2	66.50	235.25	227.45	92.30	66.50	89.00	213.80	162.20	66.50	4.42E-48
	10	135.10	225.50	221.00	41.20	69.60	83.20	230.00	165.50	48.40	2.05E-47
	30	101.60	228.80	223.40	116.30	55.70	77.00	224.30	157.10	35.30	1.28E-44
	50	137.00	220.10	229.10	155.60	33.80	102.50	227.30	66.50	47.60	2.90E-47
f_{21}	2	104.00	239.00	222.35	101.30	64.55	75.50	215.15	156.20	41.45	1.11E-43
	10	79.10	221.90	232.10	102.80	70.70	80.00	222.50	162.80	47.60	8.76E-43
	30	104.60	226.70	225.50	102.50	86.30	75.20	224.00	135.20	39.50	7.09E-40
	50	92.20	231.10	226.90	111.10	62.80	84.10	224.50	167.50	37.30	1.27E-43
f_{22}	2	151.40	90.65	58.85	153.20	193.10	165.50	77.00	152.00	177.80	5.21E-16
	10	110.30	165.20	200.30	86.30	92.00	100.70	241.70	101.30	121.70	1.09E-21
	30	83.90	212.00	224.60	92.00	67.40	82.10	231.50	168.20	57.80	1.73E-40
	50	79.90	204.10	241.30	103.90	96.70	69.40	236.80	167.20	38.20	1.25E-43
f_{23}	2	23.90	239.60	239.60	117.50	80.90	100.10	197.30	98.30	122.30	1.81E-42
	10	126.80	225.20	227.90	122.60	107.90	90.80	223.40	61.40	33.50	1.40E-41
	30	120.50	220.70	234.20	84.20	57.80	54.80	221.60	165.20	60.50	8.00E-45
	50	112.30	219.10	228.10	121.60	78.40	93.40	235.30	129.10	20.20	1.67E-41
f_{24}	2	118.10	223.40	218.60	116.60	105.50	104.30	234.50	70.70	27.80	4.04E-41
	10	116.00	236.90	215.90	132.50	106.70	117.50	223.70	16.40	53.90	1.97E-45
	30	71.60	227.30	234.50	115.70	54.50	86.60	214.70	127.70	86.90	1.29E-38
	50	80.30	232.40	212.90	93.50	83.00	67.40	231.20	145.40	73.40	5.64E-39
f_{25}	2	68.30	249.20	178.40	165.20	118.40	150.50	158.60	68.90	62.00	2.35E-32
	10	105.20	234.50	232.70	103.40	65.60	83.60	209.30	146.00	39.20	5.39E-42
	30	76.70	229.70	229.40	119.00	85.10	113.90	217.40	110.30	38.00	1.42E-39
	50	118.60	233.50	237.40	125.50	72.70	86.50	211.60	121.60	30.10	4.62E-41
f_{26}	2	88.10	227.60	227.30	114.20	90.20	83.00	221.60	152.00	15.50	4.98E-45
	10	76.40	230.60	216.50	101.00	79.70	63.80	229.40	164.90	57.20	9.01E-43
	30	66.80	230.30	226.10	95.60	74.00	94.40	220.10	158.00	54.20	1.26E-41
	50	49.00	242.20	218.20	83.80	83.50	94.60	222.10	160.00	84.10	1.52E-40
Average		97.59	220.86	217.91	119.13	82.67	95.05	210.10	123.42	59.19	
Average (rank)		3.50	8.16	7.86	4.72	2.59	3.42	7.20	4.66	**1.75**	

4.3 Constrained single objective optimisation problems

In some cases, constraint maybe included to the objective function and created constrained optimisation problems. In general the constrained optimisation problem can be represented as:-

$$optimise\ f(x), x = (x_1, x_2, \dots\dots, x_n)$$

Subject to $$g_j(x) \geq 0,\ j = 1,2, \dots., q$$

$$h_k(x) = 0, \qquad k = 1,2, \dots., m$$

where x is feasible solution if and only if q inequality constraint is more than zero and m equality constraint is equal zero. The details of technique for unconstrained and constrained optimisation problems can be found in [8].

4.3.1 Approaches for constrained single objective optimisation problems

There are several methods reported in the literature when dealing with constrained problems. These include penalty functions, decoder, special operator (representation and operator) and separation of objective function and constraint and hybrid method [154]. In this research, the penalty function method is used to handle the constrained problem. his is because it is simple and easy to use [155] .Moreover, nonlinear constrained problems can be handled with penalty functions [156] . The penalty function is based on the mathematical programming method a constrained problem is changed to an unconstrained optimization problem [157]. The general form of objective function with penalty is the following:-

$$\text{Min } \emptyset(\vec{\boldsymbol{x}}) = \boldsymbol{f}(\vec{\boldsymbol{x}}) + \boldsymbol{p}(\vec{\boldsymbol{x}}) \qquad (4.1)$$

where $\emptyset(\vec{x})$ is the expanded objective function, $f(\vec{x})$ is the original objective function and $p(\vec{x})$ is the penalty function. The main objective to get optimal solution is to gain small fitness of infeasible solution. When the problem has constrains, the constraint value is added into the objective function. By doing this, the penalty value is also considered to get a low value of fitness of infeasible solution. This low value of fitness is expected in minimization problems. Although the execution of this method is simple, it requires fine tuning of penalty coefficient value and this tuning value is different for each type of problem. The simplest method of penalty function is the death penalty. In this method, it will avoid to use infeasible solution during the search, which means that the infeasible solution is eliminated from the search process. Other types of penalty method include static penalty, adaptive penalty and dynamic penalty [158]. In this research, the static penalty function is used, where the original objective function $f(\vec{x})$ is penalised by $p(\vec{x})$**.** So the final objective will be reduced based on the value of penalty. The general form of static penalty function is given as:-

$$\boldsymbol{p(\vec{x}) = r\sum_{i=1}^{m}\big(g_i(\vec{x})\big)^2 + c\sum_{j=1}^{p}\big(h_k(\vec{x})\big)^2} \quad (4.2)$$

where r and c are penalty coefficients and represent the importance of $g_i(\vec{x})$ and $h_k(\vec{x})$. The penalty coefficient values are problem dependant. Fine-tuning of penalty coefficients r and c is crucial to ensure that valuable information can be extracted accordingly. This fine-tuning is based on trial and error method, as the penalty coefficient are problem dependent.

4.3.2 Experiments on constrained optimisation problems

This section, the proposed algorithms are tested in single objective constrained optimisation problems. Ten constrained benchmark functions from test set CEC2006 are used to evaluate the accuracy, robustness and efficiency of the algorithm. Moreover, the algorithms are further tested in five engineering design constrained problems.

4.3.2.1 Constrained benchmark functions

Ten standard benchmark functions are used to evaluate the performance of the proposed algorithms. The benchmark functions have different types of characteristics and landscapes. In the algorithms the population size was set to 30 and the number of iterations to 1600 (NFE = 48,000). In these tests, all problem inequalities were changed to equalities based on Suganthan works [159]. Table 4.14 and 4.15 show the statistical results for the algorithms. Table 4.14 shows the statistical values for the original ABC and SDA with non-hybrid proposed algorithms. As noted, CEABC outperformed other algorithms for average and best optimal point in functions g_2, g_3 . while in functions g_4, g_8 CEABC was able to provide lowest optimal point in average. The robustness of CEABC was better except in function g_2. In functions g_5, CSDA achieved the best solution among the algorithms but the average and standard deviation showed inconsistency because the SDA algorithm variant still suffers from trapping at local optima.

Table 4.14:- Numerical result of SDA and ABC variants in solving constrained benchmark functions

f	Stat	$f(x^{*})$	ABC	SDA	CSDA	CLABC	CEABC	CLABC
g_1	best	1.393E+00	1.390E+00	1.390E+00	1.390E+00	1.390E+00	1.390E+00	1.391E+00
	Avg		1.390E+00	1.390E+00	1.616E+00	1.444E+00	1.405E+00	1.437E+00
	SD		2.388E-14	2.093E-16	3.051E-01	9.607E-02	3.261E-02	5.990E-02
	worse		1.390E+00	1.390E+00	2.341E+00	1.615E+00	1.492E+00	1.531E+00
g_2	best	-3.067E+04	-3.240E+04	-3.278E+04	-3.293E+04	-3.285E+04	**-3.325E+04**	-3.277E+04
	Avg		-3.224E+04	-3.212E+04	-3.204E+04	-3.246E+04	**-3.278E+04**	-3.251E+04
	SD		1.222E+02	5.957E+02	6.921E+02	4.342E+02	5.600E+02	2.612E+02
	worse		-3.209E+04	-3.096E+04	-3.083E+04	-3.185E+04	-3.164E+04	-3.213E+04
g_3	best	-6.962E+03	-7.949E+03	1.476E+05	-4.365E+03	-7.948E+03	**-9.000E+03**	-7.948E+03
	Avg		-7.949E+03	2.736E+07	-1.357E+03	-7.555E+03	**-9.000E+03**	-7.948E+03
	SD		1.203E-03	4.102E+07	1.485E+03	5.387E+02	0.000E+00	3.631E-01
	worse		-7.949E+03	1.064E+08	-3.189E+01	-6.965E+03	-9.000E+03	-7.948E+03
g_4	best	2.431E+01	2.388E+01	1.343E+02	3.484E+01	2.441E+01	2.425E+01	2.441E+01
	Avg		2.495E+01	2.808E+02	7.631E+01	2.548E+01	**2.467E+01**	2.472E+01
	SD		9.231E-01	9.274E+01	4.726E+01	9.270E-01	4.558E-01	4.517E-01
	worse		2.679E+01	4.086E+02	1.652E+02	2.667E+01	2.533E+01	2.552E+01
g_5	best	-9.600E-02	-9.583E-02	-9.583E-02	**-2.096E+01**	-9.583E-02	-9.583E-02	-9.583E-02
	Avg		-6.915E-02	-5.685E-02	1.748E+01	-9.581E-02	-9.581E-02	**-9.580E-02**
	SD		3.443E-02	3.648E-02	5.592E+01	2.724E-05	3.147E-05	5.020E-05
	worse		-2.913E-02	-2.538E-02	1.430E+02	-9.576E-02	-9.575E-02	-9.571E-02
g_6	best	6.806E+02	6.786E+02	6.885E+02	1.442E+03	6.786E+02	6.790E+02	6.786E+02
	Avg		6.787E+02	1.642E+04	1.230E+04	**6.787E+02**	6.803E+02	6.788E+02
	SD		3.892E-02	2.993E+04	8.406E+03	1.140E-01	1.813E+00	2.856E-01
	worse		6.788E+02	7.813E+04	2.802E+04	6.789E+02	6.826E+02	6.793E+02
g_7	best	7.500E-01	**7.494E-01**	7.514E-01	7.504E-01	7.529E-01	7.536E-01	7.505E-01
	Avg		**7.576E-01**	1.115E+01	3.369E+01	7.610E-01	7.696E-01	7.674E-01
	SD		1.562E-02	1.898E+01	5.001E+01	5.119E-03	1.236E-02	1.725E-02
	worse		8.004E-01	5.156E+01	1.251E+02	7.657E-01	7.863E-01	7.947E-01
g_8	best	9.617E+02	9.618E+02	9.641E+02	9.746E+02	**9.618E+02**	9.619E+02	9.621E+02
	Avg		1.090E+03	9.709E+02	4.771E+04	9.663E+02	**9.627E+02**	9.652E+02
	SD		3.981E+02	1.205E+01	7.914E+04	3.444E+00	7.404E-01	2.916E+00
	worse		2.223E+03	9.976E+02	2.528E+05	9.700E+02	9.635E+02	9.681E+02
g_9	best	-1.905E+00	-2.097E+00	8.273E+03	8.057E+03	-2.092E+00	-2.094E+00	-2.092E+00
	Avg		-2.077E+00	8.670E+03	8.942E+03	**-2.081E+00**	-1.997E+00	-2.059E+00
	SD		2.121E-02	3.087E+02	8.346E+02	9.066E-03	7.189E-02	3.096E-02
	worse		-2.024E+00	9.240E+03	1.023E+04	-2.067E+00	-1.913E+00	-2.013E+00
g_{10}	best	-8.660E-01	-8.717E-01	-2.884E-01	2.479E+02	-8.715E-01	-8.710E-01	-8.702E-01
	Avg		-8.623E-01	8.848E+02	8.943E+03	**-8.689E-01**	-8.648E-01	-8.658E-01
	SD		1.454E-02	1.938E+03	6.648E+03	3.169E-03	5.868E-03	6.214E-03
	worse		-8.287E-01	6.284E+03	1.996E+04	-8.652E-01	-8.584E-01	-8.553E-01

Table 4.15 shows that, the ABC component in the hybrid algorithm HABCSDA helps this algorithm to dominated the top league in functions $g_1, g_4 g_6, g_9, g_{10}$ in providing the best solution. In terms of robustness by referring to the standard deviation values, HABCSDA was able to provide small deviations but the deviation from the average was large in function g_5.

Table 4.15:- *Numerical result of Hybrid variants in solving constrained benchmark functions*

		1.393E+00	ABC	SDA	SBSDA	SBA	HABCSDA
g_1	best	6.786E+02	1.390E+00	1.390E+00	1.390E+00	1.390E+00	**1.390E+00**
	Avg	6.787E+02	1.405E+00	1.390E+00	2.187E+00	1.390E+00	**1.390E+00**
	SD	3.892E-02	3.546E-02	2.093E-16	9.056E-01	2.455E-16	2.388E-14
	worse	6.788E+02	1.504E+00	1.390E+00	3.385E+00	1.390E+00	1.390E+00
g_2	best	-3.067E+04	-3.240E+04	-3.278E+04	-3.266E+04	-3.276E+04	-3.283E+04
	Avg		-3.224E+04	-3.212E+04	-3.221E+04	-3.231E+04	-3.236E+04
	SD		1.222E+02	5.957E+02	4.932E+02	3.660E+02	3.311E+02
	worse		-3.209E+04	-3.096E+04	-3.107E+04	-3.171E+04	-3.158E+04
g_3	best	-6.962E+03	-7.949E+03	1.476E+05	-8.603E+02	-7.949E+03	-7.949E+03
	Avg		-7.949E+03	2.736E+07	6.421E+03	-7.175E+03	-7.432E+03
	SD		1.203E-03	4.102E+07	9.550E+03	2.420E+03	5.078E+02
	worse		-7.949E+03	1.064E+08	3.180E+04	-2.866E+02	-6.636E+03
g_4	best	2.431E+01	2.403E+01	1.343E+02	3.361E+02	2.435E+01	**2.388E+01**
	Avg		2.514E+01	2.808E+02	6.728E+02	2.700E+01	2.495E+01
	SD		1.310E+00	9.274E+01	3.096E+02	2.567E+00	9.231E-01
	worse		2.856E+01	4.086E+02	1.175E+03	3.288E+01	2.679E+01
g_5	best	-9.600E-02	-9.583E-02	-9.583E-02	-9.583E-02	-9.583E-02	-9.583E-02
	Avg		-6.915E-02	-5.685E-02	-6.027E-02	-8.208E-02	-7.127E-02
	SD		3.443E-02	3.648E-02	3.601E-02	2.792E-02	3.731E-02
	worse		-2.913E-02	-2.538E-02	-2.513E-02	-2.914E-02	-8.259E-03
g_6	best	6.806E+02	6.786E+02	6.885E+02	6.845E+02	6.786E+02	**6.786E+02**
	Avg		6.790E+02	1.642E+04	1.824E+03	6.788E+02	6.790E+02
	SD		5.092E-01	2.993E+04	1.473E+03	2.894E-01	5.092E-01
	worse	7.500E-01	6.801E+02	7.813E+04	5.422E+03	6.794E+02	6.801E+02
g_7	best	-8.717E-01	7.495E-01	7.514E-01	7.494E-01	7.496E-01	7.495E-01
	Avg	-8.623E-01	7.687E-01	1.115E+01	8.230E-01	7.645E-01	7.687E-01
	SD	1.454E-02	2.340E-02	1.898E+01	1.085E-01	2.419E-02	2.340E-02
	worse	-8.287E-01	8.305E-01	5.156E+01	1.000E+00	8.296E-01	8.305E-01
g_8	best	9.617E+02	9.618E+02	9.641E+02	9.628E+02	9.618E+02	9.621E+02
	Avg		1.090E+03	9.709E+02	2.354E+03	9.650E+02	9.650E+02
	SD		3.981E+02	1.205E+01	2.905E+03	3.476E+00	2.486E+00
	worse		2.223E+03	9.976E+02	8.592E+03	9.723E+02	9.677E+02
g_9	best	-1.905E+00	-2.095E+00	8.273E+03	2.684E+02	-2.025E+00	**-2.097E+00**
	Avg		-2.067E+00	8.670E+03	6.689E+03	5.697E+02	-2.077E+00
	SD		3.641E-02	3.087E+02	2.324E+03	1.702E+03	2.121E-02
	worse		-1.971E+00	9.240E+03	7.847E+03	5.412E+03	-2.024E+00
g_{10}	best	-8.660E-01	-8.707E-01	-2.884E-01	4.816E+02	-8.715E-01	**-8.717E-01**
	Avg		-8.544E-01	8.848E+02	2.383E+03	-8.349E-01	-8.623E-01
	SD		2.448E-02	1.938E+03	1.472E+03	3.587E-02	1.454E-02
	worse		-8.032E-01	6.284E+03	5.707E+03	-7.561E-01	-8.287E-01

4.3.2.2 Engineering constrained problems

For further investigation of performance, the algorithms were tested in five engineering design problems. The problems are spring design [160], welded beam design [161], pressure vessel design [162], speed reducer design [163] and gear ratio for train design [164] problems. All these problems are nonlinear type. Similar with methods used in the constrained benchmark functions test, penalty function method is used to solve these engineering design problems. The best value, the worse value, standard deviation and average are recorded from 30 runs of simulation per algorithm.

(a) Welded beam

In this design problem the objective is to find the minimum cost of fabrication of welded beam. The mathematical formulation of fabrication of welded beam is:-

$$F(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

The set of constraints that needs to be considered is as follows

$$g_1(x) = \tau(x) - 13600 \le 0$$

$$g_2(x) = \sigma(x) - 30000 \le 0$$

$$g_3(x) = x_1 - x_4 \le 0$$

$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) \le 0$$

$$g_5(x) = 0.125 - x_1 \le 0$$

$$g_6(x) = \delta(x) - 0.25 \le 0$$

$$g_7(x) = 6000 - P_c \le 0$$

where:-

$$0.1 \le x_i \le 2.0, \qquad i = 1,4$$
$$0.1 \le x_i \le 10.0, \qquad i = 2,3$$

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \qquad \tau''\frac{MR}{J}, \qquad M = P\left(L + \frac{x_2}{2}\right)$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2}, \qquad \delta(x) = \frac{4PL^3}{Ex_4x_3^3}$$

$$P_C(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2} \times \left(1 - \frac{x^3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000lb, \qquad E = 30 \times 10^6 psi, \qquad L = 4in, \qquad G = 12 \times 10^6 psi$$

Table 4.16:- Result of welded beam problem

Param	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
Best	1.70E+00	1.69E+00	1.73E+00	1.73E+00	1.75E+00	1.71E+00	**1.67E+00**	1.72E+00	1.87E+00
x_1	0.50	0.327	0.832	0.664	0.543	0.600	0.489	0.834	0.583
x_2	1.48	1.945	1.114	0.966	1.144	1.138	1.400	0.769	0.945
x_3	7.16	9.166	4.779	7.986	8.592	7.485	7.750	7.615	10.000
x_4	0.33	0.205	0.736	0.266	0.231	0.301	0.280	0.303	0.202
Avg	1.95E+00	2.28E+00	2.47E+00	2.06E+00	**1.87E+00**	1.95E+00	1.88E+00	1.92E+00	2.03E+00
SD	1.79E-01	8.41E-01	6.51E-01	1.72E-01	8.55E-02	1.82E-01	3.04E-01	1.87E-01	1.81E-01
Worse	2.15E+00	4.56E+00	3.73E+00	2.32E+00	1.95E+00	2.24E+00	2.67E+00	2.37E+00	2.33E+00

Table 4.16 shows the design results of welded beam problem. As noted, the SBSDA achieved the best optimum value with small standard deviation. The design parameters x_1, x_2, x_3, x_4 were are within the predetermined range. Figure 4.4, shows the convergence of the algorithms for the welded design problem. Initial distribution of ABC was slightly different compared to others due to the random initial distribution where from the 30 trial runs for ABC algorithm 2 trials showed the initial population quite high and it resulted the ABC to start at different value. As noted, all the proposed algorithms slowly converged to the best point where the fastest was CEABC. The HABCSDA could not compete with other algorithms in solving the welded beam design. Overall, the performances of all proposed algorithms were far better than those of the original algorithms.

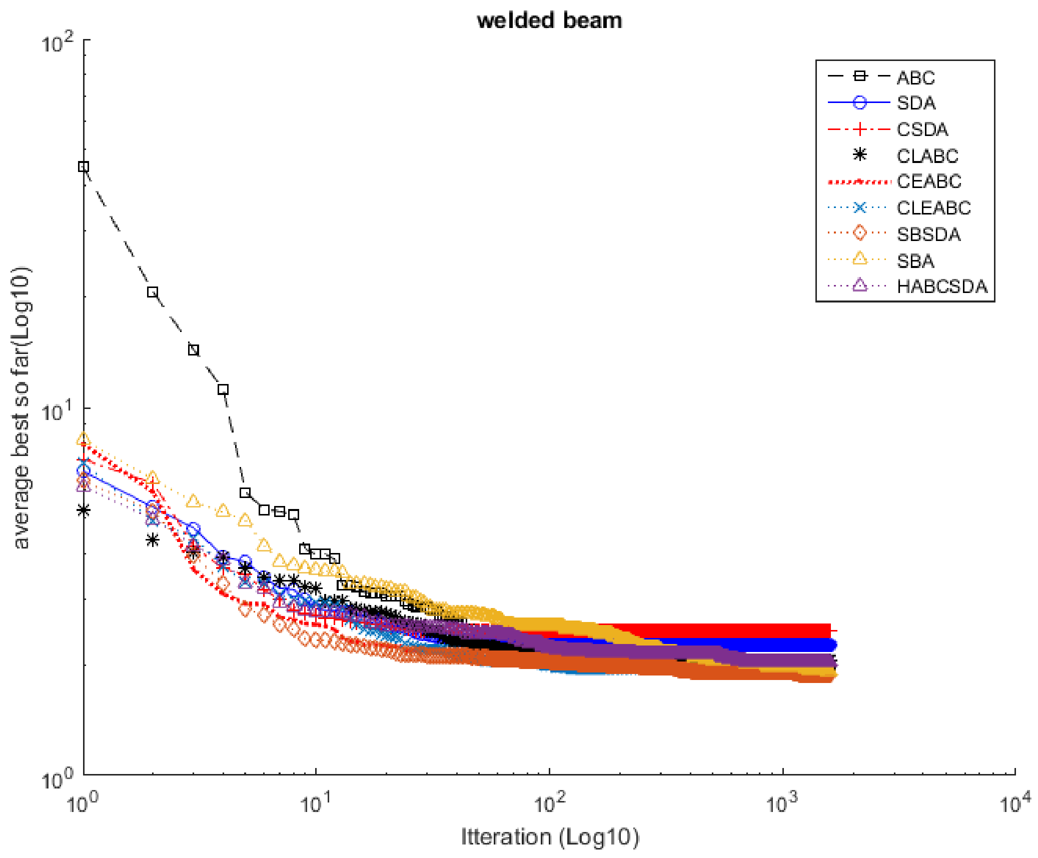


Figure 4.4 :- Convergence plot of Welded-beam design problems

(b) Gear Train design problem

In this design problem the objective is to find the best number of gear teeth to ensure the train gear ratio is at the best. The mathematical formulation of the train gear ratio is:-

$$F(x) = \left(\frac{1}{1.6931} - \frac{x_2 x_3}{x_1 x_4}\right)^2$$

The gear teeth should be within the range 12 to 60, and the constraints are

$$12 \leq x_i \leq 60, \qquad i = 1,2,3,4$$

Table 4.17: Results of gear train design problem

Param	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
best	1.65E-03	1.66E-03	3.43E-03	1.65E-03	1.65E-03	1.65E-03	2.00E-03	**7.55E-08**	1.65E-03
x_1	60.00	60.000	59.249	60.000	60.000	60.000	60.000	150.824	60.000
x_2	12.00	16.826	12.005	12.000	12.000	12.000	12.000	45.363	12.000
x_3	55.49	60.000	56.143	55.488	55.488	55.488	55.666	58.390	55.488
x_4	60.00	16.505	56.076	60.000	60.000	60.000	58.911	143.657	60.000
Avg	1.65E-03	8.49E-02	1.84E-01	1.65E-03	1.65E-03	1.65E-03	1.10E-02	**6.18E-04**	1.65E-03
SD	1.91E-18	2.13E-01	2.34E-01	**1.60E-18**	1.19E-17	2.35E-18	8.54E-03	6.35E-04	2.67E-18
worse	1.65E-03	6.11E-01	7.09E-01	1.65E-03	1.65E-03	1.65E-03	2.73E-02	1.65E-03	1.65E-03

The results in Table 4.17 show that the SBA with modified spiral trajectory performed the best in finding the best design parameter for gear train ratio. However, the smallest standard deviation from average found in this design problem was achieved with CLABC. The value of design parameter of gear design was also within the pre-setting range. Figure 4.5 shows the convergence of algorithms for the gear train design problem where all the algorithms converged near to the optimal point. As noted all algorithms converged to the optimal point in less than 10 iterations.

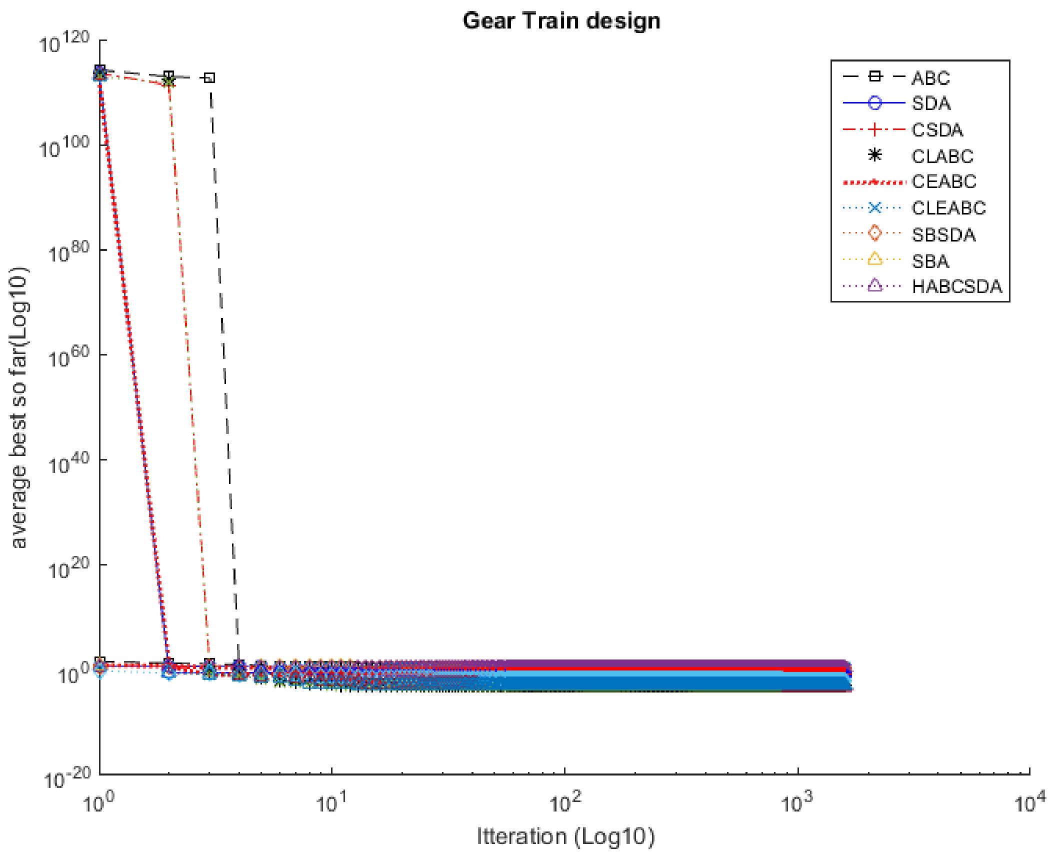


Figure 4.5:- :- Convergence plot of gear train design problems

(c) Tension/compression spring design problem

In this design problem the objective is to find the best value of coil diameter, wire diameter and number of coils to ensure the weight of spring can be minimised. The mathematical formulation of the spring design beam is:-

$$F(x) = (x_3 + 2)x_1^2 x_2$$

The challenge in this problem is that during the design the deflection, shear stress and surge frequency need to be taken into consideration and the values must be less than or equal to zero. The list of constraints are thus as follows

$$g_1(x) = 1 - \left(\frac{x_2^3 x_3}{71785 x_1^4}\right) \leq 0$$

$$g_2(x) = \frac{4x_2^2 x_1 x_2}{12566(x_1^3 x_2 - x_1^4)} + \frac{1}{5108 x_1^2} \leq 0$$

$$g_3(x) = 1 - \left(\frac{140.45 x_1}{x_2^2 x_3}\right) \leq 0$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

where

$$0.05 \leq x_1 \leq 2, \qquad 0.25 \leq x_2 \leq 1.3, \qquad 2 \leq x_3 \leq 15,$$

Table 4.18:- Results of compression spring design problem

Param	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
best	**1.27E-02**	1.28E-02	1.37E-02	**1.27E-02**	**1.27E-02**	**1.27E-02**	**1.27E-02**	1.31E-02	**1.27E-02**
x_1	0.05	0.057	0.053	0.052	0.051	0.052	0.089	0.064	0.055
x_2	0.38	0.498	0.391	0.363	0.349	0.370	0.912	0.625	0.452
x_3	9.99	14.249	12.663	10.974	11.766	10.688	15.000	4.887	7.377
x_4	1.28E-02	5.92E-02	5.38E-02	1.28E-02	1.27E-02	1.28E-02	4.70E+15	1.43E-02	1.28E-02
Avg	2.10E-04	8.77E-02	7.54E-02	6.03E-05	**1.30E-05**	7.73E-05	1.49E+16	1.35E-03	8.50E-05
SD	1.33E-02	2.42E-01	2.26E-01	1.29E-02	**1.27E-02**	1.30E-02	4.70E+16	1.74E-02	1.30E-02
worse	1.27E-02	1.28E-02	1.37E-02	1.27E-02	1.27E-02	1.27E-02	1.27E-02	1.31E-02	1.27E-02

In this design, as shown in Table 4.18, six algorithms including the original ABC and the five proposed algorithms (CLABC, CEABC, CLEABC, SBSDA, HABCSDA) achieved the same best optimal values. Among these the SBSDA had a large standard deviation. Figure 4.6 shows the convergence of algorithms for the spring design problem. The best among the top 6 performed algorithms was CEABC.

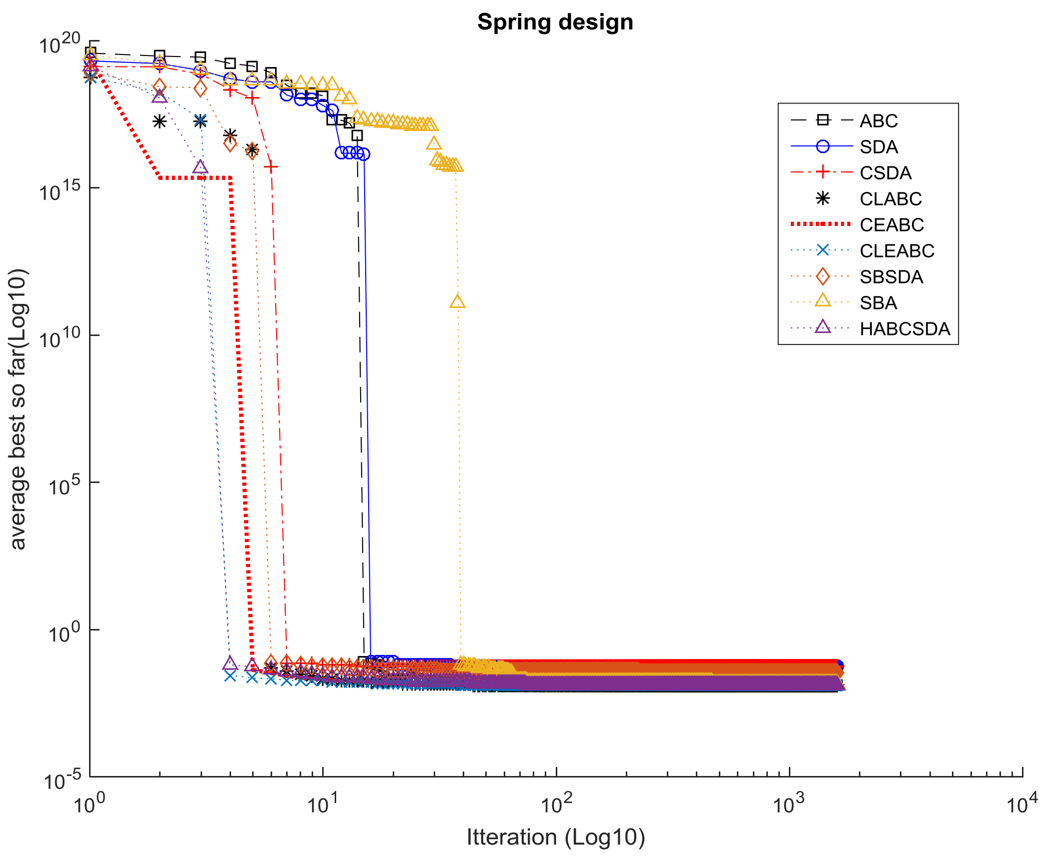


Figure 4.6 :- Convergence plot of spring design problems

(d) Pressure vessel design problem

In this design problem the objective is to find the best thickness of spherical head, spherical inner radius, length of cylinder and cylinder thickness to ensure the minimum cost of fabrication of the vessel can be achieved. The mathematical formulation of the fabrication of pressure vessel design is

$$F(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

The challenge in this problem is that during the design; there is thickness limitation where the thickness of cylinder and the spherical head must be within 0 to 100. While the inner radius of spherical head, cylinder and length of the cylinder must within 10 to 200 range. Thus, the constraints are

$$0 \leq x_i \leq 100, \qquad i = 1,2$$

$$10 \leq x_i \leq 200, \qquad i = 3,4$$

Table 4.19:-Results of pressure vessel design problem

Param	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
best	5.91E+03	6.59E+03	8.02E+03	5.95E+03	6.03E+03	2.31E+03	2.31E+03	**1.41E+03**	5.91E+03
x_1	0.81	1.768	3.160	0.822	0.833	0.810	0.894	0.854	0.794
x_2	0.40	1.649	2.677	0.402	0.406	0.427	0.442	0.423	0.391
x_3	41.90	50.991	103.638	42.074	42.575	41.975	46.338	44.150	40.831
x_4	179.41	200.000	79.716	177.479	178.488	179.086	194.233	152.915	193.066
Avg	5.99E+03	3.37E+05	8.16E+04	6.07E+03	6.15E+03	**5.68E+03**	5.12E+04	5.35E+03	6.00E+03
SD	**4.42E+01**	6.24E+05	1.19E+05	8.52E+01	9.64E+01	1.19E+03	9.10E+04	1.61E+03	7.39E+01
worse	6.05E+03	2.07E+06	3.12E+05	6.16E+03	6.23E+03	6.19E+03	2.49E+05	6.17E+03	6.14E+03

As noted in the results in Table 4.19, the ABC was the most robust algorithm in solving the pressure vessel design problem. While the bee movement in spiral manner helped SBA to reach the best optimal point, it deviated quite far from average value, thus was not robust. Although in this problem ABC was the most qualified candidate to find the best optimal point, the proposed hybrid version HABCSDA was the second best algorithm in finding the best design parameters. The convergence characteristics of the algorithms in solving this problem are shown in Figure 4.7.

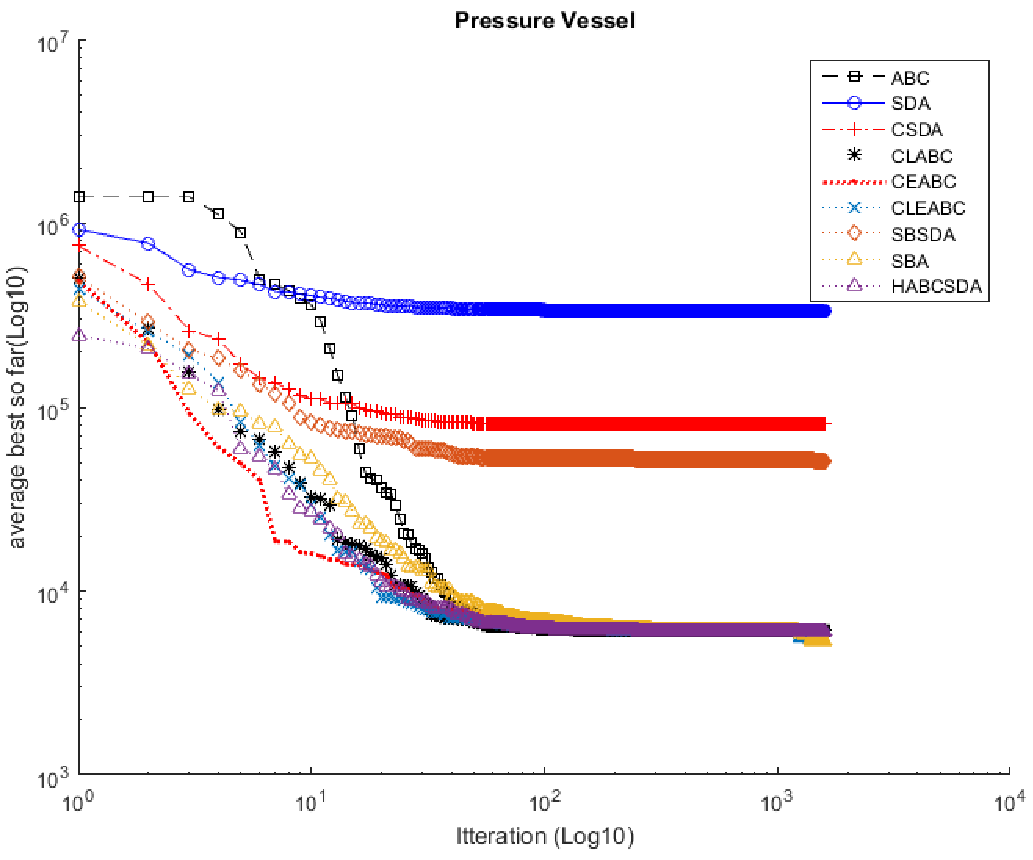


Figure 4.7:- :- Convergence plot of pressure vessel design problems

(e)　Speed reducer design problem

In this design problems the objective is to find the minimum weight of speed reducer. The mathematical formulation of weight of speed reducer is:-

$$F(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2)$$

There are 7 design parameters $(x_1, \dots, x_7)$ involved and to get the minimum of weigth of speed reducer these parameters should be within specific range of values. The list of constraints to adhere to is

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$$

$$g_3(x) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \leq 0$$

$$g_4(x) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \leq 0$$

$$g_5(x) = \frac{\left[745\left(\frac{x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right]^{\frac{1}{2}}}{110x_6^3} - 1 \leq 0$$

$$g_6(x) = \frac{\left[745\left(\frac{x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6\right]^{\frac{1}{2}}}{85x_7^3} - 1 =\leq 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 =\leq 0$$

$$g_8(x) = \frac{5x_2}{x_1} - 1 =\leq 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 =\leq 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 =\leq 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 =\leq 0$$

where:-

$$2.6 \leq x_1 \leq 3.6, \quad 0.7 \leq x_2 \leq 0.8, \quad 17 \leq x_3 \leq 28,$$
$$7.3 \leq x_4 \leq 8.3, \quad 7.3 \leq x_5 \leq 8.3, \quad 2.9 \leq x_6 \leq 3.9,$$
$$5 \leq x_7 \leq 5.5,$$

Table 4.20:- Result of speed reducer design problem

Param	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
best	**2.43E+03**	2.45E+03	2.49E+03	**2.43E+03**	**2.43E+03**	**2.43E+03**	2.45E+03	**2.43E+03**	**2.43E+03**
x_1	2.60	2.600	2.600	2.600	2.600	2.600	2.600	2.600	2.600
x_2	0.70	0.700	0.700	0.700	0.700	0.700	0.700	0.700	0.700
x_3	17.00	17.193	17.000	17.000	17.000	17.000	17.174	17.000	17.000
x_4	7.30	7.301	8.010	7.300	7.300	7.300	7.796	7.300	7.300
x_5	7.30	7.536	7.454	7.300	7.300	7.300	7.970	7.300	7.300
x_6	2.95	3.237	3.000	2.952	2.952	2.952	3.105	2.952	2.952
x_7	5.00	5.149	5.100	5.000	5.000	5.000	5.000	5.000	5.000
Avg	2.43E+03	2.64E+03	2.60E+03	2.43E+03	2.43E+03	2.43E+03	2.61E+03	2.43E+03	2.43E+03
SD	**0.00E+00**	1.75E+02	9.93E+01	**0.00E+00**	**0.00E+00**	4.79E-13	1.41E+02	4.82E-12	**0.00E+00**
worse	2.43E+03	3.00E+03	2.79E+03	2.43E+03	2.43E+03	2.43E+03	2.83E+03	2.43E+03	2.43E+03

As noted from the results in Table 4.20, the CLABC,CEABC,HABCSDA exhibited themselves as the best proposed algorithms in solving the speed reducer design problem with standard deviation of zero . and after 48,000 NFE. This implies that these algorithms are highly robust in solving this kind of problems. The worst performer was the original SDA. Figure 4.8 shows the convergence of the algorithms for speed reducer design problem. As noted, CEABC converged near to optimal point faster than others, while CSDA,SDA and SBSDA were not able to go below 2600. This problem occurred as the algorithm was trapped at local optimum in most of the trial runs.

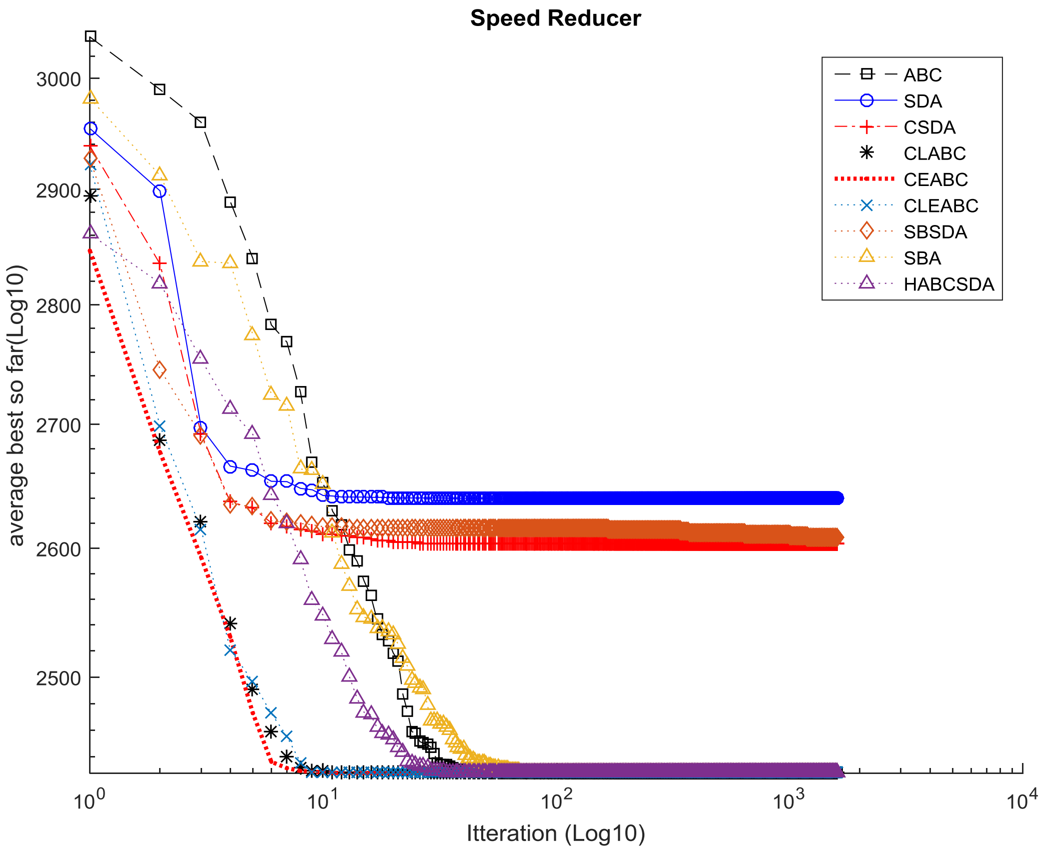


Figure 4.8:- :- Convergence plot of Speed Reducer design problems

4.3.2.3 Comparative results with other algorithms

The performances of the proposed algorithms were compared with those of other 8 algorithms (based on comparative study by Kasdirin [153]). These include teaching learning based algorithm, TLBO, Improve ant colony, IACO, coevolutionary particle swarm algorithm, CPSO, cuckoo search, CS, firefly algorithm with nonlinear spread factor, FA-NSF, firefly algorithm with exponential spread factor, FA-eSF, invasive weed optimisation with exponential spread factor, IWO-eSSF and modified invasive weed optimisation with exponential seeds, and MIWO-eSSF. The comparative results are shown in Table 4.21.

Table 4.21:- Comparative results of algorithms

	Pressure vessel		Spring design		Welded beam		Speed reducer	
	best	Std dev	best	Std dev	best	Std dev	best	Std dev
TLBO	6.06E+03	1.85E-12	1.27E-02	2.12E-06	1.73E+00	6.77E-16	2.994E+03	4.62E-13
IACO	6.06E+03	6.72E+01	1.27E-02	3.49E-05	1.73E+00	9.20E-03	NA	NA
CPSO	6.06E+03	8.65E+01	1.27E-02	5.20E-05	1.73E+00	1.29E-02	NA	NA
CS	6.06E+03	5.03E+02	NA	NA	NA	NA	3.0E+03	4.96E+00
FA-NSF	5.90E+03	2.54E+02	1.27E-02	2.37E-04	1.68E+00	1.30E-01	2.86E+03	2.66E+00
FA-eSF	5.93E+03	3.05E+02	1.27E-02	9.83E-04	1.68E+00	6.84E-02	2.86E+03	3.12E+00
IWO-eSSF	5.89E+03	2.94E+02	1.27E-02	1.42E-06	1.72E+00	1.89E-01	2.98E+03	1.64E+00
MIWO-eSSF	5.90E+03	1.62E+02	1.27E-02	3.18E-07	1.70E+00	1.70E-01	2.98E+03	2.13E+00
ABC	5.91E+03	4.42E+01	1.27E-02	1.33E-02	1.70E+00	1.79E-01	**2.43E+03**	0.00E+00
SDA	6.59E+03	6.24E+05	1.28E-02	2.42E-01	1.69E+00	8.41E-01	2.45E+03	1.75E+02
CSDA	8.02E+03	1.19E+05	1.37E-02	2.26E-01	1.73E+00	6.51E-01	2.49E+03	9.93E+01
CLABC	5.95E+03	8.52E+01	1.27E-02	1.29E-02	1.73E+00	1.72E-01	**2.43E+03**	0.00E+00
CEABC	6.03E+03	9.64E+01	1.27E-02	1.27E-02	1.75E+00	8.55E-02	**2.43E+03**	0.00E+00
CLEABC	2.31E+03	1.19E+03	1.27E-02	1.30E-02	1.71E+00	1.82E-01	**2.43E+03**	4.79E-13
SBSDA	2.31E+03	9.10E+04	1.27E-02	4.40E+16	**1.67E+00**	3.04E-01	2.45E+03	1.41E+02
SBA	**1.41E+03**	1.61E+03	1.31E-02	1.74E-02	1.72E+00	1.87E-01	**2.43E+03**	4.82E-12
HABCSDA	5.91E+03	7.39E+01	**1.27E-02**	1.30E-02	1.87E+00	1.81E-01	**2.43E+03**	0.00E+00

As noted in Table 4.21, the good features from ABC and SDA helped the proposed hybrid algorithm to reach the optimal point better than other algorithms. It is also noted that most of the proposed algorithms were better than other algorithms with good accuracy and small standard deviation. As a conclusion in this comparative study, the proposed algorithms showed potential to solve various problems with constrained parameters.

4.4 Summary

In this chapter, the performances of the proposed algorithms (CSDA, CLABC, CEABC, CLEABC, SBSDA,SBA, and HABCSDA) in comparison to those of their predecessor algorithms, ABC and SDA have been assessed. All of the algorithms have been tested and been evaluated in ten single objective standard benchmark functions and sixteen CEC2014 benchmark functions. The performance measurements have included non-parametric test, parametric test and convergence plots. The capability of the proposed algorithm in dealing with constrained problems has been tested and evaluated in ten CEC2006 benchmark functions and in five constrained engineering design.

Most of the proposed algorithms have potential to solved various unconstrained and constrained problems. The best proposed algorithm in most of the problems so far is hybrid version of SDA and ABC, HABCSDA. HABCSDA has been shown to be able to reach the best optimal point and have quick convergence in solving given problems.

Chapter 5
Multi Objective Problems

5.1 Introduction

This chapter presents the performance assessment of the proposed algorithms and their predecessor algorithms in multi-objective problems. The algorithms are used to solve two objectives problems in seven benchmark functions consisting of constrained and unconstrained problems and in one practical problem. The selected problems have various landscapes, dimensions and level of complexity. The performances of the algorithms are based on the visualization of Pareto front and three performance matrices, namely spacing, maximum spread and hyper volume.

5.2 Multi-objective optimization

The mathematical formulation for multi objective optimization problems with m objective s can be expressed as

$$\text{Minimize } f(x) = [f_1(x), f_2(x), \ldots\ldots, f_m(x)] \text{ , where } m > 1$$

$$\text{Subject to}$$

$$g_k(x) \geq 0, \quad k = 1,2, \ldots., K$$

$$h_{kj}(x) \geq 0, \quad kj = 1,2, \ldots., L$$

where x represents parameter variables in a parameter space. g_k is function of inequality constraint, h_{kj} is function of equality of constraint, K is number of inequality constraints, and L is number of equality constraints.

Multi-objective problems are difficult to solve. The reason is that optimal solutions are not unique and there will be tread-off between all objectives, where some will win and some will lose. Favourable solutions are non-dominated solutions, referred to as Pareto front optimal points [165]. Thus, the Pareto set forms the most suitable optimal solutions. In the tests here, the Pareto optimality and Pareto dominance methods are used to assess solutions of the multi-objective problems. An objective function vector $\vec{x}^*$ can be said as a non-dominated objective vector $f(\vec{x}^*)$ when no other solution, $\vec{x}$ enable to improve at least for one objective function without degrading/affecting other objectives. This set of non-dominated solutions can also be called as a Pareto optimal set and the set creates a Pareto front. For a dominated solution vector, the solution $\vec{x}^{**}$ dominates solution $\vec{x}$ where $f(\vec{x}^{**})$ is better than $f(\vec{x})$ for at least one objective function or for all objective functions.

5.3 Multi-objective approach

There are many techniques available in solving more than one objective problem. The approaches are divided into two categories; (1) Pareto based and (2) Non-Pareto based. Pareto based methods such as non-dominated sorting (NSGA-I & NSGA-II) and niched Pareto (NPGA) in genetic algorithms and simple evolutionary multi-objective (SEMO) are widely used to solve multi-objective problems. Pareto based approaches are slow convergence to the optimal Pareto front, and do not provide information between two solutions. Non-Pareto based approaches are known able to tackle multi-objective problems [166]. Weighted sum approach, aggregation method, vector evaluated, e-constrained method are examples of non-Pareto based approaches for solving multi objectives problems. Basically, a non-Pareto approach converts the objective into scalar objective function. This method is easy to use, provides good selection in Pareto front [167][168][169] and works efficiently with local search methods [170][171].

In the tests carried out here, the weight sum method is used. This method is suitable to use and is easy to implement in solving multi-objective problems [172]. This approach has been successfully implemented by researchers for solving multi-objective optimization problems [153][173]. The method is mathematically formulated as

$$F = \sum_{p=1}^{k} \omega_p F_p$$

where ω_p is real-valued weighting $(\omega_p \geq 0)$ and $\sum \omega_p = 1$.

5.4 Multi-objective benchmark problems

The algorithms are tested in seven multi-objective benchmark functions and in one practical multi-objective problem. The summary of the benchmark problems is shown in Table 5.1. The corresponding mathematical formulations and constraints are shown in appendix A.

Table 5.1:- Summary of the multi-objective benchmark functions

Function		Problems types	Search domain
MO_1	SCH 1 [124]	Unconstrained problems	$-10 \leq x \leq 10$
MO_2	Kursawe	Unconstrained problems	$-5 \leq x_i \leq 5$ $0 \leq i \leq 3$
MO_3	CTP 1	Constrained problems	$0 \leq x_1 \leq 1$ $0 \leq x_2 \leq 1$
MO_4	Constr-Ex	Constrained problems	$0.1 \leq x_1 \leq 1$ $0 \leq x_2 \leq 5$
MO_5	Binh and Korn	Constrained problems	$0 \leq x_1 \leq 5$ $0 \leq x_2 \leq 3$
MO_6	Chankong and Haimes	Constrained problems	$-20 \leq x_1 \leq 20$ $-20 \leq x_2 \leq 20$
MO_7	Osyczka and Kundu	Constrained problems	$0 \leq x_1, x_2, x_6 \leq 10$ $1 \leq x_3, x_5 \leq 5$ $0 \leq x_5 \leq 6$
MO_8	Four bar plane truss	Practical problems	Refer to table 5-2

5.5 Performance assessment

Two types of performance assessment are carried out; qualitative and quantitative. The first one is qualitative assessment is done through visualization of the Pareto front. The quantitative assessments used comprise three performance matric, namely hypervolume, spacing and maximum spread [174][175], where the true Pareto set is considered unknown. The aim of these assessments is to establish how well the non-dominated points are distributed along the Pareto front (able to cover most of the objectives). These assessments are also used for convergence performance of the algorithms, where the closeness of the Pareto front approximation to the true Pareto front is measured. The spacing and maximum spread are used to assess the coverage/diversity of the non-dominated point along the Pareto front while hypervolume test enable to evaluate the convergence and diversity of Pareto optimal points along the Pareto front.

Spacing, S – this method was developed by Shotkey in 1995 [176] , and enables to evaluate how well (diversity) the non-dominated solutions are distributed within the approximation Pareto front. Basically, S is a Euclidean distance between adjacent solutions in the approximation Pareto front [177] and is mathematically formulated as

$$S = \sqrt{\frac{1}{n+1}\sum\nolimits_{i=1}^{n}\left(\overline{d} - d_i\right)^2}$$

where $\overline{d}$ is the average distance between adjacent solutions, $d_i = \sum_{L=1}^{k}\left|f_L^i - f_L^j\right|, (i, j = 1,2, \dots, k)$, f_L^i & f_L^j is Lth objective of ith & jth solutions, n is number of non-dominated points distributed along the Pareto front. The aim is to get the smallest spacing. Smallest value of S indicates that the non-dominated solutions are distributed well on the Pareto front [178]. If the non-dominated solutions are equally distributed, the spacing value should be zero.

Maximum spread ($\boldsymbol{MaxS}$)- this performance matric was proposed by Zitzler [179]. The spread is required to be maximum, where each of the objective values should be within maximum and minimum Pareto front points. Mathematical formulations of MaxS is given as

$$MaxS = \sqrt{\sum_{i=1}^{k} \max d\,(a_i, b_i)}$$

where, i is an objective function, a_i and b_i represent maximum and minimum values for $i-th$ objective and d is Euclidean distance. To ensure the Pareto front is completely covered the reference front MaxS should converge from zero to one [153] [179].

Hypervolume ($\boldsymbol{HV}$) –HV is a popular performance measurement for multiobjective problems. HV can evaluate the convergence and diversity criterion of Pareto set. HV is determined by calculating the volume of the objective area dominated by Pareto optimal points and reference point,r. Larger HV indicates that the Pareto set is better in terms of objective tread-offs. An example and detailed procedure of measurement of HV has been presented by Lucas [180] and Auger et. el. [174][181]. Figure 5.1 shows the computation of HV for solution and reference point,r in the objective space (adopted from [174]).

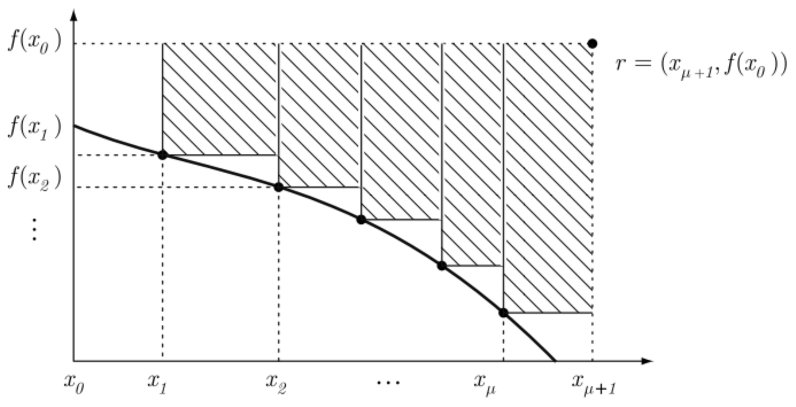


Figure 5- 1:-The computation of HV for u- solution and reference point,r in objective space (adopted from [174]*)*

5.6 Experimentation

The initial parameter settings used for the algorithms are shown in Table 5.2. The population size was set to 30 and the number of iterations to 1000. For HV measurement, the reference point needs to be set for each problem. Table 5.4 shows the HV reference point.

Table 5.2:- Multi-objective algorithms initial parameters for all algorithms

	Algorithms								
Parameters	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
r	-	0.95	0.95	-	-	-	0.95	0.95	0.95
θ	-	45^0	45^0	-	-	-	45^0	45^0	45^0
m	-	30	30	-	-		30	30	30
Limit	100	-	-	100	100	100	100		100
NF	30	-	-	30	30	30	30		30
itter	1000	1000	1000	1000	1000	1000	1000	1000	1000
$x(0)_{chao}$	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
Δ_a	-	-	-	1	1	1	-	-	-
a_2	-	-	-	0.5	0.5	0.5	-	-	-
b_1	-	-	-	0.5	0.5	0.5	-	-	-
b_2	-	-	-	0.8	0.8	0.8	-	-	-

Table 5.3:- HV reference points

Fcn	**Reference point**
MO_1	[10 9]
MO_2	[-6 8]
MO_3	[1.2 1.2]
MO_4	[1.2 1.2]
MO_5	[140 60]
MO_6	[900 100]
MO_7	[10 410]
MO_8	[1400 0.05]

Figures 5.2 to 5.9 show the Pareto sets obtained for the benchmark problems. Table 5.4 shows the performance assessment results where the bold font indicates the best value in the individual assessment metric.

Table 5.4:- Performance results of the algorithms

FCN		ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
MO_1	HV	1.26E-01	**1.43E-01**	1.32E-01	1.26E-01	1.26E-01	1.26E-01	1.30E-01	1.26E-01	1.29E-01
	MS	1.78E-03	1.85E-03	4.29E-03	1.78E-03	1.78E-03	1.78E-03	5.23E-03	1.78E-03	**9.38E-03**
	SP	4.16E+01	5.81E+01	4.97E+01	4.16E+01	4.16E+01	4.16E+01	4.43E+01	4.16E+01	**3.33E+01**
MO_2	HV	**1.77E+02**	1.71E+02	1.54E+02	**1.77E+02**	**1.77E+02**	**1.77E+02**	1.71E+02	**1.77E+02**	1.71E+02
	MS	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	SP	**0.00E+00**	2.36E+00	4.95E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.79E+01	**0.00E+00**	**0.00E+00**
MO_3	HV	5.94E-01	2.61E-01	2.71E-01	5.66E-01	5.40E-01	5.66E-01	2.68E-01	5.94E-01	**8.67E-01**
	MS	8.36E-01	8.34E-01	8.36E-01	8.36E-01	8.36E-01	8.36E-01	**8.26E-01**	8.36E-01	8.36E-01
	SP	1.78E-02	**9.23E-03**	1.53E-02	1.94E-02	2.07E-02	1.94E-02	1.73E-02	1.78E-02	2.39E-02
MO_4	HV	9.82E+00	9.81E+00	9.39E+00	9.82E+00	9.82E+00	9.82E+00	9.86E+00	9.82E+00	**1.08E+01**
	MS	5.86E-01	**5.82E-01**	5.95E-01	5.86E-01	5.86E-01	5.86E-01	5.90E-01	5.86E-01	5.86E-01
	SP	4.18E-02	5.44E-02	3.88E-02	4.18E-02	4.18E-02	4.18E-02	**3.28E-02**	4.18E-02	3.36E-02
MO_5	HV	2.16E+03	1.87E+03	1.95E+03	2.16E+03	2.16E+03	2.16E+03	**2.46E+03**	2.16E+03	2.08E+03
	MS	**4.51E-02**	1.57E-01	1.50E-01	**4.51E-02**	**4.51E-02**	**4.51E-02**	5.57E-02	**4.51E-02**	4.52E-02
	SP	1.09E-02	1.29E-02	1.22E-02	1.09E-02	1.09E-02	1.09E-02	**9.56E-03**	1.09E-02	1.16E-02
MO_6	HV	**1.75E+05**	5.07E+04	5.03E+04	**1.75E+05**	**1.75E+05**	**1.75E+05**	5.89E+04	**1.75E+05**	6.82E+04
	MS	2.88E-02	4.84E-02	4.97E-02	2.88E-02	2.88E-02	2.88E-02	**2.74E-02**	2.88E-02	3.05E-02
	SP	1.49E-01	**1.06E-01**	1.71E-01	1.49E-01	1.49E-01	1.49E-01	1.24E-01	1.49E-01	1.53E-01
MO_7	HV	2.88E+05	3.35E+05	2.26E+05	**4.45E+05**	**4.45E+05**	**4.45E+05**	2.20E+05	**4.45E+05**	2.86E+05
	MS	**7.07E-01**	7.13E-01	7.10E-01	**7.07E-01**	**7.07E-01**	**7.07E-01**	**7.07E-01**	**7.07E-01**	**7.07E-01**
	SP	5.07E+00	**2.11E-01**	3.87E-01	5.47E+00	5.47E+00	5.47E+00	4.39E-01	6.04E+00	2.09E+00
MO_8	HV	2.26E+00	3.45E+00	**3.51E+00**	2.26E+00	2.26E+00	2.26E+00	2.93E+00	2.26E+00	2.26E+00
	MS	7.07E-01	**6.42E-01**	6.46E-01	7.07E-01	7.07E-01	7.07E-01	6.43E-01	7.07E-01	7.07E-01
	SP	**0.00E+00**	4.62E-03	3.89E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.80E-03	**0.00E+00**	**0.00E+00**

As noted in Table 5.4, for MO1, HABCSDA performed very well with maximum spread and space between non-dominated solutions, where it had a good distribution along the Pareto front, while SDA showed good convergence as it achieved highest HV. Figure 5.2 shows that solutions of the algorithms were distributed well along the Pareto front for function

MO_1 except with those of SDA and CSDA, where the SDA and CSDA solutions were quite far from the Pareto front. They appear to have failed to solve this simple multi-objective problem due to the spiral trajectory.

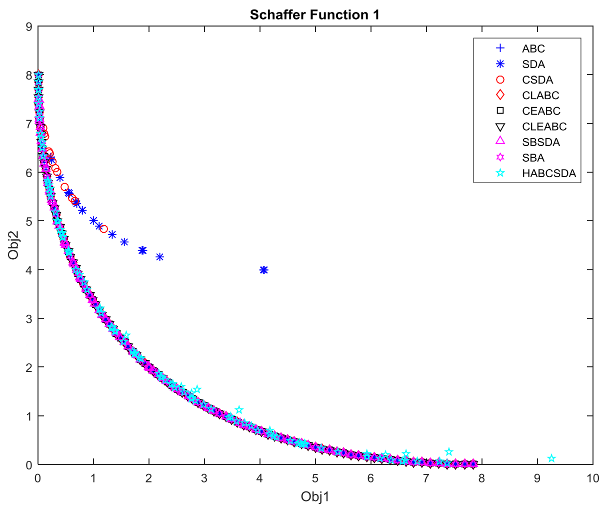


Figure 5.2:- Pareto set of algorithms for $\boldsymbol{MO_1}$

For MO2, algorithms based on ABC algorithm achieved good convergence and coverage with zero spacing, large HV and small maximum spread within the Pareto front. Zero implies that the non-dominated solutions are distributed equally along the Pareto front. Solutions achieved with SDA,CSDA and SBSDA were scattered quite far from the Pareto front. The distribution of solutions for all algorithms for MO2 can be seen in Figure 5.3.

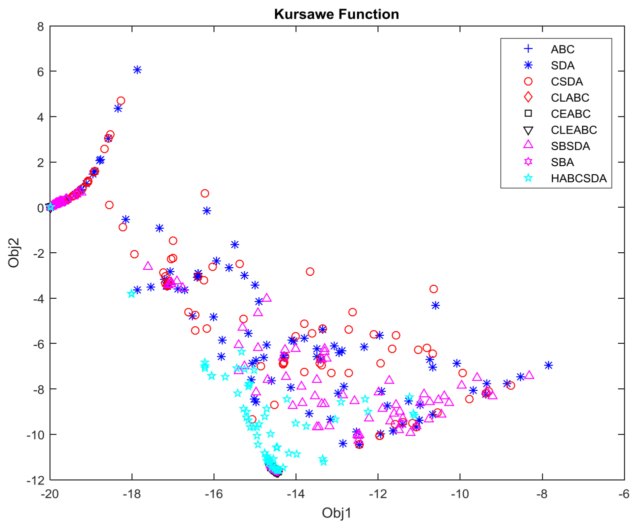


Figure 5.3:- Pareto set of algorithms for $\boldsymbol{MO_2}$

Figure 5.4 shows the Pareto set for MO3. As noted solutions with most of the algorithms were distributed along the Pareto front but the spiral movement for SDA, CSDA and SBA could not lead the solutions to the Pareto front. Although it not scattered far from the Pareto front, the solution are grouped at the right end of the Pareto front (obj1=1) and based on the numerical results in Table 5.4, SBSDA and SDA has good coverage but do not have a good convergence. Based on the HV values the HABCSDA had a good convergence.

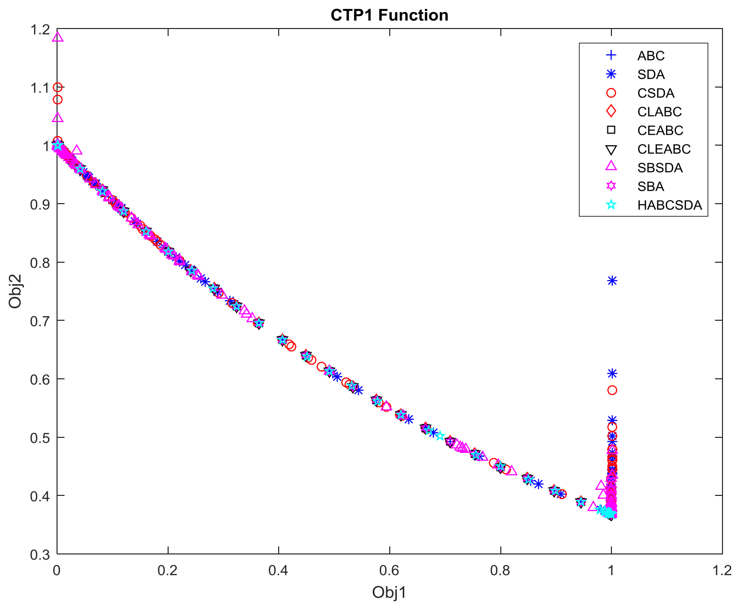


Figure 5.4:- Pareto set of algorithms for MO_3

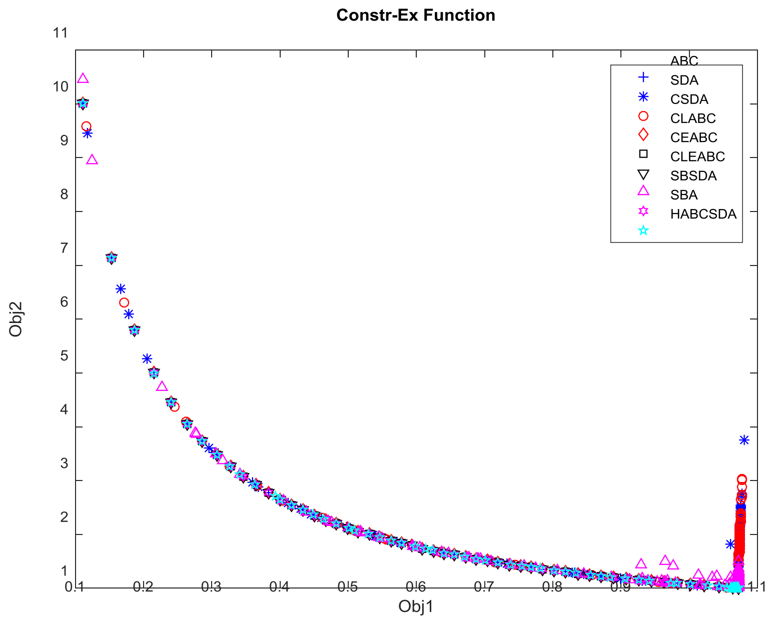


Figure 5.5:- Pareto set of algorithms for MO_4

Based on the results in Figure 5.5 and Table 5.4, SDA and SBSDA with small spacing and maximum of spread achieved non-dominated solutions distributed well along the Pareto front. HABCSDA achieved good convergence with the highest HV. Similar to MO3, the dominated solution achieved with SDA based algorithms in solving MO4 were scattered at the right end of the Pareto front. Figure 5.6 and Figure 5.7 show the Pareto set of MO5 and MO6 respectively. As noted, for MO5, the solution were distributed well but some of the solutions were pushed away from the Pareto front while for MO6 the solutions from SDA ,SBSDA and HABCSDA scattered near to the Pareto front while solutions from other algorithms were distributed well along the Pareto front. Based on the numerical results most of the proposed algorithms except SDA, CSDA and HABCSDA performed well for MO5 and MO6.

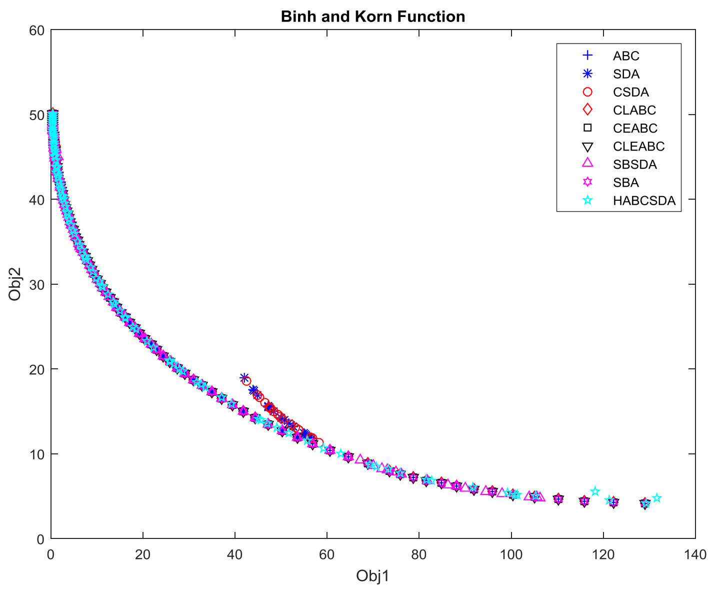


Figure 5.6:- Pareto set of algorithms for $\boldsymbol{MO_5}$

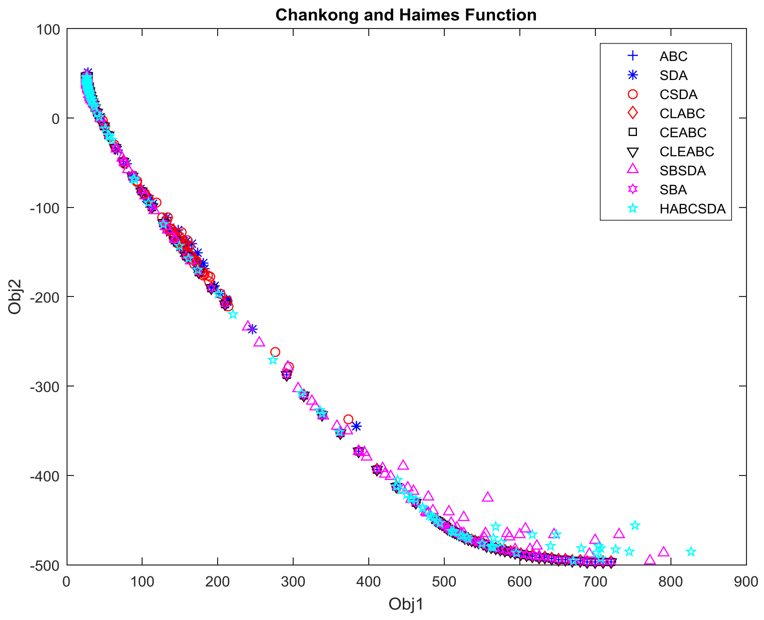


Figure 5.7:- Pareto set of algorithms for $\boldsymbol{MO_6}$

MO7 is a constrained problem and hard to solve. The Pareto set of MO7 can be seen in Figure 5.8. Based on the numerical results, the non-dominated solutions from SBA,CLEABC, CLABC and CEABC showed good diversity and good convergence. The chaotic maps could not resolve the trapping issue of algorithm at local optima and was not able to lead the CSDA to perform well in this kind of problem.

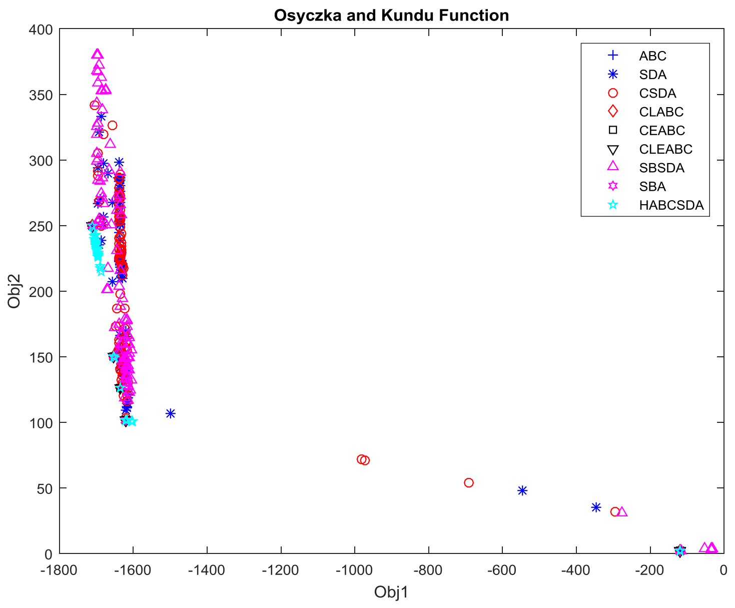


Figure 5.8:- Pareto set of algorithms for $\boldsymbol{MO_7}$

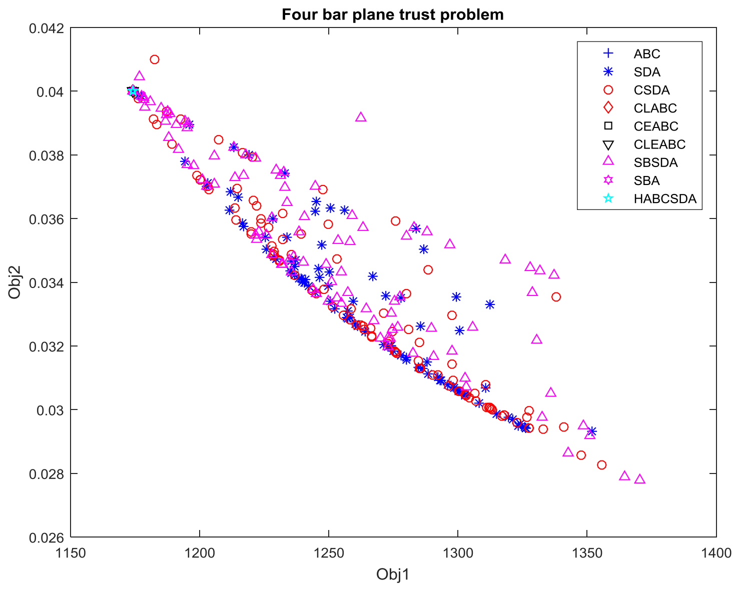


Figure 5.9:- Pareto set of algorithms for $\boldsymbol{MO_8}$

MO8 is the practical multi-objective problem. As noted in Figure 5.9, most of the solutions were scattered behind the Pareto front and the numerical results in Table 5.4 show that the spacings for ABC, CEABC, CLABC, CEABC,SBA and HABCSDA were distributed equally. The CSDA provided good convergence. It is seen that, the proposed algorithms outperformed the original algorithms in solving MO8 problem.

5.7 Summary

The proposed algorithms were tested in eight multi-objective problems. The aim of the tests were to compare the performance of the proposed algorithms with those of SDA and ABC. It has been shown through visualization of Pareto sets distribution and numerical results that the proposed algorithms have outperformed the original algorithms in most of the test problems. Thus, it can be concluded that, the proposed algorithms are able to find best non-dominated solutions with good diversity and convergence in solving multi objective problems.

Chapter 6
Engineering Applications

6.1 Introduction

The performances of the proposed algorithms were tested in chapters 4 and 5 using benchmark functions and the results showed that they outperformed the original SDA and ABC. In this chapter, the proposed algorithms are tested in four engineering/practical applications. The first of these is tuning of proportional derivative (PD) fuzzy controller for set-point tracking of hub angle of a single link manipulator system. The second application is minimization of shipping refined oil cost to Japan. The third application is to find the optimal number of flat television sets needed to sell in order to get a maximum profit margin. The fourth application is to find the minimum weight of the car in car impact designs. The applications considered have different levels of complexity and have received noticeable attention from researchers. However, SDA and ABC based algorithms have not been used in these applications. Thus, the test results of the proposed algorithms in these applications will add to current body of knowledge.

6.2 Control of flexible manipulator system

In this section, the proposed algorithm and its predecessor algorithms are used to optimize PD fuzzy controller of a single link flexible manipulator system (SLMS). The SLMS model used in this experiment has been developed by Azad [182] and Poerwanto [183] . Figure 6.1 shows a schematic representation of the SLMS rig. Detailed description the SLMS development can be found in several previous works [46] [134][184][185][186].

While conventional proportional, integral, derivative (PID) controllers have gained popularity in large sectors of industrial applications, their tuning could be tedious and time consuming. Moreover, conventional approaches of PID tuning may not yield optimal design. In case of more complex control paradigms, such as fuzzy logic and neural networks the design process become more complex.

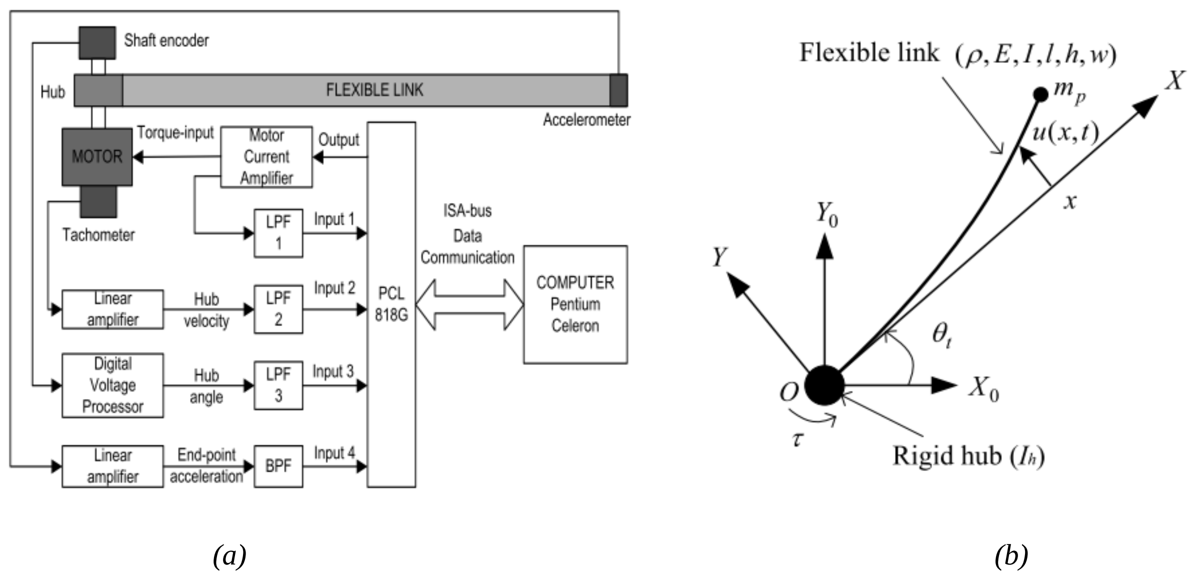


Figure 6.1: Schematic diagram(a) and schematic representation (b) of the SLMS rig (Adopted from Azad [182]*)*

Optimization algorithms have been shown as viable candidates for optimal design of controllers to their desired specifications. Kasdirin [153] has used firefly and invasive weeds algorithms to obtain optimised twenty five membership function (MF) PD- like fuzzy controller for SLMS. Supriyono [187] used bacterial foraging algorithm (BFA) to tune the PD controller for input tracking SLMS. Nasir et al. [188] used hybrid BFA and SDA to design the controller of SLMS. Adaptive BFA [189] and genetic algorithm [190][191] have been used to tune the SLMS controller. In this experiment, the main focus is to control the hub angle using PD controller and PD-like fuzzy controller. The proposed algorithms, SDA and ABC are used to tune the controller parameters.

The mathematical formulation of PID controller is:-

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$$

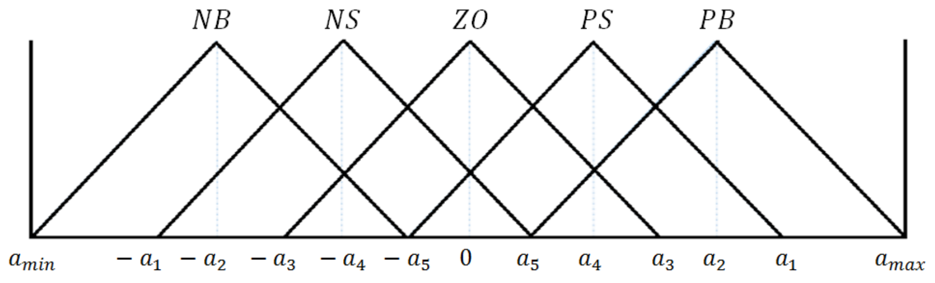
where $e(t)$ is the error. K_P, K_I and K_D are the proportional, integral and derivative gain that need to be tuned for a desired system performance. There are two types of methods commonly used for tuning PID controllers; (a) manual tuning, via heuristic trial and error approach, which proves to be tedious and depends on experience. (b) The Ziegler-Nichols method, which is based on known mathematical model of the system to be controlled. Table 6.1 shows the generic effects of PID gains on system response.

Table 6.1:- Effect of PID gains on system response

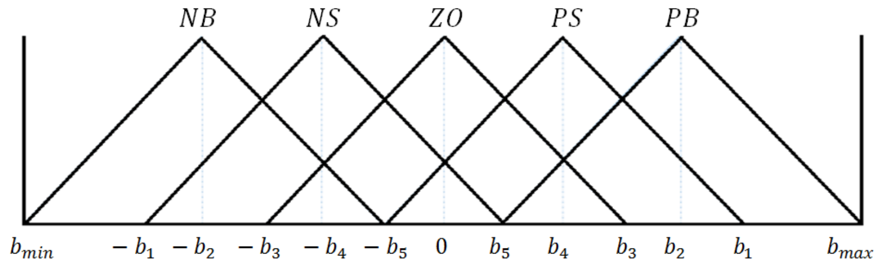
Parameter	K_P	K_I	K_D
$Rise\ time, T_r$	Decrease	Decrease	Small decrease
$Settling\ time, T_s$	Small change	Increase	Small decrease
$Steady\ state\ error, ess$	Decrease	Big Decrease	No change
$Overshoot, \%OS$	Increase	Increase	Small decrease

Although PID is a common and popular controller in many applications, due to the nonlinearity of SLMS, PD-fuzzy logic control is a more effective control mechanism for achieving accurate hub angle trajectory [153]. Thus in this experiment, the PD-Fuzzy logic control is used for SLMS. The setting and structure of PD-fuzzy controller in this experiment are based on [153].

As shown in Figure 6.2, the value of a_i and b_i, $i = 1,2 \dots 5$ is the fuzzy parameters need to be optimized by the proposed algorithms. In total, ten (10) values of fuzzy parameters together with the derivative gain K_D and proportional gain K_P need to be tune in order to get accurate trajectory of SLMS.



(a)



(b)

Figure 6.2: Fuzzy input (a) error $\boldsymbol{e(k)}$ (b) Change of error, $\Delta\boldsymbol{e(k)}$

The fuzzy logic controller with a Takagi-Sugeno fuzzy inference system (FIS) is used here. As shown in Figure 6.3, the fuzzy logic controller has 2 inputs and 1 output. The error, $e(k)$ and change of error, $\Delta e(k)$ are the inputs. The input signal to the SLMS can be representing as u_k. The $e(k)$ is the difference between reference input and hub angle. While $\Delta e(k)$ is change of error and can be formulated as $\Delta e(k) = e(k) - e(k-1)$, where k is sample number. The range of the input should be between [-1, 1]. The membership function type used in this experiment is triangle type. Five fuzzy rules are defined in each input and the output will have twenty-five fuzzy rules. The fuzzy rules base is shown in Table 6.2. To ensure the input signal of SLMS is sufficient enough, the fuzzy scaling factor K is set to 500. The constant values output of fuzzy rules are set as: - NB=-1, NS =-0.5, ZO=0, PS = 0.5 and PB=1 and the FIS output range is [0, 1]. The integral absolute error is used as an objective function. Thus the objective function is:-

$$min\ f = \int |e(k)|$$

where, the error $e(k)$ is the difference between set point angular position and actual angular position.

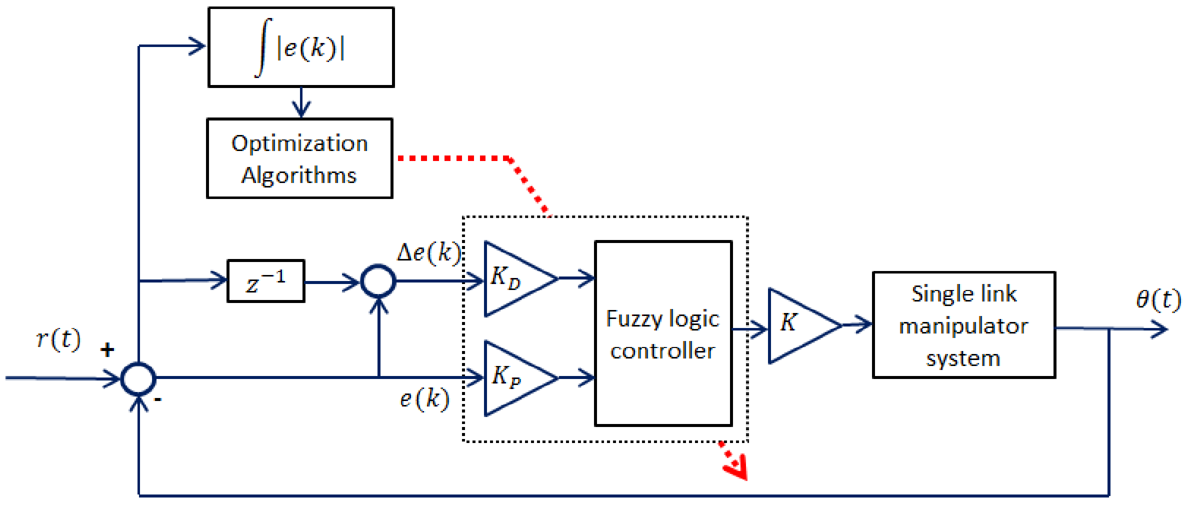


Figure 6.3:- PD fuzzy control system

Table 6.2:- Fuzzy rules

$e(k)$ \ $\Delta e(k)$	NB	NS	ZO	PS	PB
NB	PB	PB	PB	PS	ZO
NS	PB	PS	PS	ZO	NS
ZO	PS	ZO	ZO	ZO	NS
PS	PS	ZO	NS	NS	NB
PB	ZO	NS	NB	NB	NB

The reference input to SLMS used here is a bang-bang signal with positive step at 50 degrees and negative step -50 degrees and duty cycle 12 seconds. There are 12 parameters that need to be optimised. 10 from membership function values and 2 controller gains K_P and K_D. Table 6.3 and 6.4 show the system response parameters achieved, where subscript 1 and 2 refer to first and second cycles of the bang-bang input applied. Table 6.5 shows results of membership parameters and gains. Figure 6.4 shows the system response with the optimised controller, and Figure 6.5 shows the convergence plots of the algorithms.

It is noted in Table 6.3 and 6.4 that the system response with CLEABC tuned controller reached faster to the 50 degrees hub angle with zero overshoot and to -50 degrees with zero undershoot, and the time taken to settle at 0 degree was 578 ms. HABCSDA tuned controller, among the hybrid optimisation algorithms, achieved the fastest system response. Most of the proposed algorithms achieved low system response overshoot and undershoot. The CSDA tuned controller performed the worst with slow and high system response overshoot.

Table 6.3:- Numerical result of time domain response of SLMS hub angle for non-hybrid algorithms

Parameters	ABC	SDA	CSDA	CLABC	CEABC	CLEABC
Settling time (s), T_{s1}	1.179	1.518	1.018	893.649m	**743.137m**	1.139
Rise time (s), T_{r1}	409.807m	476.541m	441.718m	399.649	462.752m	**389.132m**
Overshoot, % $OS1$	**0**	12.82	2.32	3.10	2.36	**0**
Undershoot, % $US1$	**0**	49.605	0	48.993	**49.702**	**0**
Settling time (s), T_{s2}	1.545	1.718	3.957	1.192	1.260	**1.1220**
Rise time (s), T_{r2}	603.009m	673.094m	**578.440m**	631.025	662.577m	**578.440m**
Overshoot, % $OS2$	7.32	26.17	6.18	7.83	5.54	16.66
Undershoot, % $US2$	2.60	1.36	0.70	1.48	**0.48**	1.78
Settling time (s), T_{s2}	0	1.593	1.451	**718.907m**	798.463m	1.137
Rise time (s), T_{r2}	441.718m	483.786	694.128m	431.201	483.786	**399.649m**
Overshoot, % $OS2$	0	5.36	0	0.88	0.68	**0**
Undershoot, % $US2$	0	0.4	0	0.5	0.24	**0**

Table 6.4:- Numerical result of time domain response of SLMS hub angle for hybrid algorithms

Parameters	**SBSDA**	**SBA**	**HABCSDA**
Settling time (s), T_{s1}	1.168	1.079	1.105
Rise time (s), T_{r1}	389.132m	378.615m	**389.132**
Overshoot, % $OS1$	0.14	0.72	0.30
Undershoot, % $US1$	48.130	49.219	49.319
Settling time (s), T_{s2}	1.194	1.633	1.603
Rise time (s), T_{r2}	599.474m	578.440m	**578.440m**
Overshoot, % $OS2$	11.65	16.26	12.41
Undershoot, % $US2$	2.32	0.94	2.50
Settling time (s), T_{s2}	1.151	1.091	1.112
Rise time (s), T_{r2}	420.684m	431.201m	452.235m
Overshoot, % $OS2$	**0**	**0**	**0**
Undershoot, % $US2$	**0**	**0**	**0**

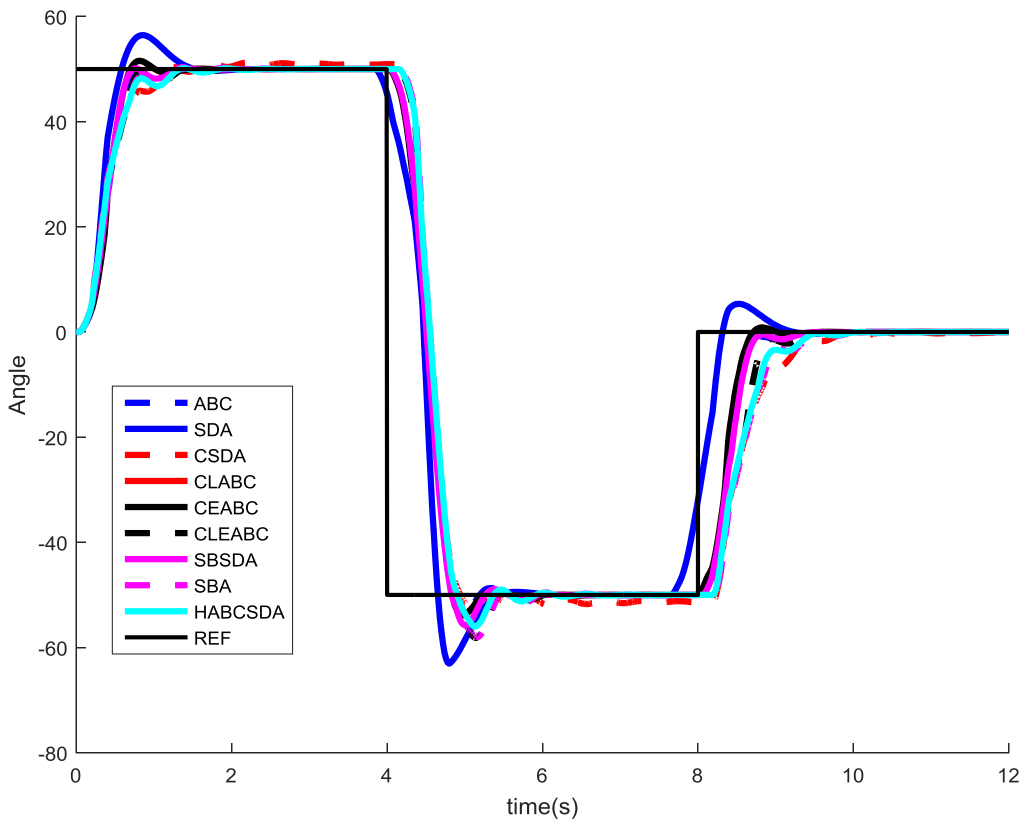


Figure 6.4:- Hub angle response of SLMS

Figure 6.5 shows the convergence plots of the algorithms for control optimisation of the SLMS. The first algorithm that reached the best optimum point in this problem was CLABC. Although CLEABC started from a distant point, it reached the optimum point in less than 15 iterations. Most of the proposed algorithms reached nearest to the best point except CSDA.

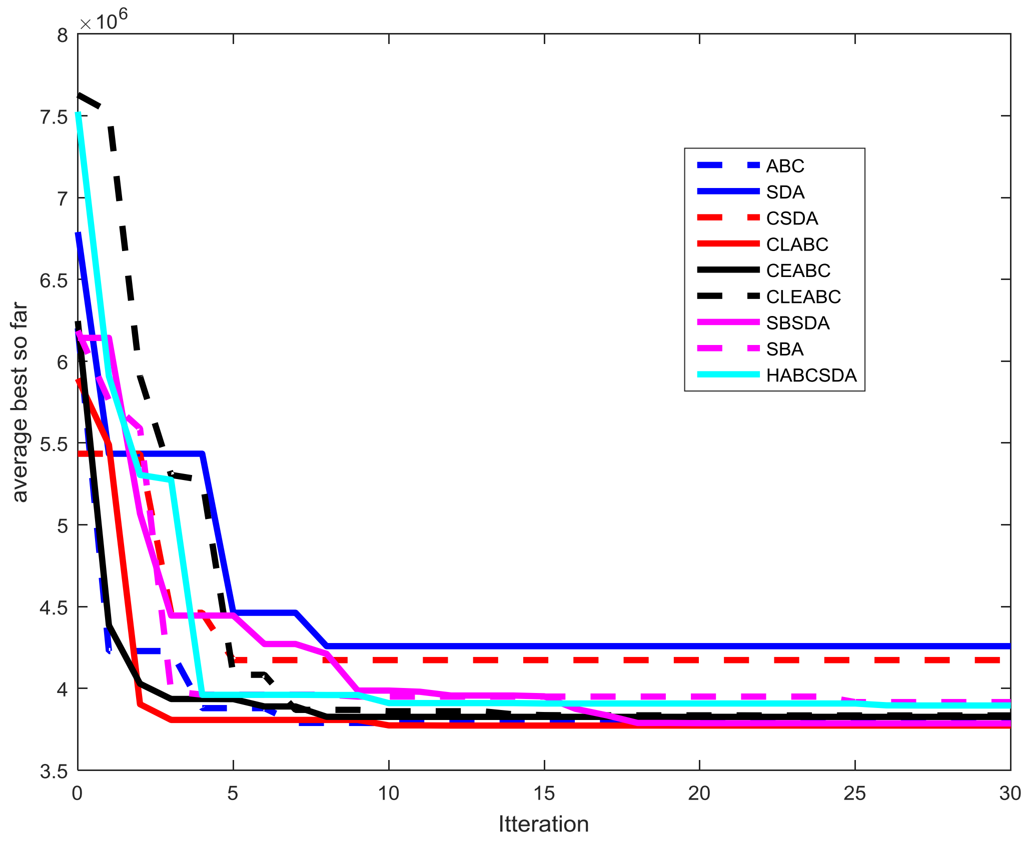


Figure 6.5:- Convergence plot algorithms for SLMS controller design

Table 6.5:- Results of optimum parameter of membership function and $\boldsymbol{K_p}$ *and* $\boldsymbol{K_D}$ *gain.*

Algorithm	**Error, $e(k)$**		**Change of Error, $\Delta e(k)$**		K_P	K_D	$f(x)$
ABC	a_1	0.4877	b_1	0.8771	0.0213	0.1134	3.79E+06
	a_2	0.7687	b_2	0.4967			
	a_3	0.6083	b_3	0.2271			
	a_4	0.1888	b_4	0.2955			
	a_5	0.3898	b_5	0.5937			
SDA	a_1	0.5010	b_1	0.3265	0.0434	0.0788	4.26E+06
	a_2	0.7203	b_2	0.6700			
	a_3	0.1795	b_3	0.1583			
	a_4	0.0741	b_4	0.4252			
	a_5	0.7197	b_5	0.0662			
CSDA	a_1	0.4263	b_1	0.7792	0.0893	0.3924	4.17E+06
	a_2	0.0528	b_2	0.7399			
	a_3	0.3771	b_3	0.9197			
	a_4	0.3031	b_4	0.1489			
	a_5	0.5606	b_5	0.4848			
CLABC	a_1	0.1213	b_1	0.4727	0.0997	0.1833	3.77E+06
	a_2	0.0613	b_2	0.5624			
	a_3	0.9630	b_3	1.0000			
	a_4	0.0685	b_4	0.6417			
	a_5	0.9197	b_5	0.7665			
CEABC	a_1	0.0331	b_1	0.2462	0.0605	0.1314	3.83E+06
	a_2	0.6517	b_2	0.4889			
	a_3	0.7151	b_3	0.5143			
	a_4	0.8238	b_4	0.7884			
	a_5	0.3152	b_5	0.0000			
CLEABC	a_1	0.6643	b_1	0.0149	0.0808	0.1651	3.83E+06
	a_2	0.4146	b_2	0.0000			
	a_3	0.3501	b_3	0.4594			
	a_4	0.3237	b_4	1.0000			
	a_5	0.9398	b_5	0.1101			
SBSDA	a_1	0.9152	b_1	0.4819	0.1387	0.9997	3.79E+06
	a_2	0.2064	b_2	0.6520			
	a_3	0.8864	b_3	0.4435			
	a_4	0.9101	b_4	0.4731			
	a_5	0.5142	b_5	0.2677			
SBA	a_1	1.0000	b_1	0.4276	0.1383	0.9901	3.92E+06
	a_2	1.0000	b_2	1.0000			
	a_3	0.0586	b_3	0.7431			
	a_4	1.0000	b_4	0.0511			
	a_5	0.8971	b_5	0.5275			
HABCSDA	a_1	0.5924	b_1	0.5677	0.0213	0.1134	3.89E+06
	a_2	0.0274	b_2	0.7706			
	a_3	0.6602	b_3	0.1895			
	a_4	0.1714	b_4	0.1073			
	a_5	0.5502	b_5	0.2138			

6.3 Transportation cost of refined oil via Malacca Straits

This objective in this problem is to reduce the cost of transportation of refined oil to Japan via Malacca Straits. This problems is based on work by Edgar [192]. The shipping cost is in dollar per kilolitre ($/kL). The algorithms used in this experiment will find the best value of tanker size (x_1) and refinary capacity (x_2) in order to get the best cost of shipping. There are additional cost needs to be considered and need to be include in total cost of shipping. There are customs charges, loading and unloading fees, sea berth cost, piping cost, warehouse cost, crude oil cost, insurance fees, interest rate cost and freight cost. Some of these costs are calculated fix per year. The mathematical formulation of the cost of transportation of refined oil via sea is given as

$$f(x) = C_c + C_i + C_x + \frac{2.09e^4(x_1)^{-0.3017}}{360} + \frac{1.064e^6 a(x_1)^{0.4925}}{52.47(x_2)(360)} + \frac{0.1049(x_1)^{0.671}}{360}$$

$$+ \frac{4.242e^4 a(x_1)^{0.7952}}{360} + \frac{1.813ip[n(x_1) + 1.2(x_2)]^{0.861}}{52.47(x_2)(360)} + \frac{5.042e^3(x_2)^{-0.1899}}{360}$$

$$+ \frac{4.25e^3[n(x_1) + 1.2(x_2)]}{52.47(x_2)(360)}$$

where,

Yearly fix cost, $a = 0.20$, crude oil price, $C_c = \frac{\$12.50}{kL}$, insurance fees, $C_i = \frac{\$0.50}{kL}$, custome fees, $C_x = \frac{\$0.90}{kL}$, interest rate, $i = 10\%$, number of ports, $i = 2$, land cost, $p = \frac{\$700}{m^2}$,

$$x_1, x_2 \geq 0$$

Table 6.6 shows that after 30 time runs, all the algorithms reduced the shipping cost of refined oil to Japan via Malacca Straits by $425.85/*kL* with the best value of tanker size (x_1) of 4.39E-03 *dwt* and refinery capacity (x_2) of 1E12 *bbl/day*. The deviation from average for ABC, CLABC, CEABC, CLEABC and HABCSDA was the smallest. This implies that the cost variation at the end of run was small and thus can be said those algorithms were stable in producing the cost.

The convergence plots in Figure 6.6 show that SDA,CSDA, SBSDA did not reach the optimum point and the apparently trapped at local optima. The CLABC was the fastest to reach the optimum cost in 150 iterations, while HABCSDA needed 171 iterations to reach the optimum point.

Table 6.6:- Statistical results of shipping cost of refined oil problems

Param	**ABC**	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
Best	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02
Avg	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02	4.26E+02
SD	2.31E-13	7.60E+11	1.11E+09	2.31E-13	2.31E-13	2.31E-13	3.87E+11	1.81E-01	2.31E-13
Worse	4.26E+02	2.97E+12	4.41E+09	4.26E+02	4.26E+02	4.26E+02	1.91E+12	4.27E+02	4.26E+02

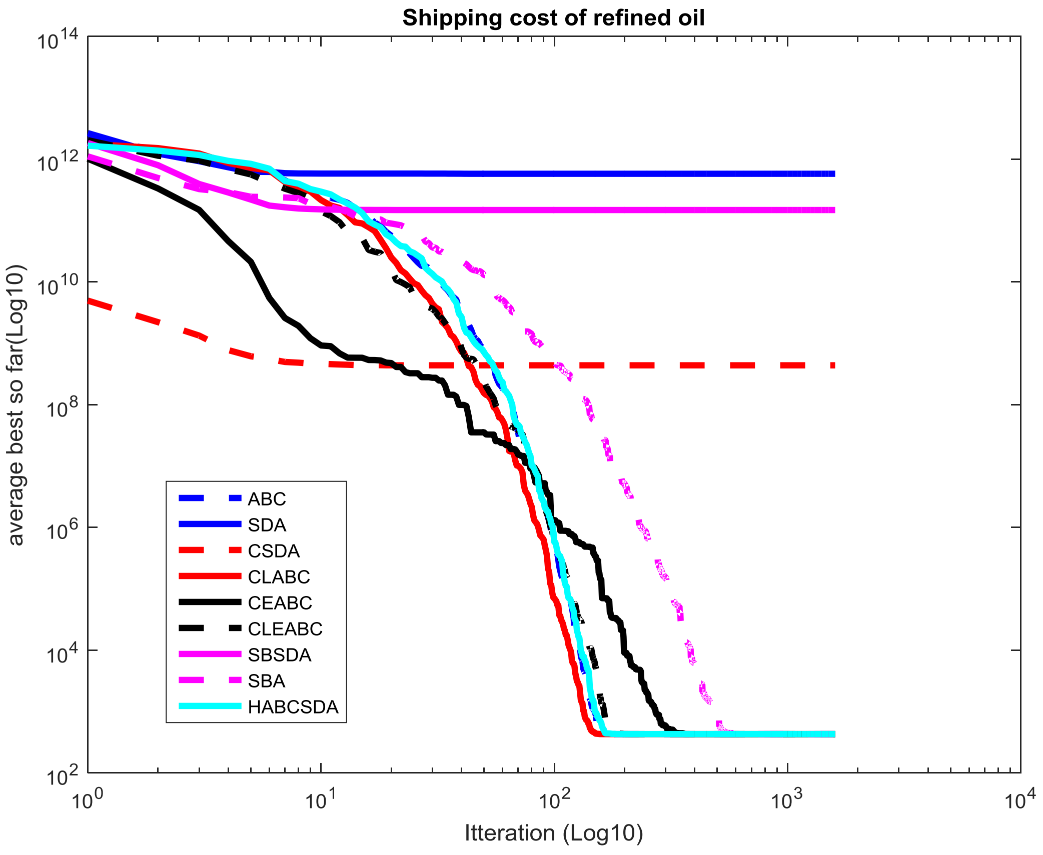


Figure 6.6:- Convergence plots of algorithms for shipping cost of refined oil problem

6.4 Profit maximization of selling television sets

In this experiment, the objective is to estimate the maximum profit gain per year by colour Television (TV) manufacture when 19” and 21” flat screen TVs are sold. This problem has been featured in the work of Yahya [173] .The price of 19” flat screen TV, p is \$339, while 21” flat screet TV is \$60 more expensive. To produce both models, the Manufacturing Company needs to spend \$195 for each 19” flat screen TV and \$225 for each 21” flat screen TV. The total operation cost for this production, x is \$400,000. Based on the supply and demand, the price of the TV is expected to be drop by \$0.013 for each 19” flat screen TV sold and the price of 21” flat screen TV is expected to be lower by \$0.014 for every unit sold. The mathematical formulation of the profit made in selling TV set is

$$f(x) = R(x) - C(x)$$

Cost to produce flat screen TV, $C(x) = 400000 + 195(x_1) + 225(x_2)$,

Total sales of flat screen TV, $R(x) = p(x_1) + q(x_2)$

Price tag for one 19” flat screen TV, $p(x) = 339 - 0.01(x_1) - 0.003(x_2)$,

Price tag for one 21” flat screen TV, $\text{q}(x) = 339 - 0.004(x_1) - 0.01(x_2)$

$$0 \leq x_1, x_2$$

Table 6.7:- Statistical results of profit maximization of selling television sets

	ABC	SDA	CSDA	CLABC	CEABC	CLEABC	SBSDA	SBA	HABCSDA
Best	\$553,641.0256								
x_1	4735.04	4735.04	4735.04	4735.04	4735.04	4735.04	4735.04	4735.04	4735.04
x_2	7042.74	7042.74	7042.74	7042.74	7042.74	7042.74	7042.74	7042.74	7042.74
Avg	553,641.0256								
SD	0								

As noted in the statistical results in Table 6.7, the best profit is \$553,641.0256. To achieve this value, the manufacture must sell 4736 sets of 19” flat screen TV and 7043 sets of 21” flat screen TV. All the proposed algorithms found the optimum profit margin in 1600 iterations. The statistical parameter resulting from the search was with zero standard deviation. As noted in the convergence plot in Figure 6.7, the HABCSDA was the fastest one to reached the optimal cost in 78 iterations while the slowest one to converge is SDA (268 iterations). CSDA, CLABC, CEABC, CLEABC, SBSDA, SBA converged to the optimal point within 97, 129, 142, 140, 49, 142 iterations respectively.

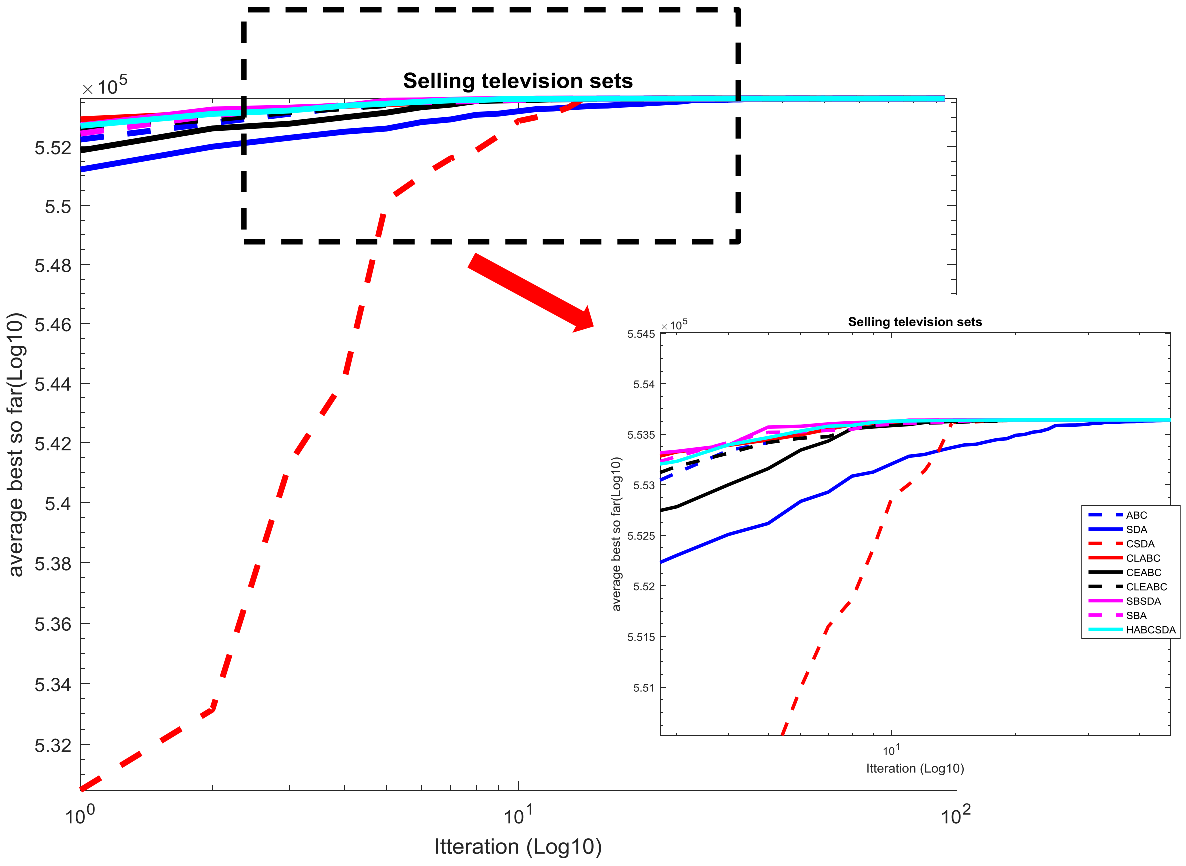


Figure 6.7:- Convergence plot of profit maximization of selling television sets

6.5 Car side impact design

This practical engineering design problem by Gu et al. [193] is about the car side impact design (Figure 6.8). The objective of design is to minimize the total car weight and to enhance the performance of car side impact crash.

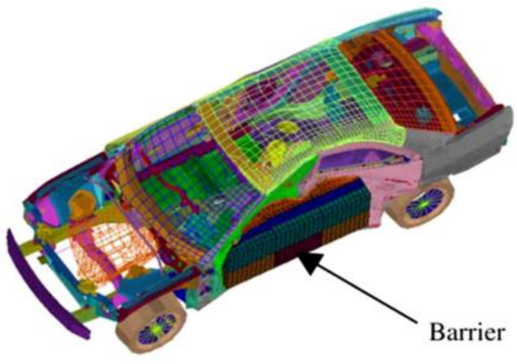


Figure 6.8:- FEM model of car side impact (adopted from Zhang et. el)[194]

The mathematical model of total car weight is given as

$$f(x) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7$$

The design parameters for this problem are B-pillar thickness (x_1), B-pillar reinforcement thickness (x_2), floor side inner tickhness (x_3), cross members thickness (x_4), door beam thickness (x_5), door beltline reinforcement thickness(x_6), roof rail thickness (x_7), B-pillar inner material (x_8), floor side inner material (x_9), height barrier (x_{10}) and hitting possition (x_{11}).According to Zhang et al. [194] and Yahya [173] the problem has ten constrained parameters to take into consideration in this design. These constraints are used to ensure the impact or injury to human inside the car during side crash is minimum [195]. Table 6.8 shows the list of constrained parameters and the corresponding mathematical models are given as

Table 6.8:- Constrained parameters list

Constrained	Descriptions
F_a	Load in abdomen
VC_u	Dummy upper chest
VC_m	Dummy middle chest
VC_l	Dummy lower chest
Δ_{ur}	Upper rib deflection
Δ_{mr}	middle rib deflection
Δ_{lr}	lower rib deflection
V_{MBP}	V-pillar at middle point
F_p	Pubic force
V_{FD}	Velocity of front door at v-pillar

$$\begin{aligned}
F_a &= 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \leq 1 \\
VC_u &= 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_{10}x_5 \\
&\quad +0.080405x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} \leq 0.32 \\
VC_m &= 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.121x_3x_9 \\
&\quad +0.0208x_3x_8 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + 0.00121x_8x_{11} \leq 0.32 \\
VC_l &= 0.074 + 0.061x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 \leq 0.32 \\
\Delta_{ur} &= 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \leq 0.32 \\
\Delta_{mr} &= 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11x_2x_8 - 0.0215x_5x_{10} \\
&\quad -9.98x_7x_8 + 22x_9x_8 \leq 32 \\
\Delta_{lr} &= 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} \leq 32 \\
V_{MBP} &= 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} \leq 9.9 \\
F_p &= 4.72 - 0.5x_4 - 0.19x_3x_2 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 \leq 4 \\
V_{FD} &= 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 \leq 15.7
\end{aligned}$$

$$0.5 \leq x_1, x_2, x_3, x_4, x_5, x_6, x_7 \leq 1.5$$

$$x_8, x_9 \in \{0.192, 0.345\}$$

$$-30 \leq x_{10}, x_{11} \leq 30$$

Table 6.9 shows the statistical results achieved with the algorithms. As noted, five of the proposed algorithms, HABCSDA, SBA, CLEABC, CEABC and CLABC outperformed other algorithms in robustness and lowest average weight of the car in this design problem. ABC also performed well in this experiment. The SDA and its variants did not reach the optimal point in this design problem. Although five algorithms reached the optimal value at the end of run, but as noted in the convergence plots in Figure 6.9, CEABC reached the optimal point after 43 iterations followed by CLEABC. The slowest to converge to the optimal point from these five top algorithms is SBA. SDA, SBSDA and CSDA got stuck at 23kg and above.

Table 6.9:- Statistical results of car side impact design problems

	ABC	**SDA**	**CSDA**	**CLABC**	**CEABC**	**CLEABC**	**SBSDA**	**SBA**	**HABCSDA**
Best	22.9047	23.4068	23.6811	22.2329	22.2329	22.2329	23.0547	22.2329	22.2329
x_1	0.5000	0.5791	0.5018	0.5000	0.5000	0.5000	0.6724	0.5000	0.5000
x_2	0.5000	1.3155	0.5000	0.5000	0.5000	0.5000	0.8888	0.5000	0.5000
x_3	0.5000	0.5471	0.5012	0.5000	0.5000	0.5000	0.5987	0.5000	0.5000
x_4	0.5000	0.6560	0.8439	0.5000	0.5000	0.5000	0.7084	0.5000	0.5000
x_5	0.5000	1.1729	0.5020	0.5000	0.5000	0.5000	0.8285	0.5000	0.5000
x_6	0.5000	0.7174	0.6330	0.5000	0.5000	0.5000	1.3689	0.5000	0.5000
x_7	0.5000	0.8386	1.0456	0.5000	0.5000	0.5000	0.5521	0.5000	0.5000
x_8	0.1920	0.1920	0.1920	0.1920	0.1920	0.1920	0.1920	0.1920	0.1920
x_9	0.3450	0.3450	0.3450	0.3450	0.3450	0.3450	0.3450	0.3450	0.3450
x_{10}	-30.0000	-3.9554	28.7004	-30.0000	-30.0000	-30.0000	-5.9247	-30.0000	-30.0000
x_{11}	30.0000	-17.1380	29.3238	30.0000	30.0000	30.0000	22.2336	30.0000	30.0000
Avg	22.9047	27.9743	27.3353	**22.2329**	**22.2329**	**22.2329**	27.4987	**22.2329**	**22.2329**
SD	0.0000	2.3922	2.3208	**0.0000**	**0.0000**	**0.0000**	2.5429	**0.0000**	**0.0000**
Worse	22.9047	31.7274	32.8483	22.2329	22.2329	22.2329	31.5280	22.2329	22.2329

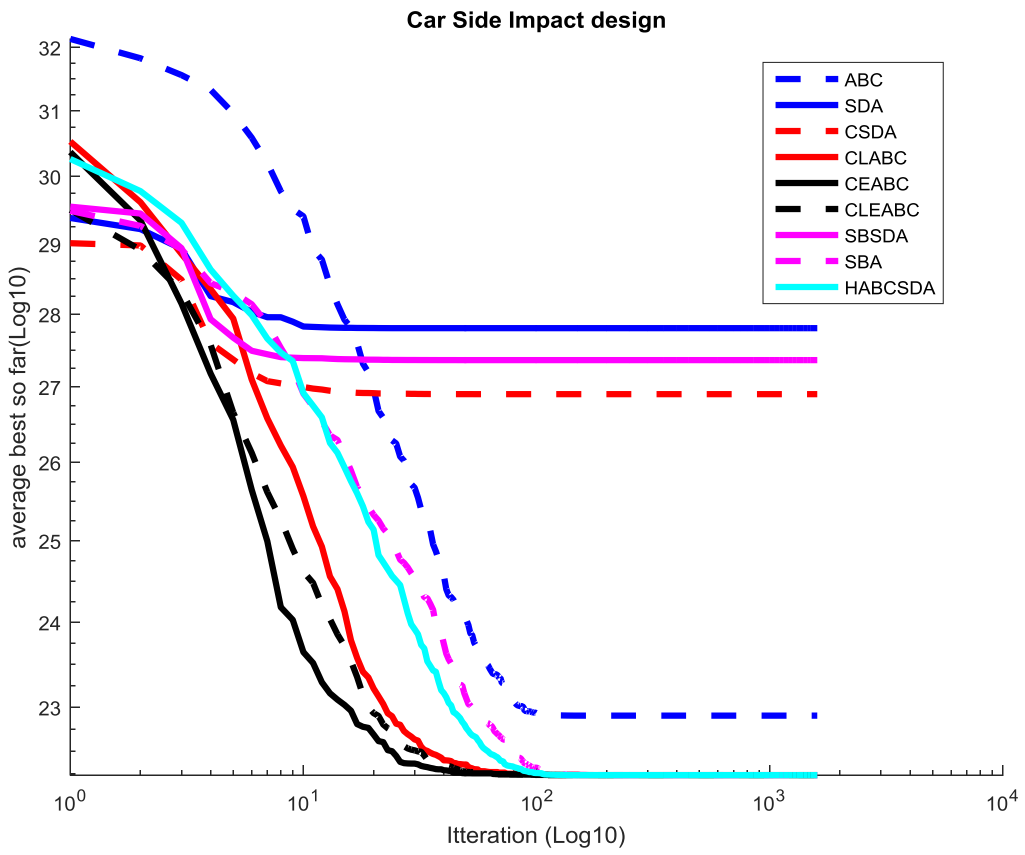


Figure 6.9:- Convergence plots of algorithms for car side impact design

6.6 Summary

Four practical/engineering problems have been used to test the proposed algorithms and their predecessor algorithms. These have included PD-fuzzy controller design for SLMS, minimization of shipping refined oil cost to Japan via Malacca Straits, profit maximization of selling flat television sets and minimization of weight in car side impact design. The results have shown that the proposed algorithms outperformed the original algorithms in solving the given practical problems.

Chapter 7
Conclusions and future work

7.1 Conclusions

An investigation of ABC and SDA algorithms and new types of algorithms has been carried out in this research. The objectives of this research have been to introduce modifications and hybridisations on the original ABC and SDA in order to enhance their performances in terms of accuracy, robustness and their convergence speed in solving various types of problems. The research has resulted eight modifications leading to seven new algorithms. The new algorithms thus arrived at have been tested and their performances have been assessed in various types of benchmark functions, including standard benchmark functions and CEC2013 benchmark functions, as well as selected complex engineering/practical applications. The results have shown that the developed algorithms outperform their predecessor algorithm in term of accuracy, convergence speed and efficiency.

The contributions of this research to knowledge can be summarised as follows:

a) In order to get a good quality of initial solution distribution, combinations of opposition based learning, random and chaotic distribution have been proposed. This novel approach enable to produce initial solutions near to the optimal value and able to speed up the algorithms to converge. This proposed initial distribution have been incorporated in all the proposed algorithms.
b) The SDA has been reformulated by replacing the spiral radius and spiral angle by chaotic patterns. Chaotic trajectory has helped the algorithm to escape from traps of local optima. This new algorithm has been referred to as CSDA.
c) Three adaptive algorithms have been proposed by incorporating chaotic maps with adaptive step sizes, using exponential and linear trajectory formulations, into the ABC algorithm. The proposed adaptive algorithms have been referred to as CLABC, CEABC and CLEABC.
d) Three hybrid algorithms have been developed based on SDA and ABC. These have been referred to as SBSDA, SBA and HABCSDA. SBSDA uses scout bee features in ABC to solve the problem of SDA getting trapped at local optima. Similarly, For SBA, The bee’s movement is ABC has been converted into spiral movement in order

to improve the local search capability. The 3rd hybrid version is HABCSDA. HABCSD algorithm is a series type hybrid strategy where the structure of algorithms is placed in a sequential order. Combination of these two algorithms in series can reduced the potential algorithm to trap at local optimum and may speed up the algorithm to converge.

e) The performances of the proposed algorithms have been assessed in single-objective and multi-objective optimisation problems. The single-objective problems have included ten unconstrained standard benchmark functions, sixteen CEC 2013 unconstrained benchmark functions, ten CEC 2006 constrained benchmark functions, five constrained engineering problems, and further five practical/engineering problems of constrained and unconstrained nature. The assessments have been carried out in comparison with the original ABC and SDA and the results have shown the superiority of the proposed algorithms in solving complex uni-modal and multi-modal problems of complex nature with great accuracy and efficiency.

In conclusion, the proposed algorithms have the capability to solve various types of problems including single objective constrained and unconstrained problems, multi-objective problems of complex and practical nature.

7.2 Recommendations for future work

For future works, the proposed algorithms can be extended to:-

a) Real time applications – It is recommended to test and assess the proposed algorithms in real time applications. The algorithms achieve accurate results in simulated environments. However, the computing requirements in real-time applications would require careful consideration of matching the computing needs of the algorithm with computing resources.
b) Although the proposed ABC algorithms are able to give accurate result, their convergence speeds are impressive, especially for real-time applications. New types of fitness evaluation and bee selection can be considered for further modifications to speed up their convergence further.
c) SDA is fast responding algorithms, but its accuracy is not impressive and gets trapped at local optima. The proposed SDA based algorithms are able to resolve these issues. However, their performance in solving high dimension are not impressive. In order to

increase the accuracy and convergence speed in high dimension problems, it is worth to investigate modifications on the square matrix of spiral trajectory pattern. This square matrix become big as the dimension gets high and consumes a lot of computing time.

References

[1] Rao, S. S. (2009). *Engineering Optimization: Theory and Practice*. John Wiley & Sons Inc

[2] Mirjalili, S., Jangir, P., Mirjalili, S. Z., Saremi, S., and Trivedi, I. N. (2017). Optimization of problems with multiple objectives using the multi-verse optimization algorithm. *Knowledge-Based Systems*, *134*, 50–71

[3] Jiao, R., Zeng, S., Alkasassbeh, J. S., & Li, C. (2017). Dynamic multi-objective evolutionary algorithms for single-objective optimization. *Applied Soft Computing Journal*, *61*, 793–805.

[4] Ochs, P., Ranftl, R., Brox, T., & Pock, T. (2016). Techniques for gradient-based bilevel optimization with non-smooth lower level problems. *Journal of Mathematical Imaging and Vision*, *56*(2), 175–194.

[5] Nocedal, J., & Wright, S. J. (1999). *Numerical Optimization*. Springer Science, 67–68.

[6] Hasan, M. A. (2004). Line search and gradient method for solving constrained optimization problems. *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume 5, 789-92. Montreal, Canada

[7] Du, K. L., and Swamy, M. N. S. (2016). Particle swarm optimization. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*, 153–173. Springer

[8] Weise, T. (2009). Global Optimization Algorithms–Theory and Application. 820 Self-Published

[9] Eshelman, L. J., Caruana, R. A., and Schaffer, J. D. (1989). Biases in the crossover landscape. *Third International Conference on Genetic Algorithms*, 10–19. Virginia, USA.

[10] Muttil, N., and Liong, S.-Y. (2004). Superior exploration–exploitation balance in shuffled complex evolution. *Journal of Hydraulic Engineering*, *130*(12), 1202–1205.

[11] Neri, F., and Cotta, C. (2012). Memetic algorithms and memetic computing

optimization: A literature review. *Swarm and Evolutionary Computation*, *2*, 1–14.

[12] Zang, H., Zhang, S., and Hapeshi, K. (2010). A review of nature-inspired algorithms. *Journal of Bionic Engineering*, *7*(SUPPL.), 232–237

[13] Hashim, M. R., and Tokhi, M. O. (2017), Optimal tuning of pd controllers using modified artificial bee colony algorithm. *Jurnal of Telecommunication Electronic Computer Engineering*. 379–384.

[14] Feng, T., Xie, Q., Hu, H., Song, L., Cui, C., and Zhang, X. (2015). Bean optimization algorithm based on negative binomial distribution, *Advances in Swarm and Computational Intelligence: 6th International Conference, ICSI 2015, Proceedings, Part I*, 82–88, *Beijing, China*

[15] Kassabalidis, I., El-Sharkawi, M. A., Marks, R. J. I., Arabshahi, P., and Gray, A. A. (2001). Swarm intelligence for routing in communication networks. *GLOBECOM'01. IEEE Global Telecommunications Conference, 6*, 3613–3617. San Antonio, TX, USA

[16] Garg, H. (2013). Solving structural engineering design optimization problems using an artificial bee colony algorithm. *Journal of Industrial and Management Optimization*, *10*(3), 777–794

[17] Yang, X. (2011). Review of metaheuristics and generalised evolutionary walk algorithm. *International Journal of Bio-Inspired Computation*, *3*(2), 77–84. Inderscience Publishers, Geneva, SWITZERLAND

[18] Chowdhury, S., Zhang, J., and Messac, A. (2012). Avoiding premature convergence in a mixed-discrete particle swarm optimization (MDPSO) algorithm. *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 23–26. Honolulu, Hawaii, USA.

[19] Learning, M., Academic, K., and Manufactured, P. (1988). Genetic algorithms and machine learning. *Machine Learning*, 95–99.

[20] Cheong, D., Kim, Y. M., Byun, H. W., Oh, K. J., and Kim, T. Y. (2017). Using genetic algorithm to support clustering-based portfolio optimization by investor information. *Applied Soft Computing Journal*, *61*, 593–602.

[21] Eberhart, R., and Kennedy, J. (1995). A new optimizer using particle swarm theory. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39–43.

[22] Karaboga, D., Gorkemli, B., Ozturk, C., and Karaboga, N. (2012). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57.

[23] Yang, X., and Gandomi, A. H. (2012). Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, *29*(5), 464–483.

[24] Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems*, *22*(3), 52–67.

[25] Dorigo, M., Maniezzo, V., and Colorni, a. (1996). The ant systems: optimization by a colony of cooperative agents. *IEEE Transactions on Man, Machine and Cybernetics-Part B*, *26*(1), 29-41.

[26] Chen, Z., Zhou, S., and Luo, J. (2017). A robust ant colony optimization for continuous functions. *Expert Systems with Applications*, *81*, 309–320.

[27] Yang, X. (2010). *Nature-Inspired Metaheuristic Algorithms Second Edition*. Luniver Press.

[28] Shareef, H., Islam, M. M., Ibrahim, A. A., and Mutlag, A. H. (2015). A nature inspired heuristic optimization algorithm based on lightning. *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, 9–14, Sabah, Malaysia.

[29] Shah, H. (2011). Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. *International Journal of Computational Science and Engineering*, *6*(1/2), 132–140. Inderscience

[30] Geem, Z. W., Kim, J. H., and Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, *76*(2), 60–68.

[31] Geem, Z. W., Yang, X. S., and Tseng, C. L. (2013). Harmony search and nature-inspired algorithms for engineering optimization. *Journal of Applied Mathematics*, *2013*, 2–4.

[32] Ahmad, S., Tokhi, M. O., and Toha, S. F. (2009). Genetic algorithm optimisation for fuzzy control of wheelchair lifting and balancing. *2009 Third UKSim European Symposium on Computer Modeling and Simulation*, 97–101. Athens, Greece

[33] Gozde, H., Taplamacioglu, M. C., and Kocaarslan, I. (2010). Application of artificial bees colony algorithm, In An Automatic Voltage Regulator (AVR) System, *International Journal on "Technical and Physical Problems of Engineering" (IJTPE)*, 2(3), 88–92.

[34] Omkar, S. N., Senthilnath, J., Khandelwal, R., Narayana Naik, G., and Gopalakrishnan, S. (2011). Artificial bee colony (ABC) for multi-objective design optimization of composite structures. *Applied Soft Computing*, *11*, 489–499.

[35] Bonde, G., and Khaled, R. (2012). Stock price prediction using genetic algorithms and evolution strategies. *The 2012 International Conference on Genetic and Evolutionary Methods*, 10-15. Nevada, USA

[36] Wanger, P., and Martin, L. (2004). Algorithms for optimizing drug therapy. *BMC Medical Informatics and Decision Making*, *4(1)*, 10.

[37] Karoboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical report-TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department.

[38] Tamura, K., and Yasuda, K. (2011). Spiral multipoint search for global optimization. *10th International Conference on Machine Learning and Applications and Workshops*, *2*(2), 470–475. Honolulu,Hawaii,USA

[39] Tamura, K., and Yasuda, K. (2011). Spiral dynamics inspired optimization. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *15*(8), 1116–1122.

[40] Tamura, K., and Yasuda, K. (2011). Primary study of spiral dynamics inspired optimization, *IEEJ Transactions on Electrical and Electronic Engineering*, 6(1), 98-100.

[41] Nasir, A. N. K., and Tokhi, M. O. (2015a). An improved spiral dynamic optimization algorithm with engineering application. *IEEE Transactions on Systems,*

Man, and Cybernetics: Systems, *45*(6), 1–1.

[42] Nasir, A. N. K., Tokhi, M. O., and Ghani, N. M. A. (2013). Novel hybrid bacterial foraging and spiral dynamics algorithms. *13th UK Workshop on Computational Intelligence (UKCI)*, 199–205. Guildford, UK.

[43] Nasir, A. N. K., Tokhi, M. O., Omar, M. E., and Ghani, N. M. A. (2014). An improved spiral dynamic algorithm and its application to fuzzy modelling of a twin rotor system. *2014 World Symposium on Computer Applications & Research (WSCAR)*, 1–6. Sousse, Tunisia.

[44] Nasir, A. N. K., Tokhi, M. O., Sayidmarie, O., and Raja IsmaiL, R. M. T. (2013). A novel adaptive spiral dynamic algorithm for global optimization. *2013 13th UK Workshop on Computational Intelligence (UKCI)*, 334–341. Guildford, UK.

[45] Nasir, A. N. K., Tokhi, M. O., Abd Ghani, N. M., and Raja Ismail, R. M. T. (2012). Novel adaptive spiral dynamics algorithms for global optimization. *2012 IEEE 11th International Conference on Cybernetic Intelligent Systems (CIS)*, 99–104. Limerick, Ireland.

[46] Nasir, A. N. K., Tokhi, M. O., Abd Ghani, N. M., and Ahmad, M. A. (2012). A novel hybrid spiral-dynamics bacterial-foraging algorithm for global optimization with application to control design. *2012 12th UK Workshop on Computational Intelligence (UKCI)*, 1–7. Guildford, UK.

[47] Nasir, A. N. K., and Tokhi, M. O. (2015). Novel metaheuristic hybrid spiral-dynamic bacteria-chemotaxis algorithms for global optimisation. *Applied Soft Computing*, *27*, 357–375.

[48] Tsai, C., Huang, W., and Chiang, M. (2014). A novel spiral optimization for clustering. *Mobile, Ubiquitous, and Intelligent Computing*, *274*, 621–628.

[49] Ashok. (2014). Design and analysis of kinematically redundant planar parallel manipulator for isotropic stiffness condition. *Master Thesis*. National Institute of Technology, India.

[50] Ouadi, A., Bentarzi, H., and Recioui, A. (2013). Optimal multiobjective design of digital filters using spiral optimization technique. *SpringerPlus*, *2*(1), 461.

[51] Tereshko, V., and Loengarov, A. (2005). Collective decision-making in honey bee foraging dynamics. *Computing and Information Systems Journal*, *9*, 1–7.

[52] Ab Wahab, M. N., Nefti-Meziani, S., and Atyabi, A. (2015). A comprehensive review of swarm optimization algorithms. *PLoS ONE*, *10*(5), 1–36.

[53] Karaboga, D., and Akay, B. (2010). Proportional- integral - derivative controller design by using artificial bee colony, harmony search, and the bees algorithms. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *224,* 869–883.

[54] Karaboga, D., and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471.

[55] Rao, R. S., Narasimham, S. V. L., and Ramalingaraju, M. (2008). Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm. *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, *2*(9), 1964–1970.

[56] Karaboga, D., Akay, B., and Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. *Modeling Decisions for Artificial Intelligence*, 318–329.

[57] Gao, W., and Liu, S. (2012). A modified artificial bee colony algorithm. *Computers & Operations Research*, *39*(3), 687–697.

[58] Karaboga, D., and Basturk, B. (2007b). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Jurnal of Global Optimization*, *39*(3), 459–471.

[59] Karaboga, D., and Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132.

[60] Karaboga, D., and Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing Journal*, 8(1), 687–697.

[61] Krishnanand, K. R., Nayak, S. K., Panigrahi, B. K., and Rout, P. K. (2009).

Comparative study of five bio-inspired evolutionary optimization techniques. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 1231–1236. India.

[62] Mala, D. J., Kamalapriya, M., Shobana, R., and Mohan, V. (2009). A non-pheromone based intelligent swarm optimization technique in software test suite optimization. *2009 International Conference on Intelligent Agent & Multi-Agent Systems*, 1–5. Chennai, India.

[63] Karaboga, D., and Akay, B. (2009). Artificial bee colony (ABC), harmony search and bees algorithms on numerical optimization. *Proceedings of Innovative Production Machines and Systems Virtual Conference, IPROMS*, 1–6. Cardiff, UK.

[64] Karaboga, D., and Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization. *12th International Fuzzy Systems Association World Congress, IFSA 2007*, 789–798, Cancun, Mexico.

[65] Mezura-Montes, E., and Velez-Koeppel, R. E. (2010). Elitist artificial bee colony for constrained real-parameter optimization. *2010 IEEE Congress on Evolutionary Computation, CEC 2010*. 1-8. Barcelona, Spain.

[66] Mezura-Montes, E., Araoz, M., and Domngez, O. (2010). Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization. *2010 IEEE Congress on Evolutionary Computation, CEC 2010*, 1-8. Barcelona, Spain.

[67] Quan, H., and Shi, X. (2008). On the analysis of performance of the improved artificial-bee-colony algorithm. *Proceedings - 4th International Conference on Natural Computation, ICNC 2008*, *7*(4), 654–658. Jinan, China.

[68] Lee, W. P., and Cai, W. T. (2011). A novel artificial bee colony algorithm with diversity strategy. *Proceedings - 2011 7th International Conference on Natural Computation, ICNC 2011*, *3*(3), 1441–1444. Shanghai, China.

[69] P-W. Tsai, J-S. Pan, B-Y. and Liao, S.-C. C. (2009). Enhanced artificial bee colony optimization. *International Jurnal of Innovative Computing, Information and Control (ICCI)*, *5*(12), 1–12.

[70] Zhu, G., and Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for

numerical function optimization. *Applied Mathematics and Computation*, *217*(7), 3166–3173.

[71] Yi, Y., and He, R. (2014). A Novel Artificial Bee Colony Algorithm. *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, (1), 271–274.

[72] Bao, L. B. L., and Zeng, J. Z. J. (2009). Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm. *2009 Ninth International Conference on Hybrid Intelligent Systems*, *1*, 411–416. Shenyang, China.

[73] Kang, F., Li, J., and Xu, Q. (2009). Structural inverse analysis by hybrid simplex artificial bee colony algorithms. *Computers and Structures*, *87*, 861–870.

[74] Jatoth, R. K., and Rajasekhar, A. (2010). Speed control of PMSM by hybrid genetic Artificial Bee Colony Algorithm. *2010 IEEE International Conference on Communication Control and Computing Technologies (ICCCCT)*, 241–246. Tamil Nadu, India.

[75] Zhong, Y., Lin, J., Ning, J., and Lin, X. (2011). Hybrid artificial bee colony algorithm with chemotaxis behavior of bacterial foraging optimization algorithm. *2011 Seventh International Conference on Natural Computation*, *2*(1), 1171–1174. Shanghai, China.

[76] Tsai, P.-W., Pan, J.-S., Shi, P., and Liao, B.-Y. (2011). A new framework for optimization based-on hybrid swarm intelligence. In B. K. Panigrahi, Y. Shi, & M.-H. Lim (Eds.), *Handbook of Swarm Intelligence: Concepts, Principles and Applications*, 421–449. Springer

[77] Li, C., and Chan, F. (2011). Complex-fuzzy adaptive image restoration -- An artificial-bee-colony-based learning approach. In N. T. Nguyen, C.-G. Kim, & A. Janiak (Eds.), *Intelligent Information and Database Systems: Third International Conference, ACIIDS 2011, Daegu, Korea, April 20-22, 2011, Proceedings, Part II* 90–99. Springer.

[78] El-Abd, M. (2011). A hybrid ABC-SPSO algorithm for continuous function optimization. *IEEE SSCI 2011 - Symposium Series on Computational Intelligence - SIS 2011: 2011 IEEE Symposium on Swarm Intelligence*, 96–101.

[79] Neelima, S., and Murthy, P. K. (2017). Optimization of association rule mining using hybridized artificial bee colony (ABC) with BAT algorithm, 832–835.

[80] Li, X., Huiyan Yang, Meihua Yang, Xian Yang, and Yang, G. (2017). Accelerating artificial bee colony algorithm with neighborhood search. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, 1549–1556. IEEE.

[81] Duan, H.-B., Xu, C.-F., and Xing, Z.-H. (2010). A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems. *International Journal of Neural Systems*, *20*(1), 39–50.

[82] Duan, H., Xing, Z., & Xu, C. (2009). An Improved quantum evolutionary algorithm based on artificial bee colony optimization, *Advances in Computational Intelligence,* 269–278. Springer

[83] Zhao, H., Pei, Z., Jiang, J., Guan, R., Wang, C., and Shi, X. (2010). A hybrid swarm intelligent method based on genetic algorithm and artificial bee colony. *Advances in Swarm Intelligence*, 558–565.

[84] Jatoth, R. K., and Rajasekhar. A. (2010). Speed control of PMSM by hybrid genetic artificial bee colony algorithm Ravi Kumar Jatoth,. *IEEE International Conference on Communication Control and Computing Technologies*, 241–246.

[85] Pulikanti S, S. (2009). An artificial bee colony algorithm for the quadratic knapsack problem. *16th International Conference, ICONIP 2009* , 196–205. Bangkok, Thailand.

[86] Marinakis, Y., Marinaki, M., and Matsatsinis, N. (2009). A hybrid discrete artificial bee colony - GRASP algorithm for clustering. *2009 International Conference on Computers & Industrial Engineering*, 548–553. Troyes, France.

[87] Banharnsakun, A., Achalakul, T., and Sirinaovakul, B. (2010). A hybrid method for solving traveling salesman problem. *Proceedings of the 2nd Congress on Nature and Biologically Inspired Computing*, 394–399. Melbourne, Australia.

[88] Alzaqebah, M., and Abdullah, S. (2011). Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems. *Combinatorial Optimization and Applications: 5th International Conference,*

COCOA 2011, 31–45. Zhangjiajie, China

[89] Shi, X., Li, Y., Li, H., Guan, R., Wang, L., and Liang, Y. (2010). An integrated algorithm based on artificial bee colony and particle swarm optimization. *2010 Sixth International Conference on Natural Computation*, (1), 2586–2590. Yantai, China.

[90] Kang, F., Li, J., Li, H., Ma, Z., and Xu, Q. (2010). An improved artificial bee colony algorithm. *2010 2nd International Workshop on Intelligent Systems and Applications*, 1–4. Wuhan, China.

[91] Shayeghi, H., Shayanfar, H. A., and Ghasemi, A. (2011). Artificial bee colony based power system stabilizer design for a turbo-generator in a single-machine power system. *International Conference on Artificial Inteligence*, 122–128.

[92] Abachizadeh, M., Yazdi, M. R. H., and Yousefi-Koma, A. (2010). Optimal tuning of PID controllers using artificial bee colony algorithm. *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 379–384. Montreal, Canada.

[93] Yan, G., and Li, C. (2011). An effective refinement artificial bee colony optimization algorithm based on chaotic search and application for PID control tuning, *Journal of Computing and Informations Systems*, 7(*9)*, 3309–3316.

[94] Mishra, A. K., Khanna, A., Singh, N. K., and Mishra, V. K. (2013). Speed control of dc motor using artificial bee colony optimization technique, *Universal Journal of Electrical and Electronic Engineering***,** *1*(3), 68–75.

[95] Karaboga, D., Okdem, S., and Ozturk, C. (2010). Cluster based wireless sensor network routings using Artificial Bee Colony Algorithm. *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*. http://doi.org/10.1109/AIS.2010.5547042

[96] Bayrakdar, M. E., and Calhan, A. (2017). Optimization of spectrum handoff with artificial bee colony algorithm. In *2017 25th Signal Processing and Communications Applications Conference (SIU)* (pp. 1–4). IEEE.

[97] Raju, R., and Kwan, H. K. (2017). FIR filter design using multi-objective artificial

bee colony algorithm. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)* (pp. 1–4). IEEE.

[98] Sonmez, M. (2011). Artificial Bee Colony algorithm for optimization of truss structures. *Applied Soft Computing Journal*, *11*(2), 2406–2418.

[99] Rekaby, A. (2013). Introducing adaptive artificial bee colony algorithm and using it in solving traveling salesman problem, *2013 Science and Information Conference (SAI)*, 502–506. London, UK.

[100] Yuxin, L. (2017). Fire detection method of mine belt conveyor based on artificial bee colony algorithm, *2017 Control and Decision Conference (CCDC),* 4778–4782. Chongqing, China.

[101] Fu, Z., Liu, Y., Hu, H., Wu, D., and Gao, H. (2017). An efficient method of white blood cells detection based on artificial bee colony algorithm. In *2017 29th Chinese Control and Decision Conference (CCDC)*, 3266–3271. IEEE.

[102] Gao, K., Yicheng Zhang, Sadollah, A., and Rong Su. (2017). Improved artificial bee colony algorithm for solving urban traffic light scheduling problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, 395–402. IEEE.

[103] Maaranen, H., Miettinen, K., and Penttinen, A. (2006). On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, *37*(3), 405.

[104] Mahdavi, S., Rahnamayan, S., and Deb, K. (2017). Opposition based learning: A literature review. *Swarm and Evolutionary Computation*.

[105] El-Abd, M. (2012). Generalized opposition-based artificial bee colony algorithm. *Proceedings of the 2012 IEEE World Congress on Evolutionary Computation*, 1–4. Brisbane, Australia.

[106] Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, *1*, 695–701. Vienna, Austria.

[107] Rojas-Morales, N., Riff Rojas, M. C., and Montero Ureta, E. (2017). A survey and classification of Opposition-Based Metaheuristics. *Computers and Industrial Engineering*, *110*, 424–435.

[108] Xu, Q., Wang, L., Wang, N., Hei, X., and Zhao, L. (2014). A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*, *29*, 1–12.

[109] Dong, N., Wu, C. H., Ip, W. H., Chen, Z. Q., Chan, C. Y., and Yung, K. L. (2012). An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection. *Computers and Mathematics with Applications*, *64*(6), 1886–1902.

[110] Wang, H., Wu, Z., Rahnamayan, S., Liu, Y., and Ventresca, M. (2011). Enhancing particle swarm optimization using generalized opposition-based learning. *Information Sciences*, *181*(20), 4699–4714.

[111] Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. A. (2008). Opposition versus randomness in soft computing techniques. *Applied Soft Computing Journal*, *8*(2), 906–918.

[112] Chuang, L.-Y., Tsai, S.-W., and Yang, C.-H. (2009). Improved catfish particle swarm optimization with embedded chaotic map. *2009 IEEE International Conference on Systems, Man and Cybernetics*, (October), 3895–3900. San Antonio, TX, USA

[113] Gao, W. feng, Liu, S. yang, and Huang, L. ling. (2012). Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Communications in Nonlinear Science and Numerical Simulation*, *17*(11), 4316–4327.

[114] Salgotra, R., Singh, U., and Saha, S. (2017). New cuckoo search algorithms with enhanced exploration and exploitation properties. *Expert Systems with Applications*, 95(1), 384-420.

[115] Mirjalili, S., and Lewis, A. (2014). Adaptive gbest-guided gravitational search algorithm. *Neural Computing and Applications*, *25*(7–8), 1569–1584.

[116] Koupaei, J. A., Hosseini, S. M. M., and Ghaini, F. M. M. (2016). A new

optimization algorithm based on chaotic maps and golden section search method. *Engineering Applications of Artificial Intelligence*, *50*, 201–214.

[117] Snaselova, P., and Zboril, F. (2015). Genetic algorithm using theory of chaos. *Procedia Computer Science*, *51*(1), 316–325.

[118] Yang, D., Li, G., and Cheng, G. (2007). On the efficiency of chaos optimization algorithms for global optimization. *Chaos, Solitons & Fractals*, *34*(4), 1366–1375.

[119] Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*. 20, 130-141.

[120] Bucolo, M., Caponetto, R., Fortuna, L., Frasca, M., and Rizzo, A. (2002). Does chaos work better than noise? *IEEE Circuits and Systems Magazine*, *2*(3), 4–19.

[121] Kellert, S. H. (1994). *In the wake of chaos: Unpredictable order in dynamical systems*. University of Chicago press.

[122] Saremi, S., Mirjalili, S., and Lewis, A. (2014). Biogeography-based optimisation with chaos. *Neural Computing and Applications*, 1077–1097.

[123] Coelho, L. dos S., and Mariani, V. C. (2012). Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. *Computers & Mathematics with Applications*, *64*(8), 2371–2382.

[124] Coelho, L. D. S., and Mariani, V. C. (2009). A novel chaotic particle swarm optimization approach using hanon map and implicit filtering local search for economic load dispatch. *Chaos, Solitons and Fractals*, *39*(2), 510–518.

[125] Ding, L., Wu, H., and Yao, Y. (2015). Chaotic artificial bee colony algorithm for system identification of a small-scale unmanned helicopter. International Journal of Aerospace Engineering, 2015, 11.

[126] Akay, B., and Karaboga, D. (2012). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, *192*, 120–142.

[127] Li, X., and Yang, G. (2016). Artificial bee colony algorithm with memory. *Applied Soft Computing*, *41*, 362–372.

[128] Xiang, W., Meng, X., Li, Y., He, R., and An, M. (2017). An improved artificial bee

colony algorithm based on the gravity model. *Information Sciences*, *429*, 49–71.

[129] Li, M., Hei, Y., and Qiu, Z. (2017). Optimization of multiband cooperative spectrum sensing with modified artificial bee colony algorithm. *Applied Soft Computing*, *57*(Supplement C), 751–759.

[130] Cui, L., Li, G., Wang, X., Lin, Q., Chen, J., Lu, N., and Lu, J. (2017). A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences*, *417*, 169–185.

[131] Ghambari, S., and Rahati, A. (2017). An improved artificial bee colony algorithm and its application to reliability optimization problems. *Applied Soft Computing*.

[132] Liang, Z., Hu, K., Zhu, Q., and Zhu, Z. (2017). An enhanced artificial bee colony algorithm with adaptive differential operators. *Applied Soft Computing*, *58*, 480–494.

[133] Xiang, W., Li, Y., He, R., Gao, M., and An, M. (2018). A novel artificial bee colony algorithm based on the cosine similarity. *Computers & Industrial Engineering*, *115*(January 2017), 54–68.

[134] Nasir, A., N., K. (2014). *Bacterial Foraging and Spiral Dynamics Based Metaheuristic Algorithms for Global Optimisation with Engineering Applications*, PhD Thesis University of Sheffield. PhD Thesis.

[135] Adeyemo, J., and Stretch, D. (2017). Review of hybrid evolutionary algorithms for optimizing a reservoir. *South African Journal of Chemical Engineering*, 25(1), 22-31.

[136] Farnad, B., Jafarian, A., and Baleanu, D. (2017). A new hybrid algorithm for continuous optimization problem. *Applied Mathematical Modelling*, 55(1), 652-673.

[137] Mafarja, M. M., and Mirjalili, S. (2017). Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing*, *260*, 302–312.

[138] Wang, Y., Lai, X., Chen, L., Ding, H., and Wu, M. (2017). A quick control strategy based on hybrid intelligent optimization algorithm for planar n -link underactuated manipulators. *Information Sciences*, *420*, 148–158.

[139] Inclan, E., Geohegan, D., and Yoon, M. (2018). A hybrid optimization algorithm to explore atomic configurations of TiO2nanoparticles. *Computational Materials Science, 141*, 1–9.

[140] Mobin, M., Mousavi, S. M., Komaki, M., and Tavana, M. (2017). A hybrid desirability function approach for tuning parameters in evolutionary optimization algorithms. *Measurement: Journal of the International Measurement Confederation, 114*, 417–427.

[141] Xia, X., Gui, L., He, G., Xie, C., Wei, B., Xing, Y., and Tang, Y. (2017). A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. *Journal of Computational Science*.

[142] Mahmoudian Esfahani, M., Sheikh, A., and Mohammed, O. (2017). Adaptive real-time congestion management in smart power systems using a real-time hybrid optimization algorithm. *Electric Power Systems Research*, *150*, 118–128.

[143] Sree Ranjini, S. R., and Murugan, S. (2017). Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications, 83*, 63–78.

[144] Noack, M. M., and Funke, S. W. (2017). Hybrid genetic deflated Newton method for global optimisation. *Journal of Computational and Applied Mathematics, 325*, 97–112.

[145] Chen, X., Zhou, Y., Tang, Z., and Luo, Q. (2017). A hybrid algorithm combining glowworm swarm optimization and complete 2-opt algorithm for spherical travelling salesman problems. *Applied Soft Computing Journal, 58*, 104–114.

[146] Jadon, S. S., Tiwari, R., Sharma, H., and Bansal, J. C. (2017). Hybrid artificial bee colony algorithm with differential evolution. *Applied Soft Computing*, *58*, 11–24.

[147] irjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., and Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163–191.

[148] Simon, D. (2013). *Evolutionary Optimization Algorithms*. Wiley.

[149] Surjanovic, S. and Bingham, D. (2013). Virtual library of simulation experiments:

test functions and datasets. Retrieved January 9, 2018, from http://www.sfu.ca/~ssurjano.

[150] Jamil, M., and Yang, X.-S. (2013). A Literature survey of benchmark functions for global optimization problems, *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2), 150–194.

[151] Abiyev, R. H., & Tunay, M. (2016). Experimental Study of Specific Benchmarking Functions for Modified Monkey Algorithm. *Procedia Computer Science*, *102*, 595–602.

[152] Liang, J. J., Qu, B. Y., and Suganthan, P. N. (2013). Problem definitions and evaluation criteria for the cec 2014 special session on constrained real-parameter optimization. In *Computational Intelligence Laboratory*.

[153] Kasdirin, H. A. (2016). *Adaptive bio-inspired firefly and invasive weed algorithms for global optimisation with application to engineering problems* , PhD Thesis, Department of Automatic Control and Systems Engineering, The University Of Sheffield,UK.

[154] Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11–12), 1245–1287.

[155] Saha, C., Das, S., Pal, K., and Mukherjee, S. (2015). A fuzzy rule-based penalty function approach for constrained evolutionary optimization. *IEEE Transactions on Cybernetics*, *PP*(99), 1. IEEE

[156] Arora, J. S. (2004). *Chapter 17-Multi Objective Optimum Design concepts and MethodsIntroduction to Optimum Design*, Introduction to Optimum Design (third edition). Academic Press, Boston.

[157] Mezura-Montes, E., and Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173–194.

[158] Rubinov, a. M., Yang, X. Q., and Bagirov, a. M. (2002). Penalty functions with a small penalty parameter. *Optimization Methods and Software*, *17*, 931–964.

[159] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *KanGAL*, 251–256.

[160] Belegundu, A. D., and Arora, J. S. (1985). A study of mathematical programming methods for structural optimization. Part I: Theory. *International Journal for Numerical Methods in Engineering*, *21*(9), 1583–1599.

[161] Reklaitis, A. R. K. M. R. (1985). Engineering optimization - Methods and applications. *Engineering Optimization*, *8*(4), 333–334.

[162] Kannan, B. K., and Kramer, S. N. (1994). An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, *116*(2), 405–411.

[163] Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, *112*(2), 223–229.

[164] Sadollah, A., Bahreininejad, A., Eskandar, H., and Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing Journal*, *13*(5), 2592–2612.

[165] Ngatchou, P., Zarei, A., and El-Sharkawi, A. (2005). Pareto multi objective optimization. *Proceedings of the 13th International Conference On, Intelligent Systems Application to Power Systems*, 84–91. Arlington, USA. IEEE

[166] Marler, R. T., and Arora, J. S. (2010). The weighted sum method for multi-objective optimization: New insights. *Structural and Multidisciplinary Optimization*, *41*(6), 853–862.

[167] Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, *181*(3), 1653–1669.

[168] Jiang, S., Zhang, J., Ong, Y. S., Zhang, A. N., and Tan, P. S. (2015). A simple and fast hypervolume indicator-based multiobjective evolutionary algorithm. *IEEE Transactions on Cybernetics*, *45*(10), 2202–2213.

[169] Jaimes, A. L., Quintero, L. V. S., and Coello, C. A. C. (2009). Ranking methods in

many-objective evolutionary algorithms. In R. Chiong (Ed.), *Nature-Inspired Algorithms for Optimisation*, 413–434. Springer

[170] Li, H., and Landa-Silva, D. (2009). An Elitist GRASP Metaheuristic for the multi-objective quadratic assignment problem. *Evolutionary Multi-Criterion Optimization: 5th International Conference, EMO 2009,* 481–494. Nantes, France.

[171] Basseur, M., and Burke, E. K. (2007), Indicator-based multi-objective local search, *IEEE Congress on Evolutionary Computation, CEC 2007,* 0–7, Singapore

[172] Naidu, K., Mokhlis, H., and Bakar, A. H. A. (2014). Multiobjective optimization using weighted sum artificial bee colony algorithm for load frequency control, *Electrical Power and Energy Systems , 55*, 657–667.

[173] Yahya, N. M. (2016). *Bats Echolocation-Inspired Algorithms for Global Optimisation Problems*. PhD Thesis, Department of Automatic Control and Systems Engineering, The University Of Sheffield, UK.

[174] Auger, A., Bader, J., Brockhoff, D., and Zitzler, E. (2012). Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science, 425*, 75–103. .

[175] Yen, G. G., and He, Z. (2013). Performance metrics ensemble for multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 18(1), 131–144.

[176] Schott, J. R. (1995). *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. Massachusetts Institute of Technology.

[177] Ara, D. R. B., and Martins-filho, J. F. (2011). A performance comparison of multi-objective optimization evolutionary algorithms for all-optical networks design. *2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM),* 89-96. Paris, France.

[178] Luo, B., Zheng, J., Xie, J., and Wu, J. (2008). Dynamic crowding distance - A new diversity maintenance strategy for MOEAs. *Proceedings - 4th International Conference on Natural Computation, ICNC 2008, 1*, 580–585. Jinan, China.

[179] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods

and applications. *TIK-Schriftenreihe*, *30*(30), 1–122.

[180] Bradstreet, L. (2011). *The Hypervolume Indicator for Multi-objective Optimisation: Calculation and Use*, PhD Thesis, ThwUniversity of Western Australia.

[181] Auger, A., Bader, J., Brockhoff, D., and Zitzler, E. (2012b). Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, *425*, 75–103.

[182] Azad, A. K. M. (1994). *Analysis and design of control mechanisms for flexible manipulator systems*. PhD Thesis, Department of Automatic Control and Systems Engineering, The University Of Sheffield,UK.

[183] Poerwanto, H. (1998). *Dynamic simulation and control of flexible manipulator systems,* PhD Thesis, Department of Automatic Control and Systems Engineering, The University Of Sheffield,UK

[184] Tokhi, M. O., Mohamed, Z., and Shaheed, M. H. (2001). Dynamic characterisation of a Fexible manipulator system. *Robotica*, *19*(19), 571–580.

[185] Supriyono, H. (2012). *Novel Bacterial Foraging Optimisation Algorithms with Application to Modelling and Control of Flexible Manipulator Systems*, PhD Thesis, Department of Automatic Control and Systems Engineering, The University Of Sheffield, UK.

[186] Tokhi, M. O., Mohamed, Z., Martins, J. M., Botto, M. A., and Sá da Costa, J. (2003). Approaches for dynamic modelling of flexible manipulator systems. *IEE Proceedings - Control Theory and Applications*, *150*(4), 401–411.

[187] Supriyono, H., Tokhi, M. O., and Zain, B. a. M. (2010). Adaptive biologically-inspired algorithm-based controller tuning for input tracking control of flexible manipulators. *2010 IEEE 9th International Conference on Cyberntic Intelligent Systems*, 1–6. Reading, UK.

[188] Nasir, A. N. K., Tokhi , M. O., Abd Ghani, N. M. and Ahmad, M. A., (2012). A novel hybrid spiral-dynamics bacterial-foraging algorithm for global optimization with application to control design, *2012 12th UK Workshop on Computational Intelligence (UKCI)*, 2012, pp. 1-7. Edinburgh

[189] Supriyono, H., Tokhi, M. O., and Zain, B. a M. (2010). Control of a single-link flexible manipulator using improved bacterial foraging algorithm. *2010 IEEE Conference on Open Systems (ICOS 2010)*, 68–73. Kuala Lumpur, Malaysia

[190] Zain, B. A. M., Tokhi, M. O., and Toha, S. F. (2009). PID-based control of a single-link flexible manipulator in vertical motion with genetic optimisation. *2009 Third UKSim European Symposium on Computer Modeling and Simulation*, 355–360. Athen, Greece.

[191] Aldebrez, F. M., Alam, M. S., and Tokhi, M. O. (2005). Input-shaping with GA-tuned PID for target tracking and vibration reduction, *2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, 485–490. Limassol, Cyprus.

[192] Edgar, T. F., Himmelblau , David M. and Lasdon Leon S., (2001). *Optimization of chemical processes* (2nd ed). New York : McGraw-Hill.

[193] Gu, L., Yang, R. J., Tho, C. H., Murkowski, M., Faruquet, O. and Li, Y. (2001). Optimisation and robustness for crashworthiness of side impact. *International Journal of Vehicle Design (IJVD)*, *26*(4), 348–360.

[194] Zhang, C., Lin, Q., Gao, L., and Li, X. (2015). Backtracking search algorithm with three constraint handling methods for constrained optimization problems. *Expert Systems with Applications*, *42*(21), 7831–7845.

[195] B.D., Y., K.K., C., R.-J., Y., and L., G. (2004). Reliability-based design optimization for crashworthiness of vehicle side impact. *Structural and Multidisciplinary Optimization*, *26*(3), 272–283.

Appendix

Appendix A:- Mathematical Formulation of benchmark problems used

Function		$\boldsymbol{f_1(x)}$	$\boldsymbol{f_2(x)}$	$\boldsymbol{g(x)}$
MO_1	SCH 1	x^2	$(x-2)^2$	-
MO_2	Kursawe	$\sum_{i=2}^{2}\left(-10e^{-0.2\sqrt{x_i^2+x_{i+1}^2}}\right)$	$\sum_{i=2}^{2}(\lvert x_i\rvert^{0.8}+5\sin x_i^3)$	-
MO_3	CTP 1	x_1	$(1+x_2)e^{\left(\frac{x_1}{1+x_2}\right)}$	$g_1=\frac{f_2}{0.858e^{(-0.541f_1)}}\geq 1,\quad g_2=\frac{f_2}{0.728e^{(-0.295f_1)}}\geq 1$
MO_4	Constr-Ex	x_1	$\frac{1+x_2}{x_1}$	$g_1=x_2+9x_1\geq 6\,,\quad g_2=-x_2+9x_1\geq 1$
MO_5	Binh and Korn	$4x_1^2+4x_2^2$	$(x_1-5)^2+(x_2-5)^2$	$g_1=(x_1-5)^2+x_2^2\leq 25,$ $g_2=(x_1-8)^2+(x_2+3)^2\geq 7.7$
MO_6	Chankong and Haimes	$2+(x_1-2)^2+(x_2-1)^2$	$9x_1-(x_2-1)^2$	$g_1=x_1^2+x_2^2\leq 225,\quad g_2=x_1-3x_2+10\leq 0$ $20\leq x_1,x_2\leq 20$
MO_7	Osyczka and Kundu	$-25(x_1-2)^2+(x_2-2)^2$ $-(x_3-1)^2-(x_4-4)^2$ $-(x_5-1)^2$	$\sum_{i=1}^{6}x_i^2$	$g_1=x_1+x_2-2\geq 0,\ g_2=6-x_1-x_2\geq 0$ $g_3=2+x_1-x_2\geq 0, g_4=2-x_1+3x_2\geq 0$ $g_5=4-(x_3-3)^2-x_4\geq 0$ $g_6=(x_5-3)^2+x_6-4\geq 0$
MO_8	Four bar plane truss	$L\left(2x_1+x_2\sqrt{2}+\sqrt{x_3}+x_4\right)$	$\frac{FL}{E}\left(\frac{2}{x_1}+\frac{2\sqrt{2}}{x_2}-\frac{2\sqrt{2}}{x_3}+\frac{2}{x_4}\right)$	$\frac{F}{\sigma}\leq x_1,x_4\leq 3\frac{F}{\sigma}$ $\sqrt{2}\frac{F}{\sigma}\leq x_2,x_3\leq 3\frac{F}{\sigma}$ $where\ F=10kN, E=2\times 10^5\,{kN}/{cm^2},$ $L=200cm\,,\sigma=10\,{kN}/{cm^2}$