Doctoral Thesis

# Extended Artificial Bee Colony Algorithms for Continuous Optimization Problems

（連続最適化問題のための拡張人工蜂コロニーアルゴリズム）

Hai Shan

Graduate School of Engineering
Hiroshima University

March 2016

# Acknowledgements

I would like to thank all the people who contributed in some way to the work described in this dissertation.

First and foremost, I would like to express deep gratitude to my academic supervisor, Prof. Ohkura, K. for the support of my doctoral course study and research, for his expertise, motivation, and immense knowledge. He supported my attendance at conferences and provided much guidance on my dissertation. Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Yamada, K., associate Prof. Iwamoto, T. and associate Prof. Matsumura, Y., for their insightful comments and professional guidance.

I would like to thank assistant Prof. Yasuda, T. for his playing crucial role in my research and life, for support with his positive energy and helpful comments. I am also thankful to the other members of my laboratory, Yu Tian, Kadota, Yufei Wei, Morikawa, Adachi, Murakawa and etc. for their cooperation, comments and encouragement.

Finally, I would like to acknowledge family and friends who supported me during my time in Japan. My special appreciation goes to my wife Chun Xia for her understanding, patience and love, parents Wang Zhu and Lao Yatu for their eternal love, brothers Li Chen, Zhao Nasitu, Namula, Wu Ying, Hai Bo and their wife and children for the encouragement, financial support and love. They supported me through my entire life, I am always very appreciated to them. I also thank all my dear friends who gave me very much support and kindness.

Hai Shan
March 10, 2016
Hiroshima, Japan

# Contents

# Chapter 1

# Introduction

## 1.1 Background

Optimization is everywhere and it is such an important paradigm by itself with a wide range of applications. For most of the applications in engineering and industry, the minimal cost and energy consumption, or maximal profit, output, performance, and efficiency are the parameters to be optimized. In reality, resources, time, and money are always limited, consequently, optimization is very important in practice (Yang, 2010b; Yang and Koziel, 2011). Many real-world problems may be formulated as optimization problems of parameters with variables in continuous optimization problems.

Optimization is collection of mathematical principles and methods used for solving quantitative problems in many disciplines. The quantitative problems in different disciplines have important mathematical elements in common. The development of optimization techniques has paralleled advances not only in computer science but also in operations research, numerical analysis, game theory, mathematical economics, control theory, and combinatorics.

In mathematics and computer science, an optimization problem is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a discrete or combinatorial optimization problem. The other one is mostly used problem as continuous optimization problem. In

continuous optimization, the variables in the model are nominally allowed to take on a continuous range of values, usually real numbers. This feature distinguishes continuous optimization from discrete or combinatorial optimization, in which the variables may be binary, integer or more abstract objects drawn from sets with finitely many elements.

The most widely used model for continuous optimization problem $R = (S, \omega, f)$ is composed of the following parts:

- $S$ is a search space on the set of $n$ continuous variables
- $\omega$ is a set of constraints among the variables
- $f$ is the objective function to be minimized: $S \in \mathfrak{R}^n \rightarrow \mathfrak{R}$

A maximization problem can be expressed by replacing the objective function $f$ by -$f$. It is called an unconstrained optimization problem when $\omega = \emptyset$ on above continuous optimization problem model. When each variable has lower and upper bounds, it becomes bounds constraints optimization problems.

In optimization problem, the best values are determined from all feasible solutions of a given problem; the aim of optimization is to obtain relevant parameter values that enable an objective function to generate the minimum or maximum value (Civicioglu and Besdok, 2013). Continuous optimization problems are usually difficult to solve when they arise from real-world optimization problems (Jones and Pevzner, 2004). The search space relates to complex landscape characteristics such as non-linear, rotated or non-separable objective functions, objective functions may have the features of ill-conditioned or smooth local irregularities and objective functions may have multiple local optima.

Test functions are commonly used to evaluate continuous optimization algorithms. The test functions defined by IEEE Congress on Evolutionary Computation (CEC) are invaluable resources for solving the optimization problems with swarm intelligence (SI) and evolutionary algorithms (EAs) algorithms. The CEC 2013 (CEC'13) (Liang et al., 2013a) test functions were defined by modifying and extending the CEC'05 (Suganthan et al., 2005) test functions. CEC'13 test function are composed of unimodal, multimodal and composition functions. The approaches or algorithms for solving this real parameter single objective optimization were designed without making use of exact equations of the test

functions. As the extended form of CEC'13 test suite, benchmark test functions of CEC 2014 (CEC'14) (Liang et al., 2013b) were developed with several novel features such as novel basic problems, composing test problems by extracting features dimension-wise from several problems, graded level of linkages, rotated trap problems. There are four types of functions defined in CEC'14, namely, unimodal, simple multimodal, hybrid and composition functions. Effective and efficient optimization algorithms are required to solve those much difficult problems.

The optimization methods started from the classical optimization methods and then developed in many types such as linear programming, non-linear programming, geometric programming, dynamic programming, integer programming and evolutionary programming. Among the optimization methods, downhill simplex method (Nelder and Mead, 1965) as linear programming method, Powell's method (Powell, 1964), Newton's method (Qi and Sun, 1993) and gradient descent method (Burachik et al., 1996) as non-linear programming methods have been widely applied as the mathematical programming optimization methods. In the past decades, different kinds of SI based algorithms or EAs have been designed and applied to solve real-parameter continuous optimization problems. Besides the SI based algorithms and EAs, hybrid evolutionary classical methods and other non-evolutionary methods such as simulated annealing (Johnson and Aragon, 1989) and tabu search (Glover et al., 1995) have been applied to solve the continuous optimization problems.

Swarm Intelligence (SI) is the discipline that deals with natural and artificial systems composed of many individuals which interact each other by collective behavior of decentralized and self-organized form. As discipline of artificial intelligence, SI has attracted the interests of many research scientists from the late 1980s. SI techniques are population based stochastic methods used in continuous optimization problems in which the collective behavior of relatively simple individuals arises from their local interactions with their environment to produce functional global patterns. Bonabeau et al. (1999) defined SI as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies." Their colonies ranging from a few animals to millions of individuals, display fascinating behaviors that combine efficiency with both flexibility and robustness (Camazine et al., 2001). The individuals in

the system are relatively homogenous (i.e., either identical or belonging to a few topologies) and the interactions among the individuals come from the communication with each other directly or through the connections with environments.

Various kinds of SI based algorithms have been proposed since the end of the last century, e.g., particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), ant colony optimization (ACO) (Dorigo et al., 1996) and artificial bee colony (ABC) (Karaboga and Basturk, 2007). Since the computational modeling of SI algorithms were proposed, SI algorithms represented as meta-heuristic approaches are applied for solving function optimization problems and real-world problems in the fields of computer sciences, engineering, economics, bioinformatics, operational research and industries. The effectiveness and efficiency of those algorithms have been testified through experiments.

Evolutionary computing (EC) (Bäck et al., 1997) is an area of computer science that inspired from natural evolution based on Darwinian evolution principles to solve computational problems. In artificial intelligence, EAs is subset of EC, generic population-based meta-heuristic optimization algorithms. Given a population of individuals within some environment that has limited resources, competition for the resources causes natural selection as evolution process, in turn, the individual with higher fitness survive. A set of candidate solutions is provided for a certain problem and selected the better ones for next generation according to quality (fitness) function with higher values. The essential metaphor of EC relates the natural evolution to trail-and-error problem solver.

The widely used forms of evolutionary algorithms (EAs) are genetic algorithm (GA) (Holland, 1975), differential evolution (DE) (Storn and Price, 1997), evolutionary programming (EP) (Fogel, 1966), genetic programming (GP) (Koza, 1992) and evolutionary strategies (ES) (Rechenberg, 1973; Schwefel, 1974). These algorithms are robust, adaptive and have been applied in a wide variety of theoretical and practical problems involving search and optimization tasks.

The ABC algorithm was introduced by Karaboga (2005) as a technical report, then its performance was measured using benchmark optimization functions (Karaboga and Basturk, 2007). The study by Karaboga and Akay (2009) showed that ABC algorithm performed significantly better or at least comparable to other SI based algorithms or EAs such as GA, DE, and PSO algorithms. Many modified versions of ABC algorithms which

modified some parameters or hybridized with another algorithms have been proposed and the convergence performances have been testified by the experiments. These ABC algorithms are superior to other algorithms in terms of its simplicity, flexibility and robustness when solving the continuous optimization problems. Along with the advantages of the improved versions of ABC algorithms, however, a few disadvantages still exist. For example, ABC algorithms have low convergence speeds when they solve unimodal functions, low exploitation abilities, and are also easily trapped in local optima when they solve complex multimodal functions. Inspired by self adaptive mechanism, incorporated with DE and PSO algorithms, an improved hybrid ABC (IHABC) algorithm will be proposed to overcome those disadvantages, which focused on the predominance of hybridizing other algorithms easily. The comparison experiments will be implemented with the parameters setup for IHABC algorithm. More specifically, CEC'13 test suite benchmark problems will be adopted and comparative experiments with IHABC, standard ABC, DE, and PSO algorithms will be implemented.

Furthermore, in order to overcome the drawbacks such as random uniform initialization loses the effectiveness when the dimension size is rather higher or lower fitness individuals have low probability for selected, levy flight-based hybrid ABC (LFHABC) and self adaptive hybrid enhanced ABC (SAHEABC) algorithms will be proposed. LFHABC algorithm will be proposed by using levy flight (Brown et al., 2007; Pavlyukevich, 2007) for initialization, chaotic opposition-based learning (OBL) (Tizhoosh, 2005) for scout bees to combine with IHABC algorithm. SAHEABC algorithm will be then proposed by utilizing the self adaptive mechanism on increasing the probability of lower fitness individuals in onlooker bees and combined it with LFHABC algorithm. The comparative experiments will be implemented for LFHABC and SAHEABC algorithms to demonstrate the effectiveness and efficiency of the proposed extended ABC algorithms. More specifically, the CEC'13 test suite benchmark problems will be adopted to show the performance of LFHABC algorithm, the CEC'14 test suite benchmark problems will be used to testify the performance of SAHEABC algorithm. Finally, comparative experiments using SAHEABC, the standard ABC, state-of-the-art ABC algorithms of BsfABC and IABC, NRGA and SHADE algorithms will be implemented and the comparison results will be discussed.

## 1.2 Goal of the Thesis

As one of the representative SI based algorithms, ABC has been proposed and modified in various versions. Although the modified ABC algorithms have achieved better performance, there are still disadvantages exist. The main goal of this thesis is achieving extended ABC algorithms which have much higher performance compared with other SI based algorithms and EAs. The extended ABC algorithms will be applied to CEC'13 and CEC'14 benchmark test functions seen as continuous optimization problems and verify the convergence performance through comparative experiments.

More specifically, the extended ABC algorithm of IHABC is proposed firstly for improving the exploitation ability and convergence speed. Comparison experiments for IHABC, ABC, DE and PSO algorithms are implemented on the CEC'13 benchmark test problems to testify the effectiveness and efficiency of IHABC algorithm. The extended ABC algorithms of LFHABC and SAHEABC are proposed in order to overcome the drawbacks that the random uniform initialization loses the effectiveness when dimension size is rather higher and lower fitness individuals have low probability for selected. The comparative experiments are implemented for LFHABC and SAHEABC algorithms to confirm the effectiveness and efficiency of the these algorithms. The CEC'13 test suite benchmark problems are adopted by LFHABC algorithm to verify the convergence performance, the CEC'14 test suite benchmark problem is utilized to testify the convergence performance of SAHEABC algorithm. Finally, the competitive performance of SAHEABC algorithm is demonstrated through implementing the comparative experiments with standard ABC, state-of-the-art ABC algorithms of BsfABC and IABC, NRGA and SHADE algorithms.

## 1.3 Structure of the Thesis

The structure of the thesis is organized as follows.

The continuous optimization problems adopted as the benchmark test functions are introduced in Chapter 2. The history review of the test functions are specifically given and the representative benchmark test functions for CEC'13 and CEC'14 are adopted as the benchmark test functions as described in more detailed ways in Section 2.2 and Section

2.3.

The bio-inspried algorithms which include SI based algorithms and EAs are introduced in Chapter 3. Artificial bee colony (ABC) and particle swarm optimization (PSO) as the main SI based algorithms are introduced firstly in Section 3.1.1 and Section 3.1.2. ABC algorithm is proposed based on the introduction of real bees' behavior of self-organization, stigmergy and division of labor. The classical EAs of genetic algorithm (GA), NRGA, differential evolution (DE) and SHADE algorithms are then given the explanations in more detail.

In Chapter 4, the representative modified versions of ABC algorithms and the state-of-the-art ABC algorithms are described firstly and the improved hybrid ABC (IHABC) is proposed in Chapter 4.2, IHABC algorithm is implemented on the CEC'13 benchmark test functions and then comparison experiments which conducted with ABC, IHABC, DE and PSO algorithms are also implemented to testify the effectiveness and efficiency of IHABC algorithm.

Chapter 5 proposes further extended levy flight hybrid ABC (LFHABC) and self adaptive hybrid enhanced ABC (SAHEABC) algorithms. LFHABC is implemented on CEC'13 benchmark test functions and then SAHEABC algorithm is implemented on CEC'14 benchmark test functions which are extended versions of CEC'13. Comparison experiments which conducted by ABC, SAHEABC, IABC, BsfABC, NRGA and SHADE algorithms are also implemented to testify the effectiveness and efficiency of SAHEABC algorithm.

Finally, Chapter 6 makes a conclusion for the thesis.

Following the contents of above mentioned chapters, CEC'13 test functions and 2-dimensional landscape figures, CEC'14 test functions and 2-dimensional landscape figures are described in Appendices A and B respectively.

# Chapter 2

# Continuous Optimization Problems

## 2.1 Background

A comprehensive collection of continuous optimization problems have been presented as benchmark functions to validate and compare different optimization algorithms. The main category for general continuous optimization problems is test functions. Test functions were proposed as artificial problems, can be used to evaluate the performances of optimization algorithms from diverse sides and difficult situations, such as velocity of convergence, precision, robustness and general performance. The test functions have the characteristics of single global minimum, multiple global minima in the presence of many local minima, long narrow valleys, quadratic ill-conditioned and smooth local irregularities. These test functions could be applied to test different algorithms in diverse scenarios (Yang and Koziel, 2011; Yang, 2010b).

The history of the test functions started from middle of last century. The test functions were proposed for evaluating the mathematic optimization and many of the functions are still nowadays utilized to evaluate the performance of continuous optimization problems. The well-known test functions presented as the standard benchmark functions were Rosenbrock function (Rosenbrock, 1960), Griewank function (Griewank, 1981), Ackey function (Ackley, 1987), Rastrigin function (Torn and Zilinskas, 1989), Schwefel function (Schwefel, 1981) and Schaffer function (Schaffer, 1984). These test functions have been used to

evaluate and testify the improvements of optimization algorithms by comparative experiments.

De Jong (1975) proposed the first sets of test functions composed of five functions which represent the common difficulties in optimization problems in an isolated manner. The brief descriptions of these five functions are shown as following:

- Sphere function: is smooth, unimodal and symmetric

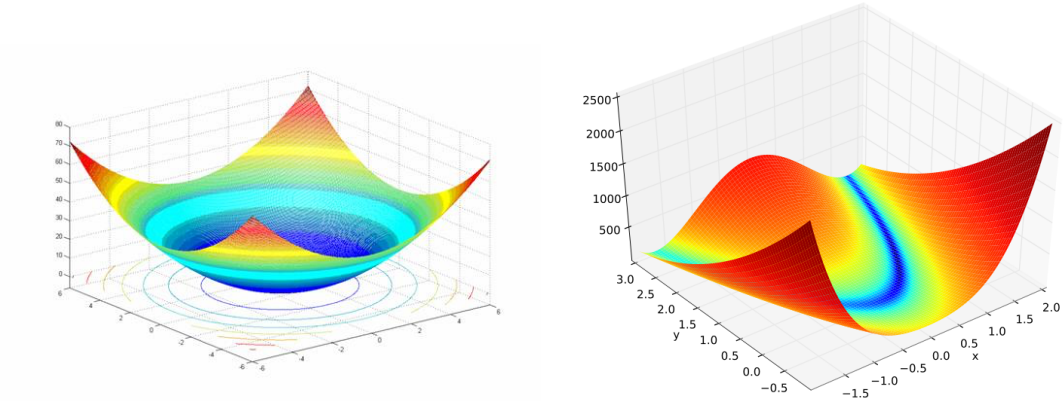
$$f_1(x) = \sum_{i=1}^{D} x_i^2 \tag{2.1}$$

  Where, $-5.11 \leq x_i \leq 5.12$, $\min(f_1(0, ..., 0))=0$, $D$ is the dimensional size.

- Rosenbrock function: having very narrow ridge, and the tip of the ridge is very sharp

$$f_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{2.2}$$

  Where, $-5.11 \leq x_i \leq 5.12$, $\min(f_2)=f_2(1, ..., 1)=0$, $D$ is the dimensional size.




(a) De Jong function $f_1$ (b) De Jong function $f_2$

Figure 2.1: 2-dimensional landscapes for De Jong test functions $f_1$ and $f_2$

- Step function: representative of the problem of flat surfaces which are obstacles for optimization algorithms

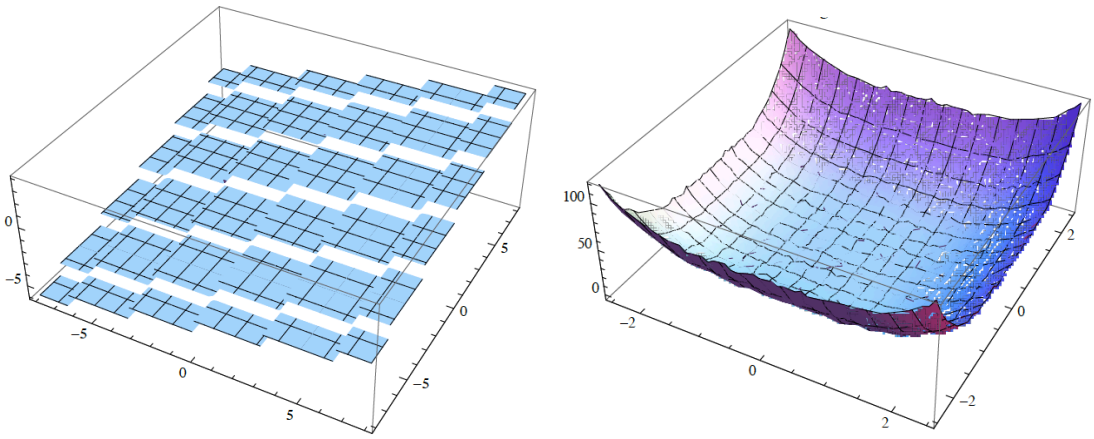
$$f_3(x) = 6 \cdot n + \sum_{i=1}^{5} [x_i] \tag{2.3}$$

Where, $-5.11 \leq x_i \leq 5.12$, min($f_3$)=$f_3$([-5.12, -5), ..., [5.12, 5))=0.

- Quartic function: simple unimodal function padded with noise

$$f_4(x) = \sum_{i=1}^{D} i x_i^4 + gauss(0, 1) \tag{2.4}$$

Where, $-1.27 \leq x_i \leq 1.28$, min($f_4$)=$f_4$(0, ..., 0)=0, $D$ is the dimensional size.




(a) De Jong $f_3$ (b) De Jong $f_4$

Figure 2.2: 2-dimensional landscapes for De Jong test functions $f_3$ and $f_4$

- Foxholes function: having many local optima

$$f_5(x) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \tag{2.5}$$

Where, $-65.535 \leq x_i \leq 65.536$,

$(a_{ij})$=

$$\begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 9 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$$

$\min(f_5)=f_5(-32, 32) \approx 1.$

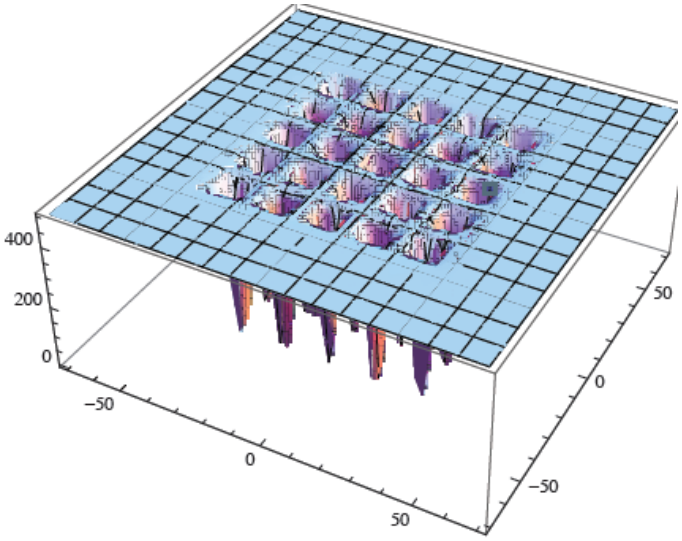


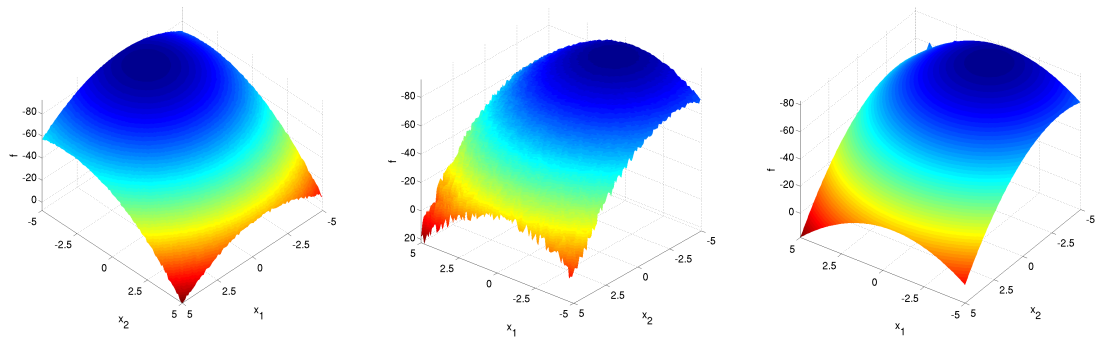

Figure 2.3: 2-dimensional landscapes for De Jong test function $f_5$

These test functions were used to test the performance of GA firstly and then adopted by various optimization algorithms to verify the effectiveness. Following De Jong's test functions, Bersini et al. (1996) organized the first international contest on evolutionary optimization (ICEO) at the IEEE international conference on evolutionary computation. A benchmark set of five functions was provided with different characteristics of uni-modality, multi-modality, separability and non-separability for different dimension sizes of five and ten. The different algorithms for the attendants were compared on the continuous optimization problems. Moreover, for testing the different algorithms on combinatorial problems, classical TSP problem composed of a set of small and large, symmetric and asymmetric instances was proposed and the comparison experiments were implemented.

Benchmark procedures for continuous optimization problems have been developed widely since the genetic and evolutionary computation conference (GECCO) and the congress on evolutionary computation (CEC) which devoted to test the reliability, efficiency and validation of optimization algorithms.

GECCO was set up to benchmark both stochastic and deterministic continuous optimization algorithms when Black Box Optimization Benchmarking (BBOB) workshops

were held in 2009, 2010, 2012 and 2013 (Hansen, 2009; El-Abd, 2010). GECCO presents the latest high-quality results in the growing field of genetic and evolutionary computation such as GA, GP, ES, EP, ACO, biological applications and more. Current BBOB functions include 24 noise-free real-parameter single objective benchmarking functions which have the characteristics of separable, low or moderate conditioning, high conditioning and unimodal, multi-modal with adequate or weak global structure. All benchmark functions are scalable with the dimension. Most functions have no specific value of their optimal solution. There are also 30 noisy functions included in BBOB functions which composed of three kinds of noise functions, namely, moderate noise, severe noise and highly multi-modal with severe noise and the description of these noise functions are shown as Table 2.1. The benchmark functions of BBOB 2010 with noise are described as referred to Finck et al. (2010). In this benchmarking suite, three different noise models are used, namely, gaussian noise, uniform noise and cauchy noise. Following the introduction of the functions shown as in Table 2.1, the 2-dimensional landscapes of these functions are illustrated as Figures 2.4-2.13. In these figures, functions F1.1-F1.3 and F2.1-F2.3 are sphere and rosenbrock functions with moderate gaussian, moderate uniform and moderate noises; functions F3.1-F3.3 and F7.1-F7.3 are sphere, rosenbrock, step ellipsoid, ellipsoid and different powers functions with moderate gaussian, moderate uniform and moderate noises; functions F8.1-F8.3 and F10.1-F10.3 are schaffer's F7, composite griewank-rosenbrock, gallagher's gaussian peaks 101-me functions with moderate gaussian, moderate uniform and moderate noises corresponding to the functions as shown in Table 2.1.

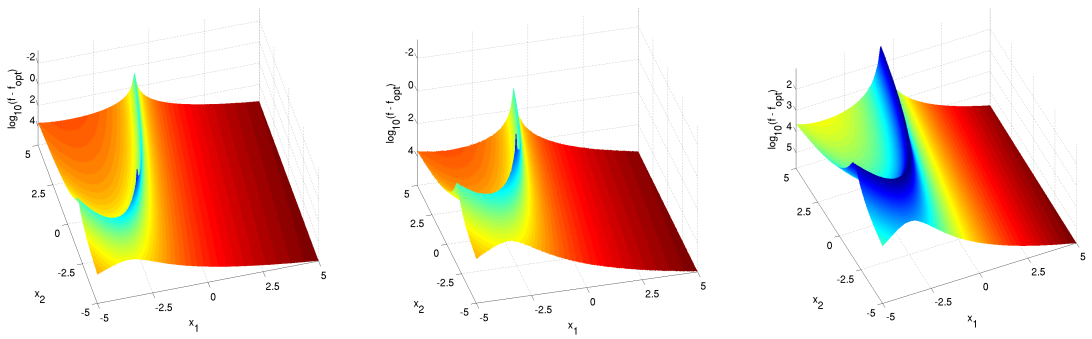


(a) F1.1 (b) F1.2 (c) F1.3

Figure 2.4: 2-dimensional landscapes for functions F1.1, F1.2 and F1.3

Table 2.1: BBOB 2010 Noisy Functions

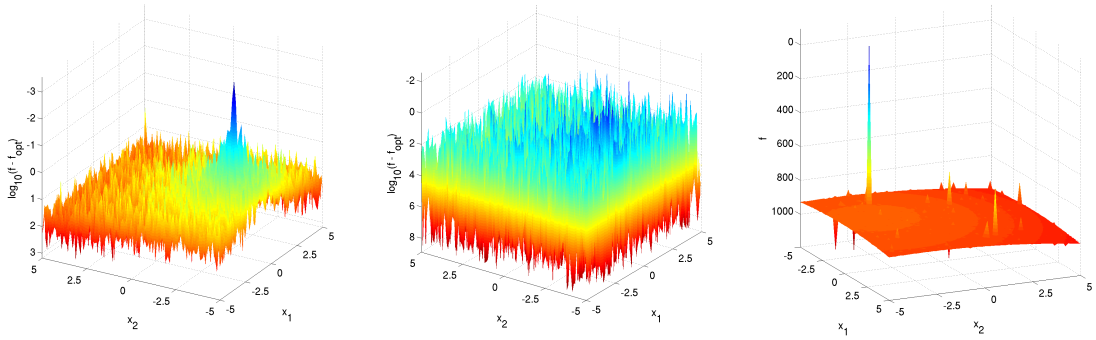
| **Func. Type** | **Func. Name** |
|---|---|
| Moderate noise | Sphere with moderate gaussian, uniform and seldom cauchy noise<br>Rosenbrock with moderate gaussian, uniform and seldom cauchy noise |
| Severe noise | Sphere with gaussian, uniform and seldom cauchy noise<br>Rosenbrock with gaussian, uniform and seldom cauchy noise<br>Step ellipsoid with gaussian, uniform and seldom cauchy noise<br>Ellipsoid with gaussian, uniform and seldom cauchy noise<br>Different Powers with gaussian, uniform and seldom cauchy noise |
| Severe noise for multi-modal | Schaffer's F7 with gaussian, uniform and seldom cauchy noise<br>Composite Griewank-Rosenbrock with gaussian, uniform, seldom cauchy noise<br>Gallagher Gaussian Peaks 101-me with gaussian, uniform, seldom cauchy noise |



(a) F2.1 (b) F2.2 (c) F2.3

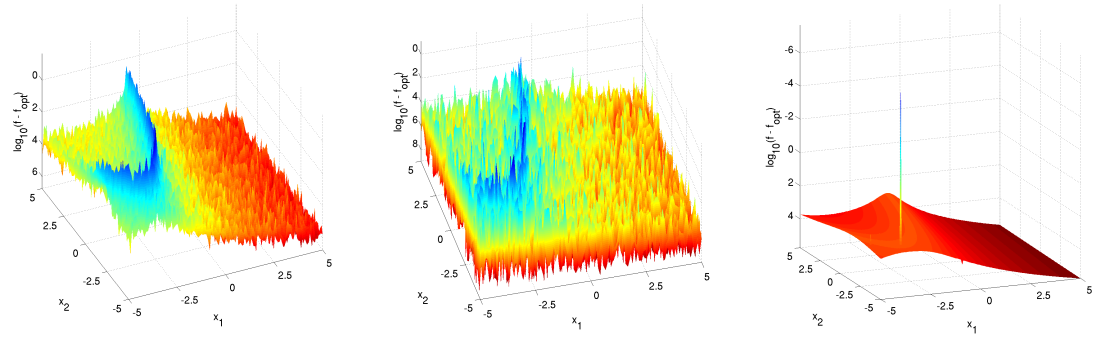
Figure 2.5: 2-dimensional landscapes for functions F2.1, F2.2 and F2.3

Availability of the benchmark of CEC provides the platform for comparing new optimization algorithms to the state-of-the-art ones. CEC benchmark is the widely used for evaluating the test functions which cover a wide range of specialized optimization problems. CEC'05 (Suganthan et al., 2005) was proposed for the special session on real parameter optimization of CEC 2005. CEC'05 benchmark test functions are composed of 25 functions including 5 unimodal and 20 multi-modal functions, each of which is freely scalable. It specifies problem dimensionality, defines the domains of variables, termination



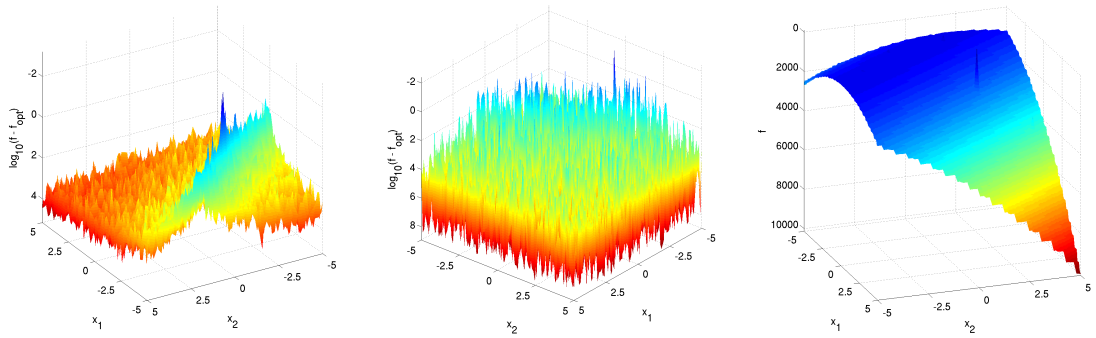
(a) F3.1 (b) F3.2 (c) F3.3

Figure 2.6: 2-dimensional landscapes for functions F3.1, F3.2 and F3.3



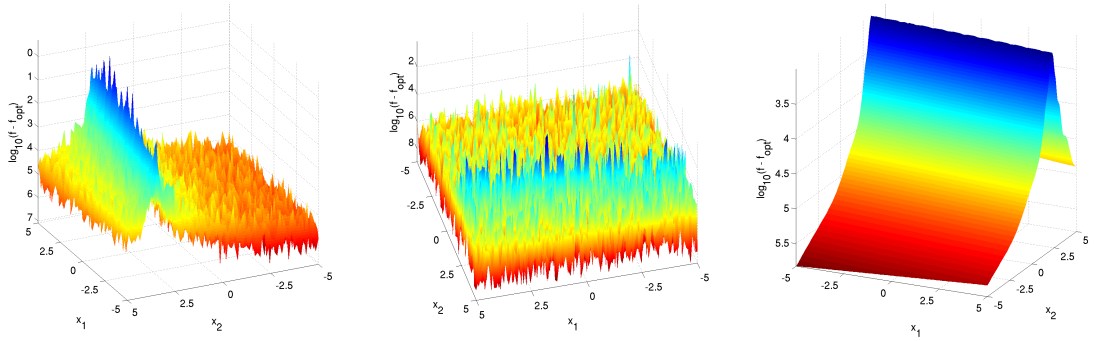
(a) F4.1 (b) F4.2 (c) F4.3

Figure 2.7: 2-dimensional landscapes for functions F4.1, F4.2 and F4.3



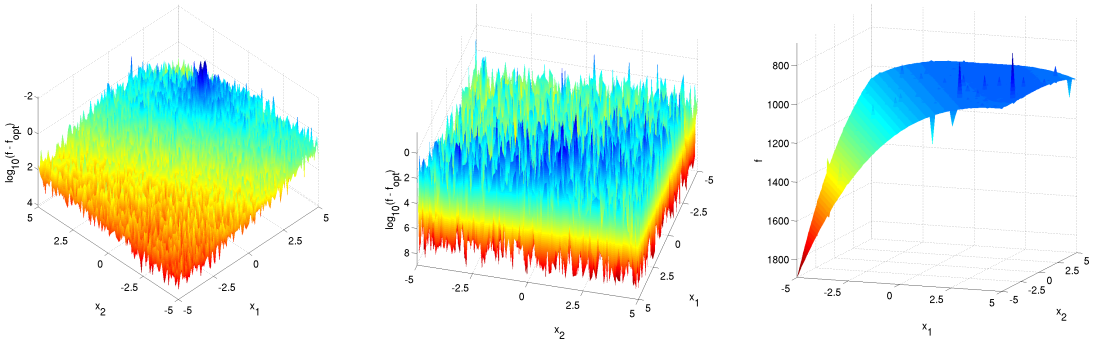
(a) F5.1 (b) F5.2 (c) F5.3

Figure 2.8: 2-dimensional landscapes for functions F5.1, F5.2 and F5.3



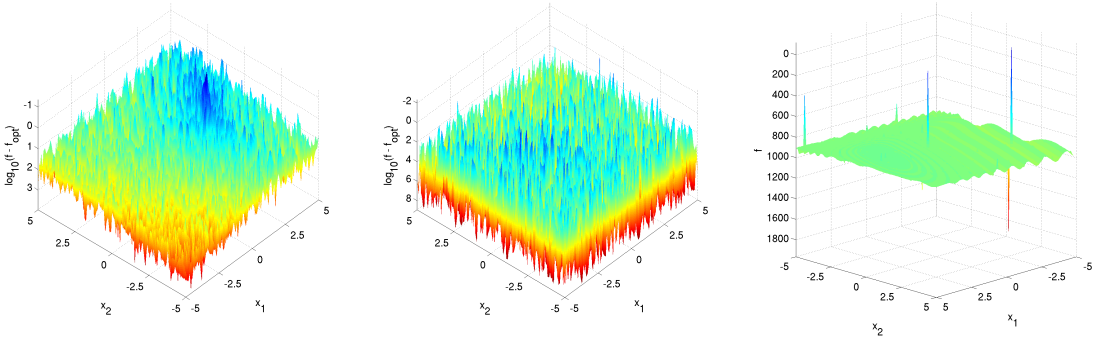

(a) F6.1 (b) F6.2 (c) F6.3

Figure 2.9: 2-dimensional landscapes for functions F6.1, F6.2 and F6.3



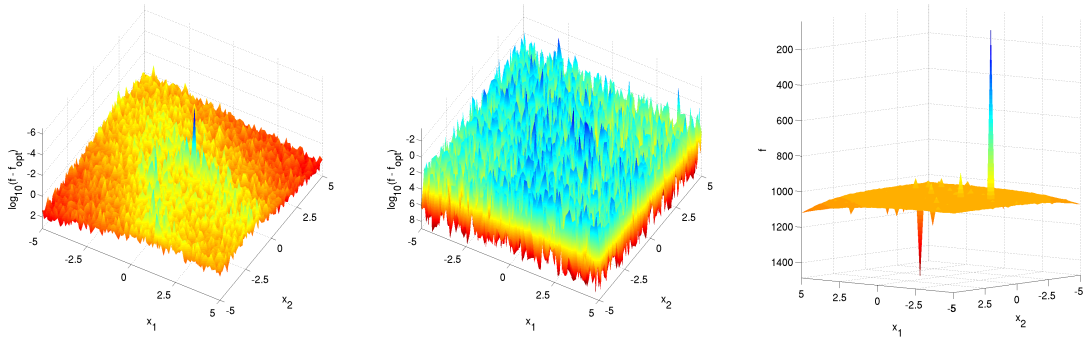

(a) F7.1 (b) F7.2 (c) F7.3

Figure 2.10: 2-dimensional landscapes for functions F7.1, F7.2 and F7.3



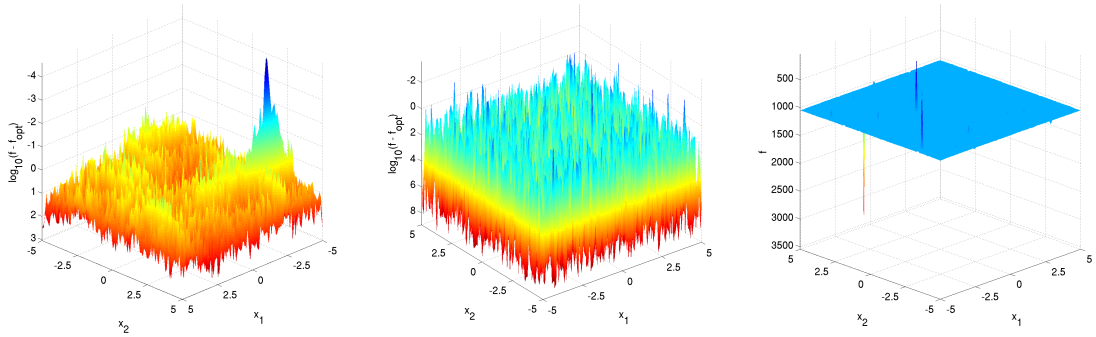

(a) F8.1 (b) F8.2 (c) F8.3

Figure 2.11: 2-dimensional landscapes for functions F8.1, F8.2 and F8.3



(a) F9.1 (b) F9.2 (c) F9.3

Figure 2.12: 2-dimensional landscapes for functions F9.1, F9.2 and F9.3



(a) F10.1 (b) F10.2 (c) F10.3

Figure 2.13: 2-dimensional landscapes for functions F10.1, F10.2 and F10.3

error values and the maximum number of function evaluations for the optimization algorithms. The published research papers using CEC'05 benchmark function set currently can be found more than 1257 citations in google scholar (as of December 2015).

CEC'13 and CEC'14 were proposed by extending CEC'05 in more diverse and complex types. These benchmark functions play an important role in the assessment of the state-of-the-art algorithms in the continuous optimization fields. CEC'13 and CEC'14 are adopted as the efficient benchmark test functions to evaluate the extended ABC algorithms proposed in Chapter 4 and Chapter 5.

Continuous optimization problems are typically solved using algorithms that generate a sequence of values of the variables, known as iterates, that converge to a solution of the

problem. In deciding how to step from one iterate to the next, the algorithm makes use of knowledge gained at previous iterates, and information about the model at the current iterate, possibly including information about its sensitivity to perturbations in the variables. The continuous nature of the problem allows sensitivities to be defined in terms of first and second derivatives of the functions that define the models. The aim of optimization is to obtain the relevant parameter values that enable an objective function to generate the minimum or maximum value (Civicioglu and Besdok, 2013).

## 2.2 Benchmark Test Functions for CEC'13

Single objective optimization algorithms are the basis of the more complex optimization algorithms such as multi-objective or constrained optimization algorithms. Research on the single objective optimization algorithms will contribute the development of the optimization algorithms in more complicated ways. In the recent years, various kinds of novel optimization algorithms have been proposed to solve real-parameter optimization problems. CEC'05 and CEC'13 test suites are the widely used benchmark test functions for real-parameter optimization problems.

CEC'13 test suite plays an important role in swarm intelligence and evolutionary computation fields because it provides a set of 28 benchmark functions with different properties that can be used to evaluate the performance of various algorithms or methods and compare those algorithms in more systematic manner by utilizing the size of problems, dimensionality of the functions, termination criteria and initialization scheme. All the test functions are minimization problems, the aim is to minimize the objective function f(x), where, $x = [x_1, x_2, ..., x_D]^T$, $D$ is the number of dimension size. The 28 numerical test functions are minimization problems classified by function characteristics into the following three groups, namely, unimodal functions, multi-modal functions and composition functions. The definitions of basic functions for these test functions and the properties of these test functions are described as following and the CEC'13 test functions and the corresponding 2-dimensional landscape figures are given in Appendix A.

1). Unimodal functions (F1-F5): the basic functions $f_1(x)$ through $f_5(x)$ are shown as the following Eqs. (2.6)-(2.10).

- Sphere function: Separable. $D$ indicates the dimensional size.

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \tag{2.6}$$

- Rotated High Conditioned Elliptic function: Non-separable, quadratic ill-conditioned, smooth local irregularities. High Conditioned Elliptic function is shown as Eq. (2.7). $D$ indicates the dimensional size.

$$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2 \tag{2.7}$$

- Rotated Bent Cigar function: Non-separable, smooth but narrow ridge. Bent Cigar function is shown as Eq. (2.8). $D$ indicates the dimensional size.

$$f_3(x) = (x_1)^2 + 10^6 \sum_{i=2}^{D} (x_i)^2 \tag{2.8}$$

- Rotated Discus function: Non-separable, asymmetrical, smooth local irregularities, with one sensitive direction. Discus function is shown as Eq. (2.9). $D$ indicates the dimensional size.

$$f_4(x) = 10^6 (x_1)^2 + \sum_{i=2}^{D} (x_i)^2 \tag{2.9}$$

- Different Powers function: Separable, sensitivities of the $z_i$-variables are different. Powers function is shown as Eq. (2.10). $D$ indicates the dimensional size.

$$f_5(x) = \sqrt{\sum_{i=1}^{D} |x_i|^{2+4\frac{i-1}{D-1}}} \tag{2.10}$$

2). multi-modal functions (F6-F20): the basic functions $f_6(x)$ through $f_{15}(x)$ are shown as the following Eqs. (2.11)-(2.20).

- Rotated Rosenbrock function: Non-separable, having a very narrow valley from local optimum to global optimum. Rosenbrock function is shown as Eq. (2.11). $D$ indicates the dimensional size.

$$f_6(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{2.11}$$

- Rotated Schaffers F7 function: Non-separable, asymmetrical, local optima's number is huge. Schaffers F7 function is shown as Eq. (2.12). $D$ indicates the dimensional size.

$$f_7(x) = (\frac{1}{D-1}\sum_{i=1}^{D-1}(\sqrt{z_i} + \sqrt{z_i}sin^2(50z_i^{0.2})))^2 \tag{2.12}$$

Where, $z_i = \sqrt{x_i^2 + x_{i+1}^2}$ for $i = 1, 2, ..., D$

- Rotated Ackley function: Non-separable, asymmetrical. Ackley function is shown as Eq. (2.13). $D$ indicates the dimensional size.

$$f_8(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D}cos(2\pi(x_i))) + 20 + e \tag{2.13}$$

- Rotated Weierstrass function: Non-separable, asymmetrical, continuous but differentiable only on a set of points. Weierstrass function is shown as Eq. (2.14). $D$ indicates the dimensional size.

$$f_9(x) = \sum_{i=1}^{D}(\sum_{k=0}^{20}[0.5^k cos(2\pi 3^k(x_i + 0.5))]) - D\sum_{k=0}^{20}[0.5^k cos(2\pi b^k \cdot 0.5)] \tag{2.14}$$

- Rotated Griewank function: Rotated, non-separable. Griewank function is shown as Eq. (2.15). $D$ indicates the dimensional size.

$$f_{10}(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{2.15}$$

- Rastrigin function: Separable, asymmetrical, local optima's number is huge. Rastrigin function is shown as Eq. (2.16). $D$ indicates the dimensional size.

$$f_{11}(x) = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10) \tag{2.16}$$

- Rotated Rastrigin function: Non-separable, asymmetrical, local optima's number is huge.

- Non-Continuous Rotated Rastrigin function: Rotated, non-separable, asymmetrical, local optima's number is huge, non-continuous.

- Schwefel function: Rotated, non-separable, asymmetrical, local optima's number is huge and second better local optimum is far from the global optimum. Schwefel function is shown as Eq. (2.17). $D$ indicates the dimensional size.

$$f_{12}(x) = 418.9829 \times D - \sum_{i=1}^{D} g(x_i) \tag{2.17}$$

$$g(x_i) = \begin{cases} x_i sin(|x_i|^{1/2}), & if \quad |x_i| \leq 500 \\ (500 - mod(x_i, 500))sin(\sqrt{|500 - mod(x_i, 500)|}) - \frac{(x_i - 500)^2}{10000D}, & if \quad x_i > 500 \\ (mod(|x_i|, 500) - 500))sin(\sqrt{|mod(|x_i|, 500) - 500|}) - \frac{(x_i + 500)^2}{10000D}, & if \quad x_i < -500 \end{cases}$$

- Rotated Schwefel function: Non-separable, asymmetrical, local optima's number is huge and second better local optimum is far from the global optimum

- Rotated Katsuura function: Non-separable, asymmetrical, continuous everywhere yet differentiable nowhere. Katsuura function is shown as Eq. (2.18). $D$ indicates the dimensional size.

$$f_{13}(x) = \frac{10}{D^2}\prod_{i=1}^{D}(1 + i\sum_{j=1}^{32}\frac{|2^j x_i - round(2^j x_i)}{2^j})^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} \tag{2.18}$$

- Rotated Lunacek Bi_Rastrigin function: Non-separable, asymmetrical, continuous everywhere yet differentiable nowhere

- Expanded Griewank's Plus Rosenbrock function: Non-separable. Expanded Griewank's Plus Rosenbrock function is shown as Eq. (2.19).

$$f_{14}(x) = f_7(f_4(x_1, x_2)) + f_7(f_4(x_2, x_3)) + ... + f_7(f_4(x_{D-1}, x_D)) + f_7(f_4(x_D, x_1)) \tag{2.19}$$

- Expanded Scaffer F6 function: Non-separable, asymmetrical. Expanded Scaffer F6 function is shown as Eq. (2.20).

$$f_{15}(x) = g(x_1, x_2) + g(x_2, x_3) + ... + g(x_{D-1}, x_D) + g(x_D, x_1) \tag{2.20}$$

Where, $g(x, y) = 0.5 + \frac{sin^2(\sqrt{x^2+y^2})-0.5}{(1+0.001(x^2+y^2))^2}$

3). Composition functions (F21-F28): composed of several basic functions as shown in Eqs. (2.21) and (2.22).

Comparing with the previous composition functions in CEC'05, the composition functions in CEC'13 merge the properties of the sub-functions better and keeps continuous around the global or local optima. The composition functions are given as Eqs. (2.21) and (2.22), where, F(x) is new composition function, $g_i$ is $i$th basic function used to construct the composition function defined as $F_i$'=$F_i - F_i^*$, $n$ is the number of basic functions, $bias_i$ defines which optimum is global optimum, $o_i$ is the new shifted optimum position for each $g_i(x)$, $\sigma_i$ and $\lambda_i$ are used to control each $g_i(x)$'s coverage range and height.

$$F(x) = \sum_{i=1}^{n}\{\omega_i * [\lambda_i g_i(x) + bias_i]\} + F^* \tag{2.21}$$

$w_i$ is the weight value for each $g_i(x)$ defined as following equation and then the weight $\omega_i = \frac{w_i}{\sum_{i=1}^{n} w_i}$ will be normalized.

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{D}(x_j - o_{ij})^2}} exp(-\frac{\sum_{j=1}^{D}(x_j - o_{ij}^2)}{2D\sigma_i^2}) \tag{2.22}$$

- Composition function 1 (n=5,Rotated)
- Composition function 2 (n=3,Unrotated)
- Composition Function 3 (n=3,Rotated)
- Composition Function 4 (n=3,Rotated)
- Composition Function 5 (n=3,Rotated)
- Composition Function 6 (n=5,Rotated)
- Composition Function 7 (n=5,Rotated)
- Composition Function 8 (n=5,Rotated)

The detailed description of the CEC'13 test suite is available in Liang et al. (2013a). CEC'13 test functions $F_1$ - $F_{20}$ are composed of $f_1$ - $f_{17}$. The function error values are defined as $F_i^* - F_i(x^*)$ and these values are calculated when the experiments implemented. The test functions defined by CEC'13 are listed on Table 2.2, the global optimal values are given on the right column of this table.

The bound constraints are used to define the lower and upper bounds on the values for each variable. In the case of the bound constraints, a solution $S \in \mathbf{S}$ is often given by $S = (x_1, x_2, ..., x_D)$ and $x_i \in [A_i, B_i]$, where $[A_i, B_i]$ is the search interval of dimension $i$, where $i \in 1, 2, ..., D$. Particularly, if all variables are within the same range, all the bound constraints change between $x_i \in [A, B]$. Bound constraints arise in many practical problems, and are frequent in the benchmark problems for continuous optimization that currently play an important role in the evaluation of continuous optimization algorithms.

Given $o = [o_1, o_2, ..., o_D]^T$, the shifted global optimum distributed randomly in the range $[-80, 80]^D$, all test functions are shifted to $o$ and are scalable. For convenience, the same bound constraint ranges are defined as $[-100, 100]^D$ for all test functions.

Table 2.2: CEC 2013 Test Functions

| Type | Func. No. | Func. Name | $F_i^* = F_i(x^*)$ |
|---|---|---|---|
| Unimodal | F1 | Sphere | -1400 |
| | F2 | Rotated High Conditioned Elliptic | -1300 |
| | F3 | Rotated Bent Cigar | -1200 |
| | F4 | Rotated Discus | -1100 |
| | F5 | Different Powers | -1000 |
| Multi-modal | F6 | Rotated Rosenbrock | -900 |
| | F7 | Rotated Schaffers F7 | -800 |
| | F8 | Rotated Ackley | -700 |
| | F9 | Rotated Weierstrass | -600 |
| | F10 | Rotated Griewank | -500 |
| | F11 | Rastrigin | -400 |
| | F12 | Rotated Rastrigin | -300 |
| | F13 | Non-Continuous Rotated Rastrigin | -200 |
| | F14 | Schwefel | -100 |
| | F15 | Rotated Schwefel | 100 |
| | F16 | Rotated Katsuura | 200 |
| | F17 | Lunacek Bi_Rastrigin | 300 |
| | F18 | Rotated Lunacek Bi_Rastrigin | 400 |
| | F19 | Expanded Griewank's Plus Rosenbrock | 500 |
| | F20 | Expanded Scaffer F6 | 600 |
| Composition | F21 | Composition Function 1 (n=5,Rotated) | 700 |
| | F22 | Composition Function 2 (n=3,Unrotated) | 800 |
| | F23 | Composition Function 3 (n=3,Rotated) | 900 |
| | F24 | Composition Function 4 (n=3,Rotated) | 1000 |
| | F25 | Composition Function 5 (n=3,Rotated) | 1100 |
| | F26 | Composition Function 6 (n=5,Rotated) | 1200 |
| | F27 | Composition Function 7 (n=5,Rotated) | 1300 |
| | F28 | Composition Function 8 (n=5,Rotated) | 1400 |

## 2.3 Benchmark Test Functions for CEC'14

CEC'14 benchmark test functions are developed based on the definition of CEC'13 test functions with several novel features which are novel basic problems, composing test problems by extracting features dimension-wise from several problems, graded level of linkages, rotated trap problems, and so on (Liang et al., 2013b).

There are some differences for CEC'14 test suite from CEC'13 test suite: each function has a shift data shown as in $o_{ii} = [o_{i1}, o_{i2}, ..., o_{iD}]^T$; rotation matrixes are assigned to each function and each basic function, the rotation matrix for each subcomponents are generated from standard normally distributed entries by Gram-Schmidt orthonormalization with condition number $c$ that is equal to 1 or 2; CEC'14 test suite has 6 hybrid functions additionally. In order to testify the effectiveness and efficiency of the proposed extended ABC algorithms in this thesis, CEC'14 is introduced and utilized by the comparative experiments.

CEC'14 benchmark test functions are classified to four groups by function properties, namely, unimodal functions, simple multi-modal functions, hybrid functions and composition functions, however, the first two groups and the last group are similar as the test functions defined by CEC'13. The definitions of basic functions for these test functions and the properties of these test functions are described as the previous Section 2.2 and the additional basic functions of HappyCat and HGBat are described as Eqs. (2.23) and (2.24). TheCEC'14 test functions and the corresponding 2-dimensional landscape figures are given in Appendix B.

- Unimodal functions (F1-F3)
- Simple multi-modal functions (F4-F16)
- Hybrid functions (F17-F22)
- Composition functions (F23-F30)

The basic additional functions $f_{16}$ and $f_{17}$ used by simple multi-modal functions $F_{13}$ and $F_{14}$ are defined as Eqs. (2.23) and (2.24).

• HappyCat function:

$$f_{16}(x) = |\sum_{i=1}^{D} x_i^2 - D|^{1/4} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5 \tag{2.23}$$

• HGBat Function:

$$f_{17}(x) = |(\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2|^{1/2} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i}^{D} x_i)/D + 0.5 \tag{2.24}$$

CEC'14 benchmark test functions are listed on Table 2.3. CEC'14 test functions and the 2-dimensional landscape figures are given in Appendix B.

The hybrid function F(x) is given as Eq. (2.25), where, $g_i(x)$ is $i$th basic function used to construct the hybrid function, $N$ is the number of basic functions, $z = [z_1, z_2, ..., z_N]$, $z_1 = [y_{S_1}, y_{S_2}, ..., y_{S_{n_1}}], z_2 = [y_{S_{n_1+1}}, y_{S_{n_1+2}}, ..., y_{S_{n_1+n_2}}], ..., z_N = [y_{S_{\sum_{i=1}^{N-1} n_i+1}}, y_{S_{\sum_{i=1}^{N-1} n_i+2}}, ..., y_{S_D}]$, $y = x - o_i$, $S = randperm(1 : D)$, $p_i$ is used to control the percentage of $g_i(x)$, $n_i$ is the dimension for each basic function $\sum_{i=1}^{N} n_i = D$, $n_1 = p_1 D$, $n_2 = p_2 D$, ..., $n_{N-1} = p_{N-1} D$, $n_N = D - \sum_{i=1}^{N-1} n_i$.

$$F(x) = g_1(M_1 z_1) + g_2(M_2 z_2) + ... + g_N(M_N z_N) + F^*(x) \tag{2.25}$$

The composition functions are given as Eqs. (2.21) and (2.22), where, functions $g_i$ are defined as $F_i$'=$F_i - F_i^*$.

## 2.4 Summary

The review for the history of test functions were described, the brief history of test functions and the test function suites were then analyzed. Availability of the benchmark of CEC provides the platform for comparing new optimization algorithms to the state-of-the-art ones. CEC benchmark is the widely used for evaluating the test functions which cover a wide range of specialized optimization problems. The representative benchmark test functions of CEC'13 and CEC'14 which were introduced based on the CEC'05 test functions were given the definitions in more detailed ways.

Table 2.3: CEC 2014 Test Functions

| **Type** | **Func. No.** | **Func. Name** | $F_i^* = F_i(x^*)$ |
|---|---|---|---|
| Unimodal | F1 | Rotated High Conditioned Elliptic | 100 |
| | F2 | Rotated Bent Cigar | 200 |
| | F3 | Rotated Discus | 300 |
| Multi-modal | F4 | Shifted and Rotated Rosenbrock | 400 |
| | F5 | Shifted and Rotated Ackley | 500 |
| | F6 | Shifted and Rotated Weierstrass | 600 |
| | F7 | Shifted and Rotated Griewank | 700 |
| | F8 | Shifted Rastrigin | 800 |
| | F9 | Shifted and Rotated Rastrigin | 900 |
| | F10 | Shifted Schwefel | 1000 |
| | F11 | Shifted and Rotated Schwefel | 1100 |
| | F12 | Shifted and Rotated Katsuura | 1200 |
| | F13 | Shifted and Rotated HappyCat | 1300 |
| | F14 | Shifted and Rotated GHBat | 1400 |
| | F15 | Expanded F4 plus F7 | 1500 |
| | F16 | Shifted and Rotated Expanded Scaffer's F6 | 1600 |
| Hybrid | F17 | Hybrid Function 1 (N=3) | 1700 |
| | F18 | Hybrid Function 2 (N=3) | 1800 |
| | F19 | Hybrid Function 3 (N=4) | 1900 |
| | F20 | Hybrid Function 4 (N=4) | 2000 |
| | F21 | Hybrid Function 5 (N=5) | 2100 |
| | F22 | Hybrid Function 6 (N=5) | 2200 |
| Composition | F23 | Composition Function 1 (N=5) | 2300 |
| | F24 | Composition Function 2 (N=3) | 2400 |
| | F25 | Composition Function 3 (N=3) | 2500 |
| | F26 | Composition Function 4 (N=5) | 2600 |
| | F27 | Composition Function 5 (N=5) | 2700 |
| | F28 | Composition Function 6 (N=5) | 2800 |
| | F29 | Composition Function 7 (N=3) | 2900 |
| | F30 | Composition Function 8 (N=3) | 3000 |

# Chapter 3

# Bio-inspired Algorithms

## 3.1 Swarm Intelligence (SI) Algorithms

In the last few decades, many algorithms were proposed by mimicking the successful characteristics of the complex systems inspired from nature. Although the nature-inspired algorithms are in their early-stage with short history, they have great potential properties of flexibility, effectiveness and efficiency compared with traditional methods such as steepest decent, linear programming and dynamic programming which generally fail to solve such large-scale problems especially with nonlinear objective functions. By far the majority of nature-inspired algorithms are based on the successful characteristics of biological systems, therefore, the largest fraction of nature-inspired algorithms are biology-inspired, or bio-inspired for short (Fister et al., 2013).

Bio-inspired algorithms are problem solving approaches based on the structure and functioning of complex natural systems in an adaptable reactive distributed manner. They are population based algorithms, usually bottom-up, decentralized approaches (Ding, 2009) which provide simple solutions to complex problems that would be hard for traditional computing approaches to solve. Bio-inspired algorithms also have good performance for the tasks with poorly defined, have improvement scope and innovation for largely unexplored fields.

A common feature of all population-based algorithms is that the population consisting of possible solutions to the problem is modified by applying some operators on the

solutions depending on the information of their fitness. Hence, the population is moved towards better solution areas of the search space. Two important classes of population based optimization algorithms are SI based algorithms and EAs.

SI based algorithms have been developed by drawing inspiration from the behavior of some social living beings, such as ant or termite colonies, bird flocks, fish schools and honey bees. These social insects or animals are composed of multiple individuals and the individuals are relatively homogeneous; the interactions among the individuals are based on the collective behaviors of decentralized and self-organization.

The fundamental principles for SI are described as follows (Mahale and Chavan, 2012).

- Proximity principle: the population should have the capablity of simple computation related to its surrounding environment;
- Quality principle: the population should be able to respond to quality factors such as food and safety in the environment;
- Diverse response principle: the population should not be concentrated in narrow region. The distribution should be designed so that each agent will be protected maximally with environmental fluctuations;
- Stability and adaptability principle: the population should not change its mode of behavior every time the environment changes because mode changing costs energy.

The agents in SI based algorithms follow very simple rules although there is no centralized control structure instructing how individual agents should behave. The interactions among such agents display impressive cognitive abilities. Instead of being centrally controlled and regulated by the orders of a single well-informed individual, taking the social insect colonies as an example, most of them operate through a process of self-organization, in which each worker's actions are governed by a set of relatively simple behavioral rules. Most of the complex patterns and behaviors of insect colonies are emergent properties of the actions of individual workers, and are not controlled by any single individual (Moussaid et al., 2009). Individual social insects organize and coordinate their actions by the mechanism of stigmergy (Bonabeau et al., 1999). Stigmergy is a mechanism of spontaneous, indirect coordination among agents or actions that influence collective behavior intelligently

(Marsh and Onof, 2008). In order to effectively coordinate their behaviors, individuals within a colony must have some means of communicating with each other. Stigmergy allows the communication among workers to take place indirectly, by means of environmental cues, thereby eliminating the need for direct interactions between individuals. The detail statements of self-organization and stigmergy will be given in Section 3.1.1.1. There are many phenomena of the behaviors through self-organization, for example, nest building and forging in honey bees, food transportation in ant colony, fish schooling and bird flocking.

Representative SI based algorithms are ABC, PSO and ACO algorithms. In addition, many other SI based algorithms have been proposed such as cuckoo search (CS) (Yang and Deb, 2009), firefly algorithm (FA) (Yang, 2010a), artificial fish school algorithm (AFSA) (Yun, 2010), bat algorithm (BA) (Yang, 2010c), ant lion optimizer (Mirjalili, 2015), dragonfly algorithm (Mirjalili, 2015), grey wolf optimizer (Mirjalili et al., 2014). As their representatives, four algorithms of ACO, CS, FA and AFSA are described as following.

• Ant colony optimization (ACO): taken inspiration from the foraging behavior of some ant species. These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving optimization problems. The first ACO algorithm is known as Ant System and was proposed in the early nineties. Since then, several other ACO algorithms have been proposed. With regard to the process of ACO algorithm, after initializing parameters and pheromone trails, the main loop consists of three main steps. First, ants construct solutions to the problem instance under consideration, biased by the pheromone information and possibly by the available heuristic information. Once the ants have completed their solutions, these may be improved in an optional local search phase. Finally, before the start of the next iteration, the pheromone trails are adapted to reflect the search experience of the ants;

• Cuckoo search (CS): based on the interesting breeding behavior of brood parasitism of certain species of cuckoos by laying their eggs in the nests of other host birds (of other species). Each cuckoo lays one egg at a time, and dumps its egg in randomly chosen nest; the best nests with high quality of eggs will carry over to the next generations; the number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird

with a probability, the host bird can build a completely new nest through throwing the egg away or abandoning the nest. CS was proposed based on the above principles and applied on optimization problems and other real world applications;

• Firefly algorithm (FA): inspired by the flashing behavior of fireflies. The main purpose for a firefly's flash is to act as a signal system to attract other fireflies. All fireflies are unisexual, so that any individual firefly will be attracted to other fireflies; attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by the brighter one; however, the apparent brightness decreases as their mutual distance increases; If there are no fireflies brighter than a given firefly, it will move randomly. The brightness should be associated with the objective function. In recent years, more than twenty variants of FA such as discrete FA, multi-objective FA, chaotic FA and hybrid FA algorithms have been developed and applied to various fields of optimization, image processing, clustering, feature selection and fault detection;

• Artificial fish school algorithm (AFSA): is inspired by the collective movement of the fish and their various social behaviors. Based on a series of instinctive behaviors, the fish always try to maintain their colonies and accordingly demonstrate intelligent behaviors. AFSA builds some artificial fish, which search an optimal solution in solution space (the environment in which AF live) by imitating fish swarm behavior. The basic behaviors of AF are prey, swarm and follow. The fish perceives the concentration of food in water to determine the movement by vision or sense and then chooses the tendency in the prey behavior; In swarm behavior, the fish will assemble in groups naturally in the moving process, which is a kind of living habits in order to guarantee the existence of the colony and avoid dangers; In the moving process of the fish swarm, when a single fish or several fish find food, the neighborhood partners will trail and reach the food quickly in the following behavior;

Various modified SI based algorithms have been also proposed and applied to optimization problems in the real-world. Their effectiveness and efficiency have been testified through experiments. The main SI algorithms are ABC and PSO involved in this thesis. The extended ABC algorithms will be proposed in Chapter 4 and Chapter 5. PSO algorithm as one of the representative SI algorithms is introduced to compare with extended ABC algorithms. The statements of ABC and PSO algorithms are given in Sections 3.1.1-

3.1.2.

### 3.1.1 Artificial Bee Colony (ABC)

**Real Bee Behavior**

Facing a life-or-death situation, a honey bee swarm engages in a complex decision-making process involving multiple, simultaneous interactions among hundreds of individuals who have no leadership at all.

The collective intelligent behavior of insect or animal groups in nature such as flocks of birds, swarms of ants and termites, schools of fish, colonies of bees have attracted the attention of researchers (Cuevas et al., 2013). Entomologists have studied this collective phenomenon to model biological swarms while engineers have applied these models as a framework for proposing various kinds of SI based algorithms.

The recent developments do not only have consequences on the study of social insects, but also provide us with powerful tools to transfer knowledge about social insects to the field of intelligent system design. In effect, a social insect colony is undoubtedly a decentralized problem-solving system, comprised of many relatively simple interacting entities (Bonabeau et al., 1999). Every insect of the colony seems to have its own responsibility, the seamless integration of the individuals does not require the central control.

A swarm of honey bee is very attractive in nature that the swarm allocate the task more dynamically and adapt to environment by collective intelligent manner.

Honey bees are one kind of the most fascinating insects on the earth. Aside from making delicious honey, honey bees are amazing because their behavior attracts researchers to study the bee colony. The honey bees have the photographic memories, space-age sensory and navigation systems, they have the intelligent behavior of selecting nest sites. Additionally, honey bees possess a number of characteristics which contribute to their making decision for success, including self-organization, stigmergy, and division of labor. They have direct and indirect communication systems which allow them to coordinate their activities and engage in decentralized problem-solving and intelligent decision-making. And, like all social insects, they have the opportunity to engage in social learning by interacting with simple other entities in their social group. While groups of insects possess collective

intelligence, the individuals within those groups have complex cognitive abilities. Self-organization, stigmergy and division of labor three component keys in bee colony described as following.

■ **Self-organization**

Self-organization (SO) is one of the key issues for generating collective intelligence. Bonabeau et al. (1997a) describe self-organization as a set of dynamical mechanisms whereby structures appear at the global level (macroscopic level) of a system from interactions among its lower-level (microscopic level) components. In a SO system, each of the covered units may respond to local stimuli individually and act together to accomplish a global task via division of labor without any centralized control. SO refers to a broad range of complex higher-order pattern-formation process in both physical and biological systems, such as sand grains assembling into ripple dunes, birds flying in swarm, fish joining together in schools, ants transporting in colony and bees working together in colony.

In short, the modeling of social insects by means of SO can help to design artificial distributed problem-solving devices to solve the problems. The paths to problem solving are not predefined but rather emergent in these systems, and result from interactions among individuals and between individuals and their environment as much as from the behavior of the individuals themselves. Taking the honey bees as an example, foraging is the most important behavior in the hive. Colony-level exploitation of nectar sources emerges automatically as each bee follows a few simple behavior rules, each bee has limited knowledge of the array of available nectar sources in the field (Seeley, 1999). Each bee needs only have knowledge of the nectar source at which she is presently feeding. Efficient exploitation of the food source can occur without any bees comparing the relative profitabilities of the nectar sources (Camazine et al., 1991). Therefore, using a swarm intelligent system to solve a problem requires a thorough knowledge not only of what individual behavior must be implemented but also of what interactions are needed to produce such global behavior.

According to Bonabeau et al. (1999), positive feedback, negative feedback, multiple interactions and fluctuations are the four characteristics for SO. In terms of honey bees, the basic elements on SO relies are as follows:

- Positive feedback: as the nectar amount of food sources increases, the number of onlookers visiting them also increases
- Negative feedback: the exploration process of a food source abandoned by bees is stopped
- Fluctuations: the scouts carry out a random search process for discovering new food sources
- Multiple interactions: bees share their information about food source positions with their nest mates on the dance area

■ **Stigmergy**

In stigmergic labor, it is the product of work previously accomplished, rather than direct communication among nest mates, that induces the insects to perform additional labor (Wilson, 1971). The individuals could act independently without direct interaction or sophisticated communications. Therefore, stigmergic process is a sequence of indirect stimulus or responses behaviors and contributes to the coordination between insects through the environment.

Michener (1974) described many activities in bee colonies that result in nest structures, conditions of brood or stored food, to which other bees respond. Referring to this as social interactions by indirect communicating, where the construct is made for other primary objectives, not for signaling, although the information content becomes essential for colony integration. In nectar source decision making in honey bees, it is less clear if only direct communication through the recruitment dances of the bees produce the SO behavior, or if also the indirect communication given by the waiting time for downloading the honey is affecting the collective behavior (Seeley et al., 1991). In the process of recruiting other bees, waggle dance works as the stigmergy mechanism. It is an advertisement for the food source of the dancer. Another forager can leave own food source and watch out for the well advertised food source (Seeley, 1992). A forager randomly follows dances of multiple recruiting foragers and seems to respond randomly as well. Especially she does not compare several dances. A dance does not seem to contain any information that helps to

choose a food source (Wedde et al., 2004). Upon their return from a foraging exploration, bees inform the distance, direction, and quality of a flower site to their fellow foragers by making waggle dances. In this way, if a good flower site is exploited, then the numbers of foragers at this site are reinforced.

Stigmergy is often associated with flexibility: when the environment changes because of an external perturbation, the insects respond appropriately to that perturbation, as if it were a modification of the environment caused by the colony's activities. In terms of honey bees, the bee colony can collectively respond to the perturbation with individuals exhibiting the same behavior. They show how stigmergy can easily be made operational because of the simplicity of the behaviors involved.

### ■ Division of Labor

The division of labor is the specialization of cooperating individuals who perform specific tasks and roles. Division of labor is one of the most well explored phenomena in the study of the behavior of social insects (Robinson, 1992). As the SO and stigmergy, division of labor is also the key property for honey bees to obtain swarm intelligent behaviors.

The division of labor is accomplished by the behavioral flexibility of individual workers, which can be characterized by individual specialization and role plasticity (Holldobler and Edward, 2009). Individual specialization makes honey bees to implement different search modes referring to the individual preference for different tasks. Role plasticity makes the honey bees to convert the characters caused by the changing environment. Colony of bees responds to the internal and external changing conditions by adjusting the ratios of individual workers engaged in the foraging tasks. Individuals have the ability to learn, behave flexibility and achieve the effective task allocation and survives through mutual cooperation with other individuals in changing environment. In addition, the individuals have the property of robustness and endow the colony with the ability to function even though some individuals may fail to perform their task. With these properties, honey bees make the individuals have the ability to switch tasks quickly and accurately, in the meantime, make them have the plasticity ability to adapt to changing environments dynamically.

**Standard ABC Algorithm**

The social insect colonies include ants, termites, bees, and wasps, and the other animal societies. They live together in self-organized cluster, they engage in a variety of complex tasks not practiced by the multitude of solitary insects. Besides the above mentioned three components keys, complex hive building, communication, guarding, navigation, environment control and foraging are the behaviors that honey bees have developed to survive successfully among the social insects. The food foraging is one of the major tasks for the honey bees as charming characteristic.

According to Karaboga et al. (2014), the inspiration from the bees' behavior related to the food sources, employed foragers and unemployed foragers are concluded as follows:

- Food Sources: In order to select a food source, a forager bee evaluates several properties related with the food source such as its closeness to the hive, richness of the energy, taste of its nectar, and the ease or difficulty of extracting this energy. For the simplicity, the quality of a food source can be represented by only one quantity although it depends on various parameters as mentioned above;

- Employed foragers: An employed forager is employed at a specific food source which she is currently exploiting. She carries information about this specific source and shares it with other bees waiting in the hive. The information includes the distance, the direction and the profitability of the food source;

- Unemployed foragers: A forager bee that looks for a food source to exploit is called unemployed. It can be either a scout who searches the environment randomly or an onlooker who tries to find a food source by means of the information given by the employed bee. The mean number of scouts is about 5-10%.

The food foraging is organized by the bee colony by recruiting bees for different jobs. The recruitment is accomplished by the foraging bees through exchanging the information which is the most important occurrence in the formation of collective knowledge. While examining the entire hive it is possible to distinguish some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is

the dancing area where the forager bees perform dance to communicate with their fellow bees inside the hive and recruit them. Bees that decide foraging without any guidance from other bees are called scouts. Bees that attend to the waggle dance at the dance floor can decide which food source to go based on its quality. The quality of a food source is proportional to the quantity of nectar found there, and this information is transmitted by changing the intensity of the waggle dance and through antennae contacts. The better the food source, the more intense is the dance and the contacts (Reinhard and Srinivasan, 2009). There is a greater probability of onlookers choosing more profitable sources since more information is circulating about the more profitable sources. Employed foragers share their information with a probability which is proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to profitability of a food source (Tereshko and Loengarov, 2005).

Figure 3.1 illustrates the basic foraging behavior of the bees (Camazine et al., 1991; Karaboga et al., 2014). It is assumed that there are two discovered food sources of A and B in the fields of M and N. Food sources FS exist in the fields of M and N besides A and B. Potential foragers $P_1$ and $P_2$ are set as unemployed foragers. They have no information about the food sources around the hive. There are two possible selections for such a unemployed foraging bee:

1. It can become a scout bee and begin to search the nectar around the hive spontaneously (shown as S on Fig. 3.1)

2. It can be recruited as a follower and begin to search for a food source by watching the waggle dances (shown as R on Fig. 3.1)

After searching the food source, the bees can memorize the location of the nectars and then start to exploit them instantly. The bees will become employed forager bees and then they take the nectars from the food source fields and returns to the hive and unload them. After unloading the nectars, the forager bees may become one of the following bees:

- It might become an free follower after relinquishing the food source and go back to dance area (FF)

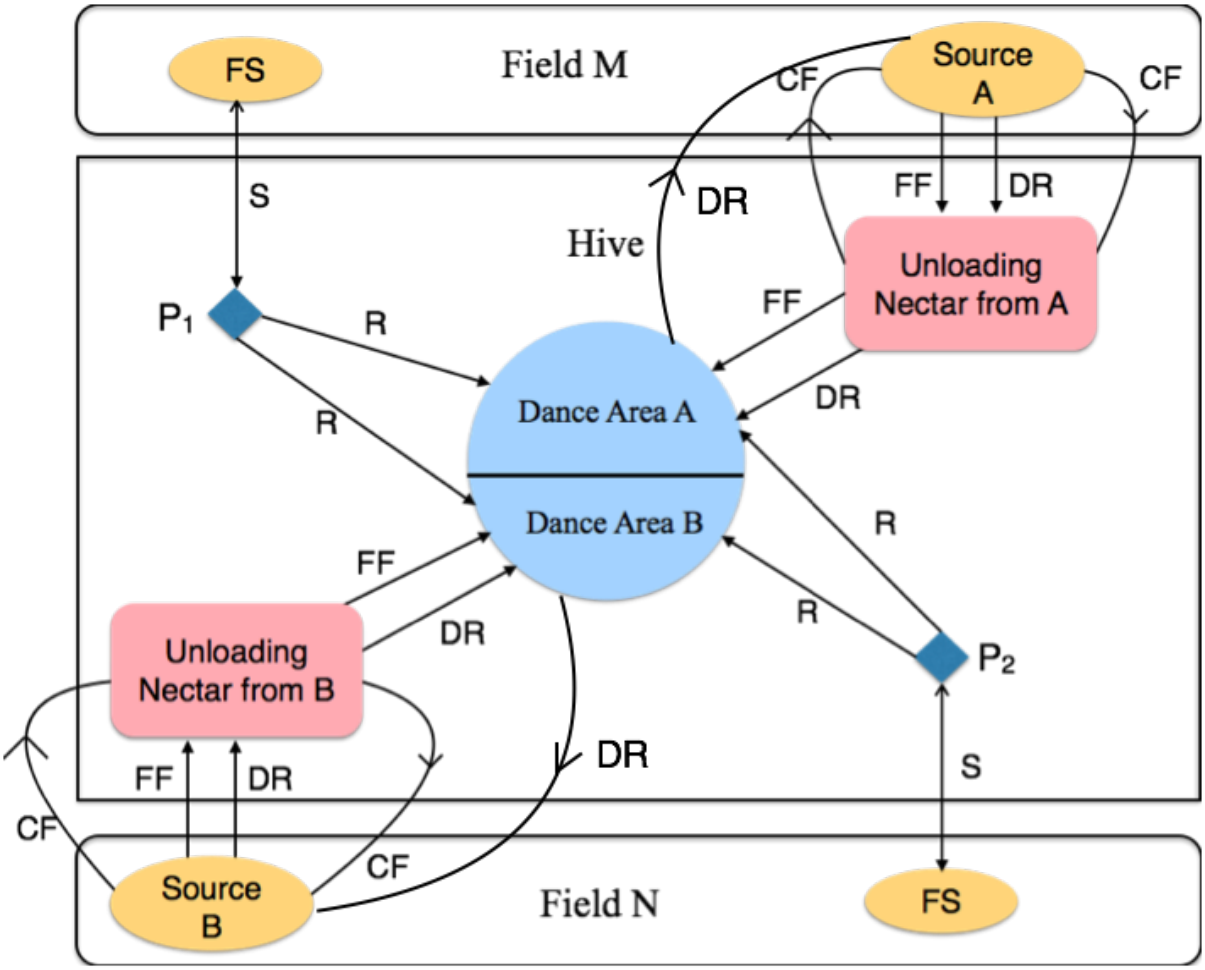



Figure 3.1: Flowchart for foraging behavior of honey bees

- It might dance and then recruit hive mates and return to the same food source field (DR)
- It might continue to forage from the same food source field without recruiting hive mates (CF)

ABC algorithm is a swarm based meta-heuristic algorithm introduced by Karaboga (2005) that mimics the food foraging behavior of honey bees following the fundamental principles as mentioned at the beginning of this section. Then ABC has successfully been applied to numerical optimization problems as mentioned in Chapter 1.

In the ABC algorithm, the artificial bee colony associated with the foraging task comprises three kinds of bees: employed bees, onlooker bees, and scout bees. In ABC algorithm, the food sources represent the candidate solutions or possible solutions and the nectar amount of the food source represents the fitness associated with the solution for the optimization problem. And each of the food sources is exploited by only one employed bee hence the number of food sources are equal to the number of employed bees.

In the initial phase of foraging, bee explores the environment randomly in search of food sources. When the forager bee finds the food source it becomes employed bee. Employed bees search for food source sites by modifying the site in their memory, evaluating the nectar amount of each new source, and memorizing the more productive site through a selection process. These bees share information related to the quality of the food sources they exploit in the "dance area". The number of employed bees is equal to the number of food sources for the hive. Onlooker bees search for food sources based on the information coming from employed bees within the hive. As such, more beneficial sources have higher probability to be selected by onlookers. Onlooker bees choose the food sources by watching the waggle dances performed by employed bees. Waggle dances are performed by individual foragers on returning to the nest and convey information about the direction and distance of the forage site visited by the dancing bees. When the food source is abandoned, a new food source is randomly selected by a scout bee to replace the abandoned source. The number of food sources in ABC algorithm is equivalent to the number of solutions in a population for an optimization problem. The number of nectar sites of a food source represents the fitness cost of the associated solution. The flowchart of ABC is shown as Fig. 3.2.

The main steps of ABC algorithm are shown as following:

1. Initialize the population of solutions $x_{ij}$ with Eq. (3.1);

$$x_{ij} = x_{min,j} + rand[0,1](x_{max,j} - x_{min,j}) \tag{3.1}$$

   Where, $i \in 1, 2, ..., SN$ and $j \in 1, 2, ..., D$ are randomly selected indexes, $SN$ is the number of food source, and $D$ is the dimension size

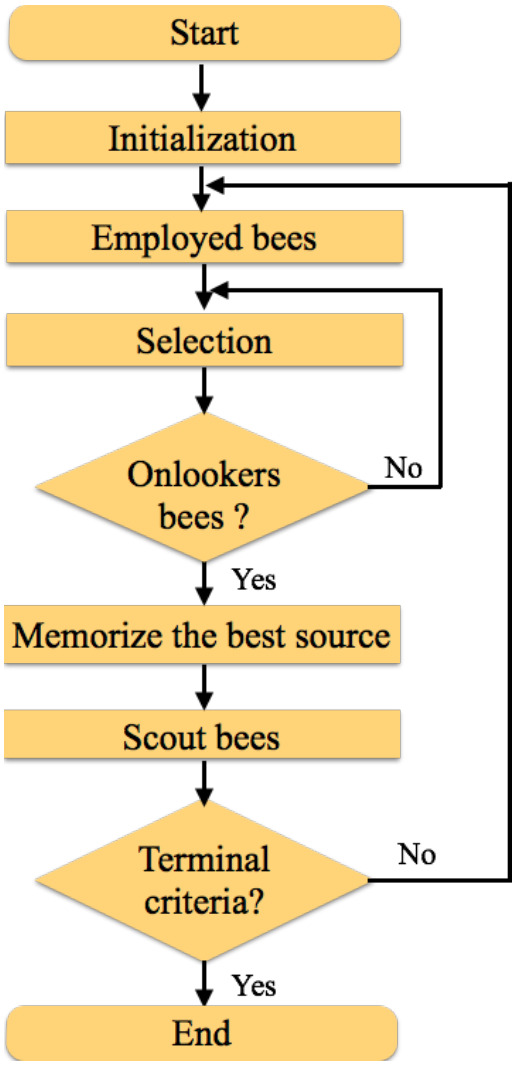



Figure 3.2: Flowchart for ABC algorithm

2. Evaluate the population;

3. Initialize *cycle* to 1, this number is counting the iteration cycle of this algorithm;

4. Produce new solutions $v_i$ for the employed bees by using $x_{ij}$ mentioned in Eq. (3.1);

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3.2}$$

Where $\phi_{ij}$ is uniformly distributed random number in the range [-1,1]; $i, k \in 1, 2, ..., SN$

are randomly selected indexes with $k$ different from $i$, and $j \in 1, 2, ..., D$ is a randomly selected index

Then the solutions are evaluated according to fitness value $fit_i$ in minimization problem, where $f_i$ is the cost value of solution $v_i$.

$$fit_i = \begin{cases} 1/(1 + f_i), & if \quad f_i \geq 0 \\ 1 + |f_i|, & if \quad f_i < 0 \end{cases} \tag{3.3}$$

5. Apply the greedy selection process for the employed bees according to the fitness values of current and the greatest one;

6. If the solution is not improved, then modify the value of *trail*, which is incremented by one in each fail or is set zero in each successful try carried out by an employed bee;

7. Calculate probability values $P_i$ for the solutions using Eq. (3.4);

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{3.4}$$

Where $fit_i$ is the fitness value of solution $i$ as defined in Eq. (3.3).

8. Produce new solutions $v_i$ for the onlooker bees from solutions $x_i$ using Eq. (3.1) which is selected depending on $P_i$ then evaluate them;

9. Apply the greedy selection process for the onlooker bees according to the fitness values of current value and the greatest one;

10. If the solution is not improved, then the value of *trail* is modified, which is incremented by one in each fail or is set to zero in each successful try carried out by an onlooker bee;

11. Determine the abandoned solution through the number of *limit* for the scout bee, if it exists, then replaced with a new random solution using Eq. (3.1);

12. Memorize the best solution achieved so far;

13. Let *cycle* = *cycle* +1;

14. Repeat *cycles* 4-13 until *cycle* reaches a predefined maximum cycle number (*MCN*).

ABC algorithm has been applied to several fields in various ways, for example, optimization problem (Karaboga and Basturk, 2008), training neural networks (Karaboga and Akay, 2007); applied to protein structure prediction (Benítez and Lopes, 2010); solving sensor deployment problem (Udgata et al., 2009); applied to engineering design optimization (Akay and Karaboga, 2010a); solving wireless sensor network problem (Okdem et al., 2011); solving redundancy allocation problem (Yeh and Hsieh, 2011); applied to data mining (Celik et al., 2011) and solving job shop scheduling (Yin et al., 2011). As one of the prominent SI algorithms in recent years, ABC algorithm will be adopted to solve these benchmark problems defined by the CEC'13 and CEC'14 test functions in the thesis. As an example for applying ABC to optimization problem, it is shown as following:

To minimize $f(x) = x_1^2 + x_2^2$, where, $-5 \leq x_1, x_2 \leq 5$

Control parameters of ABC Algorithm are set as: Colony size (CS) = 6, dimension of the problem, D = 2, Limit for scout, L = (CS*D)/2 = 6.

First, initialize the positions $x$ of 3 food sources (CS/2) of employed bees, randomly using uniform distribution in the range (-5, 5):

$$\begin{pmatrix} 1.4112 & -2.5644 \\ 0.4756 & 1.4338 \\ -0.1824 & -1.0323 \end{pmatrix}$$

$f(x)$ values are:

$$\begin{pmatrix} 8.5678 \\ 2.2820 \\ 1.0990 \end{pmatrix}$$

According to the fitness function Eq. (3.3), initial fitness vector is:

$$\begin{pmatrix} 0.1045 \\ 0.3047 \\ 0.4764 \end{pmatrix}$$

The maximum fitness value is 0.4764, it indicates the quality of the best food source. One cycle of ABC algorithm is conducted and the position, function value and the fitness vector are calculated according to Eqs. (3.1-3.4):

// For employed bees step:

1st employed bee: $v_0 = (2.1644, -2.5644)$, where $\phi_0 = 0.805$, f($v_0$)=11.261, fitness value is 0.0816; solution 0 could not be improved because 0.0816 ¡ 0.1045 according to greedy selection between $x_0$ and $v_0$, and then increase the trial counter.

2nd employed bee: $v_1 = (0.4756, 1.6217)$, where $\phi_0 = 0.0762$, f($v_1$)=2.856, fitness value is 0.2593; solution 1 could not be improved because 0.2593 ¡ 0.3047 according to greedy selection between $x_1$ and $v_1$, and then increase the trial counter.

3rd employed bee: $v_2 = (-0.0754, -1.0323)$, where $\phi_0 = -0.0672$, f($v_0$)=1.0714, fitness value is 0.4828; solution 2 is improved because 0.4828 ¿ 0.4764 according to greedy selection between $x_0$ and $v_0$, set its trial counter as 0 and replace the solution $x_2$ with $v_2$. Then the positions are reset as following:

$$\begin{pmatrix} 1.4112 & -2.5644 \\ 0.4756 & 1.4338 \\ -0.0754 & -1.0323 \end{pmatrix}$$

$f(x)$ values are:

$$\begin{pmatrix} 8.5678 \\ 2.2820 \\ 1.0714 \end{pmatrix}$$

Fitness vector is:

$$\begin{pmatrix} 0.1045 \\ 0.3047 \\ 0.4828 \end{pmatrix}$$

Probability values $P_i$ is calculated by means of their fitness using Eq. (3.4):

$$\begin{pmatrix} 0.1172 \\ 0.3416 \\ 0.5412 \end{pmatrix}$$

// For onlooker bees step:

1st onlooker bee: i=2, $v_2 = (-0.0754, -2.2520)$, f($v_2$)=5.0772 and the fitness value is 0.1645; solution 2 could not be improved because 0.1645 ¡ 0.4828 according to greedy selection between $x_2$ and $v_2$, and then increase the trial counter.

2nd onlookerw bee: i=1, $v_1 = (0.1722, 1.4338)$, f($v_1$)=2.0855 and the fitness value is 0.3241; solution 1 is improved because 0.3241 ¿ 0.3047 according to greedy selection between $x_1$ and $v_1$, and set its trial counter as 0 and replace the solution $x_1$ with $v_1$. Then the positions are reset as following:

$$\begin{pmatrix} 1.4112 & -2.5644 \\ 0.1722 & 1.4338 \\ -0.0754 & -1.0323 \end{pmatrix}$$

$f(x)$ values are:

$$\begin{pmatrix} 8.5678 \\ 2.0855 \\ 1.0714 \end{pmatrix}$$

Fitness vector is:

$$\begin{pmatrix} 0.1045 \\ 0.3241 \\ 0.4828 \end{pmatrix}$$

3rd employed bee: i=2, $v_2 = (0.0348, -1.0323)$, f($v_2$)=1.0669 and the fitness value is 0.4838; solution 2 is improved because 0.4828 ¿ 0.4828 according to greedy selection between $x_2$ and $v_2$, set its trial counter as 0 and replace the solution $x_2$ with $v_2$. Then the positions are reset as following:

$$\begin{pmatrix} 1.4112 & -2.5644 \\ 0.1722 & 1.4338 \\ 0.0348 & -1.0323 \end{pmatrix}$$

$f(x)$ values are:

$$\begin{pmatrix} 8.5678 \\ 2.0855 \\ 1.0669 \end{pmatrix}$$

Fitness vector is:

$$\begin{pmatrix} 0.1045 \\ 0.3241 \\ 0.4838 \end{pmatrix}$$

// Memorize the best: Best = (0.0348, -1.0323)

// For scout bee step:

TrialCounter =

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

There is no abandoned solution since L = 6. If there is an abandoned solution (trial counter value is higher than L), generate a new solution randomly to replace with the abandoned one.

// Cycle = Cycle+1

The above procedure is continued until the termination criterion is reached.

### 3.1.2 Particle Swarm Optimisation (PSO)

Particle swarm optimization (PSO) proposed by Kennedy and Eberhart (1995), is a SI oriented stochastic, population based global optimization technique. The PSO uses a simple mechanism that mimics swarm behavior in birds flocking and fish schooling to guide the particles to search for global optimal solutions. Due to its unique searching mechanism, simple concept, computational efficiency, and easy implementation, it has rapidly progressed in recent years and with many successful applications seen in solving real world optimization problems (Li and Engelbrecht, 2007; Krohling and Coelho, 2006; Liu et al., 2007).

In PSO, the term particles refer to population members which are mass-less and volume-less, could be called "points". Velocities and accelerations are more properly applied to particles with an arbitrarily small mass and volume.

A swarm is similar to population, while a particle is similar to an individual. In simple terms, the particles are "flown" through a multidimensional search space, where the position of each particle is adjusted according to its own and neighbor's experience. Each particle in the swarm has four vectors, its current position, best position found so far, the best position found by its neighborhood so far and its velocity. Each particle adjusts its position in the search space based on the best position reached by itself personal best ($p$best) and best position reached by its neighbor ($g$best) during the search process.

In the PSO algorithm, the cognitive component (individual learning) of a particle, which is proportional to its $p$best position, and the social component (cultural transmission) generated by the swarm are generated while a particle is developing a new situation. This situation enables the PSO algorithm to effectively find the global optimum solutions by searching through the local solutions.

The standard PSO algorithm starts from the initialization step composed of a population of random number solutions. With the fitness value for evaluation, particles move iteratively through D-dimensional search space to find new solutions.

In each iteration, each particle updates its velocity and position as follows:

$$v_{i,d} = \omega v_{i,d} + c_1 r_1 (p_{i,d} - x_{i,d}) + c_2 r_2 (p_{g,d} - x_{i,d}) \tag{3.5}$$

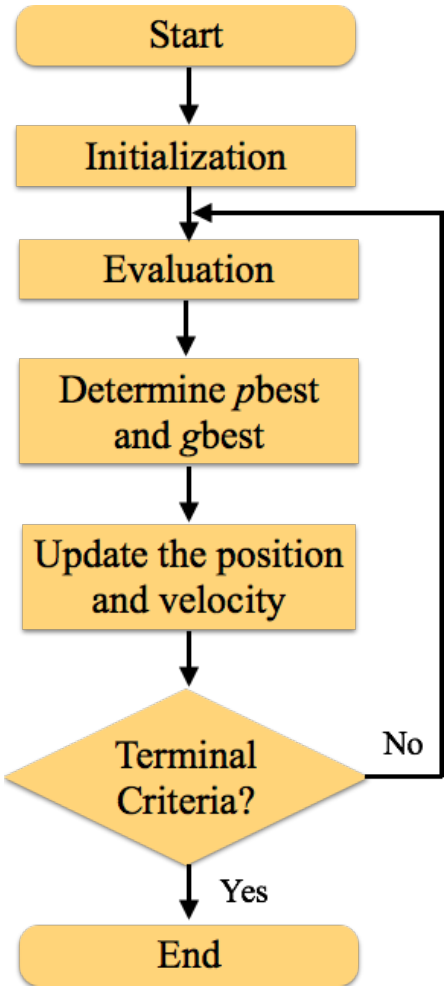



Figure 3.3: Flowchart of PSO algorithm

$$x_{id} = x_{id} + v_{id} \tag{3.6}$$

Where $v$ represents particle velocity and $x$ represents particle position, $i \in 1, 2, ..., M$, $d \in 1, 2, ..., D$ , $M$ is the total number of particles in the swarm, $\omega$ is the inertia weight, $r_1, r_2$ are random numbers in the range [0,1], $c_1, c_2$ are acceleration coefficients, $p_{i,d}$ is the personal best ($p$best) and $p_{g,d}$ is the global best ($g$best).

The flowchart of PSO algorithm is shown as Fig. 3.3. The process of the PSO flowchart is explained as following:

1. Create a population of agents (particles) with uniformly distributed random numbers;
2. Evaluate each particle's position and velocity;
3. Determine the *p*best and *g*best;
4. Update the particle's position and velocity;
5. Repeat the steps 2-4 until terminal criteria are met, if it is not, then return to step 2 and evaluate the new population.

PSO algorithm is attractive that there are few parameters to train, therefor, it has been used for wide range of applications which focused on a specific requirement. The applications vary from evolving artificial neural networks, system design, classification, pattern recognition, scheduling, signal processing, decision making to multi-objective optimization problem.

## 3.2 Evolutionary Algorithms (EAs)

EA algorithms are stochastic search and optimization heuristics using computational models of evolutionary processes derived from biology with beginning their existence during the late 1960s and early 1970s. The evolutionary process is the key element in the design and implementation of computer based problem solving systems. Different from the traditional calculus based optimization strategies, EAs are based on a population of encoded tentative solutions which are processed with some evolutionary operators such as recombination and mutation to find a optimal solution. The search and optimization process follows the principle of the Darwinistic evolution theory of "survival of the fittest" to generate successively better results over generations to finally approximate the optimal solutions. The EAs use three main principles of natural evolution: reproduction, natural selection and diversity of species, maintained by the differences of each generation with the previous.

To solve optimization problems with an evolutionary heuristic, the individuals of a population have to represent a possible solution of a given problem and the selection probability is set proportional to the quality of the represented solution (Streichert, 2002). Evolution of the population then takes place after the repeated application of the above steps.

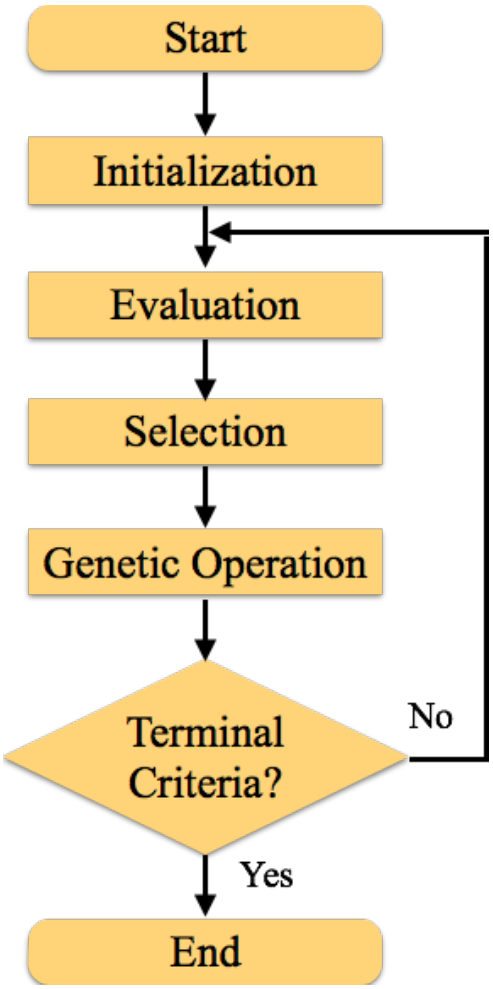



Figure 3.4: The general flowchart for EA

An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, natural selection and the survival of the fittest. The general flowchart of EA is shown as in Fig. 3.4. The process of the EA flowchart is explained as following:

1. The initial population is generated randomly;

2. Evaluate the fitness value of the individuals in the initial population;

3. The individuals with higher quality (better fitness) value are selected according to the evaluation;

4. New individuals are generated from parents through the genetic operators with crossover and (or) mutation;

5. Repeat the steps 2-4 until terminal criteria are met, if it is not, then return to step 2 and evaluate the new population.

As the classical EAs, GA, DE and their extended versions of Non-uniform Real-coded GA (NUGA) and SHADE algorithms have been proposed. The interpretations of NUGA and SHADE algorithms are given in the following sections and then adopted as the algorithms used to compare with extended ABC algorithms in Chapter 4 and Chapter 5.

### 3.2.1 Genetic Algorithm (GA)

Genetic algorithm (GA) is a stochastic optimization algorithm based on the concepts of biological evolutionary theory (Holland, 1975). GA starts off with a population of randomly generated chromosomes and advances toward better chromosomes by applying genetic operators, modeled on the genetic processes occurring in nature. The chromosomes represent set of genes, which code the independent variables. Every chromosome represents a solution of a given problem. An individual, its vector of variables, are used as another word for a chromosome.

A set of different chromosomes (individuals) forms a generation, the chromosomes are evaluated as possible solutions. Based on these evaluations, an offspring population is created using a mechanism of selection and applying genetic operators such as crossover and mutation. The selection of individuals is performed by survival of the fittest. There are several kinds of selection mechanisms as fitness proportional selection, ranking selection, roulette wheel selection and tournament selection. For the selection mechanisms, the user has to provide means to determine the relative fitness of the individuals. The individuals with higher fitness are selected for next step of crossover. Crossover is the first step for reproduction process. In this process, the genes of the parents are used to form an entirely new chromosome. There are one-point crossover, two-point cross over and uniform crossover for the crossover operator. Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. Mutation operator has serval types as bit string, flip bit, uniform, non-uniform and gaussian mutations.

The operation of a standard GA is described as following steps:

1. Randomly generate an initial population;
2. Compute the fitness of the population;
3. Create an intermediate population by using a selection operator;
4. Generate a new population by applying crossover and mutation operators;
5. Increase the generation number and go back to step 3 when termination condition is not met.

GA is a very effective way of quickly finding a reasonable solution to a complex problem, it can do an excellent job of searching through a large and complex search space. You may know exactly what you want a solution to do but have no idea how you want it to go about doing it. This is where GA thrives. GA produces solutions that solve the problem in ways you may never have even considered. The applications of GA are involving various fields in the real-world, for example, state assignment problem, economics, scheduling, traveling salesman problem and computer-aided design. Specifically, GA is often applied as an approach to solve global optimization problems.

### 3.2.2 Non-uniform Real-coded GA (NRGA)

From the statistical data, the best so far solution obtained is $x^{b,t}$ (where $b$ denotes the best and $t$ is the current iteration number). The individual is bounded between values $a$ and $b$ (with $a$ as lower bound $x_L$ and $b$ as upper bound $x_U$). The location of the individual after performing three basic operations of selection, crossover and mutation is $x$ (Yashesh et al., 2014).

The mapping function is:

$$m(\zeta) = k\zeta^{\eta} \tag{3.7}$$

where $\zeta = (x - a)/(b - a)$. Using this mapping equation, $x$ is mapped to $x^*$. $x^*$ is nearer to the best ever solution $x^{b,t}$.

$$k = (\eta + 1)(\frac{b-a}{x^{b,t} - a})^{\eta} \tag{3.8}$$

When $x \in (a, x^{b,t})$, the mapped value $x^*$ for one-dimensional space is :

$$x^* = a + [(x - a)(x^{b,t} - a)^{\eta}]^{1/(\eta+1)} \tag{3.9}$$

The point $x$ used above is one of the vector form $X$ with its coordinates as $x_i$ (where $i = 1, 2, ..., n$ ).

Two kinds of methods for mapping were implemented. First method was the variable-wise mapping approach, where the components of $X$ (i.e. $x_i$ ) were pushed towards the corresponding components of $X^{b,t}$ (i.e. $x_i^{b,t}$ ) just like in one-dimensional case.

$$x_i^* = x_{iL} + [(x_i - x_{iL})(x_i^{b,t} - x_{iL})^{\eta}]^{1/(\eta+1)} \tag{3.10}$$

The second method was vector-wise mapping. The parameterized value of the mapped point $X^*$ is $d^*$ using the following equation:

$$d^* = a + [(-a)(1 - a)^{\eta}]^{1/(\eta+1)} \tag{3.11}$$

$$X^* = X + d^*(X^{b,t} - X) \tag{3.12}$$

It is clearly noticed that higher the value of $\eta$, more will be the pushing done and so the solutions will start accumulating nearer and nearer to the best ever point $X^{b,t}$. If the value of $\eta$ is set too large right from the start, then the amount of exploration will be reduced and there will be higher chances of getting premature convergence.

NRGA was proposed to improve the performance by controlling the diversity of population and avoiding premature convergence through the experiments based on CEC'14 real-parameter numerical optimization (Yashesh et al., 2014).

### 3.2.3 Differential Evolution (DE)

DE is a population-based stochastic search technique proposed by Storn and Price (1997). It has several attractive features of simple, but powerful, effective and efficient for global optimization in continuous search domain. It is significantly faster and robust for solving numerical optimization problems. DE has been successfully applied in diverse fields such as mechanical engineering (Rogalsky et al., 2000; Joshi and Sanderson, 1999), and training neural network (Ilonen et al., 2003). In DE, there exist many trial vector generation strategies out of which a few may be suitable for solving a particular problem. Moreover, three crucial control parameters involved in DE, i.e., population size, scaling factor, and crossover rate, may significantly influence the optimization performance of the DE. Therefore, to successfully solve a specific optimization problem at hand, it is generally required to perform a time-consuming trial-and-error search for the most appropriate strategy and to tune its associated parameter values.

The general mutation type for DE is DE/x/y/z, where x represents a string denoting the base vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exp indicates for exponential and bin stands for binomial). There are 5 types of mutations, namely, “DE/rand/1”, “DE/best/1”, “DE/current-to-best/1”, “DE/rand/2” and “DE/best/2” shown as following:

- “DE/rand/1”

$$V_{i,G} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) \tag{3.13}$$

- “DE/best/1”

$$V_{i,G} = x_{best,G} + F \cdot (x_{r_1,G} - x_{r_2,G}) \tag{3.14}$$

- “DE/current-to-best/1”

$$V_{i,G} = x_{i,G} + F \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r_1,G} - x_{r_2,G}) \tag{3.15}$$

- "DE/rand/2"

$$V_{i,G} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) + F \cdot (x_{r_4,G} - x_{r_5,G}) \tag{3.16}$$

- "DE/best/2"

$$V_{i,G} = x_{best,G} + F \cdot (x_{r_1,G} - x_{r_2,G}) + F \cdot (x_{r_3,G} - x_{r_4,G}) \tag{3.17}$$

Where, the indices $r_1, ..., r_5$ are randomly selected from [1,$N$] such that they differ from each other as well as $i$, $x_{best,G}$ is the best individual in population in generation $G$. The parameter $F \in [0, 1]$ controls the magnitude of the differential mutation operator.

DE algorithm aims at evolving a population of $D$-dimensional parameter vectors seen as individuals, which encode the candidate solutions $x_{i,G}$ as a population for each generation $G$, where $i = 1, 2, ..., NP$. The initial vector population is chosen randomly. As a rule, a uniform probability distribution is provided for all random decisions unless otherwise stated. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector on the mutation step. The mutated vector's parameters are then mixed with the parameters of another predetermined vector, the target vector, to generate the trial vector. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. The last step is selection. Basic operating steps for DE algorithm are described as follows:

■ **Mutation**

For each target vector $x_{i,G}$, where $i = 1, 2, ..., NP$, $G$ is the generation number.

A mutant vector $V_{i,G}$ for "DE/rand/1" is generated as follows:

$$V_{i,G} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) \tag{3.18}$$

The random indexes $r_1, r_2, r_3 \in 1, 2, ..., NP$ are mutually different integers. $F$ is scale factor.

■ **Crossover**

In order to increase the diversity of the perturbed parameter vectors, crossover operator is introduced to generate a trial vector $u_{i,G}$:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} \;\; if \;\; rand[0,1] \leq CR \;\; or \;\; j = j_{rand} \\ x_{j,i,G} \;\; otherwise \end{cases} \tag{3.19}$$

In Eq. (3.19), *rand*[0,1] is a uniform random number generator with outcome [0,1]. $CR$ is the crossover constant selected from [0,1], $j$ is dimension size chosen from $j \in 1, 2, ..., D$ , and $j_{rand}$ is a random integer from [0, $D$-1] generated for each $i$.

■ **Selection**

To decide whether the target or trial vector survives to the next generation, the trial vector $u_{i,G}$ is compared to the target vector $x_{i,G}$ using the greedy criterion. If vector $u_{i,G}$ yields a smaller cost function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G}$; otherwise, the old value $x_{i,G}$ is retained.

$$x_{i,G+1} = \begin{cases} u_{i,G} \;\; if \;\; f(u_{i,G}) < f(x_{i,G}) \\ x_{i,G} \;\; otherwise \end{cases} \tag{3.20}$$

### 3.2.4 SHADE

**• 1. JADE**

JADE ( Zhang and Sanderson, 2009; Tanabe and Fukunaga, 2013) is a DE variant that has a mutation strategy "DE/current-to-$p$best" with external archive, and that independently controls $CR$ and $F$ for each individual component in an adaptive manner. An individual component has same representative ways as that in the classical DE.

■ **Mutation Strategy**

The mutation vector is generated by the strategy "DE/current-to-$p$best/1".

$$V_{i,G} = x_{i,G} + F_i \cdot (x_{pbest,G} - x_{i,G}) + F_i \cdot (x_{r_1,G} - x_{r_2,G}) \tag{3.21}$$

Where $x_{r_1,G}$ is an individual component randomly selected from $P$ and $x_{r_2,G}$ is an individual selected randomly from the union set $P \cup A$. When the archive is not used, $\mathbf{A}= \emptyset$. $x_{pbest,G}$ is an individual component randomly chosen from the top $100p\%$ individuals in $P$ with $p \in [0, 1]$. $p$ is a constant.

■ **External Archive**

JADE used an optional external archive in order to keep diversity. Parent vectors are preserved when they are worse than trial vectors. $P$ indicates the population. $\mathbf{A}$ is defined as the archive that stores the individuals that are discarded at the selection process. Therefore, $\mathbf{A} = \emptyset$ at the beginning. Archive size $\mid \mathbf{A} \mid$ is set to same as the population size $NP$. The randomly chosen variables are deleted to keep for newly inserted elements then the size of archive exceeds $\mid \mathbf{A} \mid$.

■ **Parameter Adaptation**

Two variables $\mu_{CR}$ and $\mu_F$, are provided for initial process with value of 0.5. At each generation $G$ and for each individual $i$, two parameters, $CR_i$ and $F_i$, are calculated as follows:

$$CR_i = randn_i(\mu_{CR}, 0.1) \tag{3.22}$$

$$F_i = randc_i(\mu_F, 0.1) \tag{3.23}$$

$CR_i$ is a random number generated by normal distribution of mean $\mu_{CR}$ and standard deviation 0.1, then it is converted to [0,1]. Similarly, $F_i$ is a random number generated by Cauchy distribution with location parameter $\mu_F$ and scale parameter 0.1, then it is converted to 1 when $F_i > 1$, otherwise it is regenerated.

■ **Crossover**

Crossover is applied in the same way as in Eq. (3.19) with the modification of $CR_i$.

$$u_{j,i,G} = \begin{cases} v_{j,i,G} \;\; if \;\; rand[0,1] \leq CR_i \;\; or \;\; j = j_{rand} \\ x_{j,i,G} \;\; otherwise \end{cases} \tag{3.24}$$

■ **Selection**

Selection is produced by Eq. (3.20). $CR_i$ and $F_i$ for generating $u_{i,G}$ are stored in $S_{CR,G}$ and $S_{F,G}$, respectively, in the case when $x_{i,G+1} = u_{i,G}$. $x_{i,G}$ is stored in $\mathbf{A}$ when the archive option is used.

The following parameter adaptations are produced when $S_{CR,G}$ and $S_{F,G}$ are not empty.

$$\mu_{CR,G+1} = (1-c) \cdot \mu_{CR,G} + c \cdot mean_A(S_{CR,G}) \tag{3.25}$$

$$\mu_{F,G+1} = (1-c) \cdot \mu_{F,G} + c \cdot mean_L(S_{F,G}) \tag{3.26}$$

Where $c \in (0,1)$ is a constant, $mean_A(S_{CR,G})$ is the arithmetic mean of $S_{CR,G}$ and $mean_L(S_{F,G})$ is the Lehmer mean value calculated by following equation:

$$mean_L(S_{F,G}) = \frac{\sum_{F \in S_{F,G}} F^2}{\sum_{F \in S_{F,G}} F} \tag{3.27}$$

Then, generation is renewed to calculate the next generation.

• **2. SHADE**

SHADE (Tanabe and Fukunaga, 2013) is proposed as an extended JADE, but it adopts a history of parameter adaptation of JADE. The new modification that controls the greediness constant $p$ is replaced with a variable $p_i$ to reduce the number of parameters.

■ **Parameter Adaptation**

In SHADE, $M_{CR}$ and $M_F$ as the memory areas, are provided for storing the parameters used. Each memory has a maximum memory size of $H$.

$$M_{CR} = M_{CR,0}, M_{CR,1}, ..., M_{CR,H-1} \tag{3.28}$$

$$M_F = M_{F,0}, M_{F,1}, ..., M_{F,H-1} \tag{3.29}$$

At the initial step, all values are set to 0.5, then they are updated in the search process using following equation:

$$M_{CR,k,G+1} = \begin{cases} mean_{WA}(S_{CR,G}) \;\; if \;\; S_{CR,G} \neq 0 \\ M_{CR,k,G} \;\; otherwise \end{cases} \tag{3.30}$$

$$M_{F,k,G+1} = \begin{cases} mean_{WL}(S_{F,G}) \;\; if \;\; S_{F,G} \neq 0 \\ M_{F,k,G} \;\; otherwise \end{cases} \tag{3.31}$$

Where $k \in [0, H-1]$ is the index which indicates the current place in the memory. $M_{CR,k,G}$ means the $k$th element in memory area $M_{CR}$ means the value of $CR$ at generation $G$ . $k$ is set to 0 at the initialization and increased following update. $k$ is set to 0 again when it is increased to $k = H-1$. $mean_{WA}(S_{CR,G})$ and $mean_{WL}(S_{F,G})$ are weighted means calculated by Eqs. (3.32)-(3.35), respectively.

$$mean_{WA}(S_{CR,G}) = \sum_{q=1}^{|S_{CR,G}|} \omega_q \cdot S_{CR,G,q} \tag{3.32}$$

$$\omega_q = \frac{\Delta f_q}{\sum_{q=1}^{|S_{CR,G}|} \Delta f_q} \tag{3.33}$$

$$\Delta f_q = \mid f(u_{q,G}) - f(x_{q,G}) \mid \tag{3.34}$$

$$mean_{WL}(S_{F,G}) = \frac{\sum_{q=1}^{|S_{F,G}|} \omega_q \cdot S_{F,G,q}^2}{\sum_{q=1}^{|S_{F,G}|} \omega_q \cdot S_{F,G,q}} \tag{3.35}$$

Therefore, $CR_i$ and $F_i$ are calculated as follows in the same way of Eqs. (3.36) and (3.37) as defined in JADE:

$$CR_i = randn_i(M_{CR,r_i}, 0.1) \tag{3.36}$$

$$F_i = randc_i(M_{F,r_i}, 0.1) \tag{3.37}$$

Where, $r_i \in [0, H-1]$ is random number.

■ **Mutation**

SHADE has similar mutation strategy "DE/current-to-$p$best/1" as adopted in JADE, except that a constant $p$ is replaced with:

$$p_i = rand[p_{min}, 0.2] \tag{3.38}$$

Where $p_{min}$ is selected by the calculation of mutation.

## 3.3 Summary

Swarm intelligence based algorithms and evolutionary algorithms were introduced as bio-inspired algorithms which proposed by mimicking the successful characteristics of the complex systems inspired from nature. A swarm of honey bee is very attractive in nature that the swarm allocates the task more dynamically and adapts to environment by collective intelligent manner. Like all social insects, the honey bees have the opportunity to engage in social learning by interacting with simple other entities in their social group, while the group of insects possesses collective intelligence, the individuals within the group have complex cognitive abilities. Self-organization, stigmergy and division of labor were interpreted in the first section as three component keys in bee colony. Standard ABC algorithm was inspired from the behavior of honey bees related to the food sources, employed foragers and unemployed foragers with regards to the food foraging behavior. The ABC algorithm was described in detailed explanations with the flowchart.

Following the introduction of ABC algorithm, PSO algorithm as one of the classical SI based algorithms, GA and its extended version NRGA, DE and its extended version SHADE which belong to EAs were given the explanations in more detail. These SI based algorithms and EAs will be used to compare with the proposed extended ABC algorithms in Chapter 4 and Chapter 5 and comparative analyses will be conducted to evaluate the performance of them.

# Chapter 4

# Improved Hybrid ABC (IHABC) Algorithm

## 4.1 Previous Research about ABC Algorithms

### 4.1.1 Modified Versions of ABC Algorithms

The standard ABC algorithm was proposed based on the results of some standard benchmark problems, however, as an initial proposal, it still has a considerable performance gap with respect to state-of-the-art algorithms. In particular, it was found to have poor performance on composite and non-separable functions, and have a slow convergence rate toward high quality solutions. To improve the performance, the ABC algorithm has been extended in a number of ways recently. Some of the improved versions of ABC algorithms are taken as examples shown as Table 4.1.

In addition, ABC algorithm requires fewer training parameters, therefore combining it with other algorithms is easier. ABC algorithm was combined with some other SI based algorithms or EAs in order to enhance the performance of ABC. Many hybridized ABC algorithms have been proposed and some of them are taken as the examples as shown Table 4.2.

Table 4.1: Improved versions of ABC algorithms

| Algorithm Name | Modifications and Improvements | References |
|---|---|---|
| Modified ABC | using several selection strategies such as disruptive selection strategy, tournament selection strategy and rank selection strategy in order to improve the population diversity and avoid the premature | Bao and Zeng (2009) |
| Chaotic ABC | many chaotic maps for parameters adapted from the original ABC were introduced to improve its convergence performance | Alatas (2010) |
| G-best guided ABC | incorporate the information of global best solution into the solution search equation to improve the exploitation | Zhu and Kwong (2010) |
| Incremental ABC | integrate the population growth and local search with standard ABC algorithm | Aydin et al. (2011) |
| Best-so-far ABC | enhance the exploration and exploitation processes by biasing the solution direction toward the best-so-far position | Banharnsakun et al. (2011) |
| ABC with two variants | improve the performance of the algorithm by using new methods for the position update of the artificial bees | Diwold et al. (2011) |
| Enhanced ABC | overcome the slow convergence speed of the ABC algorithm by controlling the frequency of perturbation and magnitude of perturbation | Akay and Karaboga (2012) |
| Improved ABC | improve the performance of the algorithm by using a modified solution search equation with chaotic initialization | Gao and Liu (2012) |

### 4.1.2 State-of-the-art Modified ABC Algorithms

In this thesis, the state-of-the-art ABC algorithms are introduced to compare the their performance with proposed extended ABC algorithms. These state-of-the-art ABC algorithms are representative with promising experimental performance proposed in various journal papers. Best-so-far ABC (BsfABC) and incremental ABC (IABC) are described in following sections as the representative ABC algorithms.

#### Best-so-far ABC (BsfABC) Algorithm

To enhance the exploitation and exploration processes, best-so-far ABC (BsfABC) algorithm was proposed by Banharnsakun et al. (2011). In this BsfABC algorithm, three major changes were introduced. All onlooker bees use the information from all employed bees

Table 4.2: Hybridized ABC algorithms

| Algorithm Name | Modifications and Improvements | References |
|---|---|---|
| Novel hybrid ABC and Quantum Evolutionary Algorithm | avoid the premature convergence and find the optimal value by adopting ABC to increase the local search capacity as well as the randomness of the populations | Duan et al. (2010) |
| Novel bi-group differential artificial bee colony | improve the performance of ABC by combined with DE through opposition-based learning for initial individuals and performing the evolutions of two sub-groups | Bao and Zeng (2011) |
| Hybrid ABC assisted DE | enable significant reduction in population size required with less convergence time and balance between local and global search ability | Li et al. (2011) |
| Efficient and robust ABC | improve the convergence performance of ABC through combinatorial solution search equation for accelerating the search process, chaotic search technique and reverse selection based on roulette wheel, in addition, chaotic initialization for enhancing the global convergence ability | Xiang et al. (2013) |
| Local global variant ABC | harness the local and global variant of PSO into ABC, the variant can get high quality solutions efficiently through solving a set of thirteen well known constrained benchmarks problems and three chemical engineering problems | Sharma and Pant (2013) |
| Enhanced ABC | improve the performance of exploitation and exploration ability by introducing self-adaptive searching strategy and artificial immune network operators | Chen and Xiao (2014) |

to make a decision on a new candidate food source. Thus, the onlookers can compare information from all candidate sources and are able to select the best-so-far position which will lead to optimal solution. The new method used to calculate a candidate food source is shown as Eq. (4.1).

$$v_{ij} = x_{ij} + \phi \times f_b(x_{ij} - x_{bj}) \tag{4.1}$$

where $v_{ij}$ is the new candidate food source for onlooker bee position $i$ and dimension $j$, $j = 1, 2, ..., D$, $x_{ij}$ is the selected food source position $i$ in a selected dimension $j$, $\phi$ is a random number selected from [-1,1], $f_b$ is the fitness value of the best food source so far, $x_{bj}$ is the best-so-far food source in selected dimension $j$. A global search ability for the

scout bee was introduced as shown in Eq. (4.2) for resolving the problem of trapping in local optimum.

$$v_{ij} = x_{ij} + \phi_{ij}[\omega_{max} - \frac{iteration}{MCN}(\omega_{max} - \omega_{min})]x_{ij} \tag{4.2}$$

where $v_{ij}$ is a new feasible solution of a scout bee that is modified from the current position of an abandoned food source ($x_{ij}$) and $\phi_{ij}$ is a random number between [-1,1]. $MCN$ denotes maximum cycle number. The values of $\omega_{max}$ and $\omega_{min}$ represent the maximum and minimum percentage of the position adjustment for the scout bee. The value of $\omega_{max}$ and $\omega_{min}$ are fixed to 1 and 0.2, respectively.

### Incremental ABC (IABC) Algorithm

Aydin et al. (2011) proposed another modified ABC algorithm named incremental ABC (IABC) algorithm. IABC algorithm begins with few food sources. New food sources are placed biasing their location towards the location of the best-so-far solution. This is implemented as Eq. (4.3).

$$x'_{new,j} = x_{new,j} + rand[0,1](x_{gbest,j} - x_{new,j}) \tag{4.3}$$

where $x_{new,j}$ is the randomly generated new food source location, $x'_{new,j}$ is the updated location of the new food source, $x_{gbest,j}$ refers to best-so-far food source location.

Another modification is applied by the scout bees step in IABC. The difference is a replacement factor parameter, $R_{factor}$, that controls how much of the new food source locations will be closer to the best-so-far food source. This modified rule is specified with Eq. (4.4).

$$x'_{new,j} = x_{gbest,j} + R_{factor}(x_{gbest,j} - x_{new,j}) \tag{4.4}$$

The other difference between the standard ABC and IABC is that employed bees search in the vicinity of $x_{gbest,j}$ instead of a randomly selected food source. This modification boosts the exploitation behavior of the algorithm and helps to converge quickly towards good solutions.

The BsfABC and IABC algorithms are introduced to compare with SAHEABC algorithm in Chapter 5, and the comparative performance of these algorithms will be evaluated based on the CEC'14 test suite experiments.

## 4.2 Introduction of IHABC Algorithm

Exploration and exploitation are two key components for meta-heuristic algorithms ( Blum and Roli, 2003; Rashedi, 2009). Exploration is the ability to expand the search space more thoroughly and to help generate diverse solutions, and exploitation is the ability to find optima around better solutions. Too much exploration will increase the probability of finding the true optimum globally, but it may often reduce the speed of process with much lower convergence rate. On the other hand, too much exploitation will make the optimization process converse quickly, but it may lead to prematurely converge with local optimum, sometime it may get wrong solution. Moreover, it will also reduce the probability of finding the global optima. Exploration and exploitation play key roles in SI based algorithms and EAs, they coexist in the evolutionary process of algorithms such as PSO, DE, and ABC, in the meantime, they contradict each other.

Along with the advantages of the improved versions of ABC algorithms as analyzed in Section 4.1, however, a few disadvantages still exist. For example, ABC algorithms have low convergence speeds when they solve some unimodal and composition functions, have low exploitation abilities, and are also trapped in local optima easily when they solve complex multimodal functions (Karaboga and Akay, 2009). Inspired by self adaptive mechanism, incorporated with DE and PSO algorithms, an improved hybrid ABC (IHABC) algorithm is proposed to overcome these disadvantages, and achieve better performance with more higher convergence speed, exploitation ability and avoid trapped in local optimum through the characteristic of predominance of hybridizing other algorithms easily. As the methods adopted for the proposed IHABC algorithm, self adaptive mechanism and hybridization with DE and PSO algorithms are described as following.

In the standard ABC algorithm, a random perturbation is added to the current solution to produce a new solution. This random perturbation is weighted by $\phi_{ij}$ selected from [-1,1] and is a uniformly distributed real random number in the standard ABC. Too large or too

small values will affect the convergence speed. Therefore, a self adaptive mechanism is introduced to balance the exploration ability and the convergence speed of the algorithm for employed bees. The self adaptive ABC approach has a very simple structure and is easy to implement. $\phi_{ij}$ is changed with the cycle number according to a random value called *rand* in the range [0,1] for food searching process of employed bee. $\phi_{ij}$ is defined as $\phi_{ij} = 1 - 2 * rand$ in the standard ABC and changed to self adaptive mechanism shown as Eq. (4.5).

$$\phi_{ij} = \begin{cases} -\mathrm{e}^{\frac{-0.12cycle}{MCN}}, 0 \leq rand \leq 0.5 \\ \mathrm{e}^{\frac{-0.12cycle}{MCN}}, 0.5 < rand \leq 1 \end{cases} \tag{4.5}$$

The DE algorithm has been proved to be a simple yet powerful and efficient population based algorithm for many global optimization problems. To further improve the performance of the DE algorithm, researchers have suggested different schemes of DE (Das and Suganthan, 2011). Like other evolutionary algorithms, DE also relies on an initial random population generation and then improves its population via mutation, crossover and selection processes. The mutation equations for DE were shown as Eqs. (3.13)-(3.17) in Chapter 3. The searching food source process in ABC algorithm is similar to the mutation process of DE. Moreover, the best solution in the current population is very useful for improving convergence performance for DE. As one scheme of the mutations of DE, "DE/best/1" can effectively maintain the population diversity. Therefore, the "DE/best/1" mutation strategy is combined with the searching food process of ABC algorithm to produce a new search equation (4.6) and improved the convergence ability.

$$v_{ij} = x_{best,j} + \phi_{ij}(x_{ij} - x_{kj}) \tag{4.6}$$

Where $i, k \in 1, 2, ..., SN$ are randomly selected indexes and $k$ different from $i$; $j \in 1, 2, ..., D$ is a randomly selected index and $\phi_{ij}$ was given in Eq. (4.5).

According to the previous research of Kong et al. (2013), the search ability of the ABC algorithm is good at exploration, but poor in terms of exploitation which affects the convergence speed. Specifically, the relationship between employed bees and onlooker bees are seen as relationship between exploration and exploitation, respectively. Employed bees

explore new food sources and send information to onlooker bees, and onlooker bees exploit the food sources explored by employed bees.

In the standard ABC algorithm, much time is required to find the food source due to poor exploitation abilities. PSO is incorporated into ABC algorithm in order to improve the exploitation ability of ABC algorithm. The equations of PSO are shown as Eqs. (3.5) and (3.6). The modification for onlooker bee search solution is shown as Eq. (4.7) by taking advantage of the search mechanism of PSO.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(x_{best,j} - x_{ij}) \tag{4.7}$$

Where $i, k \in 1, 2, ..., SN$ are randomly selected indexes and $k$ different from $i$; $j \in 1, 2, ..., SN$ is a randomly selected index; $x_{best,j}$ is the $j$th element of the best solution so far, and $\varphi_{ij} \in [-1, 1]$ and $\psi_{ij} \in [0, 1.5]$ are uniformly distributed random numbers.

The main modifications for IHABC algorithm on the standard ABC are as follows.

- The modifications with Eqs. (4.5) and (4.6) substitute the Eq. (3.2) on step 4 of main steps of standard ABC algorithm in Section 3.1 to balance the exploration ability and the convergence speed of the algorithm for employed bees
- The modification with Eq. (4.7) substitutes the Eq. (3.2) on step 8 of main steps of standard ABC algorithm in Section 3.1 to improve the exploitation ability of ABC algorithm

## 4.3 Experimental Setup

ABC and IHABC algorithms are evaluated for all 28 test functions defined by CEC'13 test suite with parameters selected by comparing experiments on three dimension sizes, i.e., 10 (10D), 30 (30D) and 50 (50D) respectively. The 28 test functions are executed 51 times with respect to each test function at each problem dimension size. The search range is defined as $[-100, 100]^D$.The algorithms are terminated when the $MCN$ is reached for function evaluations or the error value is smaller than $10^{-8}$. In the experiments, firstly, the numbers of $MCN$ are set for 10,000, 30,000, and 50,000 on 10D, 30D and 50D respectively for standard ABC and IHABC algorithms.

Table 4.3: Parameter adjustment experimental results

| Limit/NP | 20 | 50 | 100 | 150 | 200 | 300 |
|---|---|---|---|---|---|---|
| 50 | -/-/- | -/-/- | -/-/- | -/-/- | -/-/- | -/-/- |
| 100 | -/-/- | -/-/- | 10/30/- | 10/-/50 | -/-/- | -/-/- |
| 150 | -/-/- | 10/30/- | 10/30/50 | 10/-/50 | 10/-/50 | -/-/- |
| 250 | -/-/- | 10/30/- | 10/30/- | -/30/50 | -/30/50 | -/-/- |
| 400 | -/-/- | 10/-/- | -/30/50 | -/-/50 | -/-/- | -/-/- |

Table 4.3 shows the parameter adjustment experimental results. If the IHABC algorithm reaches the value of *Limit*, the position of food source can not be improved further and a new position will be produced. $NP$ is the number of population size. In this table, "-/-/-" indicates the comparative performance of ABC algorithm on 10D, 30D and 50D. "-" indicates that its performance is significantly worse than others on that dimension size.

According to Table 4.3, it is observed that ABC is not sensitive to the parameter choice of much lower or higher population size and limit for 28 test functions defined by CEC'13. Parameters of *limit* and $NP$ are selected for 150 and 100 respectively by comparison experiment. The food source number is set to half of the $NP$ for 50, in the meantime, the employed bee number and onlooker bee number is set to same as food source number. The number of scout bee is set to1 for each cycle.

After the comparison of ABC and IHABC algorithms, comparative experiments for ABC, IHABC, DE (Qin and Li, 2013) and PSO (Stephen et al., 2013) algorithms are conducted. In the experiments, the numbers of $MCN$ are set for 100,000, 300,000, and 500,000 on 10D, 30D, and 50D respectively. The experimental results are analyzed statistically using Wilcoxon rank sum test with significance level of 0.05.

## 4.4 Experimental Results

Table 4.4 illustrate the mean values of ABC and IHABC algorithms for 10,000, 30,000, and 50,000 evaluations on 10D, 30D and 50D respectively.

After conducting comparison experiments between ABC and IHABC algorithms with

Table 4.4: Function evaluations for ABC and IHABC on 10D, 30D and 50D

| F./Eva. | ABC(10D) | IHABC(10D) | ABC(30D) | IHABC(30D) | ABC(50D) | IHABC(50D) |
|---|---|---|---|---|---|---|
| F1 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 |
| F2 | 2.07e+06 | 1.63e+05 | 7.08e+06 | 2.67e+06 | 1.23e+07 | 8.94e+06 |
| F3 | 4.96e+06 | 5.09e+05 | 2.63e+08 | 6.01e+06 | 7.43e+08 | 3.41e+07 |
| F4 | 1.11e+04 | 1.21e+03 | 6.83e+04 | 4.00e+04 | 1.31e+05 | 9.42e+04 |
| F5 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 |
| F6 | 3.82e+00 | 7.13e+00 | 1.29e+01 | 3.31e+01 | 3.94e+01 | 4.72e+01 |
| F7 | 3.56e+01 | 1.09e+01 | 1.01e+02 | 5.25e+01 | 6.51e+01 | 9.11e+01 |
| F8 | 2.03e+01 | 2.02e+01 | 2.07e+01 | 2.09e+01 | 2.11e+01 | 2.10e+01 |
| F9 | 5.41e+00 | 2.61e+00 | 2.79e+01 | 2.82e+01 | 5.66e+01 | 5.58e+01 |
| F10 | 1.01e+00 | 6.35e-01 | 1.62e-01 | 3.13e-01 | 1.28e-01 | 3.11e-01 |
| F11 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 | 0.00e+00 |
| F12 | 2.75e+01 | 7.27e+00 | 2.41e+02 | 5.82e+01 | 6.42e+02 | 2.40e+02 |
| F13 | 3.40e+01 | 8.05e+00 | 2.19e+02 | 1.20e+02 | 6.97e+02 | 3.57e+02 |
| F14 | 9.59e-02 | 3.25e-03 | 7.51e-01 | 0.00e+00 | 2.53e+00 | 2.50e-03 |
| F15 | 6.67e+02 | 4.80e+02 | 4.53e+03 | 3.67e+03 | 9.64e+03 | 7.09e+03 |
| F16 | 6.49e-01 | 6.49e-01 | 1.22e+00 | 1.65e+00 | 1.68e+00 | 2.44e+00 |
| F17 | 9.50e+00 | 1.01e+01 | 3.04e+01 | 3.04e+01 | 5.08e+01 | 5.08e+01 |
| F18 | 3.73e+01 | 2.04e+01 | 2.77e+02 | 1.20e+02 | 6.93e+02 | 3.11e+02 |
| F19 | 4.27e-02 | 2.49e-02 | 8.75e-01 | 2.04e-01 | 5.32e-01 | 1.23e+00 |
| F20 | 3.26e+00 | 2.10e+00 | 1.43e+01 | 1.29e+01 | 2.44e+01 | 2.33e+01 |
| F21 | 1.68e+02 | 3.30e+02 | 1.70e+02 | 3.15e+02 | 2.15e+02 | 7.57e+02 |
| F22 | 1.31e+01 | 2.30e+01 | 2.51e+01 | 7.17e+01 | 1.54e+01 | 6.27e+01 |
| F23 | 1.04e+03 | 5.61e+02 | 4.48e+03 | 4.20e+03 | 9.96e+03 | 9.30e+03 |
| F24 | 1.32e+02 | 1.10e+02 | 2.88e+02 | 2.51e+02 | 3.06e+02 | 3.33e+02 |
| F25 | 1.64e+02 | 1.24e+02 | 3.04e+02 | 2.81e+02 | 4.00e+02 | 3.87e+02 |
| F26 | 1.38e+02 | 1.07e+02 | 2.00e+02 | 2.00e+02 | 2.01e+02 | 2.01e+02 |
| F27 | 3.79e+02 | 3.59e+02 | 4.00e+02 | 4.00e+02 | 4.01e+02 | 4.01e+02 |
| F28 | 1.82e+02 | 2.73e+02 | 1.70e+02 | 3.00e+02 | 4.00e+02 | 4.00e+02 |

the function errors of mean value for evaluations, the number of better (+) , similar (≈) and worse (-) performance of 28 test functions defined by CEC'13 are listed in Table 4.5. With the dimension size increasing, the number of functions with better performance for IHABC algorithm is decreasing. From Table 4.5, it is concluded that the number of functions with better and similar performance of IHABC is much better than ABC, the both algorithms have better performance for the multimodal functions.

Table 4.5: Comparison performance of IHABC versus ABC

| IHABC VS. ABC | 10D | 30D | 50D |
|---|---|---|---|
| + | 18 | 13 | 11 |
| ≈ | 4 | 8 | 8 |
| - | 6 | 7 | 9 |

Figures 4.1-4.6 illustrate the convergence performance of the both algorithms ABC and IHABC with function evaluations on 10D, 30D and 50D respectively. According to Figs. 4.1-4.4, the performance of IHABC for functions F2, F3, F12 and F13 is much better than ABC on 30D and 50D, better than ABC on 10D, but not so significantly. The much better performance could be seen for functions F4 on 10D, F7 on 10D and 30D, F10 on 10D, F14 on 30D and 50D, F18 on 10D and 50D, F20 on 10D and F23 on 10D, respectively according to Table 4.4. The performance of IHABC is better than ABC but not so significantly for the functions F15, F25 on 10D, 30D, 50D, F4 on 30D, F16 on 50D, F19 on 10D and 30D, F20 and F23 on 30D and 50D and F26, F27 on 10D, respectively according to Table 4.4.

There is no significant difference for the convergence performance of function F20 with both ABC and IHABC algorithms on 10D, 30D and 50D according to Fig. 4.5. The similar performance could be seen for functions F24, F25, F26, and F27 on 10D, 30D and 50D according to Table 4.4. The convergence performance is much similar for both ABC and IHABC algorithms on 10D, 30D and 50D from Figure 4.6. The similar performance can be seen for functions F8 and F24 on 10D, 30D, 50D, F10, F26, F27 on 30D and 50D, F13 on 10D and F17 on 30D, respectively according to Table 4.4.

The best results could be seen for functions F1, F5, and F11 on 10D, 30D and 50D because all the function errors of the mean value reached zero.

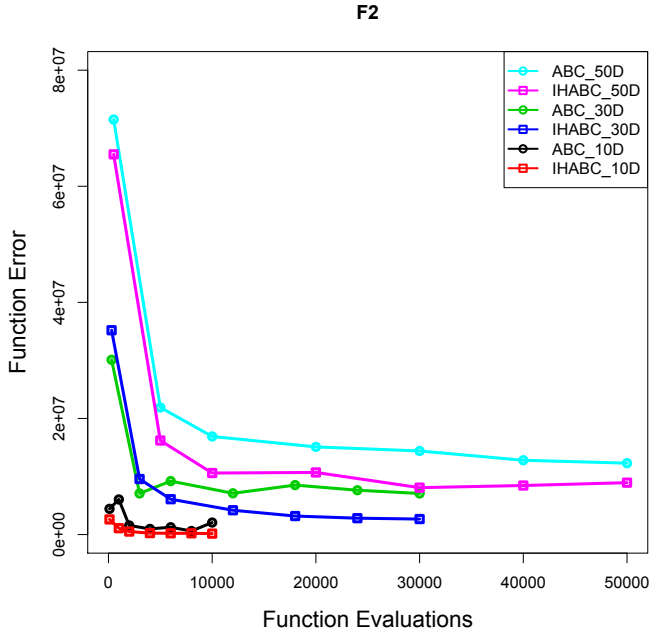



Figure 4.1: Convergence for F2 on D10, D30, and D50

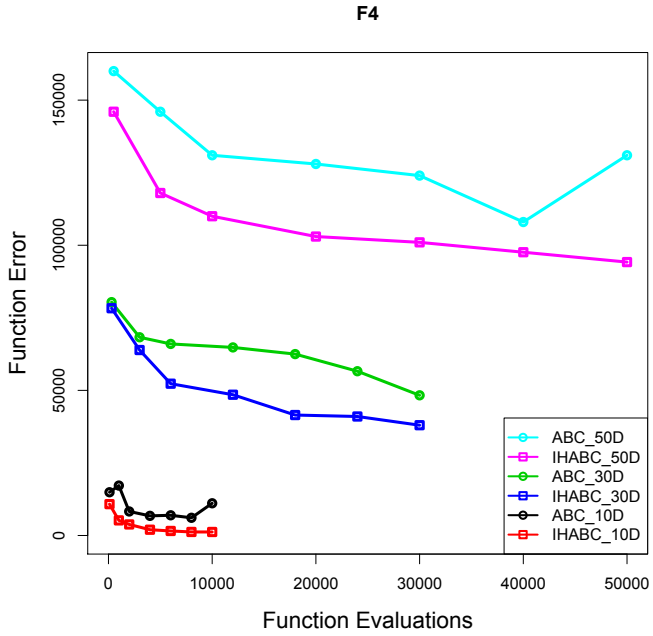



Figure 4.2: Convergence for F4 on D10, D30, and D50

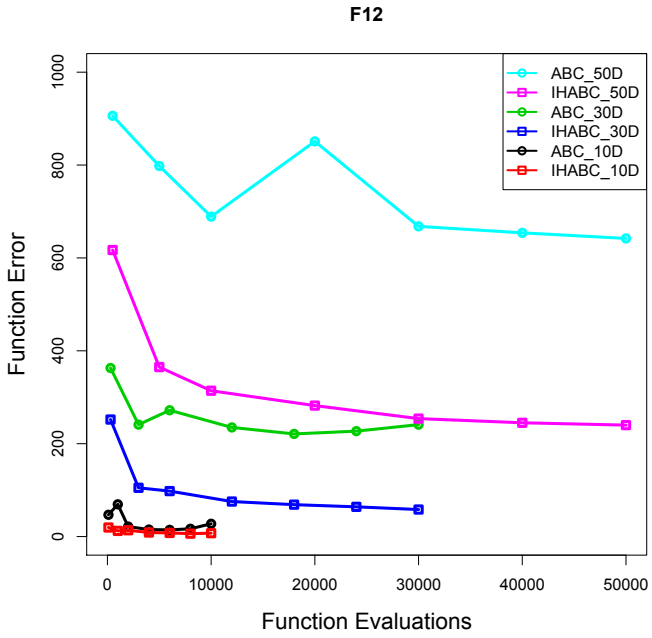



Figure 4.3: Convergence for F12 on D10, D30, and D50

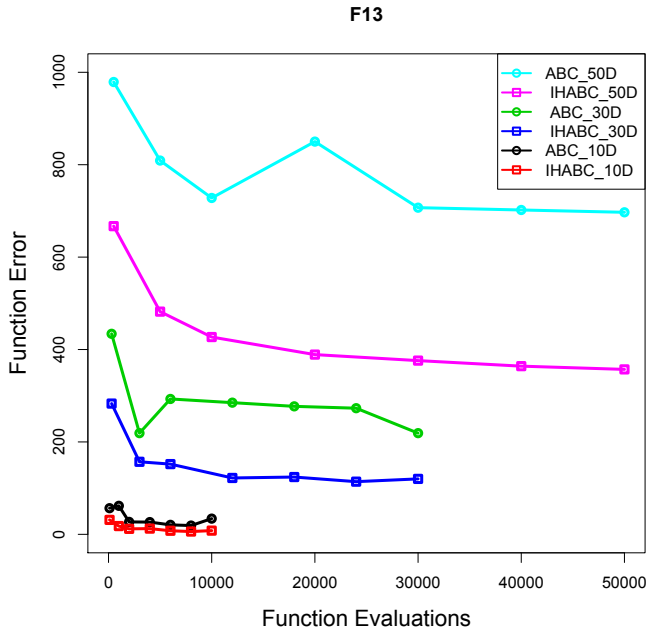



Figure 4.4: Convergence for F13 on D10, D30, and D50

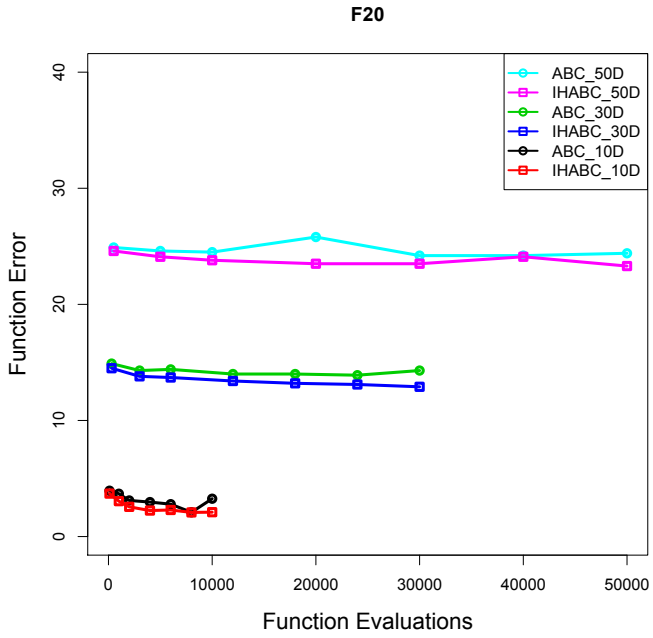



Figure 4.5: Convergence for F20 on D10, D30, and D50

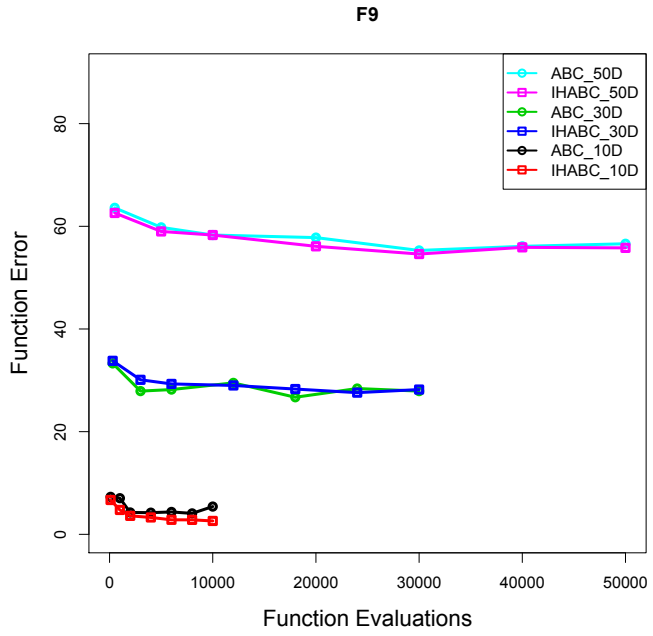



Figure 4.6: Convergence for F9 on D10, D30, and D50

According to the above Tables 4.4-4.5 and Figs. 4.1-4.6 obtained from experimental results, ABC and IHABC algorithms were not much effective for unimodal functions except for function F1, however, the performance of IHABC is much better than ABC algorithm. For all the evaluations, functions F1, F5, and F11 got the best performance with the function errors of mean value reached zero and the performance of functions F8, F9 and F24 were much similar for all evaluation stages. It was also observed that IHABC algorithm outperformed on those functions with the properties of non-separability, having many local optima and second local optimum is far from the global optimum for the unimodal and multi-modal functions.

After the comparison experiment of standard ABC and IHABC algorithms, the comparative experiments of IHABC, ABC, DE and PSO algorithms are implemented with the function errors of mean values by the $MCN$ numbers for 100,000, 300,000, and 500,000 evaluations on 10D, 30D and 50D, respectively. The parameters are set as same as shown in Section 4.3. After the comparison experiment, the convergence performance with the numbers of better (+) , similar ($\approx$) and worse (-) of the function errors of mean value of the IHABC algorithm versus ABC, DE and PSO algorithms are statistically analyzed as demonstrated in Table 4.6 on 28 test functions defined by CEC'13.

Table 4.6: Comparison performance with function errors of mean value for IHABC algorithm versus ABC, DE, and PSO algorithms

| IHABC (10D) VS. | ABC | DE | PSO |
|---|---|---|---|
| + | 7 | 19 | 24 |
| $\approx$ | 15 | 6 | 4 |
| - | 6 | 3 | 0 |
| IHABC (30D) VS. | ABC | DE | PSO |
| + | 8 | 17 | 22 |
| $\approx$ | 13 | 5 | 4 |
| - | 7 | 6 | 2 |
| IHABC (50D) VS. | ABC | DE | PSO |
| + | 10 | 14 | 15 |
| $\approx$ | 10 | 6 | 9 |
| - | 8 | 8 | 4 |

With the dimension size increases, the number of functions with better and similar convergence performance of the IHABC decreased compared to ABC, DE and PSO algorithms according to Table 4.6. It is concluded that the better and similar convergence performance of IHABC algorithm are quite competitive to ABC, DE, and PSO algorithms. Especially, the convergence performance is very significant when it is compared to PSO algorithm.

Figures 4.7-4.16 illustrate the box-plots for function errors of mean value of IHABC, ABC, DE, and PSO algorithms on 10D, 30D and 50D, respectively. The numbers of "1, 2, 3, 4" indicate that the function errors of mean value of IHABC, ABC, DE, and PSO algorithms on 10D, in the same way, the numbers of "5, 6, 7, 8" , and "9, 10, 11, 12" indicate that the function errors of mean value of IHABC, ABC, DE, and PSO algorithms on 30D, and 50D, respectively. Figs. 4.7 and 4.8 illustrate the box-plots for function errors of the mean value of unimodal functions. According to Fig. 4.7, the convergence performance of the IHABC algorithm is same as ABC, DE, and PSO algorithms for function F1 on 10D, 30D, and 50D, it is also observed that function F1 reached the best value of zero for function error of mean value. For Fig. 4.8, the convergence performance of IHABC algorithm is competitive to ABC, but not better than DE, and PSO algorithms for function F4 on 30D and 50D; DE algorithm gets the best convergence performance on 10D, 30D and 50D. For the remaining unimodal functions, the performance of IHABC is the best on 10D and better than ABC on 30D, DE is the best on 30D and 50D for function F2, the performance of all algorithms are similar on 10D and 30D, but PSO performed the worst on 30D and 50D for function F3. As a whole, the performance of IHABC was much competitive to others on lower dimensional sizes of 10 and 30, but DE reached the best performance on 50D for unimodal functions.

Figures 4.9-4.13 show the box-plots for function errors of mean value of multi-modal functions. According to Fig. 4.9, the convergence performance of IHABC is quite similar to ABC, DE, and PSO algorithms for function F8 on 10D, 30D, and 50D. With regard to Fig. 4.10, it is observed that the convergence performance of IHABC algorithm for function F13 is competitive to other algorithms, but the ABC algorithm is not better than DE and PSO algorithms on 50D. Moreover, DE algorithm is better than ABC and PSO algorithms on 30D and 50D. The convergence performance sequence for these algorithms is IHABC, ABC, DE and PSO for function F14 on 10D, 30D, and 50D with respect to Fig.

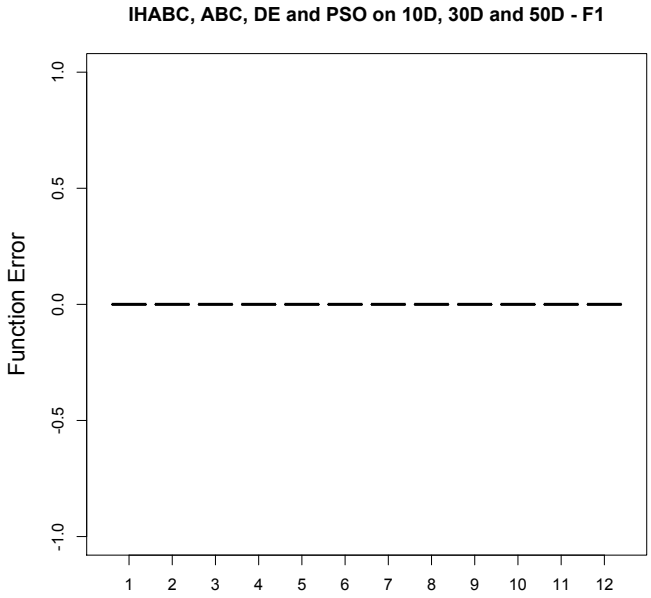



Figure 4.7: Box-plot of F1 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

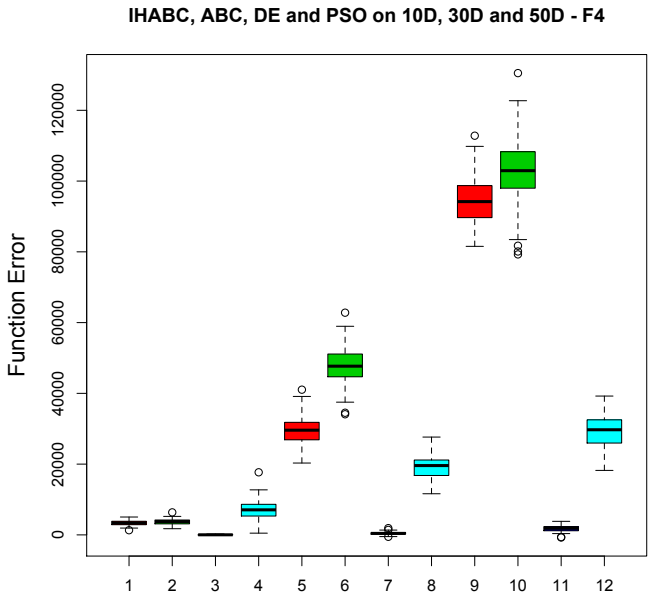



Figure 4.8: Box-plot of F4 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

4.11, especially, ABC and IHABC algorithms reached the best performance.

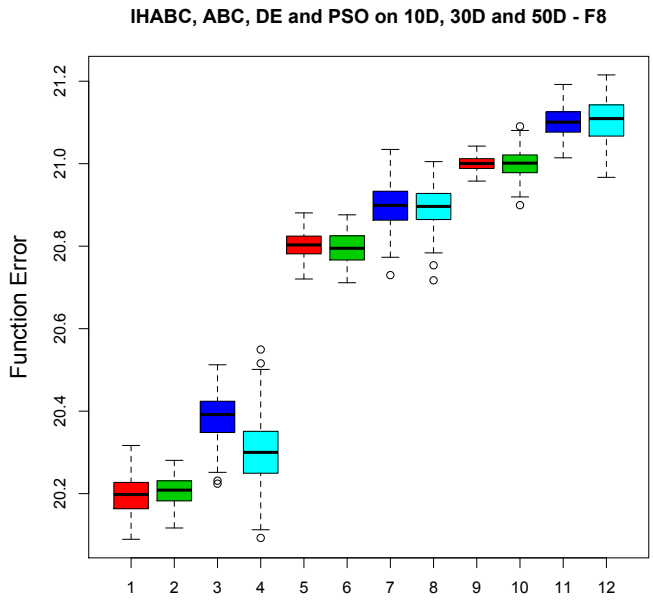



Figure 4.9: Box-plot of F8 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

According to Fig. 4.12, it can be seen that the convergence performance of the IHABC and ABC algorithms are very similar and better than DE and PSO algorithms for function F17. The convergence performance of DE is better than PSO algorithm on 30D and 50D, but worse on 10D. For Fig. 4.13, the convergence performance of IHABC is better than ABC, but worse than DE and PSO algorithms on 30D and 50D for function F18. For the remaining multimodal functions, it is observed that the convergence performance of IHABC and ABC algorithms are competitive to DE and PSO algorithms, DE is much better than PSO algorithm. Especially, functions F5 and F11 achieve the best performance as function F1, functions F9 and F20 reach similar performance as function F8 for all four algorithms. IHABC is much competitive to others on 10D and 50D, but ABC outperforms than IHABC on 30D for function F6, DE and PSO have similar performance on 10D, 30D and 50D. The performance of IHABC is better than ABC and PSO on 10D, 30D and 50D, similar as DE on 10D but worse than DE on 30D and 50D for function F7. With regard to function F10, the performance is much better or similar for all algorithms on all

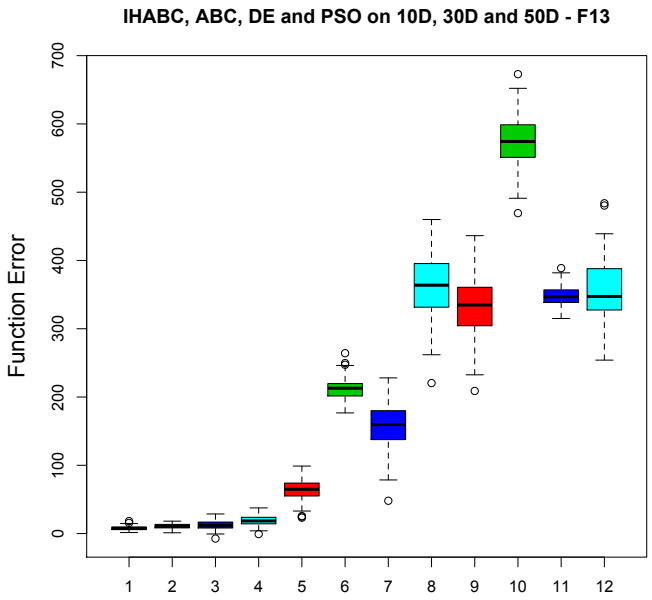



Figure 4.10: Box-plot of F13 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

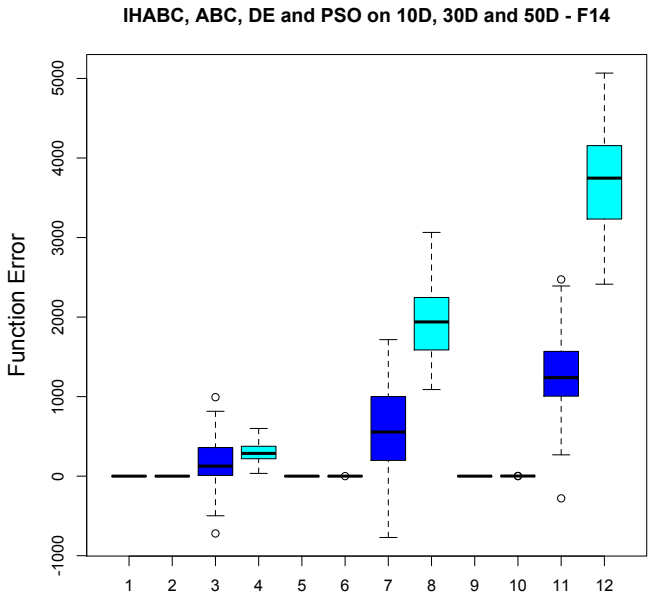



Figure 4.11: Box-plot of F14 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

dimension sizes, but PSO is the worst. IHABC performs the best among them on 10D and 30D, however, PSO reaches the best performance on 50D. Regarding to functions F15 and F16, the performance of IHABC and ABC is similar on 10D, ABC outperforms than IHABC on 30D and 50D, but both ABC and IHABC algorithms are better than DE and PSO algorithms. IHABC and ABC algorithms get the best performance on 10D, 30D and 50D, but PSO is the worst.

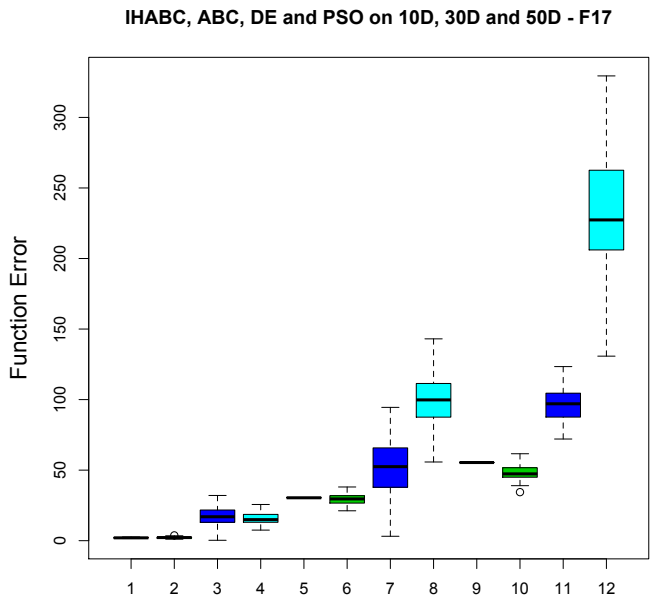



Figure 4.12: Box-plot of F17 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

Figures 4.14-4.16 illustrate the box-plots for function errors of mean value of composition functions. According to Fig. 4.14, the convergence performance of IHABC and ABC algorithms is very similar, but much better than DE and PSO algorithms, the convergence performance of DE algorithm is competitive to PSO algorithm on 10D, 30D, and 50D for function F22. Figure 4.15 shows that IHABC gets better convergence performance than ABC algorithm on 10D and 50D for function F24, however, DE reaches the best on 30D and 50D. In addition, IHABC and ABC algorithms are very similar and much competitive to DE and PSO algorithms. According to Fig. 4.16, HABC is very similar to ABC algorithm, DE performs better than PSO algorithm on 10D, 30D and 50D for function F27. For

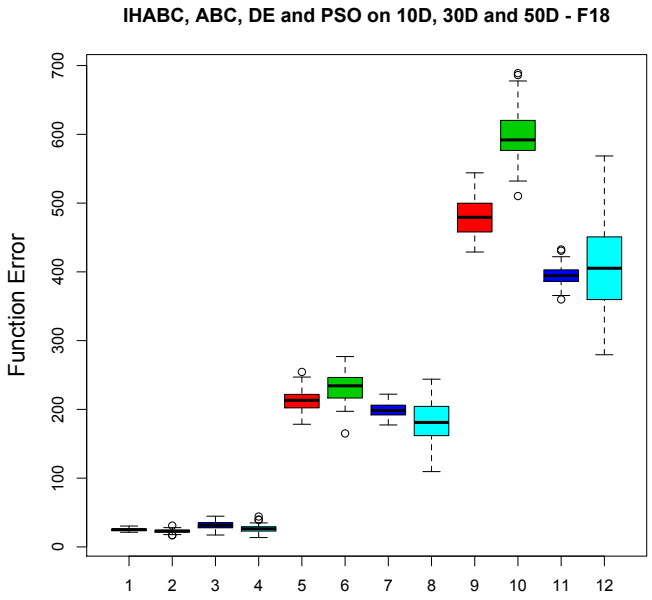



Figure 4.13: Box-plot of F18 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

the remaining composition functions, it is concluded that the convergence performance of IHABC algorithm is better or similar to ABC algorithm; DE algorithm is competitive to PSO algorithm. However, PSO performs well for functions F21 and F23 on 10D, 30D and 50D. IHABC reaches the best performance for function F25 on 10D, function F26 on 10D, 30D and 50D as well as function F28 on 50D.

According to the above analyses with Table 4.6 and Figs. 4.7-4.16 obtained from the experimental results, it was concluded that IHABC and ABC algorithms were not so effective on unimodal functions except for function F1, but IHABC outperformed than ABC algorithm. For multi-modal and composition functions, the convergence performance of IHABC was the best as a whole, especially on lower dimensional sizes of 10 and 30. The convergence performance of ABC was better than DE and PSO algorithms. DE outperformed than PSO algorithm. Functions F1, F5, and F11 reached the best convergence performance with function errors of mean value of zero for all 10D, 30D, and 50D. It was also observed that the performance of IHABC algorithm was better than ABC, DE and PSO on those functions with the properties of non-separability, asymmetry, having many

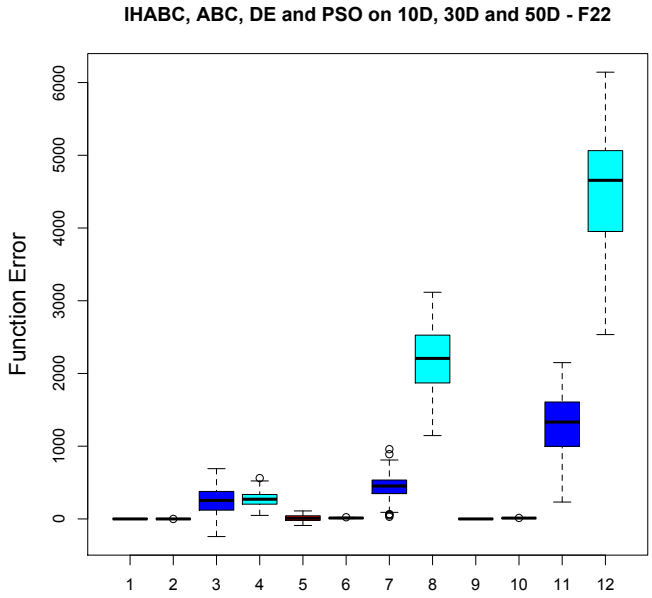



Figure 4.14: Box-plot of F22 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

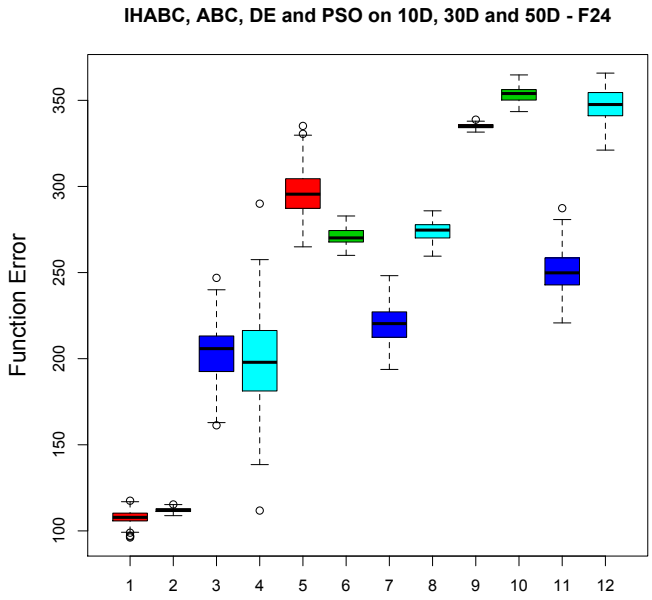



Figure 4.15: Box-plot of F24 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

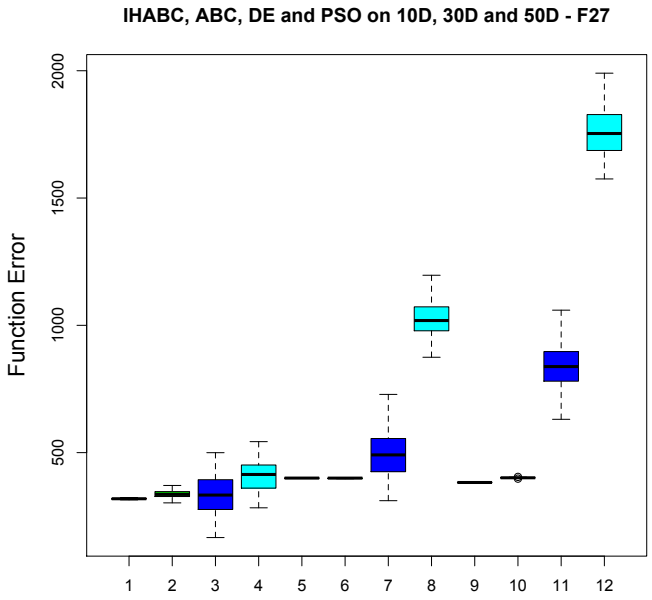



Figure 4.16: Box-plot of F27 for IHABC, ABC, DE and PSO on 10D, 30D and 50D

local optima and second local optimum is far from the global optimum for unimodal and multi-modal functions.

## 4.5 Summary

The previous research about the modified versions of ABC algorithms and the state-of-the-art modified ABC algorithms of BsfABC and IABC algorithms were introduced firstly.

In order to overcome the disadvantages such as low convergence speeds when they solve unimodal functions, low exploitation abilities and easily trapped in local optima when they solve complex multimodal functions, an improved hybrid ABC (IHABC) algorithm was proposed inspired by self adaptive mechanism, incorporated with DE and PSO algorithms.

Comparison experiments of ABC and IHABC algorithms were conducted for 28 test functions defined by CEC'13 test suite. After the comparison of ABC and IHABC algorithms, additional comparative experiments for ABC, IHABC, DE and PSO algorithms

were implemented with the same parameter setups then the experimental results were analyzed statistically.

As the experimental results, IHABC and ABC algorithms were not effective on unimodal functions except for function F1, but IHABC outperformed than ABC algorithm, the convergence performance of IHABC was the best as a whole for multi-modal and composition functions, especially on 10D and 30D. ABC algorithm outperformed than DE and PSO algorithms DE is better than PSO algorithm. Functions F1, F5, and F11 reached the best convergence performance with function errors of mean value of zero on 10D, 30D, and 50D.

# Chapter 5

# Levy Flight-based Hybrid ABC (LFHABC) and Self Adaptive Hybrid Enhanced ABC (SAHEABC) Algorithms

## 5.1 Background

ABC algorithm has the characteristics which make it more attractive in SI algorithm: it has few control parameters of population size, limit and $MCN$ number; it has the advantages of simple, flexible and robust; it has fast convergence speed and it is easy to hybrid with other SI algorithms. In order to overcome the drawbacks such as low convergence speeds when solving some unimodal and composition functions, low exploitation abilities, trapped in local optima easily when they solve complex multimodal functions. The extended ABC algorithm of IHABC was proposed in Chapter 4 and the comparative experiment's results showed that IHABC achieved the competitive performance of convergence speed and the ability of controlling the balance of exploration and exploitation. However, a few shortcomings still exist, for example, random uniform initialization loses the effectiveness for higher dimensional size although plays a more important role in higher dimensional problems (up to 50 dimensions), and lower fitness individuals have low probability to be selected as onlooker bees according to the study by Richards and Ventura (2004).

To overcome these shortcomings, a levy flight-based hybrid ABC (LFHABC) algorithm

is proposed by utilizing levy flight (Brown et al., 2007; Pavlyukevich, 2007) for initialization, chaotic opposition-based learning (OBL) for scout bee step (Tizhoosh, 2005) and then combing them with IHABC algorithm in order to increase the population diversity with extending the search space information and accelerate convergence speed by using chaotic OBL. Based on LFHABC algorithm, a self adaptive hybrid enhanced ABC algorithm (SAHEABC) is proposed by modifying the probability on onlooker bees in order to increase exploitation ability and convergence ability by selecting probability for the lower fitness individuals.

Comparative experiments are implemented for LFHABC and SAHEABC algorithms to demonstrate the effectiveness of the algorithms; more specifically, CEC'13 test suite benchmark problems are adopted by LFHABC algorithm and CEC'14 test suite benchmark problems are utilized by SAHEABC algorithm and then the performance of the proposed extended ABC algorithms are statistically analyzed. In addition, the standard ABC, IABC, BsfABC algorithms are used to compare with SAHEABC algorithm, and the comparative performance of these algorithms will be evaluated based on the CEC'14 test suite experiments. Finally, additional extended comparison experiments are conducted using NRGA and SHADE algorithms with above mentioned algorithms.

## 5.2 Introduction of LFHABC and SAHEABC Algorithms

In LFHABC algorithm, modifications of levy flight initialization, self adaptive mechanism for employed bees and onlooker bees, and chaotic opposition based learning (OBL) for scout bee are introduced for improving the convergence performance of standard ABC algorithm based on the benchmark optimization problems.

Population initialization is a crucial step in SI algorithms because it can affect convergence speed and the quality of the final solution. If information about the solution is unavailable, then random initialization is the most commonly used method for generating an initial population. ABC algorithm produces the candidate solution from its parent by a simple operation based on taking the difference of randomly determined parts of the parent and a randomly chosen solution from the population, so it does not get better convergence for initialization step. In order to increase the population diversity with extending

the search space information, a levy flight distribution is introduced. In the past, the flight behavior of animals and insects that exhibit important properties of levy flight have been analyzed in various studies. This levy flight behavior has been applied to optimization and search algorithms, and reported results show its importance in the field of solution search algorithms (Yang and Deb, 2009, 2013). Recently, Yang proposed new meta-heuristic algorithms, such as CS using levy flight. Levy flight is a random walk in which the step lengths have a heavy-tailed probability distribution. Random step lengths drawn from a levy flight distribution (Yang, 2010a; Viswanathan et al., 1996) are shown as Eq. (5.1).

$$L(s) \sim |s|^{-1-\beta} \tag{5.1}$$

Where $\beta$ $(0 < \beta \leq 2)$ is an index and $s$ is the step length.

Initialization for LFHABC using levy flight is calculated as shown in Eq. (5.2).

$$x_{ij}^{t+1} = x_{i,j}^{t} + \alpha * levy(\beta) \tag{5.2}$$

Where $i \in 1, 2, ..., SN$ and $j \in 1, 2, ..., D$ are randomly selected indexes, $t$ is the iteration number and set for 50, $\alpha$ is uniformly distributed number selected from $U[0, 1]$.

The levy flight is produced using Eq. (5.3).

$$Levy(\beta) \sim 0.01 \left(\frac{u}{|v|}\right)^{-\frac{1}{\beta}} (x_{i,j}^{t} - x_{best,j}^{t}) \tag{5.3}$$

Where $i \in 1, 2, ..., SN$ and $j \in 1, 2, ..., D$ are randomly selected indexes, $x_{best,j}$ is best solution found so far, $u$ and $v$ are derived from normal distributions shown as Eqs. (5.4) and (5.5).

$$u \sim N(0, \sigma_u^2) \quad v \sim N(0, \sigma_v^2) \tag{5.4}$$

Where

$$\sigma_u = \left(\frac{\Gamma(1+\beta) sin(\pi\beta/2)}{\beta\Gamma[(1+\beta)/2] 2^{(\beta-1)/2}}\right)^{1/\beta}, \quad \sigma_v = 1 \tag{5.5}$$

The concept of opposition based learning (OBL) was introduced by Tizhoosh (Tizhoosh, 2005), and has been applied to accelerate reinforcement learning and back-propagation

learning in neural networks (Viswanathan et al., 2006). The main idea behind OBL is to improve the chance to start with a closer (fitter) solution by checking the opposite solution simultaneously. By doing this, the closer one to solution which says as guess or opposite guess can be chosen as initial solution. In fact, according to probability theory, the guess is farther to solution than opposite guess in 50% of cases; for these cases staring with opposite guess can accelerate convergence speed. According to Rahnamayan et al. (2008), OBL was introduced to DE and improved the convergence performance. Therefore, to accelerate convergence speed, chaotic OBL initialization approach is introduced for scout bees. Here, a sinusoidal iterator is selected, and its equation is defined as Eq. (5.6).

$$ch_{kj} = sin(\pi ch_{k-1,j}) \tag{5.6}$$

Where $ch_k \in [0, 1]$ , $k \in 1, 2, ..., Max$, $j \in 1, 2, ..., D$

The initialization population for scout bees is produced by Eq. (5.7).

$$x_{ij} = x_{min,j} + ch_{kj}(x_{max,j} - x_{min,j}) \tag{5.7}$$

and the chaotic OBL initialization for scout is shown as Eq. (5.8).

$$ox_{ij} = x_{min,j} + x_{max,j} - x_{ij} \tag{5.8}$$

Where $ox$ indicates the opposition-based population.

$SN$ individuals are selected from the set $\{X(SN) \cup OX(SN)\}$ as the initial scout bees population, $X$ is the population composed of $x_{ij}$ and $OX$ is the population composed of $ox_{ij}$.

In order to evaluate the performance of LF and chaotic OBL initialization, the random initialization and LF with chaotic OBL initialization have been tested on the CEC'13 test functions in terms of the population diversity. Population diversity is a measurement of the cover degree ( Kuang et al., 2014), which is defined as follows:

$$Diversity = \frac{1}{SN}\sum_{i=1}^{SN}\sqrt{\frac{1}{D}\sum_{j=1}^{D}(x_{i,j} - \bar{x_j})^2} \tag{5.9}$$

where, $SN$ denotes the number of food sources, which is equal to the number of employed bees or onlooker bees. $D$ is the dimension of the problem, and $\bar{x}$ is the center position of the colony.

The main modifications for LFHABC algorithm based on the IHABC algorithm are described as follows.

- The modification with Eqs. (5.2)-(5.5) substitute the Eq. (3.1) on step 4 of main steps of standard ABC algorithm in Section 3.2 to increase the population diversity with extending the search space information
- The amendments with Eqs. (5.6) to (5.8) exchange the Eq. (3.1) on step 11 of main steps of standard ABC algorithm in Section 3.2 to accelerate convergence speed by using chaotic OBL initialization for scout bee.

## 5.3 Experimental Setup and Results for LFHABC

Comparative experiments are conducted on both the standard ABC and LFHABC algorithms for 28 test functions defined by CEC'13 test suite with selected parameters of *limit* for 150 and $NP$ for 100 as same as used in IHABC algorithm. However, The maximum evaluation sizes are set to 100,000, 300,000, and 500,000 on 10D, 30D and 50D in the experiments, respectively. Each function of 28 test functions is executed 51 times with respect to 10D, 30D and 50D.The food source number is set to half of the $NP$ for 50, in the meantime, the employed bee number and onlooker bee number is set to same as food source number. The number of scout bee is set to 1 for each cycle.

When conducting the comparison experiments for ABC and LFHABC algorithms, the diversities for both ABC and LFHABC algorithms are calculated using Eq. (5.9) and the results are shown as Table 5.1 on 10D, 30D and 50 respectively. From this table, it is concluded that the diversity of LFHABC is better than ABC in 10D, 30D and 50D, the greater diversity provides more higher opportunity of finding more food sources and accelerates the convergence speed. For testifying the effectiveness of LFHABC algorithm, additional comparison experiments of LFHABC, standard ABC, DE, and PSO algorithms are implemented. The statistical analyses are conducted with Wilcoxon rank sum test by significance

level of 0.05.

Tables 5.2-5.4 illustrate the function errors of mean value of LFHABC, ABC, DE, and PSO algorithms for 100,000, 300,000, and 500,000 evaluations of *MCN* on 10D, 30D and 50D, respectively. The symbols of ”+”, ”≈” and ”-” indicate better, similar or worse performance of LFHABC algorithm compared to ABC, DE and PSO algorithms. After implementing comparative experiments on LFHABC, ABC, DE, and PSO algorithms, the numbers of better, similar, and worse performance of mean values of these algorithms are listed in Table 5.5 for 28 test functions.

When the dimension size increases, the number of functions in LFHABC algorithm with better and similar performance decreased compared to the other algorithms for the function errors of mean value according to Table 5.5. From the experimental results with Tables 4.6 and 5.5, it is concluded that the number of the functions with better and similar performance for LFHABC algorithm is more than IHABC algorithm on 10D, 30D and 50D. Especially, when it comes to compare with PSO algorithm, the performance for both IHABC and LFHABC are so significantly competitive.

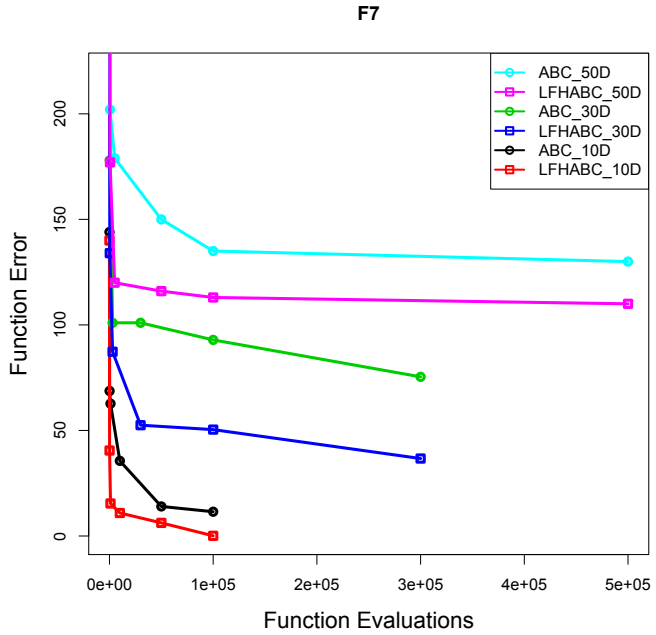



Figure 5.1: Comparative convergence for F7 on 10D, 30D and 50D

Table 5.1: Diversities for ABC and LFHABC on 10D, 30D and 50D

| F. / Div. | ABC(10D) | LFABC(10D) | ABC(30D) | LFABC(30D) | ABC(50D) | LFABC(50D) |
|---|---|---|---|---|---|---|
| F1 | 1.39 | 1.35 | 0.45 | 0.48 | 0.40 | 0.12 |
| F2 | 4.48 | 5.27 | 0.89 | 3.50 | 1.32 | 1.55 |
| F3 | 1.71 | 1.83 | 0.84 | 2.46 | 1.08 | 1.13 |
| F4 | 3.30 | 4.90 | 1.37 | 3.96 | 1.69 | 0.33 |
| F5 | 0.61 | 0.58 | 0.22 | 0.62 | 0.23 | 0.09 |
| F6 | 1.41 | 2.57 | 0.86 | 1.83 | 2.36 | 0.74 |
| F7 | 4.13 | 5.66 | 0.93 | 3.38 | 0.97 | 1.78 |
| F8 | 3.84 | 5.43 | 1.65 | 4.21 | 1.50 | 2.66 |
| F9 | 3.11 | 4.56 | 1.06 | 5.90 | 1.42 | 2.86 |
| F10 | 3.16 | 3.62 | 0.42 | 0.40 | 0.36 | 0.46 |
| F11 | 1.71 | 1.28 | 0.57 | 0.74 | 0.48 | 0.36 |
| F12 | 2.51 | 2.60 | 0.73 | 0.97 | 1.08 | 1.18 |
| F13 | 3.10 | 2.85 | 0.80 | 0.99 | 0.99 | 1.22 |
| F14 | 1.32 | 1.95 | 0.51 | 0.82 | 1.12 | 1.17 |
| F15 | 3.57 | 5.33 | 1.72 | 2.12 | 1.57 | 2.45 |
| F16 | 2.87 | 4.95 | 1.42 | 1.33 | 1.44 | 2.98 |
| F17 | 0.58 | 0.66 | 0.33 | 0.29 | 0.25 | 0.47 |
| F18 | 1.17 | 1.18 | 0.86 | 0.71 | 0.89 | 1.26 |
| F19 | 3.16 | 4.02 | 0.80 | 0.45 | 0.80 | 0.58 |
| F20 | 4.28 | 6.17 | 1.23 | 2.69 | 2.03 | 3.18 |
| F21 | 2.92 | 3.14 | 2.80 | 1.42 | 0.83 | 1.35 |
| F22 | 3.76 | 3.99 | 0.45 | 0.40 | 4.01 | 1.61 |
| F23 | 3.29 | 4.49 | 1.52 | 2.45 | 1.85 | 3.16 |
| F24 | 4.37 | 5.18 | 1.03 | 1.72 | 2.55 | 3.31 |
| F25 | 3.09 | 5.52 | 1.68 | 2.43 | 1.84 | 2.50 |
| F26 | 4.27 | 4.68 | 0.84 | 1.38 | 1.75 | 1.99 |
| F27 | 2.40 | 4.77 | 1.14 | 1.93 | 2.15 | 2.77 |
| F28 | 2.47 | 2.62 | 1.37 | 1.35 | 0.88 | 0.91 |

Table 5.2: Function errors of mean value for LFHABC, ABC, DE and PSO on 10D

| F./Eva. | LFHABC | ABC | DE | PSO |
|---|---|---|---|---|
| F1 | 0.00e+00 | 0.00e+00(=) | 0.00e+00(=) | 0.00e+00(=) |
| F2 | 2.20e+04 | 4.91e+05(+) | 2.42e+03(-) | 2.83e+05(+) |
| F3 | 7.27e+03 | 1.42e+05(+) | 1.41e+00(-) | 6.19e+05(+) |
| F4 | 3.50e+03 | 3.57e+03(≈) | 2.71e+01(-) | 6.76e+03(+) |
| F5 | 0.00e+00 | 0.00e+00(=) | 0.00e+00(=) | 0.00e+00(=) |
| F6 | 1.62e-03 | 3.77e-02(≈) | 3.29e+00(+) | 4.03e+00(+) |
| F7 | 3.72e+00 | 1.15e+01(+) | 1.44e-03(-) | 1.01e+01(+) |
| F8 | 2.02e+01 | 2.02e+01(=) | 2.04e+01(≈) | 2.03e+01(≈) |
| F9 | 2.61e+00 | 2.68e+00(≈) | 1.14e+00(-) | 3.47e+00(+) |
| F10 | 4.96e-01 | 4.68e-01(≈) | 4.92e-02(-) | 4.58e-01(≈) |
| F11 | 0.00e+00 | 0.00e+00(=) | 1.14e+00(+) | 4.83e+00(+) |
| F12 | 5.67e+00 | 9.23e+00(+) | 8.24e+00(+) | 1.09e+01(+) |
| F13 | 5.08e+00 | 1.03e+01(+) | 1.21e+01(+) | 1.83e+01(+) |
| F14 | 0.00e+00 | 0.00e+00(=) | 2.20e+02(+) | 2.91e+02(+) |
| F15 | 3.07e+02 | 2.71e+02(-) | 1.13e+03(+) | 5.93e+02(+) |
| F16 | 5.10e-01 | 3.29e-01(≈) | 1.01e+00(+) | 6.18e-01(≈) |
| F17 | 2.02e+00 | 2.13e+00(≈) | 1.78e+01(+) | 1.55e+01(+) |
| F18 | 2.23e+01 | 2.25e+01(≈) | 3.16e+01(+) | 2.64e+01(+) |
| F19 | 1.00e-06 | 1.00e-06(=) | 1.07e+00(+) | 6.56e-01(+) |
| F20 | 1.92e+00 | 2.13e+00(+) | 2.36e+00(+) | 2.73e+00(+) |
| F21 | 8.00e+01 | 1.43e+01(-) | 3.73e+02(+) | 2.59e+02(+) |
| F22 | 0.00e+00 | 1.73e-03(≈) | 2.23e+02(+) | 2.65e+02(+) |
| F23 | 3.50e+02 | 4.53e+02(+) | 9.77e+02(+) | 6.99e+02(+) |
| F24 | 1.09e+02 | 1.12e+02(≈) | 2.02e+02(+) | 2.00e+02(+) |
| F25 | 1.18e+02 | 1.19e+02(≈) | 2.02e+02(+) | 2.03e+02(+) |
| F26 | 1.07e+02 | 9.39e+01(-) | 1.67e+02(+) | 1.18e+02(+) |
| F27 | 3.20e+02 | 3.36e+02(+) | 3.37e+02(+) | 4.09e+02(+) |
| F28 | 4.80e+01 | 5.49e+01(+) | 2.92e+02(+) | 2.61e+02(+) |

Table 5.3: Function errors of mean value for LFHABC, ABC, DE and PSO on 30D

| F./Eva. | LFHABC | ABC | DE | PSO |
|---|---|---|---|---|
| F1 | 0.00e+00 | 0.00e+00(=) | 0.00e+00(=) | 0.00e+00(=) |
| F2 | 4.83e+06 | 4.89e+06(≈) | 1.54e+05(-) | 5.02e+06(+) |
| F3 | 1.73e+07 | 5.07e+07(+) | 3.65e+06(-) | 2.85e+08(+) |
| F4 | 4.36e+04 | 4.73e+04(≈) | 4.62e+02(-) | 1.92e+04(-) |
| F5 | 0.00e+00 | 0.00e+00(=) | 3.09e-05(≈) | 0.00e+00(=) |
| F6 | 1.11e-02 | 3.29e+00(+) | 1.98e+01(+) | 2.38e+01(+) |
| F7 | 5.94e+01 | 7.54e+01(+) | 1.58e+00(-) | 6.45e+01(+) |
| F8 | 2.08e+01 | 2.08e+01(=) | 2.09e+01(≈) | 2.09e+01(≈) |
| F9 | 2.41e+01 | 2.45e+01(≈) | 9.17e+00(-) | 2.70e+01(+) |
| F10 | 2.18e-01 | 1.10e-01(≈) | 7.62e-02(≈) | 1.55e+00(+) |
| F11 | 0.00e+00 | 0.00e+00(=) | 1.42e+01(+) | 5.65e+01(+) |
| F12 | 1.17e+02 | 1.65e+02(+) | 1.14e+02(≈) | 9.05e+01(-) |
| F13 | 1.55e+02 | 2.15e+02(+) | 1.53e+02(≈) | 1.48e+02(≈) |
| F14 | 0.00e+00 | 1.65e-01(+) | 5.72e+02(+) | 1.95e+03(+) |
| F15 | 3.48e+03 | 2.91e+03(-) | 7.01e+03(+) | 4.00e+03(+) |
| F16 | 1.33e+00 | 9.26e-01(-) | 2.45e+00(+) | 1.70e+00(+) |
| F17 | 3.04e+01 | 2.92e+01(≈) | 5.62e+01(+) | 1.01e+02(+) |
| F18 | 1.98e+02 | 2.32e+02(+) | 1.99e+02(≈) | 1.88e+02(≈) |
| F19 | 9.53e-02 | 1.36e-01(≈) | 3.93e+00(+) | 6.29e+00(+) |
| F20 | 1.23e+01 | 1.25e+01(≈) | 1.19e+01(≈) | 1.18e+01(≈) |
| F21 | 2.10e+02 | 1.35e+02(-) | 3.07e+02(+) | 2.10e+02(=) |
| F22 | 4.00e+01 | 1.28e+01(-) | 4.44e+02(+) | 2.19e+03(+) |
| F23 | 4.15e+03 | 3.61e+03(-) | 7.11e+03(+) | 4.53e+03(+) |
| F24 | 2.45e+02 | 2.71e+02(+) | 2.17e+02(-) | 2.74e+02(+) |
| F25 | 2.71e+02 | 2.89e+02(+) | 2.48e+02(-) | 2.87e+02(+) |
| F26 | 2.00e+02 | 2.00e+02(=) | 2.37e+02(+) | 2.29e+02(+) |
| F27 | 4.00e+02 | 4.00e+02(=) | 4.95e+02(+) | 1.02e+03(+) |
| F28 | 1.08e+02 | 1.23e+02(+) | 3.00e+02(+) | 2.96e+02(+) |

Table 5.4: Function errors of mean value for LFHABC, ABC, DE and PSO on 50D

| F./Eva. | LFHABC | ABC | DE | PSO |
|---|---|---|---|---|
| F1 | 0.00e+00 | 0.00e+00(=) | 0.00e+00(=) | 0.00e+00(=) |
| F2 | 1.51e+07 | 9.86e+06(-) | 4.91e+05(-) | 1.13e+07(-) |
| F3 | 2.99e+07 | 3.53e+07(+) | 1.97e+07(-) | 3.25e+09(+) |
| F4 | 9.85e+04 | 1.04e+05(+) | 1.83e+03(-) | 2.96e+04(-) |
| F5 | 0.00e+00 | 0.00e+00(=) | 1.34e-04(≈) | 0.00e+00(=) |
| F6 | 1.17e-01 | 2.98e+01(+) | 4.46e+01(+) | 4.65e+01(+) |
| F7 | 1.10e+02 | 1.30e+02(+) | 1.24e+01(-) | 1.17e+02(≈) |
| F8 | 2.10e+01 | 2.10e+01(=) | 2.11e+01(≈) | 2.11e+01(≈) |
| F9 | 5.11e+01 | 5.24e+01(≈) | 2.94e+01(-) | 5.57e+01(+) |
| F10 | 2.26e-01 | 1.16e-01(≈) | 1.37e-01(≈) | 1.01e+01(+) |
| F11 | 0.00e+00 | 0.00e+00(=) | 3.64e+02(+) | 1.20e+02(+) |
| F12 | 3.60e+02 | 5.35e+02(+) | 2.94e+02(-) | 2.24e+02(-) |
| F13 | 3.26e+02 | 5.79e+02(+) | 3.50e+01(-) | 3.54e+02(+) |
| F14 | 0.00e+00 | 1.42e+00(+) | 1.25e+03(+) | 3.69e+03(+) |
| F15 | 8.29e+03 | 6.64e+03(-) | 1.38e+04(+) | 8.85e+03(+) |
| F16 | 1.90e+00 | 1.47e+00(-) | 3.25e+00(+) | 2.49e+00(+) |
| F17 | 5.08e+01 | 4.89e+01(≈) | 9.52e+01(+) | 2.32e+02(+) |
| F18 | 2.00e+02 | 6.03e+02(+) | 3.95e+02(+) | 4.03e+02(+) |
| F19 | 4.38e-01 | 5.03e-01(≈) | 5.89e+00(+) | 1.83e+01(+) |
| F20 | 2.27e+01 | 2.35e+01(≈) | 2.18e+01(≈) | 2.15e+01(≈) |
| F21 | 4.55e+02 | 2.06e+02(-) | 7.05e+02(+) | 3.13e+02(-) |
| F22 | 0.00e+00 | 1.17e+01(+) | 1.23e+03(+) | 4.54e+03(+) |
| F23 | 8.32e+03 | 7.88e+03(-) | 1.36e+04(+) | 9.70e+03(+) |
| F24 | 3.35e+02 | 3.53e+02(+) | 2.50e+02(-) | 3.48e+02(+) |
| F25 | 3.70e+02 | 3.91e+02(+) | 2.92e+02(-) | 3.74e+02(≈) |
| F26 | 2.00e+02 | 2.01e+02(≈) | 3.21e+02(+) | 3.18e+02(+) |
| F27 | 3.83e+02 | 4.01e+02(+) | 8.34e+02(+) | 1.75e+03 (+) |
| F28 | 4.00e+02 | 4.00e+02(=) | 5.16e+02(+) | 4.00e+02(=) |

Table 5.5: Comparison performance with function errors of mean value of LFHABC algorithm to ABC, DE and PSO algorithms

| LFHABC (10D) VS. | ABC | DE | PSO |
|---|---|---|---|
| + | 9 | 19 | 23 |
| ≈ | 16 | 3 | 5 |
| - | 3 | 6 | 0 |
| LFHABC (30D) VS. | ABC | DE | PSO |
| + | 10 | 13 | 19 |
| ≈ | 13 | 8 | 7 |
| - | 5 | 7 | 2 |
| LFHABC (50D) VS. | ABC | DE | PSO |
| + | 12 | 14 | 17 |
| ≈ | 11 | 5 | 7 |
| - | 5 | 9 | 4 |

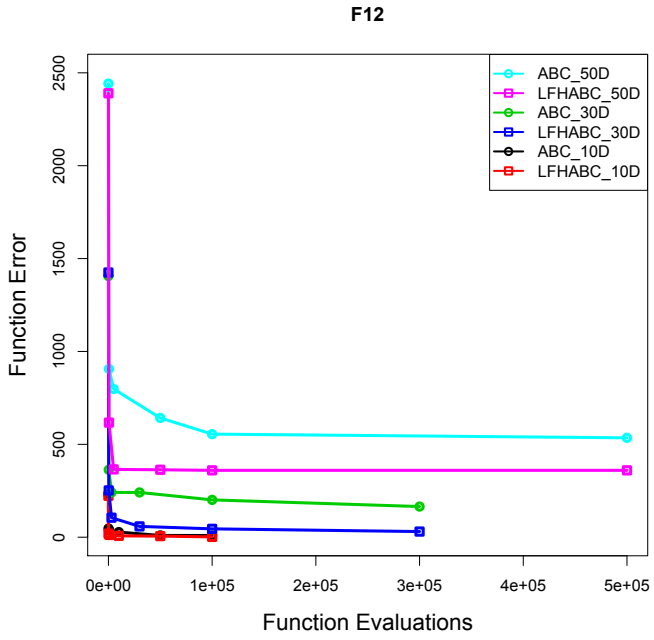



Figure 5.2: Comparative convergence for F12 on 10D, 30D and 50D

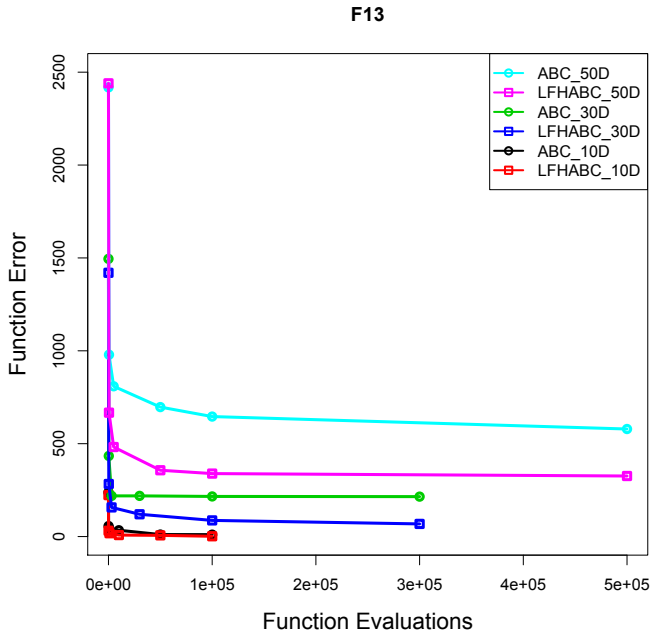



Figure 5.3: Comparative convergence for F13 on 10D, 30D and 50D

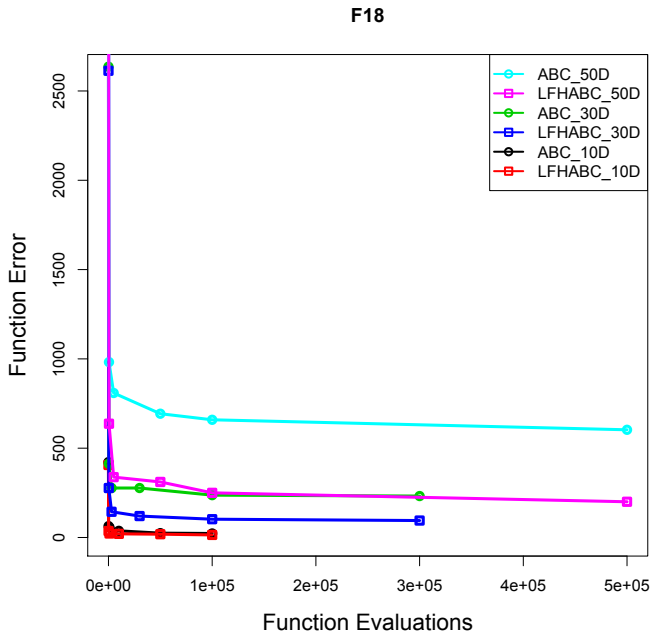



Figure 5.4: Comparative convergence for F18 on 10D, 30D and 50D

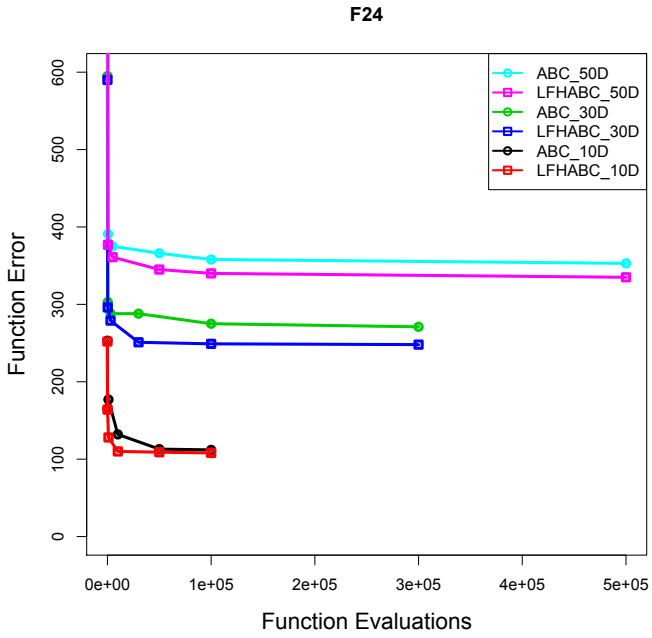



Figure 5.5: Comparative convergence for F24 on 10D, 30D and 50D

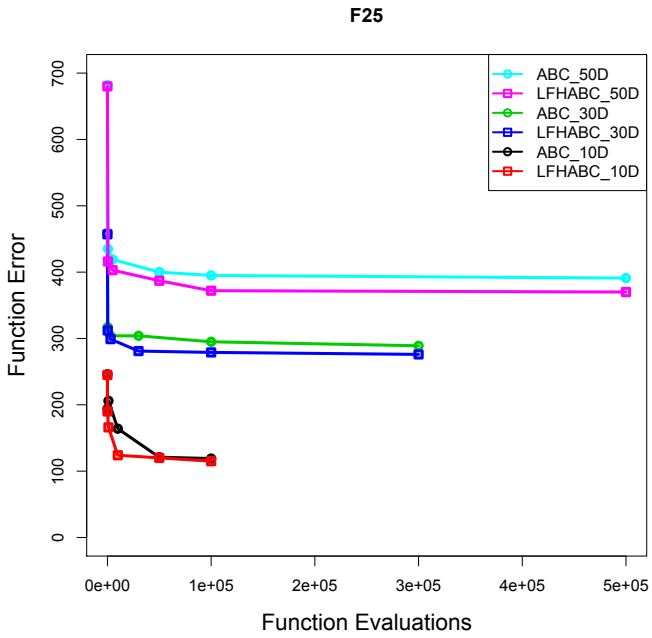



Figure 5.6: Comparative convergence for F25 on 10D, 30D and 50D

According to Figs. 5.1-5.4, the performance of LFHABC is much better than that of ABC algorithm. Moreover, functions F2, F3, F4, and F6 achieve very competitive performance by both LFHABC and ABC algorithms from Tables 5.2-5.4.

For Figs. 5.5-5.6, the performance of LFHABC is better than that of ABC algorithm, but is not obvious. Further, functions F14, F19, and F22 achieve the similar performance for both LFHABC and ABC algorithms according to Tables 5.2-5.4.

Very similar performance is achieved for both LFHABC and ABC algorithms as shown in Fig. 5.7, and functions F20, F26, and F27 achieve similar results according to Tables 5.2-5.4. With regard to function F8, convergence performance is much similar for both LFHABC and ABC algorithms as shown in Figure 5.8. Further, functions F1, F5, and F11 achieve the best performance by both LFHABC and ABC algorithms because all function errors of mean value reached zero.

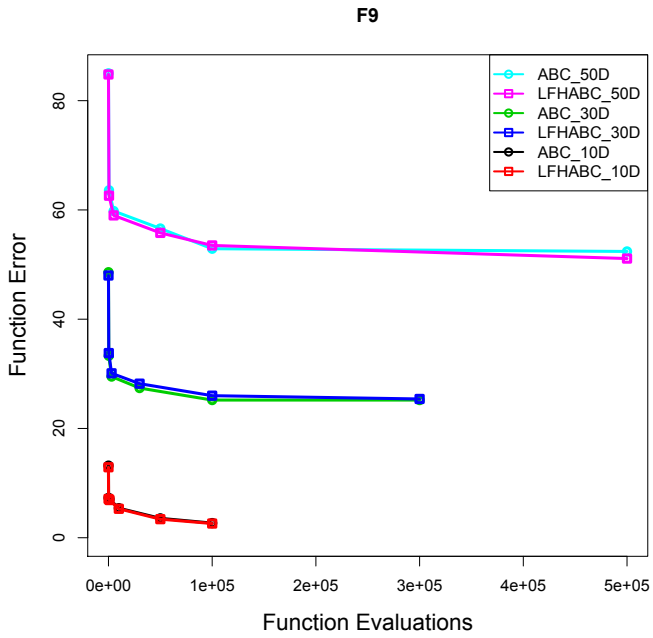



Figure 5.7: Comparative convergence for F9 on 10D, 30D and 50D

With regard to functions F10, F15, F16, F17, F21, F23, and F28, the performance of LFHABC is not better than that of the ABC algorithm on the same dimension sizes. More specifically, the performance of LFHABC is not better than that of ABC algorithm for

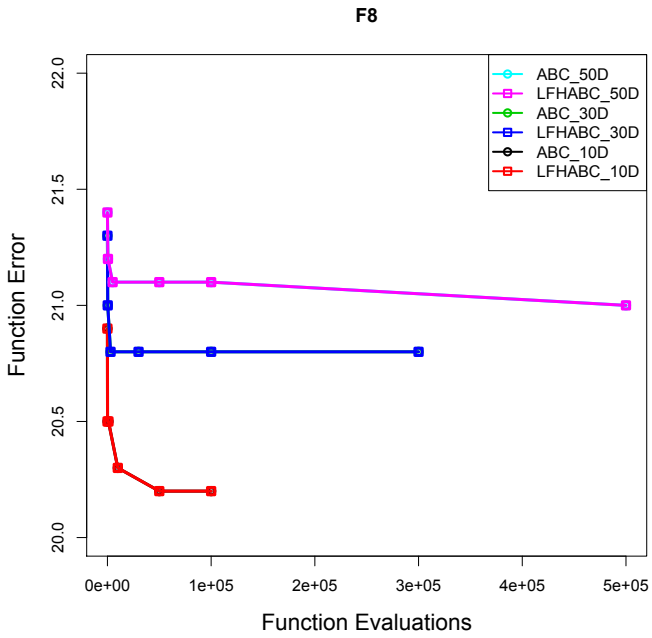



Figure 5.8: Comparative convergence for F8 on 10D, 30D and 50D

functions F15 and F21 on 10D, 30D, 50D, functions F10, F16, and F17 on 30D, 50D as well as function F28 on 10D and 30D.

According to the above Tables 5.2-5.5 and Figs. 5.1-5.8 obtained from the experimental results, LFHABC and the standard ABC algorithms were not very effective at solving unimodal functions except for function F1. For all evaluations, functions F1, F5, and F11 achieved the best performance because the function errors of mean value reached zero. Moreover, function F8 had the same performance for all evaluation stages. At the meantime, the performance of LFHABC was better than IHABC algorithm according to Table 4.6 and Table 5.5.

It was also observed that the performance of LFHABC was better than ABC algorithm on those functions with the properties of non-separability, having many local optima and second local optimum is far from the global optimum, having vary narrow valley from local optimum to global optimum, continuous everywhere yet differentiable nowhere and having different properties around the local optima.

## 5.4 Experimental Setup and Results for SAHEABC

In order to increase the probability of the lower fitness individuals to be selected through modifying $P_i$ used in Eq. (3.3) on step 4 of standard ABC algorithm in Section 3.2, self adaptive mechanism is used to modify the value of $P_i$. The probability value $P_i$ varies with the updated cycle number, it also depends on the summation value of fitness and maximum value of fitness. $P_i$ is given in Eq. (5.10).

$$P_i = \mathrm{e}^{\frac{-0.18cycle}{MCN}} \frac{fit_i}{fit_{max}} + (1 - \mathrm{e}^{\frac{-0.18cycle}{MCN}}) \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{5.10}$$

Where $MCN$ is the maximum cycle number, $fit_i$ is the fitness value of solution $i$, $fit_{max}$ is the maximum fitness value.

SAHEABC algorithm is proposed based on LFABC algorithm, in addition, modifying Eq. (3.3) on step 4 of standard ABC algorithm in Section 3.2 with Eq. (5.10).

The standard ABC, BsfABC, IABC and SAHEABC algorithms are evaluated for all 30 test functions defined by CEC'14 test suite with the selected parameters of *limit* for 150 and $NP$ for 100 as same as used in IHABC algorithm. The food source number is set to half of the $NP$ for 50, in the meantime, the employed bee number and onlooker bee number is set to same as food source number. The number of scout bee is set to 1 for each cycle. The 30 test functions are executed 25 times with respect to each test function at each problem dimension size. The algorithms are terminated when the $MCN$ is reached for function evaluations or when the error value gets smaller than $10^{-8}$. In the experiments, $MCN$ is set for 10,000, 30,000, and 50,000 on 10D, 30D and 50D, respectively. The comparative experiments for the standard ABC, BsfABC, IABC and SAHEABC algorithms are implemented and also statistical analyses are made with Wilcoxon rank sum test by significance level of 0.05.

Tables 5.7-5.9 illustrate the function errors of mean value for ABC, BsfABC, IABC, and SAHEABC algorithms for 10,000, 30,000, and 50,000 evaluations on 10D, 30D and 50D, respectively. The symbols of "+", "≈ " and "-" indicate better, similar or worse performance of SAHEABC algorithm compared to ABC, BsfABC and IABC algorithms.

After implementing comparative experiments on the standard ABC, BsfABC, IABC

Table 5.6: Function errors of mean value for the SAHEABC, ABC, BsfABC and IABC and the signs for SAHEABC compared with ABC, BsfABC and IABC on 10D

| F. | SAHEABC | ABC Sign | BsfABC Sign | IABC Sign |
|---|---|---|---|---|
| F1 | 4.35e+04 | 6.68e+04 ≈ | 5.87e+04 + | 4.58e+04 ≈ |
| F2 | 2.66e+01 | 1.21e+01 - | 4.56e+00 - | 2.17e+01 ≈ |
| F3 | 6.75e+01 | 3.34e+01 - | 2.73e+02 + | 1.11e+02 + |
| F4 | 4.41e-03 | 1.00e-02 + | 8.04e-03 + | 6.53e-03 + |
| F5 | 1.35e+01 | 8.44e+00 ≈ | 1.83e+01 ≈ | 1.22e+01 ≈ |
| F6 | 8.87e-01 | 1.05e+00 ≈ | 1.93e-01 + | 1.46e-01 + |
| F7 | 5.94e-04 | 1.42e-03 ≈ | 4.95e-04 ≈ | 1.31e-02 ≈ |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ |
| F9 | 3.12e+00 | 4.47e+00 + | 7.44e+00 + | 6.03e+00 + |
| F10 | 0.00e+00 | 3.01e-01 ≈ | 3.80e-02 + | 0.00e+00 ≈ |
| F11 | 5.84e+01 | 9.04e+01 + | 6.52e+01 + | 7.74e+01 + |
| F12 | 1.30e-01 | 1.02e-01 - | 1.09e-01 - | 1.104e-01 - |
| F13 | 8.21e-02 | 8.51e-02 ≈ | 7.21e-02 ≈ | 1.30e-01 + |
| F14 | 6.42e-02 | 1.11e-01 + | 1.06e-01 + | 1.20e-01 + |
| F15 | 5.19e-01 | 4.01e-01 ≈ | 7.25e-01 + | 6.17e-01 + |
| F16 | 1.38e+00 | 1.68e+00 + | 1.87e+00 + | 1.65e+00 + |
| F17 | 2.95e+04 | 6.33e+04 + | 4.25e+04 + | 3.92e+04 ≈ |
| F18 | 5.55e+01 | 1.07e+02 + | 8.53e+01 + | 1.03e+02 + |
| F19 | 2.84e-01 | 1.86e-01 ≈ | 2.34e-01 ≈ | 2.76e-01 ≈ |
| F20 | 2.58e+01 | 4.20e+01 + | 9.32e+01 + | 2.36e+01 ≈ |
| F21 | 9.21e+02 | 3.13e+03 + | 2.46e+03 + | 1.83e+03 + |
| F22 | 6.31e-02 | 1.31e-01 + | 1.46e+00 + | 4.76e-01 + |
| F23 | 1.50e+02 | 1.05e+02 ≈ | 1.05e+02 ≈ | 5.64e+00 - |
| F24 | 1.10e+02 | 1.10e+02 ≈ | 1.17e+02 + | 1.15e+02 + |
| F25 | 1.17e+02 | 1.25e+02 + | 1.34e+02 + | 1.27e+02 + |
| F26 | 9.50e+01 | 9.99e+01 + | 1.00e+02 + | 9.82e+01 + |
| F27 | 4.54e+00 | 6.24e+00 + | 1.21e+01 + | 5.73e+00 + |
| F28 | 3.45e+02 | 3.32e+02 - | 3.66e+02 + | 3.28e+02 - |
| F29 | 2.47e+02 | 2.47e+02 - | 2.31e+02 - | 2.49e+02 - |
| F30 | 4.68e+02 | 5.15e+02 + | 4.99e+02 + | 5.20e+02 + |

Table 5.7: Function errors of mean value for the SAHEABC, ABC, BsfABC and IABC and the signs for SAHEABC compared with ABC, BsfABC and IABC on 30D

| F. | SAHEABC | ABC Sign | BsfABC Sign | IABC Sign |
|---|---|---|---|---|
| F1 | 6.66e+06 | 2.42e+06 - | 7.14e+06 + | 3.01e+06 - |
| F2 | 7.59e+00 | 1.15e+01 + | 1.39e+01 + | 3.20e+01 + |
| F3 | 2.42e+02 | 9.40e+01 - | 2.18e+02 ≈ | 3.02e+02 ≈ |
| F4 | 2.08e-01 | 2.69e-01 + | 2.48e-01 + | 2.25e-01 + |
| F5 | 2.02e+01 | 2.02e+01 ≈ | 2.00e+01 ≈ | 2.02e+01 ≈ |
| F6 | 1.30e+01 | 1.18e+01 - | 1.50e+01 + | 1.26e+01 ≈ |
| F7 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ |
| F9 | 3.99e+01 | 6.320+01 + | 4.85e+01 + | 5.52e+01 + |
| F10 | 0.00e+00 | 1.53e-01 + | 3.21e-02 + | 7.21e-01 + |
| F11 | 1.70e+03 | 1.54e+04 + | 2.09e+03 + | 1.49e+03 - |
| F12 | 2.68e-01 | 1.89e-01 - | 1.80e-01 - | 1.95e-01 - |
| F13 | 2.051e-01 | 1.94e-01 ≈ | 1.81e-01 - | 2.79e-01 + |
| F14 | 1.53e-01 | 1.72e-01 + | 1.65e-01 + | 1.82e-01 + |
| F15 | 6.22e+00 | 5.83e+00 ≈ | 9.53e+00 + | 8.03e+00 + |
| F16 | 9.03e+00 | 9.32e+00 + | 1.00e+01 + | 9.61e+00 + |
| F17 | 1.17e+06 | 1.25e+06 + | 2.99e+06 + | 1.47e+06 + |
| F18 | 9.34e+03 | 5.21e+02 - | 2.76e+02 - | 4.44e+03 - |
| F19 | 5.88e+00 | 5.62e+00 ≈ | 6.50e+00 + | 6.27e+00 + |
| F20 | 2.17e+03 | 2.77e+03 + | 2.45e+03 + | 3.08e+03 + |
| F21 | 2.35e+05 | 8.50e+04 - | 3.59e+05 + | 1.082e+05 - |
| F22 | 1.59e+02 | 1.19e+02 ≈ | 3.28e+02 + | 1.20e+02 ≈ |
| F23 | 3.15e+02 | 3.15e+02 ≈ | 3.15e+02 ≈ | 2.97e+02 - |
| F24 | 1.79e+02 | 2.53e+02 + | 2.25e+02 + | 2.22e+02 + |
| F25 | 2.06+02 | 2.05e+02 ≈ | 1.95e+02 ≈ | 1.86e+02 ≈ |
| F26 | 1.00e+02 | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ |
| F27 | 4.07e+02 | 4.04e+02 ≈ | 4.07e+02 ≈ | 3.89e+02 ≈ |
| F28 | 8.09e+02 | 8.76e+02 + | 8.32e+02 + | 8.61e+02 + |
| F29 | 9.96e+02 | 8.70e+02 - | 7.73e+02 - | 9.14e+02 ≈ |
| F30 | 1.42e+03 | 1.58e+03 + | 2.43e+03 + | 2.06e+03 + |

Table 5.8: Function errors of mean value for the SAHEABC, ABC, BsfABC and IABC and the signs for SAHEABC compared with ABC, BsfABC and IABC on 50D

| F. | SAHEABC | ABC Sign | BsfABC Sign | IABC Sign |
|---|---|---|---|---|
| F1 | 1.27e+07 | 7.51e+06 - | 1.41e+07 ≈ | 8.53e+06 - |
| F2 | 9.40e+03 | 2.30e+02 - | 3.00e+02 - | 1.18e+02 - |
| F3 | 5.85e+03 | 3.55e+03 - | 6.40e+03 + | 6.63e+03 + |
| F4 | 5.52e+00 | 1.60e+01 + | 1.13e+01 + | 8.32e+00 + |
| F5 | 2.03e+01 | 2.03e+01 ≈ | 2.03e+01 ≈ | 2.03e+01 ≈ |
| F6 | 2.74e+01 | 2.81e+01 ≈ | 3.32e+01 + | 2.81e+01 ≈ |
| F7 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ |
| F9 | 1.12e+02 | 1.54e+02 + | 1.39e+02 + | 1.25e+02 + |
| F10 | 5.92e-03 | 5.20e-01 + | 1.82e+00 + | 3.15e-01 + |
| F11 | 3.34e+03 | 3.80e+03 + | 4.55e+03 + | 3.71e+03 + |
| F12 | 3.33e-01 | 2.34e-01 - | 2.25e-01 - | 2.49e-01 - |
| F13 | 2.61e-01 | 2.82e-01 ≈ | 2.70e-01 ≈ | 3.39e-01 + |
| F14 | 1.84e-01 | 2.03e-01 + | 2.11e-01 + | 2.17e-01 + |
| F15 | 1.55e+01 | 1.58e+01 ≈ | 2.47e+01 + | 2.13e+01 + |
| F16 | 1.79e+01 | 1.94e+01 + | 1.84e+01 ≈ | 1.89e+01 + |
| F17 | 3.68e+06 | 2.58e+06 ≈ | 5.82e+06 + | 2.74e+06 ≈ |
| F18 | 1.03e+04 | 1.71e+03 - | 5.36e+02 - | 1.08e+04 ≈ |
| F19 | 1.38e+01 | 1.94e+01 + | 1.64e+01 + | 1.45e+01 ≈ |
| F20 | 1.88e+04 | 1.29e+04 - | 2.88e+04 + | 1.26e+04 - |
| F21 | 1.92e+06 | 1.44e+06 - | 3.93e+06 + | 1.40e+06 - |
| F22 | 6.06e+02 | 5.01e+02 - | 8.42e+02 + | 5.30e+02 ≈ |
| F23 | 3.44e+02 | 3.44e+02 ≈ | 3.44e+02 ≈ | 3.44e+02 ≈ |
| F24 | 2.56e+02 | 2.59e+02 ≈ | 2.57e+02 ≈ | 2.58e+02 ≈ |
| F25 | 2.12e+02 | 2.12e+02 ≈ | 2.12e+02 ≈ | 2.12e+02 ≈ |
| F26 | 1.00e+02 | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ |
| F27 | 4.15e+02 | 4.21e+02 + | 4.42e+02 + | 4.19e+02 ≈ |
| F28 | 1.22e+02 | 1.47e+03 + | 1.28e+03 + | 1.38e+02 + |
| F29 | 1.82e+03 | 1.11e+03 - | 1.38e+03 - | 1.55e+03 - |
| F30 | 9.09e+03 | 9.28e+03 + | 1.02e+04 + | 9.23e+03 + |

and SAHEABC algorithms, the numbers of better, similar, and worse performance of function errors of mean value for these algorithms are listed in Table 5.10 for 30 test functions statistically with symbols “+”, “≈”, “-”.

Table 5.9: Comparison performance of function errors of mean value for SAHEABC to ABC, BsfABC, and IABC algorithms.

| SAHEABC (10D) VS. | ABC | BsfABC | IABC |
|---|---|---|---|
| + | 14 | 21 | 17 |
| ≈ | 11 | 6 | 9 |
| - | 5 | 3 | 4 |
| SAHEABC (30D) VS. | ABC | BsfABC | IABC |
| + | 12 | 18 | 14 |
| ≈ | 11 | 8 | 10 |
| - | 7 | 4 | 6 |
| SAHEABC (50D) VS. | ABC | BsfABC | IABC |
| + | 10 | 16 | 11 |
| ≈ | 11 | 10 | 13 |
| - | 9 | 4 | 6 |

Figures 5.9-5.16 illustrate the convergence performance for logarithmic values of mean value of function error on ABC, BsfABC, IABC, and SAHEABC algorithms with increasing function evaluations on 10D, 30D and 50D respectively.

According to Figs. 5.9-5.11, the performance of SAHEABC algorithm is better than ABC, BsfABC, and IABC algorithms on 10D. The convergence performance of SAHEABC on functions F11 as shown in Fig. 5.9 and F18 as shown in Fig. 5.10 are much better compared to ABC, BsfABC, and IABC algorithms on 10D. The functions F6 and F9 achieve similar performance according to Tables 5.7-5.9.The comparative convergence of function F30 as shown in Fig. 5.11 is better than ABC, BsfABC, and IABC algorithms on 10D, functions F4, F9, F10, F14,F16, F17, F20-F22, and F25-F27 are found to achieve similar performance according to Tables 5.7-5.9. The best result is achieved by standard and state-of-the-art ABC algorithms for function F8, moreover, the function F8 has the best performance because the function errors of mean value reached zero. For functions F7 and F10, SAHEABC algorithm almost achieves the best performance with the function error of

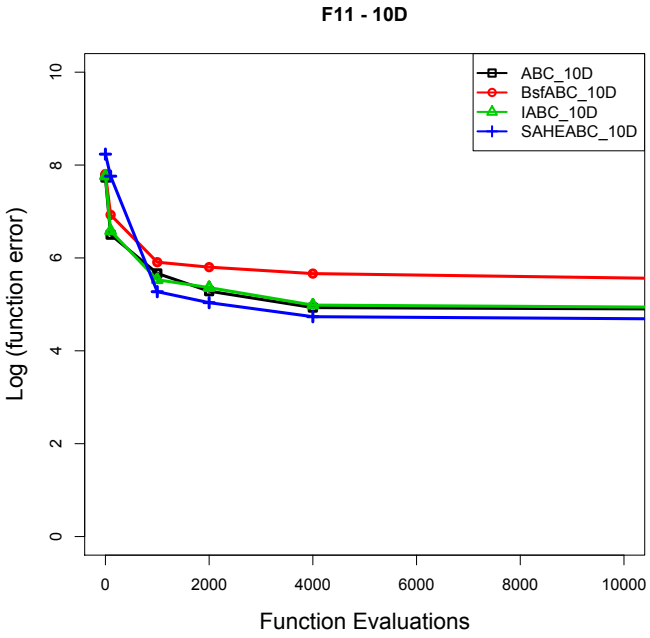



Figure 5.9: Comparative convergence for F11 on 10D

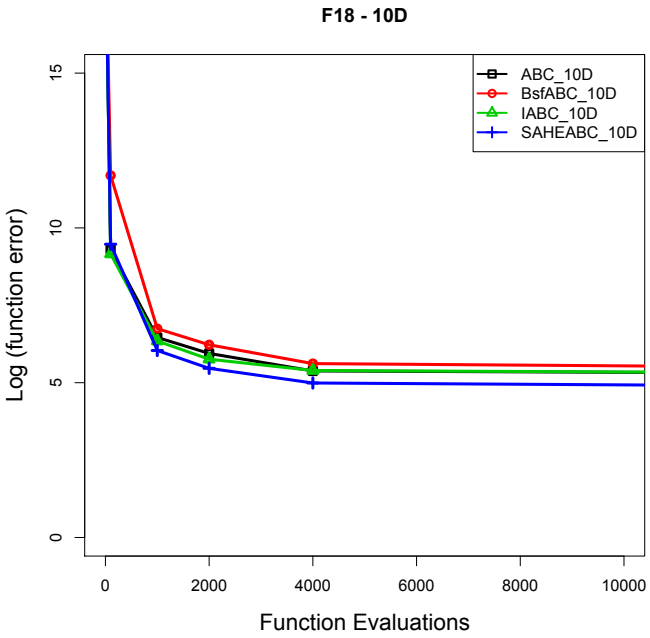



Figure 5.10: Comparative convergence for F18 on 10D

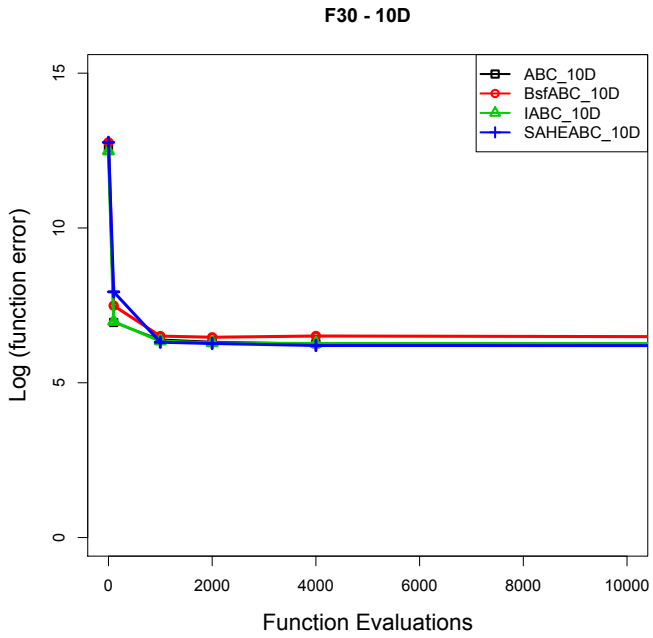



Figure 5.11: Comparative convergence for F30 on 10D

mean value of zero.

With regard to Figs. 5.12-5.13, the convergence performance of SAHEABC is better than ABC, BsfABC, and IABC algorithms on 30D. The convergence speed of functions F2 as shown in Fig. 5.12 and F9 as shown in Fig. 5.13 for SAHEABC is much faster compared to ABC, BsfABC, and IABC algorithms on 30D. The convergence ability of function F24 as shown in Fig. 5.14 for SAHEABC is significantly better than ABC, BsfABC, and IABC algorithms on 30D. The functions F4, F10-F12, F16, F17, F20, F28 and F30 achieve similar performance according to Tables 5.7-5.9. The best results are got by ABC, BsfABC, IABC, and SAHEABC algorithms on functions F7 and F8 which reached the best performance because the function errors of mean value reached zero. SAHEABC algorithm almost achieves the best performance with function error of mean value of zero for function F10. ABC, BsfABC, IABC, and SAHEABC algorithms have similar results for functions F23, F25-F27 on D30 according to Tables 5.7-5.9.

According to Figs. 5.15-5.16, the competitive convergence of SAHEABC is achieved compared to ABC, BsfABC, and IABC algorithms on 50D. The convergence speed for

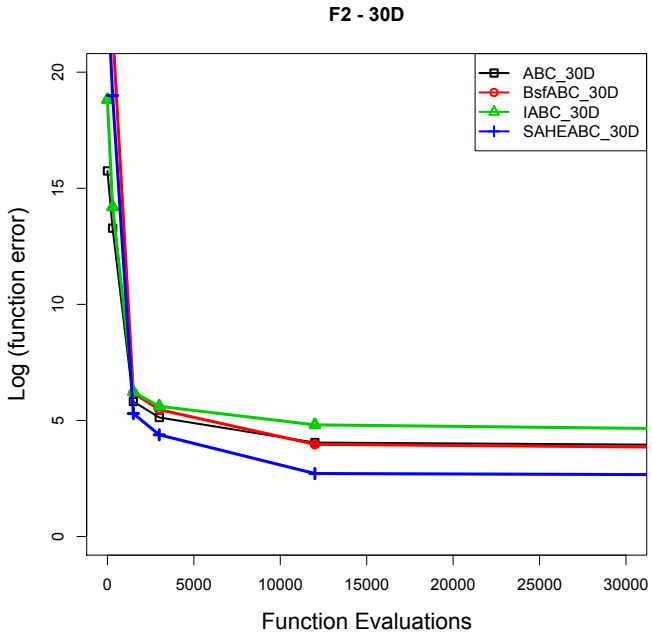



Figure 5.12: Comparative convergence for F2 on 30D

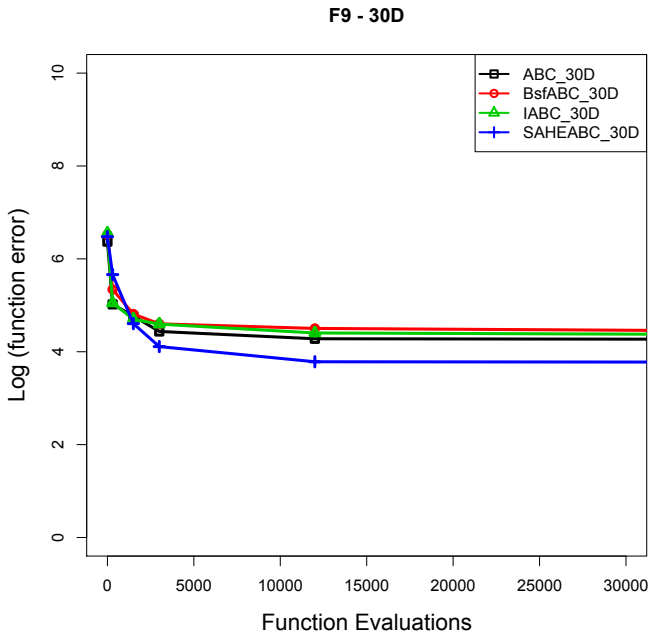



Figure 5.13: Comparative convergence for F9 on 30D

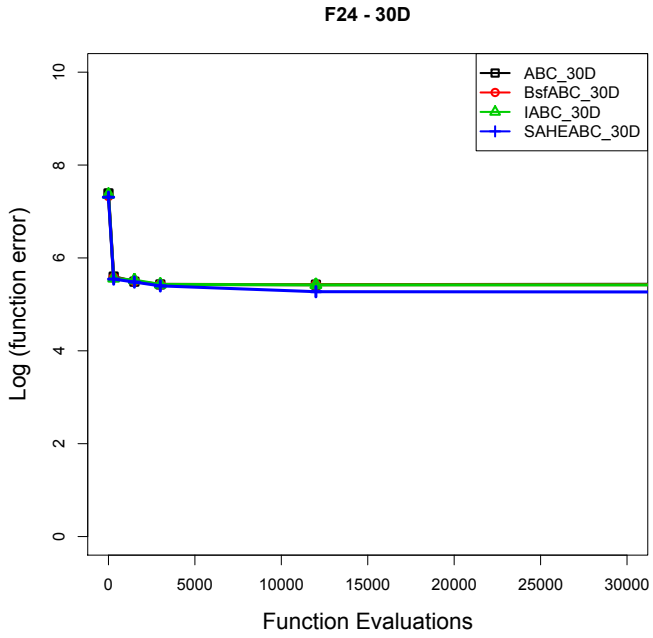



Figure 5.14: Comparative convergence for F24 on 30D

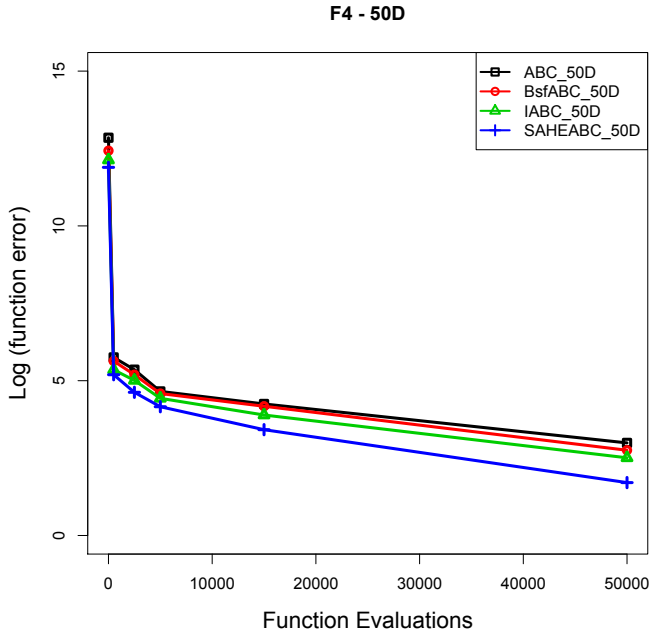



Figure 5.15: Comparative convergence for F4 on 50D

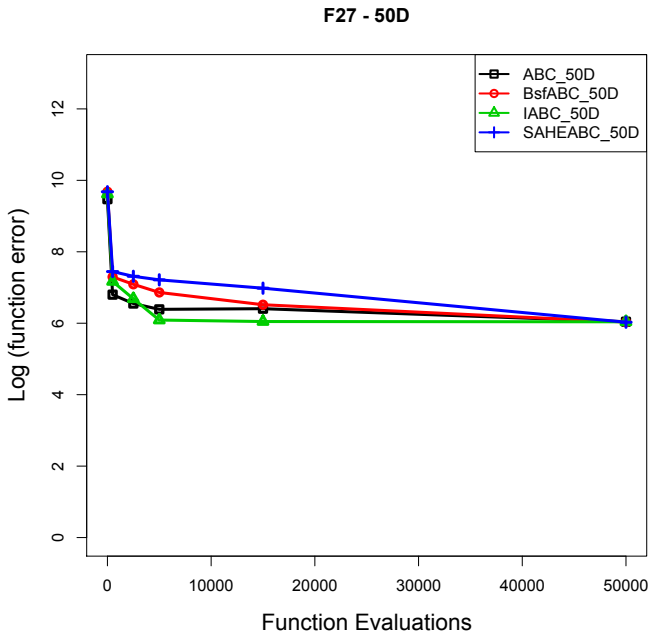



Figure 5.16: Comparative convergence for F27 on 50D

function F4 as shown in Fig. 5.15 is much better than ABC, BsfABC, and IABC algorithms on 50D, functions F9 and F10 have the same results as well as function F4; The convergence performance of function F27 as shown in Fig. 5.16 is better than ABC, BsfABC, and IABC algorithms on 50D, but not so obviously. The functions F9, F11, F14, F16, F19, F28, and F30 are observed to have the similar results according to the Tables 5.7-5.9. The best results are achieved by ABC, BsfABC, IABC, and SAHEABC algorithms on functions F7 and F8 which reached the best performance because the function errors of mean values reached zero. For function F10, SAHEABC algorithm almost achieves the best performance with the mean value of function error of zero. The all four compared algorithms have the similar performance for functions F5, F6, F13, F15, F17 and F23-26 on D50.

Figures 5.17-5.24 show the boxplots with function errors of mean value for ABC, BsfABC, IABC, and SAHEABC algorithms on 10D, 30D, and 50D respectively. The numbers of “1, 2, 3, 4” indicate that the function errors of mean value of ABC, BsfABC, IABC, and SAHEABC algorithms on 10D, in the same way, the numbers of “5, 6, 7, 8” , and “9,

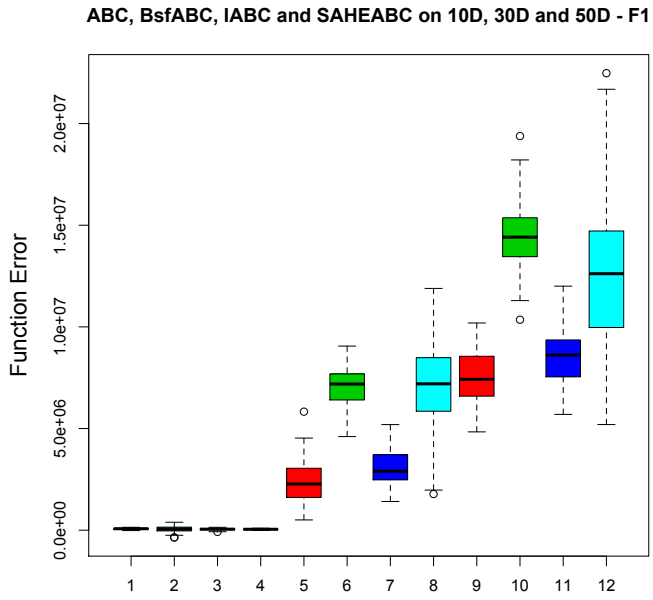



Figure 5.17: Boxplot of comparative convergence for F1

10, 11, 12" indicate that the function errors of mean value of ABC, BsfABC, IABC, and SAHEABC algorithms on 30D, and 50D, respectively. According to Fig. 5.17, the performance of SAHEABC is better than BsfABC algorithm for F1 on 10D and 30D, but its performance is not better than others except for BsfABC algorithm on 10D, 30D and 50D. For the remaining functions of unimodal functions, it is observed that SAHEABC achieves the best performance for function F2 on D30, function F3 on 10D and 50D compared to BsfABC and IABC algorithms. From the analyses of the unimodal functions F1-F3, it is concluded that the the performance of SAHEABC algorithm is effective on dimension size of 10.

Figures 5.18-5.20 illustrate the boxplots of simple multi-modal functions F4, F5, and F14 with mean values of function error of ABC, BsfABC, IABC, and SAHEABC algorithms on 10D, 30D, and 50D, respectively. According to Fig. 5.18, the performance of SAHEABC is the best, IABC is the second, BsfABC is better than ABC for function F4 on 10D, 30D, and 50D. It is observed that the performance of ABC is the best for function

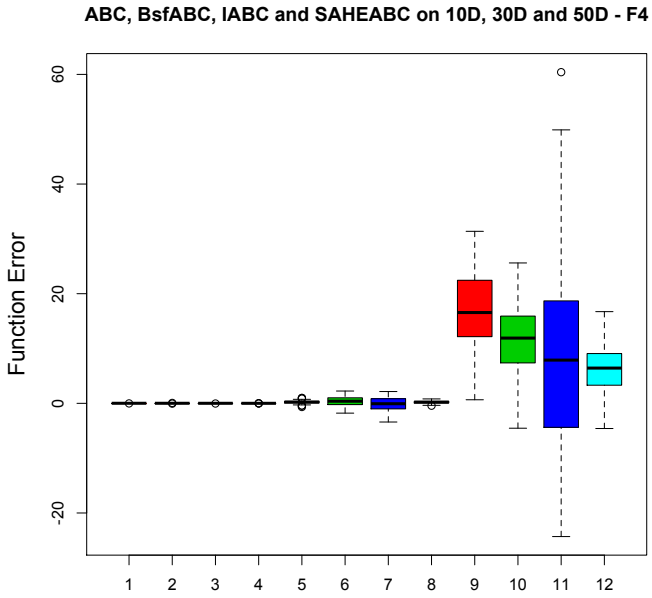



Figure 5.18: Boxplot of comparative convergence for F4

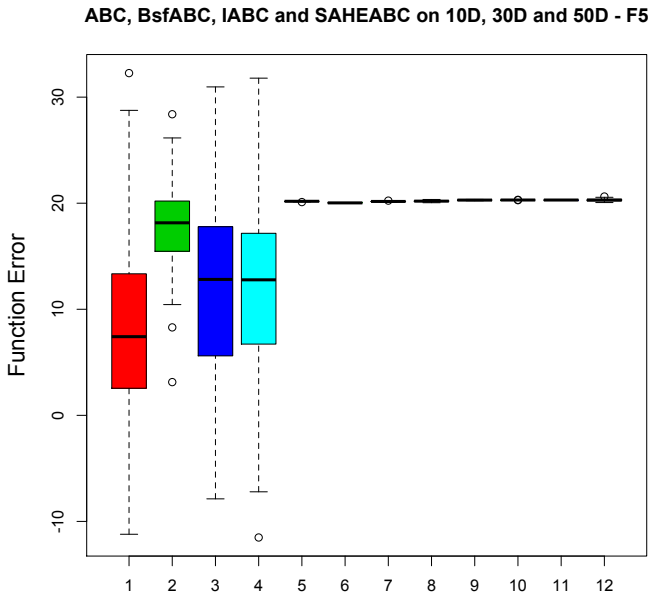



Figure 5.19: Boxplot of comparative convergence for F5

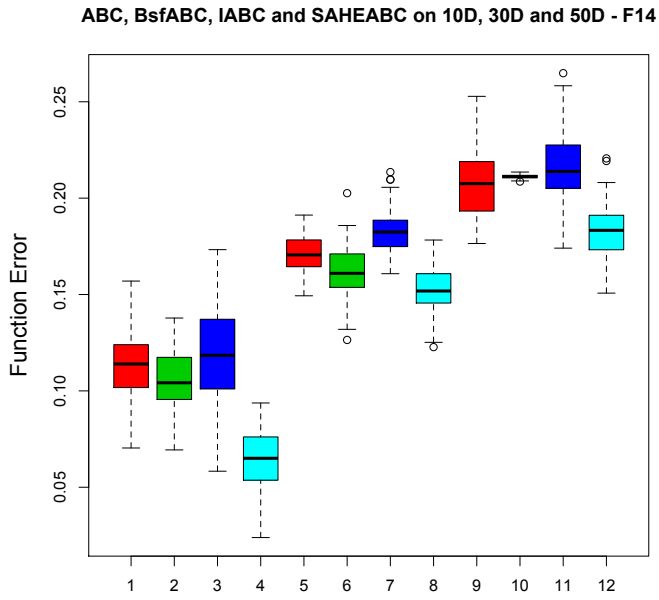



Figure 5.20: Boxplot of comparative convergence for F14

F5 on 10D, however, all four compared algorithms on function F5 achieve the same performance on 30D and 50D according to Fig. 5.19. With regard to Fig. 5.20, it could be seen that the performance of SAHEABC for function F14 is better than others on 10D, 30D and 50D; BsfABC is better than ABC algorithm on 10D and 30D; but ABC is not worse than BsfABC algorithm on 50D; IABC algorithm performs the worst on 10D, 30D, and 50D. For the remaining functions of simple multi-modal functions, the SAHEABC reaches the best performance for functions F4, F9 and F14 on 10D, 30D and 50D; functions F11and F16 on 10D; function F10 on 30D as well as functions F10 and F11 on 50D. Functions F7 and F8 achieve the best performance on 10D, 30D, and 50D, because the function errors of mean value reached zero. Based on the analyses of simple multi-modal functions, it could be concluded that SAHEABC is much competitive when it comes to compare with ABC, BsfABC, and IABC algorithms.

Figures 5.21-5.22 show the boxplots of hybrid functions for F19 and F20 with the function errors of mean value for ABC, BsfABC, IABC, and SAHEABC algorithms on 10D, 30D, and 50D, respectively. According to Fig. 5.21, the performance of SAHEABC

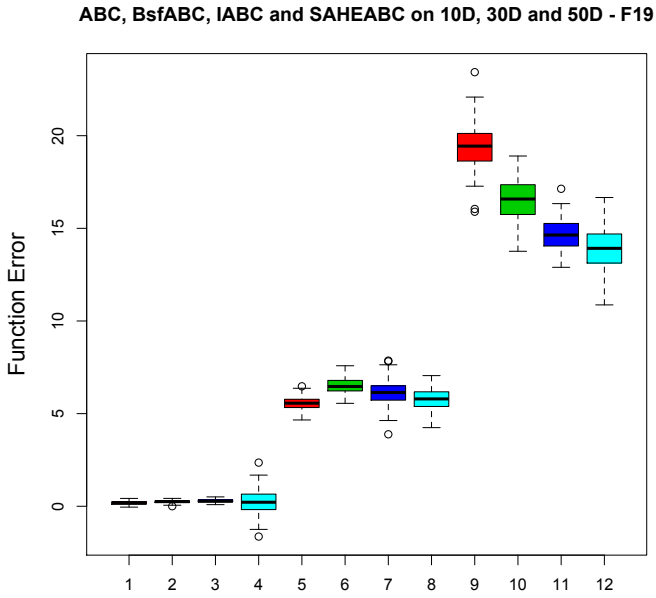



Figure 5.21: Boxplot of comparative convergence for F19

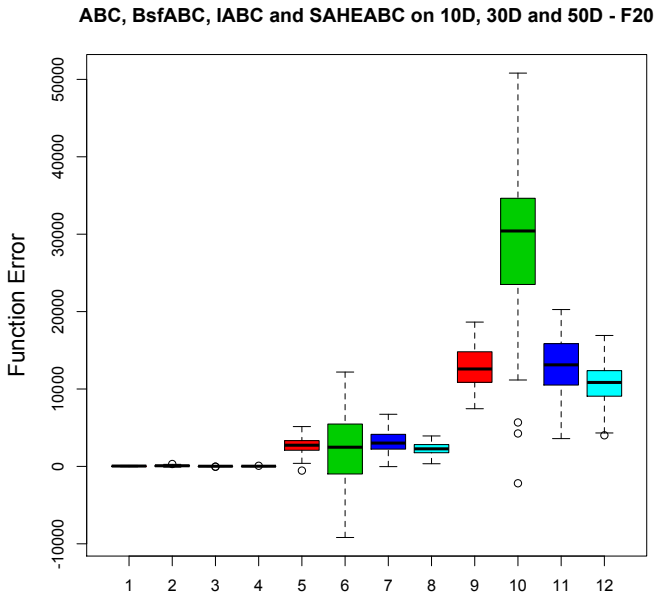



Figure 5.22: Boxplot of comparative convergence for F20

is the best compared to ABC on 10D, BsfABC on 30D and 50D, IABC on 30D, in addition, ABC outperforms than others on 10D and 30D, but not so obvious. With regard to Fig. 5.22, SAHEABC is the best on 10D and 30D. For the remaining hybrid functions, it is concluded that SAHEABC is the best for functions F17 as well as F18, F20 and F21 on 10D compared to ABC and BsfABC algorithms; functions F17, F21 as well as F22 on 30D and 50D compared to BsfABC algorithm according to Tables 5.6-5.8.

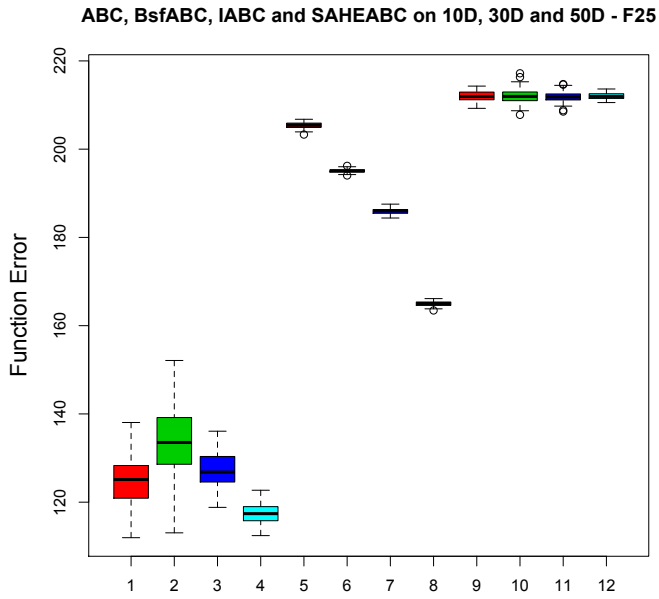



Figure 5.23: Boxplot of comparative convergence for F25

Figures 5.23-5.24 illustrate the boxplots of composition functions of F25 and F27 with the function errors of mean value for ABC, BsfABC, IABC, and SAHEABC algorithms on 10D, 30D, and 50D, respectively. According to Fig. 5.23, SAHEABC is the best on 10D. The convergence performances for all compared algorithms are very similar on 30D and 50D. SAHEABC algorithm reaches the best performance on 10D, 30D, and 50D from Fig. 5.24. For the remaining composition functions, it could be concluded that the performance of SAHEABC is the best except for function F29 on 10D, 30D, and 50D; function F23 on 10D and 30D compared to IABC algorithm; function 28 on 10D compared to ABC and IABC algorithms. SAHEABC algorithm achieves the similar performance for functions

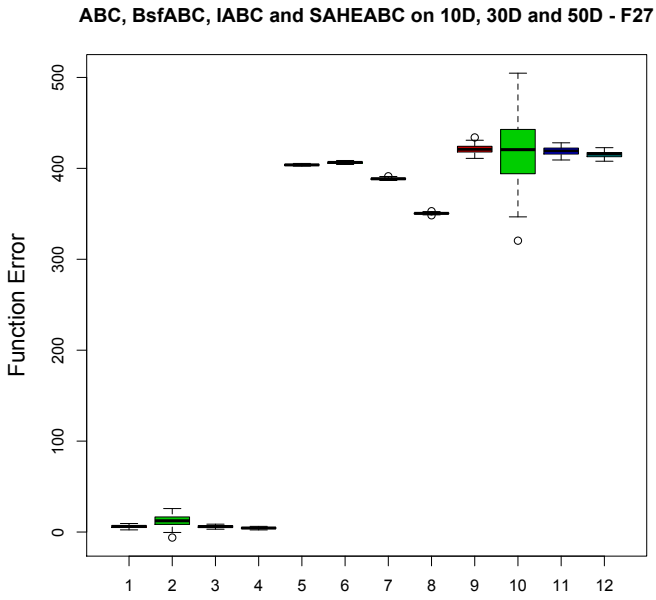



Figure 5.24: Boxplot of comparative convergence for F27

F23, F24 on 50D with function F26 on 30D and 50D.

According to the above Tables 5.7-5.10 and Figs. 5.9-5.24, SAHEABC was effective for unimodal functions on 10D. It was much effective for simple multi-modal functions. For hybrid functions and composition functions, the performance of SAHEABC algorithm was the best among the four compared algorithms as a whole, but not so significantly. However, the performance of ABC, BsfABC, and IABC algorithms were better than others for several functions on certain dimension size as the analyses mentioned above. For all 10D, 30D and 50D, functions F7 and F8 reached the best performance because the function errors of mean value reached zero.

## 5.5 Experimental Results of SAHEABC Algorithm Compared with ABC, BsfABC, IABC, SHADE and NRGA Algorithms

As mentioned in Chapter 3, ABC, BsfABC, IABC, SHADE and NRGA algorithms are conducted to compare with the SAHEABC algorithm in this section. Comparative experiments are conducted on SAHEABC, ABC, BsfABC, IABC, SHADE and NRGA algorithms for 30 test functions defined by CEC'14 test suite with selected parameters of *limit* for 150 and $NP$ for 100. However, The maximum evaluation sizes are set to 100,000, 300,000, 500,000 and 1, 000, 000 on 10D, 30D, 50D and 100D in the experiments, respectively. Each function of 30 test functions is executed 51 times with respect to 10D, 30D, 50D and 100D.The food source number is set to half of the $NP$ for 50, in the meantime, the employed bee number and onlooker bee number is set to same as food source number. The number of scout bee is set to 1 for each cycle.

The function errors of mean value for the SAHEABC, ABC, BsfABC, IABC, SHADE and NRGA algorithms and the signs of SAHEABC algorithm to ABC, BsfABC, IABC, SHADE and NRGA algorithms are given in Table 5.11-5.14 on 10D, 30D, 50D and 100D respectively. The comparison results are shown in Table 5.15.

From the Tables 5.11-5.15, the number of functions with better convergence performance of the SAHEABC decreased when the dimension size increased compared to ABC, BsfABC, IABC, SHADE and NRGA algorithms. It is concluded that convergence performance of SAHEABC is much competitive to BsfABC, IABC and NRGA algorithms. Especially, the convergence performance is very significant as compared to NRGA algorithm.

The convergence performance of SAHEABC compared to ABC, BsfABC, IABC, SHADE and NRGA algorithms for function errors of mean value are shown in Figs 5.25-5.28 on 10D, 30D, 50D and 100D respectively. The comparisons are conducted based on the function types, namely, unimodal, simple multi-modal, hybrid and composition functions.

According to Fig. 5.25, SAHEABC performs better than ABC, BsfABC, IABC, SHADE and NRGA algorithms on 10D, especially, it is much obvious for multi-modal and hybrid

Table 5.10: Function errors of mean value for the SAHEABC, ABC, BsfABC, IABC, SHADE and NRGA with the signs of SAHEABC compared with these algorithms on 10D

| F. | SAHEABC | ABC sign | BsfABC sign | IABC sign | SHADE sign | NRGA sign |
|---|---|---|---|---|---|---|
| F1 | 1.23e+04 | 1.98e+04 + | 2.45e+05 + | 1.39e+04 ≈ | 0.00e+00 - | 2.79e+04 + |
| F2 | 8.62e-01 | 7.55e-01 ≈ | 1.32e-01 - | 1.68e+00 + | 0.00e+00 + | 9.15e+02 + |
| F3 | 5.73e+00 | 4.01e+00 - | 7.01e+01 + | 1.42e+01 + | 0.00e+00 - | 1.52e+03 + |
| F4 | 1.11e-03 | 4.10e-03 + | 1.59e-03 + | 5.06e-03 + | 2.94e+01 + | 1.54e+01 + |
| F5 | 5.37e+00 | 1.15e+00 - | 5.41e+00 ≈ | 1.41e+00 - | 1.47e+01 + | 1.96e+01 + |
| F6 | 2.10e-01 | 5.23e-01 + | 1.28e+00 + | 8.28e-01 + | 0.00e+00 - | 2.45e+00 + |
| F7 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 1.85e-04 ≈ | 3.69e-03 ≈ | 2.03e-01 + |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 5.58e+00 + |
| F9 | 2.02e+00 | 2.81e+00 + | 4.33e+00 + | 3.87e+00 + | 3.14e+00 + | 8.69e+00 + |
| F10 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 1.52e-06 ≈ | 1.19e+02 + |
| F11 | 1.48e+01 | 1.83e+01 + | 6.13e+01 + | 1.94e+01 + | 7.35e+01 + | 5.76e+02 + |
| F12 | 1.10e-01 | 6.82e-02 - | 7.14e-02 - | 7.06e-02 - | 1.55e-01 ≈ | 1.24e-01 ≈ |
| F13 | 5.84e-02 | 6.37e-02 + | 4.97e-02 ≈ | 9.54e-02 + | 7.85e-02 + | 1.58e-01 + |
| F14 | 3.71e-02 | 7.62e-02 + | 6.41e-02 + | 7.34e-02 + | 1.01e-01 + | 2.54e-01 + |
| F15 | 2.97e-01 | 2.57e-01 ≈ | 4.83e-01 + | 4.27e-01 + | 4.89e-01 + | 1.02e+00 + |
| F16 | 7.25e-01 | 1.15e+00 + | 1.45e+00 + | 1.27e+00 + | 1.57e+00 + | 2.75e+00 + |
| F17 | 7.32e+03 | 9.67e+03 + | 1.05e+05 + | 6.23e+03 ≈ | 5.93e+00 - | 1.61e+04 + |
| F18 | 1.37e+01 | 2.60e+01 + | 1.63e+01 + | 3.73e+01 + | 1.76e-01 - | 7.42e+03 + |
| F19 | 7.70e-02 | 9.98e-02 + | 1.01e-01 + | 1.21e-01 + | 2.34e-01 + | 2.09e+00 + |
| F20 | 3.50e+00 | 5.23e+00 + | 1.33e+01 + | 6.01e+00 + | 2.50e-01 - | 1.72e+03 + |
| F21 | 1.43e+02 | 3.31e+02 + | 1.38e+04 + | 2.75e+02 + | 3.79e-01 - | 4.82e+03 + |
| F22 | 2.13e-02 | 6.54e-02 + | 5.22e-02 + | 2.05e-01 + | 3.29e-01 + | 3.76e+01 + |
| F23 | 1.77e+00 | 2.75e+00 + | 1.87e+01 + | 2.02e+00 + | 3.29e+02 + | 3.29e+02 + |
| F24 | 1.00e+02 | 8.95e+01 - | 9.17e+01 - | 8.34e+01 - | 1.09e+02 + | 1.31e+02 + |
| F25 | 1.12e+02 | 1.19e+02 ≈ | 1.25e+02 + | 1.19e+02 ≈ | 1.43e+02 + | 1.84e+02 + |
| F26 | 6.73e+01 | 8.68e+01 + | 1.00e+02 + | 9.54e+01 + | 1.00e+02 + | 1.00e+02 + |
| F27 | 2.94e+00 | 3.91e+00 + | 6.15e+00 + | 3.84e+00 + | 1.49e+02 + | 2.81e+02 + |
| F28 | 3.56e+02 | 1.88e+02 - | 3.15e+02 - | 1.97e+02 - | 3.91e+02 + | 4.77e+02 + |
| F29 | 2.42e+02 | 2.28e+02 ≈ | 2.17e+02 - | 2.17e+02 - | 2.22e+02 - | 4.13e+02 - |
| F30 | 4.61e+02 | 4.79e+02 ≈ | 5.01e+02 + | 4.83e+02 + | 4.78e+02 ≈ | 1.73e+03 + |

Table 5.11: Function errors of mean value for the SAHEABC, ABC, BsfABC, IABC, SHADE and NRGA with the signs of SAHEABC compared with these algorithms on 30D

| F. | SAHEABC | ABC sign | BsfABC sign | IABC sign | SHADE sign | NRGA sign |
|---|---|---|---|---|---|---|
| F1 | 3.44e+06 | 1.68e+06 - | 4.29e+06 + | 1.79e+06 - | 3.36e+02 - | 1.31e+06 - |
| F2 | 5.06e-01 | 1.05e+00 + | 3.11e-01 - | 9.11e-01 + | 0.00e+00 - | 9.30e+03 + |
| F3 | 4.04e+01 | 1.24e+01 - | 4.96e+01 + | 3.26e+01 - | 0.00e+00 - | 4.92e+03 + |
| F4 | 2.27e-02 | 1.04e-01 + | 9.65e-02 + | 2.58e-01 + | 0.00e+00 - | 9.36e+01 + |
| F5 | 2.02e+01 | 2.01e+01 ≈ | 2.00e+01 ≈ | 2.01e+01 ≈ | 2.01e+01 ≈ | 2.00e+01 ≈ |
| F6 | 1.00e+01 | 1.03e+01 ≈ | 1.34e+01 + | 1.13e+01 + | 1.15e+00 + | 1.79e+01 + |
| F7 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 2.96e-04 ≈ | 1.65e-02 + |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 3.02e+01 + |
| F9 | 1.48e+01 | 5.05e+01 + | 6.36e+01 + | 6.60e+01 + | 1.48e+01 ≈ | 4.57e+01 + |
| F10 | 0.00e+00 | 9.24e-02 + | 1.53e-01 + | 3.13e-02 + | 1.17e-02 + | 1.28e+03 + |
| F11 | 1.01e+03 | 1.31e+03 + | 1.62e+03 + | 1.30e+03 + | 1.49e+03 + | 3.42e+03 + |
| F12 | 2.46e-01 | 1.46e-01 - | 1.42e-01 - | 1.59e-01 - | 1.67e-01 + | 1.62e-01 + |
| F13 | 1.72e-01 | 1.66e-01 ≈ | 1.45e-01 ≈ | 2.28e-01 + | 2.12e-01 + | 2.82e-01 + |
| F14 | 1.03e-01 | 1.45e-01 + | 1.44e-01 + | 1.65e-01 + | 2.36e-01 + | 1.87e-01 + |
| F15 | 4.79e+00 | 4.16e+00 + | 6.71e+00 + | 6.84e+00 + | 2.58e+00 - | 1.41e+01 + |
| F16 | 7.53e+00 | 8.87e+00 + | 9.22e+00 + | 9.09e+00 + | 9.01e+00 + | 1.15e+01 + |
| F17 | 4.63e+05 | 5.25e+05 + | 2.05e+06 + | 5.24e+05 + | 1.05e+03 - | 3.36e+05 ≈ |
| F18 | 6.00e+03 | 2.53e+02 - | 8.05e+01 - | 1.58e+03 - | 5.42e+01 - | 5.50e+02 - |
| F19 | 6.26e+00 | 4.99e+00 - | 5.60e+00 - | 5.35e+00 - | 4.43e+00 - | 1.40e+01 + |
| F20 | 8.90e+02 | 1.09e+01 + | 6.61e+03 + | 1.24e+03 + | 1.20e+01 - | 1.20e+04 + |
| F21 | 1.20e+05 | 3.92e+04 - | 3.24e+05 + | 5.93e+04 - | 2.48e+02 - | 2.12e+05 + |
| F22 | 3.99e+01 | 4.18e+01 + | 1.62e+02 + | 4.83e+01 + | 1.03e+02 + | 4.21e+02 + |
| F23 | 3.15e+02 | 3.15e+02 ≈ | 3.15e+02 ≈ | 2.92e+02 - | 3.15e+02 ≈ | 3.15e+02 ≈ |
| F24 | 1.57e+02 | 1.99e+02 + | 2.15e+02 + | 2.08e+02 + | 2.25e+02 + | 2.29e+02 + |
| F25 | 2.04e+02 | 2.05e+02 ≈ | 2.04e+02 ≈ | 2.05e+02 ≈ | 2.03e+02 ≈ | 2.11e+02 ≈ |
| F26 | 1.00e+02 | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ |
| F27 | 4.04e+02 | 4.03e+02 ≈ | 3.06e+02 - | 3.53e+02 - | 3.24e+02 - | 5.89e+02 + |
| F28 | 7.58e+02 | 8.19e+02 + | 8.93e+02 + | 8.27e+02 + | 8.36e+02 + | 1.60e+03 + |
| F29 | 8.39e+02 | 8.23e+02 ≈ | 4.77e+02 + | 8.33e+02 ≈ | 7.23e+02 - | 1.33e+03 + |
| F30 | 1.23e+03 | 1.26e+03 ≈ | 1.79e+03 + | 1.63e+03 + | 1.91e+03 + | 3.23e+03 + |

Table 5.12: Function errors of mean value for the SAHEABC, ABC, BsfABC, IABC, SHADE and NRGA with the signs of SAHEABC compared with these algorithms on 50D

| F. | SAHEABC | ABC sign | BsfABC sign | IABC sign | SHADE sign | NRGA sign |
|---|---|---|---|---|---|---|
| F1 | 8.93e+06 | 4.50e+06 - | 7.83e+06 - | 6.61e+06 - | 1.78e+04 - | 2.13e+06 - |
| F2 | 2.80e+03 | 9.54e+01 - | 3.04e+00 - | 5.17e+01 - | 0.00e+00 - | 4.62e+03 + |
| F3 | 3.15e+03 | 2.17e+03 ≈ | 3.41e+03 ≈ | 2.10e+03 - | 0.00e+00 - | 1.13e+04 + |
| F4 | 4.95e-02 | 2.71e+00 + | 1.68e+00 + | 9.12e+00 + | 2.58e+01 + | 1.33e+02 + |
| F5 | 2.03e+01 | 2.03e+01 ≈ | 2.00e+01 ≈ | 2.03e+01 ≈ | 2.01e+01 ≈ | 2.00e+01 ≈ |
| F6 | 2.37e+01 | 2.56e+01 + | 3.10e+01 + | 2.66e+01 + | 4.12e+00 - | 3.56e+01 + |
| F7 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 2.07e-03 + | 1.31e-02 + |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 6.70e+01 + |
| F9 | 9.12e+01 | 1.39e+02 + | 1.54e+02 + | 1.71e+02 + | 3.26e+01 - | 9.32e+01 ≈ |
| F10 | 0.00e+00 | 2.13e-01 + | 5.56e-01 + | 1.12e-01 + | 1.25e-02 + | 2.57e+03 + |
| F11 | 3.05e+03 | 3.35e+03 + | 3.88e+03 + | 3.41e+03 + | 3.44e+03 + | 6.18e+03 + |
| F12 | 2.92e-01 | 1.86e-01 - | 1.83e-01 - | 1.97e-01 - | 1.60e-01 - | 2.07e-01 ≈ |
| F13 | 2.19e-01 | 2.60e-01 + | 2.26e-01 ≈ | 2.80e-01 + | 3.25e-01 + | 4.72e-01 + |
| F14 | 1.51e-01 | 1.77e-01 ≈ | 1.75e-01 ≈ | 2.03e-01 + | 2.92e-01 + | 3.16e-01 + |
| F15 | 1.50e+01 | 1.22e+01 - | 1.77e+01 + | 1.74e+01 + | 5.79e+00 - | 9.51e+01 + |
| F16 | 1.74e+01 | 1.74e+01 ≈ | 1.77e+01 ≈ | 1.71e+01 ≈ | 1.74e+01 ≈ | 2.06e+01 + |
| F17 | 2.85e+06 | 1.53e+06 ≈ | 1.48e+06 - | 1.85e+06 - | 2.49e+03 - | 3.47e+05 - |
| F18 | 7.41e+03 | 1.36e+03 - | 2.16e+02 - | 4.03e+03 - | 1.71e+02 - | 1.03e+03 - |
| F19 | 9.05e+00 | 1.13e+01 + | 1.31e+01 + | 1.21e+01 + | 9.16e+00 ≈ | 2.99e+01 + |
| F20 | 5.20e+03 | 7.24e+03 + | 3.61e+04 + | 7.79e+03 + | 1.85e+02 - | 1.72e+04 + |
| F21 | 1.33e+06 | 5.55e+05 - | 2.40e+06 + | 9.47e+05 - | 1.33e+03 - | 4.68e+05 - |
| F22 | 4.81e+02 | 3.30e+02 - | 6.89e+02 + | 5.10e+02 + | 4.85e+02 ≈ | 1.05e+03 + |
| F23 | 3.44e+02 | 3.44e+02 ≈ | 3.44e+02 ≈ | 3.21e+02 ≈ | 3.44e+02 ≈ | 3.44e+02 ≈ |
| F24 | 2.55e+02 | 2.57e+02 ≈ | 2.57e+02 ≈ | 2.57e+02 ≈ | 2.74e+02 + | 2.73e+02 + |
| F25 | 2.10e+02 | 2.10e+02 ≈ | 2.10e+02 ≈ | 2.10e+02 ≈ | 2.07e+02 ≈ | 2.19e+02 + |
| F26 | 1.00e+02 | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.00e+02 ≈ | 1.22e+02 + |
| F27 | 4.18e+02 | 4.14e+02 ≈ | 4.32e+02 ≈ | 4.12e+02 ≈ | 4.36e+02 + | 1.19e+03 + |
| F28 | 1.17e+03 | 1.41e+03 + | 1.54e+03 + | 1.30e+03 + | 1.14e+03 ≈ | 4.84e+03 + |
| F29 | 1.48e+03 | 9.55e+02 - | 1.22e+03 + | 1.45e+03 ≈ | 8.59e+02 - | 2.46e+03 + |
| F30 | 8.47e+03 | 8.71e+03 ≈ | 1.00e+04 + | 8.73e+03 ≈ | 9.53e+03 + | 1.84e+04 + |

Table 5.13: Function errors of mean value for the SAHEABC, ABC, BsfABC, IABC, SHADE and NRGA with the signs of SAHEABC compared with these algorithms on 100D

| F. | SAHEABC | ABC sign | BsfABC sign | IABC sign | SHADE sign | NRGA sign |
|---|---|---|---|---|---|---|
| F1 | 5.11e+07 | 3.10e+07 ≈ | 3.91e+07 ≈ | 3.43e+07 ≈ | 1.40e+05 - | 3.24e+07 ≈ |
| F2 | 2.39e+04 | 5.92e+02 - | 2.34e+03 - | 3.45e+02 - | 0.00e+00 - | 1.46e+04 ≈ |
| F3 | 1.00e+04 | 7.62e+03 - | 7.81e+03 - | 9.26e+03 ≈ | 2.29e-03 - | 2.70e+04 + |
| F4 | 3.05e+01 | 1.05e+02 + | 1.04e+02 + | 1.05e+02 + | 1.17e+02 + | 3.96e+02 + |
| F5 | 2.06e+01 | 2.05e+01 ≈ | 2.04e+01 ≈ | 2.05e+01 ≈ | 2.02e+01 ≈ | 2.00e+01 ≈ |
| F6 | 7.15e+01 | 7.33e+01 ≈ | 8.00e+01 + | 7.44e+01 + | 2.79e+01 - | 9.76e+01 + |
| F7 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 1.28e-03 + | 2.22e-02 + |
| F8 | 0.00e+00 | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 0.00e+00 ≈ | 2.00e+02 + |
| F9 | 3.40e+02 | 5.50e+02 + | 6.02e+02 + | 6.35e+02 + | 9.69e+01 - | 2.45e+02 + |
| F10 | 8.92e-03 | 1.56e+00 + | 3.29e+00 + | 1.58e+00 + | 1.15e-02 + | 6.33e+03 + |
| F11 | 7.19e+03 | 9.89e+03 + | 1.09e+04 + | 1.00e+04 + | 9.68e+03 ≈ | 1.37e+04 + |
| F12 | 5.38e-01 | 3.47e-01 ≈ | 4.16e-01 ≈ | 3.73e-01 ≈ | 2.29e-01 - | 3.80e-01 ≈ |
| F13 | 3.11e-01 | 3.01e-01 ≈ | 3.23e-01 ≈ | 3.87e-01 ≈ | 4.13e-01 + | 5.01e-01 + |
| F14 | 2.15e-01 | 2.24e-01 ≈ | 2.18e-01 ≈ | 2.28e-01 ≈ | 2.06e-01 ≈ | 1.63e-01 ≈ |
| F15 | 5.114e+01 | 4.84e+01 ≈ | 6.05e+01 + | 6.37e+01 + | 1.83e+01 - | 4.53e+02 + |
| F16 | 4.01e+01 | 4.00e+01 ≈ | 4.00e+01 ≈ | 3.99e+01 ≈ | 3.96e+01 ≈ | 4.36e+01 + |
| F17 | 1.53e+07 | 7.41e+06 - | 9.62e+06 - | 9.69e+06 - | 1.16e+04 - | 2.17+06 + |
| F18 | 4.60e+04 | 1.19e+03 - | 7.05e+02 - | 3.41e+04 ≈ | 5.96e+02 - | 6.32e+02 - |
| F19 | 3.05e+01 | 3.47e+01 + | 3.85e+01 + | 3.79e+01 + | 9.64e+01 + | 9.93e+01 + |
| F20 | 5.70e+04 | 3.66e+04 ≈ | 9.84e+04 + | 3.93e+04 ≈ | 5.61e+02 - | 7.17e+04 ≈ |
| F21 | 5.81e+06 | 4.03e+06 ≈ | 6.99e+06 ≈ | 3.82e+06 ≈ | 3.36e+03 - | 1.92e+06 ≈ |
| F22 | 2.05e+03 | 1.31e+03 - | 1.82e+03 ≈ | 1.30e+03 - | 1.43e+03 - | 2.29e+03 ≈ |
| F23 | 3.48e+02 | 3.48e+02 ≈ | 3.48e+02 ≈ | 3.48e+02 ≈ | 3.48e+02 ≈ | 3.70e+02 ≈ |
| F24 | 3.22e+02 | 3.29e+02 ≈ | 3.30e+02 ≈ | 3.29e+02 ≈ | 3.95e+02 ≈ | 3.76e+02 ≈ |
| F25 | 2.32e+02 | 2.37e+02 ≈ | 2.43e+02 ≈ | 2.38e+02 ≈ | 2.63e+02 ≈ | 2.27e+02 ≈ |
| F26 | 1.00e+02 | 1.01e+02 ≈ | 1.00e+02 ≈ | 1.01e+02 ≈ | 2.00e+02 + | 2.00e+02 + |
| F27 | 4.11e+02 | 4.45e+02 + | 5.47e+02 + | 4.46e+02 + | 8.62e+02 + | 2.35e+03 + |
| F28 | 2.21e+03 | 3.14e+03 + | 3.37e+03 + | 2.89e+03 + | 2.35e+03 + | 1.21e+04 + |
| F29 | 1.85e+03 | 2.03e+02 - | 2.58e+03 + | 2.22e+03 + | 1.24e+03 ≈ | 3.81e+03 + |
| F30 | 1.09e+04 | 1.58e+04 + | 1.42e+04 + | 2.69e+04 + | 8.80e+03 - | 3.45e+04 + |

Table 5.14: Comparison performance with function errors of mean value for SAHEABC to ABC, BsfABC, IABC, SHADE and NRGA algorithms

| SAHEABC (10D) VS. | ABC | BsfABC | IABC | SHADE | NRGA |
|---|---|---|---|---|---|
| + | 17 | 20 | 19 | 17 | 28 |
| ≈ | 8 | 5 | 6 | 5 | 1 |
| - | 5 | 5 | 5 | 8 | 1 |
| SAHEABC (30D) VS. | ABC | BsfABC | IABC | SHADE | NRGA |
| + | 12 | 18 | 16 | 11 | 23 |
| ≈ | 12 | 7 | 5 | 12 | 5 |
| - | 6 | 5 | 9 | 7 | 2 |
| SAHEABC (50D) VS. | ABC | BsfABC | IABC | SHADE | NRGA |
| + | 9 | 13 | 12 | 9 | 22 |
| ≈ | 13 | 12 | 11 | 9 | 4 |
| - | 8 | 5 | 7 | 12 | 4 |
| SAHEABC (100D) VS. | ABC | BsfABC | IABC | SHADE | NRGA |
| + | 8 | 12 | 11 | 8 | 18 |
| ≈ | 16 | 14 | 15 | 9 | 11 |
| - | 6 | 4 | 4 | 13 | 1 |

functions, the number of functions with worse performance is very small. SAHEABC is better than BsfABC, IABC and NRGA algorithms for unimodal functions and better than ABC, IABC, SHADE and NRGA algorithms on composition functions.

From Fig. 5.26, it is concluded that SAHEABC performs significantly better than ABC, BsfABC, IABC, SHADE and NRGA algorithms on 30D for multi-modal and composition functions. SAHEABC is better than BsfABC and NRGA algorithms for unimodal functions and significantly better than NRGA algorithm on all functions.

According to Fig. 5.27, SAHEABC outperforms significantly than ABC, BsfABC, IABC, SHADE and NRGA algorithms on 50D for multi-modal and composition functions, similar or worse for hybrid functions. SAHEABC is only better than NRGA algorithm for unimodal functions. The convergence performance of SAHEABC is better than BsfABC algorithm on hybrid functions.

With regard to Fig. 5.28, it is concluded that SAHEABC outperforms than ABC, BsfABC, IABC, SHADE and NRGA algorithms for multi-modal and composition functions

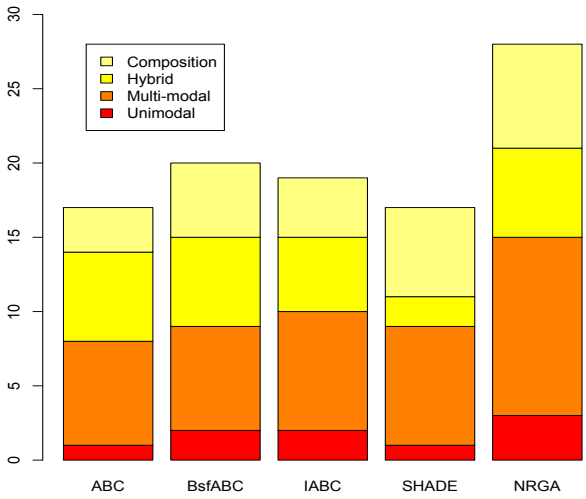



Figure 5.25: Convergence performance of SAHEABC compared to ABC, BsfABC, IABC, SHADE and NRGA algorithms with function errors of mean value on 10D

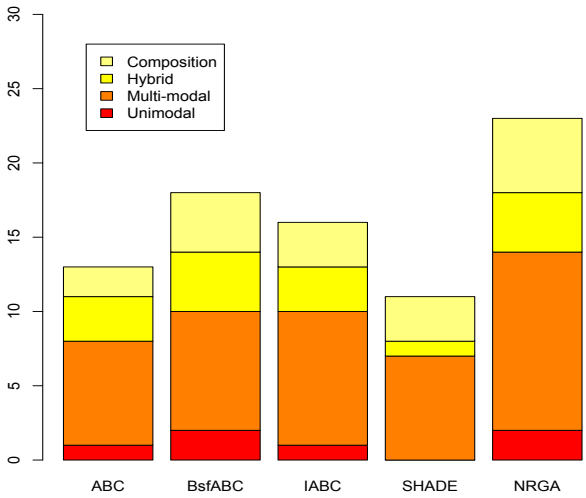



Figure 5.26: Convergence performance of SAHEABC compared to ABC, BsfABC, IABC, SHADE and NRGA algorithms with function errors of mean value on 30D

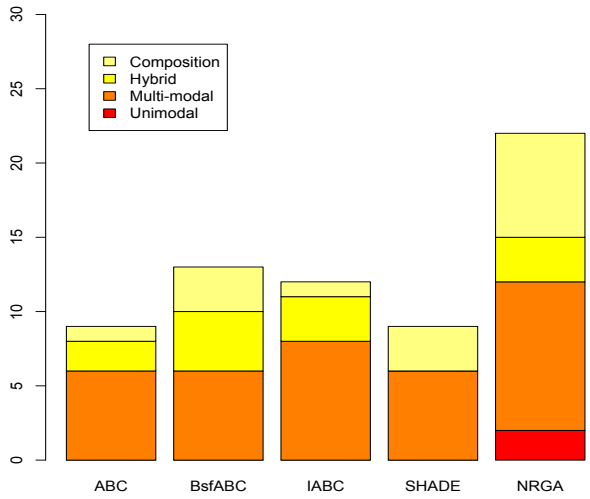



Figure 5.27: Convergence performance of SAHEABC compared to ABC, BsfABC, IABC, SHADE and NRGA algorithms with function errors of mean value on 50D

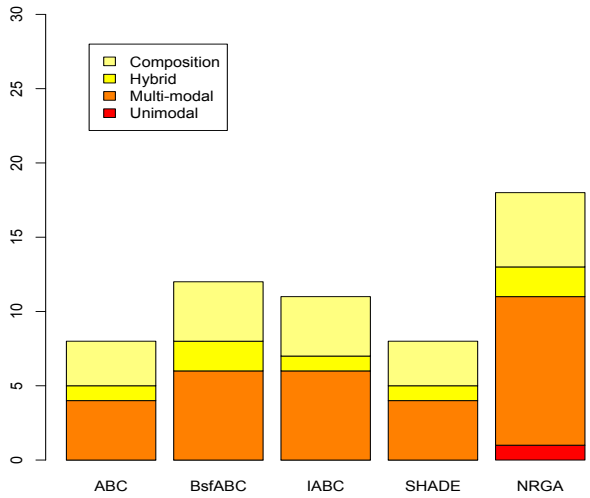



Figure 5.28: Convergence performance of SAHEABC compared to ABC, BsfABC, IABC, SHADE and NRGA algorithms with function errors of mean value on 100D

on 100D. For unimodal and hybrid functions, SAHEABC is better than NRGA algorithm.

According to the analyses of above Tables 5.11-5.15 and Figs 5.25-5.28, SAHEABC was much effective for multi-modal functions, however, it was not so obvious for unimodal functions. For hybrid functions and composition functions, the performance of SAHEABC was the best among the compared algorithms as a whole, but not so significantly for hybrid functions. SHADE performed the best for unimodal functions among all the algorithms. NRGA was the worst as a whole for all the functions. For all 10D, 30D, 50D and 100D, functions F7 and F8 reached the best performance because the function errors of mean value got zero. It was also observed that SAHEABC was more effective than ABC, BsfABC, IABC, SHADE and NRGA algorithms on those functions with the properties of non-separability, having many local optima and second local optimum is far from the global optimum, having a narrow valley from local optimum to global optimum, having different properties around different local optima and different variables subcomponents.

## 5.6 Summary

A few disadvantages still exist for ABC algorithm, e.g., random uniform initialization loses the effectiveness for higher dimensional size although plays a more important role in higher dimensional problems (up to 50 dimensions), and lower fitness individuals have low probability to be selected as onlooker bees. To overcome the disadvantages, a levy flight-based hybrid ABC (LFHABC) algorithm was proposed. Based on LFHABC algorithm, a self adaptive hybrid enhanced ABC algorithm (SAHEABC) was proposed by modifying the probability on onlooker bees to increase the selected probability of lower fitness individuals.

Comparative experiments were implemented for LFHABC and SAHEABC algorithms to demonstrate the effectiveness and efficiency of these algorithms. More specifically, CEC'13 test suite were adopted by LFHABC algorithm and CEC'14 test suite were utilized by SAHEABC algorithm and then the performance of the extended ABC algorithms were statistically analyzed by the comparison experiments with ABC, IABC, BsfABC, SHADE and NRGA algorithms.

As the comparison experimental results of SAHEABC with ABC, BsfABC, IABC,

SHADE and NRGA, SAHEABC algorithm was much effective for multi-modal functions. For hybrid functions and composition functions, the performance of SAHEABC algorithm was the best, but not so significantly for hybrid functions. It was also observed that SAHE-ABC outperformed than ABC, BsfABC, IABC, SHADE and NRGA algorithms on those functions with the properties of non-separability, separability, having many local optima and second local optimum is far from the global optimum, having a narrow valley from local optimum to global optimum, having different properties around different local optima and variables subcomponents.

# Chapter 6

# Conclusion

Swarm Intelligence (SI) is the collective behavior of decentralized, self-organized systems with natural or artificial ways as a discipline of artificial intelligence. Artificial bee colony (ABC) is one of the SI based algorithms inspired by the food foraging collective behavior of honey bees. ABC has attracted the researchers to make great contributions by modifying it in various ways. The modified ABC and hybridized ABC algorithms are superior to other algorithms in terms of its simplicity, flexibility and robustness when solving the numerical optimization problems, along with the advantages of the improved versions of ABC algorithm. The extended ABC algorithms of improved hybrid ABC (IHABC), levy flight-based hybrid ABC (LFHABC) and self adaptive enhanced hybrid ABC (SAHEABC) were proposed in the thesis in order to overcome the disadvantages, e.g., low convergence speed when they solving unimodal or composition functions, low exploitation abilities, and are also easily trapped in local optima when solving multi-modal functions, low diversity for initialization and lower fitness individuals hard to be selected as onlooker bees .

The conclusions of the chapters in the thesis are made as following.

The backgrounds of continuous optimization problems, swarm intelligence and evolutionary computation, goal of the thesis and the structure of the thesis were introduced in Chapter 1.

In Chapter 2, the review for the history of test functions were described, the brief history of test functions and the test function suites were then analyzed. Availability of the benchmark of CEC provides the platform for comparing new optimization algorithms to the

state-of-the-art ones. CEC benchmark is the widely used for evaluating the test functions which cover a wide range of specialized optimization problems. The representative benchmark test functions of CEC'13 and CEC'14 which were introduced based on the CEC'05 test functions were given the definitions in more detailed ways.

In Chapter 3, swarm intelligence (SI) based algorithms and evolutionary algorithms (EAs) were introduced as bio-inspired algorithms which proposed by mimicking the successful characteristics of the complex systems inspired from nature. Like all social insects, the honey bees have the opportunity to engage in social learning by interacting with simple other entities in their social group, while the group of insects possesses collective intelligence, the individuals within the group have complex cognitive abilities. Self-organization, stigmergy and division of labor were interpreted firstly as component keys in bee colony. The ABC algorithm was described in detailed explanations. PSO, GA, NRGA, DE and SHADE were given the explanations in more detail.

In Chapter 4, the previous research about the modified versions of ABC algorithms and the state-of-the-art modified ABC algorithms of BsfABC and IABC algorithms were introduced firstly. IHABC was proposed inspired by self adaptive mechanism, incorporated with DE and PSO algorithms in order to overcome the disadvantages such as low convergence speeds when they solve unimodal functions, low exploitation abilities and easily trapped in local optima when they solve complex multimodal functions. Comparative experiments for ABC, IHABC, DE and PSO algorithms were implemented for test functions defined by CEC'13. As the experimental results, IHABC and ABC algorithms were effective on unimodal function F1, IHABC outperformed than ABC algorithm, the convergence performance of IHABC was the best as a whole for multi-modal and composition functions, especially on 10D and 30D. ABC algorithm outperformed than DE and PSO algorithms DE is better than PSO algorithm. Functions F1, F5, and F11 reached the best convergence performance with function errors of mean value of zero on 10D, 30D, and 50D.

In Chapter 5, LFHABC and SAHEABC were proposed in order to overcome the disadvantages, e.g., random uniform initialization loses the effectiveness for higher dimensional size although plays a more important role in higher dimensional problems, and lower fitness individuals have low probability to be selected as onlooker bees. Comparative experiments were implemented for LFHABC and SAHEABC algorithms to demonstrate the

effectiveness of these algorithms. More specifically, CEC'13 was adopted by LFHABC and CEC'14 was utilized by SAHEABC. Through the comparison experiments of ABC, IABC, BsfABC, SHADE and NRGA algorithms, the results showed that SAHEABC was much effective for multi-modal functions, SAHEABC was the best for hybrid functions and composition functions, but not so significantly for hybrid functions. It was also observed that SAHEABC outperformed than ABC, BsfABC, IABC, SHADE and NRGA algorithms on those functions with the properties of non-separability, separability, having many local optima and second local optimum is far from the global optimum, having a narrow valley from local optimum to global optimum, having different properties around different local optima and variables subcomponents.

Chapter 6 made a conclusion for the thesis.

This thesis focuses on the improving the performance of bio-inspired optimization algorithms for solving continuous optimization problems. For continuous optimization problems, the variables in the model are nominally allowed to take on a continuous range of values, usually real numbers. As the widely used benchmark test functions, CEC'13 and CEC'14 were adopted in the thesis for proving the effectiveness of the proposed extended ABC algorithms. Through numerous comparative experiments implemented, IHABC algorithm was competitive or superior to recent state-of-the-art algorithms for unimodal, multimodal and composition continuous optimization problems defined by CEC'13, LFHABC and SAHEABC algorithms were competitive or superior to recent state-of-the-art algorithms for unimodal, multimodal, hybrid and composition continuous optimization problems defined by CEC'14.

In the thesis, the extended ABC algorithms are only adopted on continuous optimization problems defined by CEC'13 and CEC'14. The proposed extended ABC algorithms had the competitive performance compared to recent state-of-the-art algorithms so that they will be applied to solve the continuous-discrete optimization problems or real-world problems with more complex properties such as large-scale dynamic problems. The extended ABC algorithms still has the characteristics for hybridizing with other approaches easily, therefore, they may be combined with other state-of-the-art algorithms to develop extended ABC algorithms with much higher performance.

# Bibliography


Ackley, D. A connectionist machine for genetic hill climbing. Kluwer, Boston, 1987.

Akay, B., Karaboga, D. Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of Intelligent Manufacturing 23(4), pp. 1001-1014, 2010a.

Akay, B., Karaboga, D. A modified ABC for real-parameter optimization. Information Sciences 192, pp. 120-142, 2012.

Alatas, B. Chaotic bee colony algorithms for global numerical optimization. Expert Systems with Applications 37, pp. 5682-5687, 2010.

Aydin, D., Liao, T.J., Marco A. Improving performance via population growth and local search: the case of the ABC algorithm. IRIDIA Technical Report TR/IRIDIA/2011-015, 2011.

Back, T. Evolution strategies: An alternative evolutionary algorithm. Artificial Evolution. Springer Berlin Heidelberg, 1996.

B*ä*ck, T., Fogel, D. B. and Michalewicz, Z. Handbook of evolutionary computation. IOP Publishing Ltd., 1997.

Banharnsakun, A., Achalakul, T., Sirinaovakul, B. The best-so-far selection in ABC algorithm. Applied Soft Computing 11, pp. 2888-2901, 2011.

Bao, L. and Zeng, J. C. Comparison and analysis of the selection mechanism in the artificial bee colony algorithm. Ninth International Conference on Hybrid Intelligent Systems (HIS'09), Vol. 1, 2009.

Bao, Li, and Zeng, J. C. A bi-group differential artificial bee colony algorithm. Control Theory & Applications 28(2), pp. 266-272, 2011.

Benítez, C.V., Lopes, H. Artificial bee colony algorithm approaches for protein structure prediction using the 3dhp-sc model. Intelligent Distributed Computing IV. pp. 255-264, 2010.

Bersini, H. et al. Results of the first international contest on evolutionary optimization (1st ICEO). In Proc. of IEEE Inte. Conf. on Evol. Comp., ICEC'96, Piscataway, NJ, pp. 611-615, 1996.

Bonabeau, E., Theraulaz, G., Deneubourg, J. L., Aron, S. and Camazine, S., Self-organization in social insects. Trends in Ecology and Evolution 12:188-193, 1997a.

Bonabeau, E., Sobkowski, A., Theraulaz, G., Deneubourg, J. L., Adaptive Task Allocation Inspired by a Model of Division of Labor in Social Insects, BCEC. 1997b.

Bonabeau, E., Dorigo, M., Theraulaz, G. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, pp. 1-21, 1999.

Blum, C. and Roli, A. Metaheuristic in combinatorial optimization: overview and conceptual camparison. ACM Comu. Surv. 35, pp. 268-308, 2003.

Brown, C.T., Liebovitch, L.S., Glendon, R. L*é*vy flights in Dobe Ju /' hoansi foraging patterns. Human Ecology 35, pp. 129-138, 2007.

Burachik, R. et al. Full convergence of the steepest descent method with inexact line searches. Optimization 32, pp. 137-146, 1996.

Camazine, S. and Sneyd, J. A model of collective nectar source selection by honey bees: self-organization through simple rules. Journal of theoretical Biology 149(4), pp. 547-571, 1991.

Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulaz, G., Bonabeau, E. Selforganization in biological systems, Princeton Univ. press, New Jersey, 2001.

Celik, M., Karaboga, D., Koyl, F. Artificial bee colony data miner (abc-miner). Proc. of IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), pp. 96-100, 2011.

Chen, T. G. and Xiao, R. B. Enhancing artificial bee colony algorithm with self-adaptive searching strategy and artificial immune network operators for global optimization. The Scientific World Journal 2014, 2014.

Civicioglu, P., Besdok, E. A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artificial Intelligence Review 39(4), pp. 315-346, 2013.

Cuevas, E. et al. A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Systems with Applications, 40(16), pp. 6374-6384, 2013.

Das, S., and Suganthan, P. N. Differential evolution: a survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1), pp. 4-31, 2011.

De Jong, K. A. Analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, 1975.

Ding, J. Advances in Network Management, Taylor & Francis, pp. 164-169, 2009.

Diwold, K. et al. Performance evaluation of artificial bee colony optimization and new selection schemes. Memetic Computing 3(3), pp. 149-162, 2011.

Dorigo, M., Vittorio M. and Alberto, C. Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26(1), pp. 29-41, 1996.

Dorigo, M., Birattari, M., Stützle, T. Ant colony optimization: artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine, 1(4), 2006.

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

El-Abd, M. Black-box optimization benchmarking for noiseless function testbed using artificial bee colony algorithm. Proc. of the 12th annual conference companion on Genetic and evolutionary computation. pp. 1719-1724, 2010.

Engelbrecht, A. P. Computational intelligence: an introduction. John Wiley & Sons, 2007.

Finck, S., et al. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2010.

Fister J. I., Yang, X. S., Fister, I., Brest, J. and Fister, D. A brief review of nature-inspired algorithms for optimization. Elektřotehniski Vestnik 80(3), pp. 1-7, 2013.

Fogel, L. J., Owens, A. J. and Walsh, M. J. Artificial intelligence through simulated evolution. 1966.

Fogel, D. B. Introduction to evolutionary computation. Evolutionary computation 1, pp. 1-3, 2000.

Gao, W., Liu, S. A modified artificial bee colony algorithm. Computers & Operations Research 39, pp. 687-697, 2012.

Glover, F., James P. K., and Manuel, L. Genetic algorithms and tabu search: hybrids for optimization. Computers & Operations Research 22(1), pp. 111-134, 1995.

Griewank, A. O. Generalized descent for global optimization. Journal of optimization theory and applications 34(1), pp. 11-39, 1981.

Hansen, N. et al. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. 2009.

Holland, J.H. Adaption in Nature and Artificial System. University of Michigan Press, Ann Arbor, MI, 1975.

Holldobler, B., and Edward, O. W. The superorganism: the beauty, elegance, and strangeness of insect societies. WW Norton & Company, 2009.

Ilonen, J., Kamarainen, J. K. and Lampinen, J. Differential evolution training algorithm for feed-forward neural networks. Neural Process. Lett. 7(1), pp. 93-105, 2003.

Johnson, D. S., Aragon, C. R., Mcgeoch, L. A., Schevon, C. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. Operations research 37(6), pp. 865-892, 1989.

Johnson, C. G. Genetic programming with fitness based on model checking. Genetic Programming. Springer Berlin Heidelberg, pp. 114-124, 2007.

Jones, N. C. and Pevzner, P. An introduction to bioinformatics algorithms. MIT press, 2004.

Joshi, R. and Sanderson, A. C. Minimal representation multisensor fusion using differential evolution. IEEE Trans. Syst. Man Cybern. A, Syst. Humans 29(1), pp. 63-76, 1999.

Karaboga, D. An idea based on honey bee swarm for numerical optimization. Tech. Rep. TR06, Erciyes Univ. Press, Erciyes, 2005.

Karaboga, D. and Akay, B. An artificial bee colony (abc) algorithm on training artificial neural networks. 15th IEEE Signal Processing and Communications Applications, pp.1-4, 2007.

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

Karaboga, D. and Basturk, B. On the performance of artificial bee colony (ABC) algorithm. Applied Soft computing 8, pp. 687-697, 2008.

Karaboga, D. and Akay, B. A comparative study of artificial Bee colony algorithm. Applied Mathematics and Computation 214(1), pp. 108-132, 2009.

Karaboga, D. et al. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review 42(1), pp. 21-57, 2014.

Kennedy, J., Eberhart, R. Particle swarm optimization. IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.

Kim, J. H.. and Myung, H. Evolutionary programming techniques for constrained optimization problems. Evolutionary Computation, IEEE Transactions on 1(2), pp. 129-140, 1997.

Kong, X. Y., Liu, S. Y., and Wang, Z. A new hybrid artificial bee colony algorithm for global optimization. International Journal of Computer Science 10(1), 2013.

Koza, J. R. Genetic programming: on the programming of computers by means of natural selection. Vol. 1. MIT press, 1992.

Krohling, R. and Coelho, L. S. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 36(6), pp. 1407-1416, 2006.

Kuang, F. J., et al. A novel chaotic artificial bee colony algorithm based on tent map. Evolutionary Computation (CEC), 2014 IEEE Congress on. IEEE, 2014.

Li, Y. C., Wang, Y.L. and Li, B. A hybrid artificial bee colony assisted differential evolution algorithm for optimal reactive power flow. Electrical Power and Energy Systems 52, pp. 25-33, 2013.

Li, X. D. and Engelbrecht, A. P. Particle swarm optimization: An introduction and its recent developments. In Proc. Genetic Evol. Comput. Conf. pp. 3391-3414, 2007.

Liang, J.J., Qu, B.Y., Suganthan, P.N., Hernández-Díaz, A.G. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical Report 201212, 2013a.

Liang, J.J., Qu, B.Y., Suganthan, P.N., 2013b. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real parameter numerical optimization. Technical Report 201311.

Liu, B., Wang, L. and Jin, Y. H. An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Trans. Syst., Man, Cybern. B, Cybern. 37(1), pp. 18-27, 2007.

Mahale, R. A., and Chavan, S. D. A survey: evolutionary and swarm based bio-inspired optimization algorithms. International Journal of Scientific and Research Publications 2(12), pp. 1-6, 2012.

Marsh, L. and Onof, C. Stigmergic epistemology, stigmergic cognition. Cognitive Systems Research 9(1), pp. 136-149, 2008.

Michener, C. D. The Social Behavior of the Bees: a comparative study. Harvard University Press, 1974.

Miller, P. Smart swarm. London, UK: Collins, 2010.

Mirjalili, S, Mirjalili, S. M. and Lewis, A. Grey wolf optimizer. Advances in Engineering Software 69, pp. 46-61, 2014.

Mirjalili, S. The ant lion optimizer. Advances in Engineering Software 83, pp. 80-98, 2015.

Mirjalili, S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, pp.1-21, 2015.

Moussaid, M., Garnier, S., Theraulaz, G. and Helbing, D. Collective Information Processing and Pattern Formation in Swarms, Flocks, and Crowds. Topics in Cognitive Science 1, pp. 469-497, 2009.

Nelder, J. A. and Mead, R. A simplex method for function minimization. The Computer Journal 7(4), pp. 308-313, 1965.

Ohkura, K., Yasuda, T., Kadota, M. and Matsumura, Y. An Extended SHADE and Its Evaluations. Proc. of 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES 2014) 2, pp. 493-504 , 2014.

Ohkura, K., Yoshiyuki M. and Kanji U. Robust evolution strategies. Simulated Evolution and Learning. Springer Berlin Heidelberg, pp. 10-17, 1999.

Okdem, S., Karaboga, D., Ozturk, C. An application of wireless sensor network routing based on artificial bee colony algorithm. IEEE Congress on Evolutionary Computation, pp. 326-330, 2011.

Pavlyukevich, I. L*é*vy flights, non-local search and simulated annealing. Computational Physics 226, pp. 1830-1844, 2007.

Powell, M. JD. An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal 7(2), pp. 155-162, 1964.

Qin, A. K. and Li, X. Differential evolution on the CEC-2013 single-objective continuous optimization testbed. Proc. of IEEE Congress on Evolutionary Computation, Cancun, Mexico, pp. 1099-1106, 2013.

Qi, L. Q. and Sun, S. A nonsmooth version of Newton‘s method. Mathematical programming 58(1-3), pp. 353-367, 1993.

Rahnamayan, S., Hamid, R.T. and Magdy, M.A.S. Opposition-based differential evolution. IEEE Trans. of Evolution Computation 12(1), pp. 64-79, 2008.

Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. GSA: a gravitational search algorithm. Information Sciences 179(13), pp. 2232-2248, 2009.

Rechenberg, I. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-holzbog, Stuttgart, 1973.

Reinhard, J. and Srinivasan, S. The role of scents in honey bee foraging and recruitment. Food Exploitation by Social Insects: Ecological, Behavioral, and Theoretical Approaches. CRC Press, 1st ed., pp.165-182, 2009.

Richards, M. and Ventura, D. Choosing a starting configuration for PSO. IEEE International Conference on Neural Networks 23, pp. 2309-2312, 2004.

Robinson, G. E. Regulation of division of labor in insect societies. Annual review of entomology 37(1), pp. 637-665, 1992.

Rogalsky, T., Kocabiyik, S. and Derksen, R.W. Differential evolution in aerodynamic optimization. Canadian Aeronautics and Space Journal 46(4), pp. 183-190, 2000.

Rosenbrock, H. H. An automatic method for finding the greatest or least value of a function. The Computer Journal 3(3), pp. 175-184, 1960.

Schaffer, J. D. (1984). Some experiments in machine learning using vector evaluated genetic algorithms. Ph. D. Thesis, Nashville, TN: Vanderbilt University.

Schwefel, H. P. Numerische Optimierung von Computer-Modellen (PhD thesis), 1974.

Schwefel, H. P. Numerical optimization of computer models. John Wiley & Sons, Inc., 1981.

Seeley, T. D., Camazine, S. and Sneyd, J. Collective decision-making in honey bees: how colonies choose among nectar sources. Behavioral Ecology and Sociobiology 28(4), pp. 277-290, 1991.

Seeley, T. D. and Towne, W. F. Tactics of dance choice in honey bees: Do foragers compare dances? Behav. Ecol. Sociobiol. 30, pp. 59-69, 1992.

Seeley, T. D. and Buhrman, S. C. Group decision making in swarms of honey bees. Behavioral Ecology and Sociobiology 45(1), pp. 19-31, 1999.

Sharma, T. K. and Pant, M. Blend of Local and Global Variant of PSO in ABC, World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 113-119, 2013.

Stephen, C., James, M. and Antonio, B. R. Standard Particle Swarm Optimization on the CEC2013 Real-Parameter Optimization Benchmark Functions. Tech. Rep., School of Information Technology, York University, 2013.

Storn, R., Price, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), pp. 341-359, 1997.

Suganthan, P. N. et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report 2005005, 2005.

Tanabe, R. and Fukunaga, A. Success-history based parameter adaptation for differential evolution. In 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 71-78, 2013.

Tereshko, V. and Loengarov, A. Collective decision-making in honeybee foraging dynamics. Computing and Information Systems 9(3), 2005.

Tizhoosh, H.R. Opposition-based learning: a new scheme for machine intelligence. Proc. of International Conference on Comp. Intell. for Modeling, Control and Automation 1, pp. 695-701, 2005.

Torn, A. and Zilinskas, A. Global optimization. Springer-Verlag New York, Inc., 1989.

Udgata, S.K., Sabat, S.L., Mini, S. Sensor deployment in irregular terrain using artificial bee colony algorithm. IEEE Congress on Nature & Biologically Inspired Computing, pp. 1309-1314, 2009.

Ventresca, M. and Tizhoosh, H.R. Improving the convergence of back-propagation by opposite transfer functions. Proc. of IEEE World Congress on Computation Intelligence, pp. 9527-9534, 2006.

Viswanathan, G.M., Afanasyev, V. and Buldyrev, S.V. Levy flight search patterns of wandering albatrosses. Nature 381, 1996.

Wedde, H. F., Farooq, M. and Zhang. Y. BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. Ant colony optimization and swarm intelligence. Springer Berlin Heidelberg, pp. 83-94, 2004.

Wilson E. O. The insect societies . Cambridge, Harvard University Press, 1971.

Xiang, W. L. and An, M. Q. An efficient and robust artificial bee colony algorithm for numerical optimization. Computers & operations research 40, pp. 1256-1265, 2013.

Yang, X. S, and Deb, S. Cuckoo search via L*é*vy flights. Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on IEEE, 2009.

Yang, X. S. Firefly algorithm, Levy flights and global optimization. Research and development in intelligent systems XXVI. Springer London, pp. 209-218, 2010a.

Yang, X. S. Engineering optimization: an introduction with metaheuristic applications. New Jersey: John Wiley and Sons, pp. 153-161, 2010.

Yang, X. S. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer Berlin Heidelberg, pp. 65-74, 2010.

Yang, X. S. and Koziel, S. Computational optimization and applications in engineering and industry. Springer, Germany, 2011.

Yang, X.S. and Deb, S. Multiobjective cuckoo search for design optimization. Computers & Operations Research 40(6), pp. 1616-1624, 2013.

Yashesh, D., Deb, K., and Bandaru, S. Non-uniform mapping in real-coded genetic algorithms. 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, pp. 2237-2244, 2014.

Yeh, W.C., Hsieh, T.J. Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Computers & Operations Research 38(11), pp. 1465-1473, 2011.

Yin, M.H., Li, X.T., Zhou, J.P. An efficient job shop scheduling algorithm based on artificial bee colony. Scientific Research and Essays 6(12), pp. 2578-2596, 2011.

Yun, C. Artificial fish school algorithm applied in a combinatorial optimization problem. International Journal of Intelligent Systems and Applications (IJISA) 1, pp. 37-43, 2010.

Zhang, J., Sanderson, A. C. Jade: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13(5), pp.945-958, 2009.

Zhu, G., Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation 217, pp. 3166-3173, 2010.

Zou, W. P. et al. Clustering approach based on von neumann topology artificial bee colony algorithm. International Conference on Data Mining (DMIN'11), 2011.

# Appendix A

# CEC'13 Test Functions

$M_1, M_2, ..., M_{10}$: orthogonal (rotation) matrix generated from standard normally distributed entries by Gram-Schmidt orthonormalization. $\Lambda^{\alpha}$: a diagonal matrix in $D$ dimensions with the $i$th diagonal element as $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}$, $i = 1, 2, ..., D$.

$$T_{asy}^{\beta} = x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}} \qquad if \quad x_i > 0 \tag{A.1}$$

$$T_{osz} = sign(x_i)exp(\hat{x}_i + 0.049(sin(c_1\hat{x}_i) + sin(c_2\hat{x}_i))) \tag{A.2}$$

Where, $\hat{x}_i = \begin{cases} log(\mid x_i \mid) & if \quad x_i \neq 0 \\ 0 & otherwise \end{cases}$ $signx_i = \begin{cases} -1 & if \quad x_i < 0 \\ 0 & if \quad x_i = 0 \\ 1 & otherwise \end{cases}$

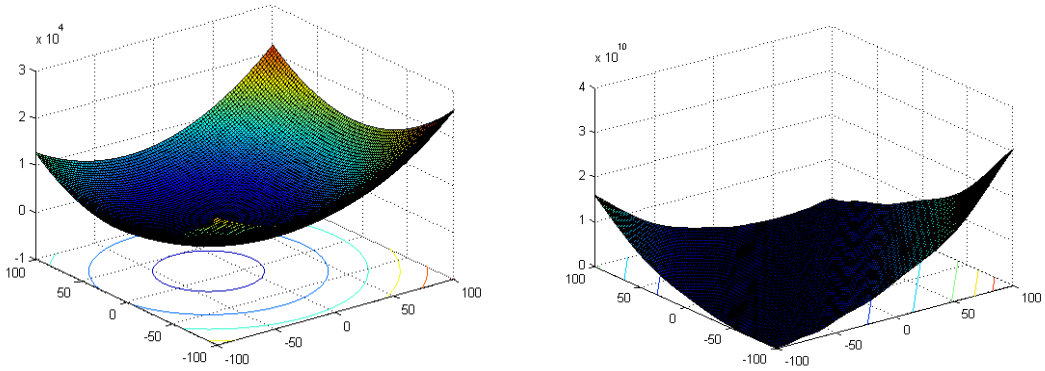
$c_1 = \begin{cases} 10 & if \quad x_i > 0 \\ 5.5 & otherwise \end{cases}$ $c_2 = \begin{cases} 7.9 & if \quad x_i > 0 \\ 3.1 & otherwise \end{cases}$

In above equations, for $i = 1, 2, ..., D$.

**■ Unimodal and Multimodal Functions**

1. Sphere Function

$$F_1(x) = f_1(x - o) + F_1^* \tag{A.3}$$




(a) F1 (b) F2

Figure A.1: 2-dimensional landscapes

2. Rotated High Conditioned Elliptic

$$F_2(x) = f_2(T_{OSZ}(M_1(x - o))) + F_2^* \tag{A.4}$$

3. Rotated Bent Cigar Function

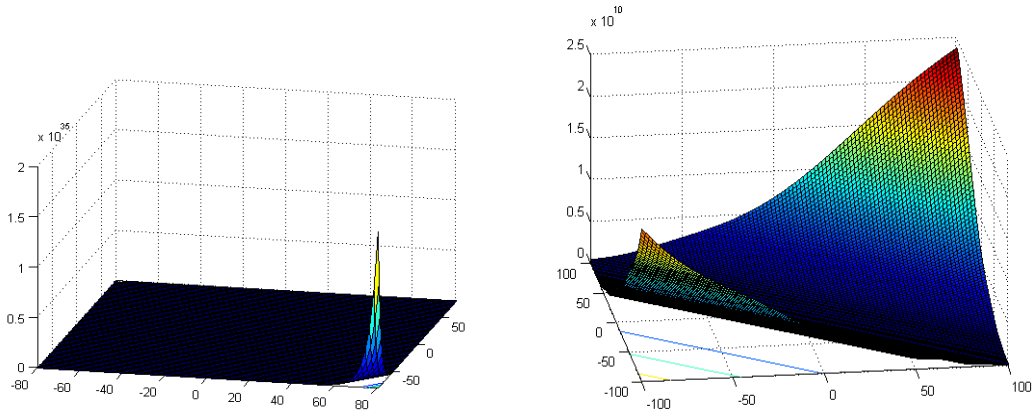
$$F_3(x) = f_3(M_2 T_{asy}^{0.5}(M_1(x - o))) + F_3^* \tag{A.5}$$

4. Rotated Discus Function

$$F_4(x) = f_4(T_{osz}(M_1(x - o))) + F_4^* \tag{A.6}$$

5. Different Powers Function

$$F_5(x) = f_5(x - o) + F_5^* \tag{A.7}$$

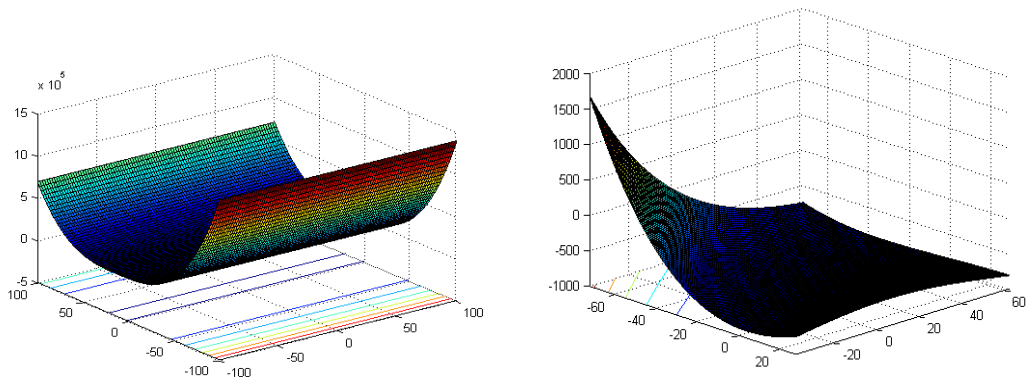



(a) F3 (b) F4

Figure A.2: 2-dimensional landscapes

6. Rotated Rosenbrock's Function

$$F_6(x) = f_6(M_1(\frac{2.048(x-o)}{100}) + 1) + F_6^* \tag{A.8}$$




(a) F5 (b) F6

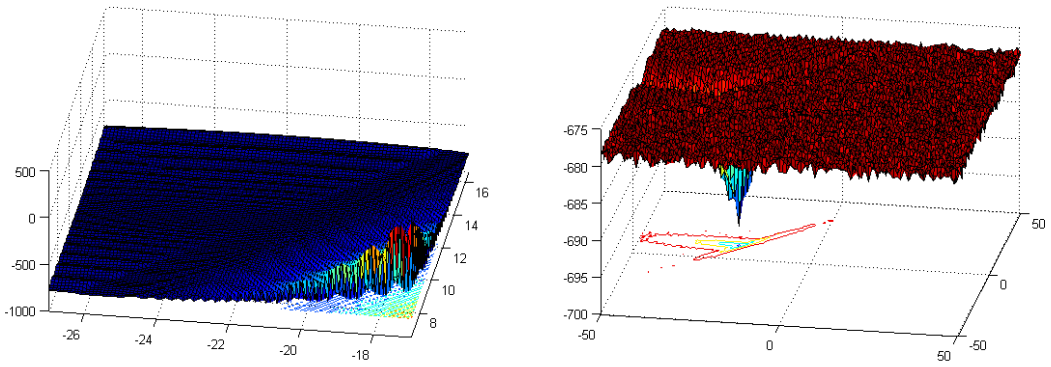
Figure A.3: 2-dimensional landscapes

7. Rotated Schaffers F7 Function

$$F_7(x) = f_7(\wedge^{10} M_2 T_{asy}^{0.5}(M_1(x-o))) + F_7^* \tag{A.9}$$

8. Rotated Ackley's Function

$$F_8(x) = f_8(\wedge^{10} M_2 T_{asy}^{0.5}(M_1(x - o))) + F_8^* \tag{A.10}$$

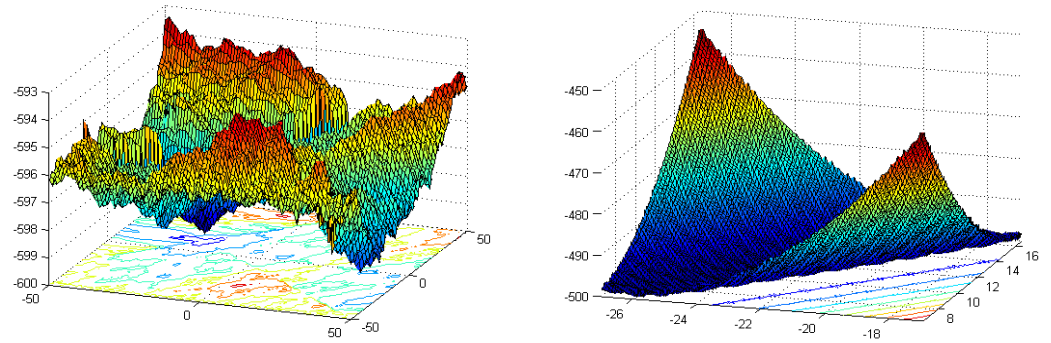



(a) F7 (b) F8

Figure A.4: 2-dimensional landscapes

9. Rotated Weierstrass Function

$$F_9(x) = f_9(\wedge^{10} M_2 T_{asy}^{0.5}(M_1 \frac{0.5(x - o)}{100})) + F_9^* \tag{A.11}$$




(a) F9 (b) F10

Figure A.5: 2-dimensional landscapes

10. Rotated Griewank's Function

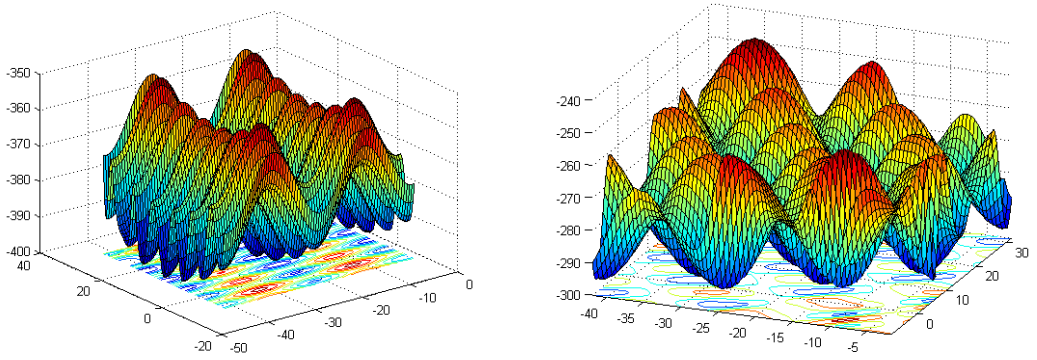
$$F_{10}(x) = f_{10}(wedge^{100} M_1 \frac{600(x-o)}{100}) + F^*_{10} \tag{A.12}$$

11. Rastrigin's Function

$$F_{11}(x) = f_{11}(\wedge^{10} T^{0.2}_{asy}(T_{osz}(\frac{5.12(x-o)}{100}))) + F^*_{11} \tag{A.13}$$

12. Rotated Rastrigin's Function

$$F_{12}(x) = f_{11}(M_1 \wedge^{10} M_2 T^{0.2}_{asy}(T_{osz}(M_1 \frac{5.12(x-o)}{100}))) + F^*_{12} \tag{A.14}$$




(a) F11 (b) F12

Figure A.6: 2-dimensional landscapes

13. Non-continuous Rotated Rastrigin's Function

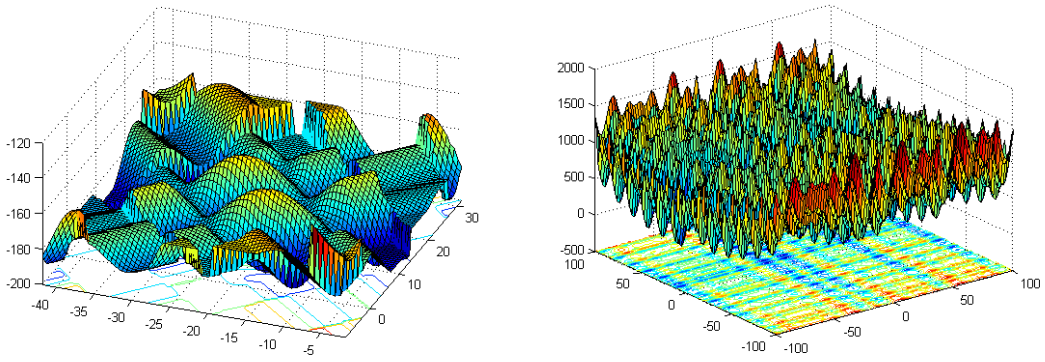
$$F_{13}(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + F^*_{13} \tag{A.15}$$

Where, $\hat{x} = M_1 \frac{5.12(x-o)}{100}$, $y_i = \begin{cases} \hat{x}_i, & if \quad |\hat{x}_i| \leq 0.5 \\ round(2\hat{x}_i)/2, & if \quad |\hat{x}_i| > 0.5 \end{cases}$ for $i$=1, 2,..., $D$

$z = M_1 \wedge^{10} M_2 T^{0.2}_{asy}(T_{osz}(y))$

14. Schwefel's Function

$$F_{14}(x) = f_{12}(\wedge^{10}(\frac{1000(x-o)}{100}) + 4.209687462275036e + 002) + F^*_{14} \quad (A.16)$$




(a) F13 (b) F14

Figure A.7: 2-dimensional landscapes

15. Rotated Schwefel's Function

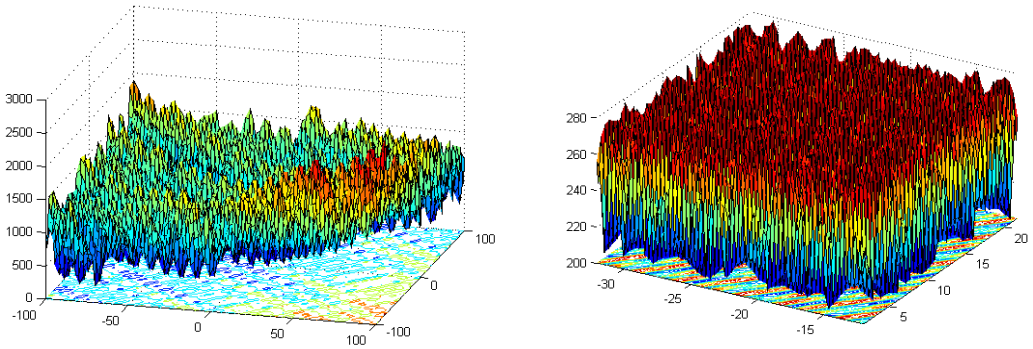
$$F_{15}(x) = f_{12}(\wedge^{10} M_1(\frac{1000(x-o)}{100}) + 4.209687462275036e + 002) + F^*_{15} \quad (A.17)$$

16. Rotated Katsuura Function

$$F_{16}(x) = f_{13}(M_2 \wedge^{100} (M_1 \frac{5(x-o)}{100})) + F^*_{16} \quad (A.18)$$

17. Lunacek bi-Rastrigin Function

$$F_{17}(x) = \min(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2) + 10(D - \sum_{i=1}^{D} cos(2\pi z_i)) + F^*_{17} \quad (A.19)$$




(a) F15 (b) F16

Figure A.8: 2-dimensional landscapes

Where, $\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d = 1,$

$\hat{x}_i = 2sign(x_i^*)y_i + \mu_0, y = \frac{10(x-o)}{100}, z = \wedge^{100}(\hat{x} - \mu_0),$ for $i = 1, 2, ..., D$

18. Rotated Lunacek bi-Rastrigin Function

$$F_{18}(x) = \min(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2) + 10(D - \sum_{i=1}^{D} cos(2\pi z_i)) + F_{18}^* \quad \text{(A.20)}$$

Where, $\mu_0, \mu_1, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d, \hat{x}_i, y$ are same as defined in $F_{17}(x)$.

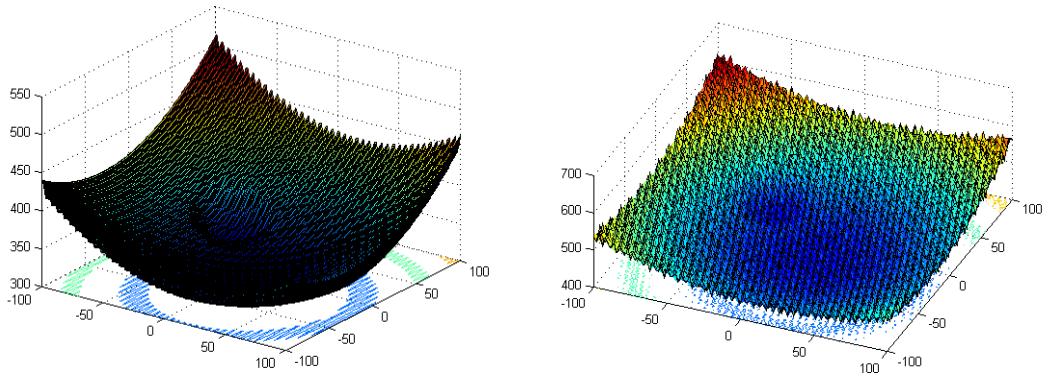
$z = M_2 \wedge^{100} (M_1\hat{x} - \mu_0)$

19. Rotated Expanded Griewank's plus Rosenbrock's Function

$$F_{19}(x) = f_{14}(M_1(\frac{5(x-o)}{100}) + 1) + F_{19}^* \quad \text{(A.21)}$$

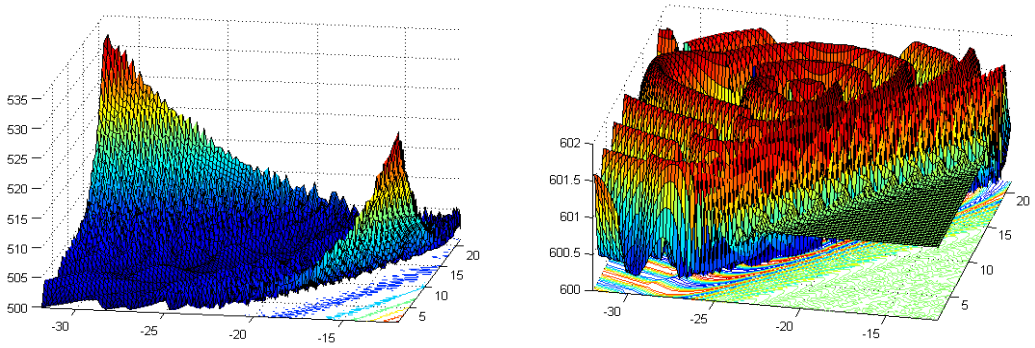
20. Rotated Expanded Scaffer's F6 Function

$$F_{20}(x) = f_{15}(M_2 T_{asy}^{0.5}(M_1(x - o))) + F_{20}^* \quad \text{(A.22)}$$




(a) F17 (b) F18

Figure A.9: 2-dimensional landscapes




(a) F19 (b) F20

Figure A.10: 2-dimensional landscapes

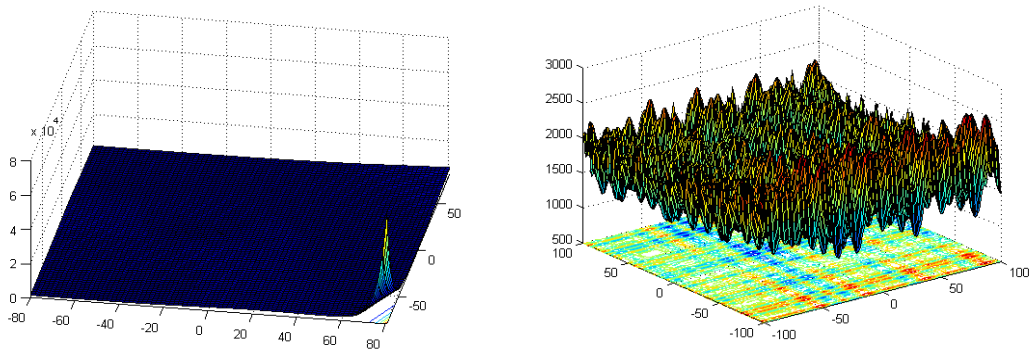
■ **Composition Functions**

21. Composition Function 1

    n=5, $\omega$=[10, 20, 30, 40, 50], bias=[0, 100, 200, 300, 400], $\lambda$=[1, 1e-6, 1e-26, 1e-6, 0.1], $g_1$: Rotated Rosenbrock, $g_2$: Rotated Different Powers, $g_3$: Rotated Bent Cigar, $g_4$: Rotated Discus, $g_5$: Sphere

22. Composition Function 2

    n=3, $\omega$=[20, 20, 20], $\lambda$=[1, 1, 1], bias=[0, 100, 200], $g_{1-3}$: Schwefel




(a) F21 (b) F22

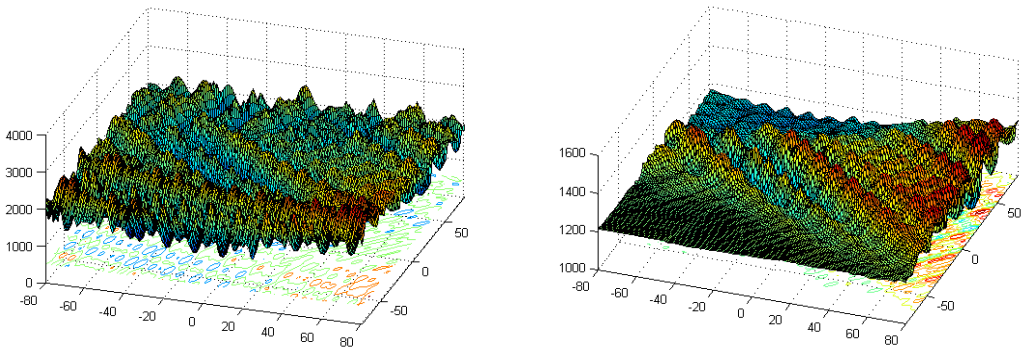
Figure A.11: 2-dimensional landscapes

23. Composition Function 3

    n=3, $\omega$=[20, 20, 20], $\lambda$=[1, 1, 1], bias=[0, 100, 200], $g_{1-3}$: Rotated Schwefel

24. Composition Function 4

    n=3, $\omega$=[20, 20, 20], $\lambda$=[0.25, 1, 2.5], bias=[0, 100, 200], $g_1$: Rotated Schwefel, $g_2$: Rastrigin, $g_3$: Weierstrass




(a) F23 (b) F24

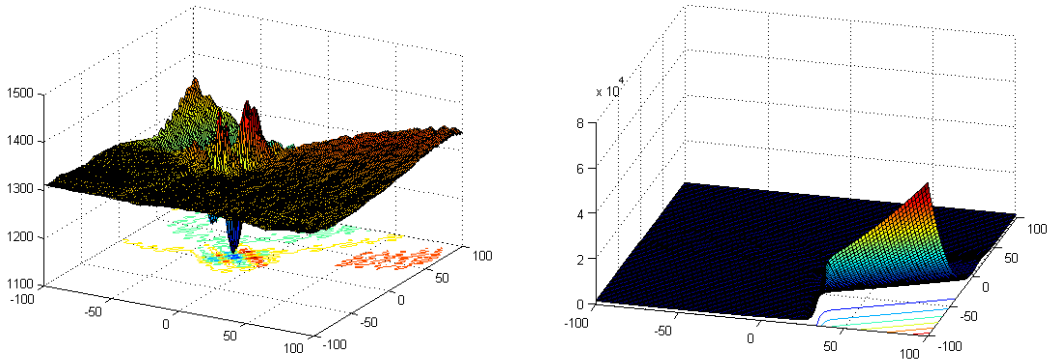
Figure A.12: 2-dimensional landscapes

25. Composition Function 5

    All settings are same as composition function 4, except for $\omega$=[10, 30, 50]

26. Composition Function 6

    n=5, $\sigma$=[10, 10, 10, 10, 10], $\lambda$=[0.25, 1, 1e-7, 2.5, 10], bias=[0, 100, 200, 300, 400], $g_1$: Rotated Schwefel, $g_2$: Rotated Rastrigin, $g_3$: Rotated High Conditioned Elliptic $g_4$: Rotated Weierstrass, $g_5$: Rotated Griewank




(a) F25 (b) F26

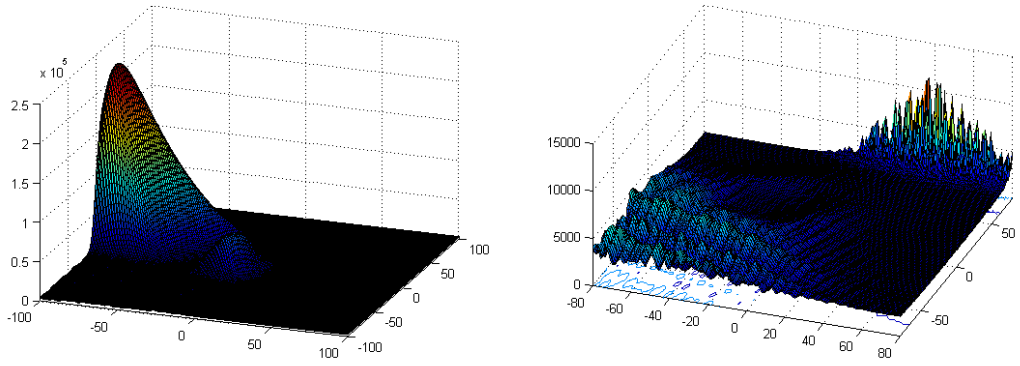
Figure A.13: 2-dimensional landscapes

27. Composition Function 7

    n=5, $\omega$=[10, 10, 10, 20, 20], $\lambda$=[100, 10, 2.5, 25, 0.1], bias=[0, 100, 200, 300, 400], $g_1$: Rotated Griewank, $g_2$: Rotated Rastrigin, $g_3$: Rotated Schwefel, $g_4$: Rotated Weierstrass, $g_5$: Sphere

28. Composition Function 8

    n=5, $\omega$=[10, 20, 30, 40, 50], $\lambda$=[2.5, 2.5e-3, 2.5, 5e-4, 0.1], bias=[0, 100, 200, 300, 400], $g_1$: Rotated Expanded Griewank plus Rosenbrock, $g_2$: Rotated Schaffers $F_7$, $g_3$: Rotated Schwefel, $g_4$: Rotated Expanded Scaffer's $F_6$, $g_5$: Sphere




(a) F27 (b) F28

Figure A.14: 2-dimensional landscapes

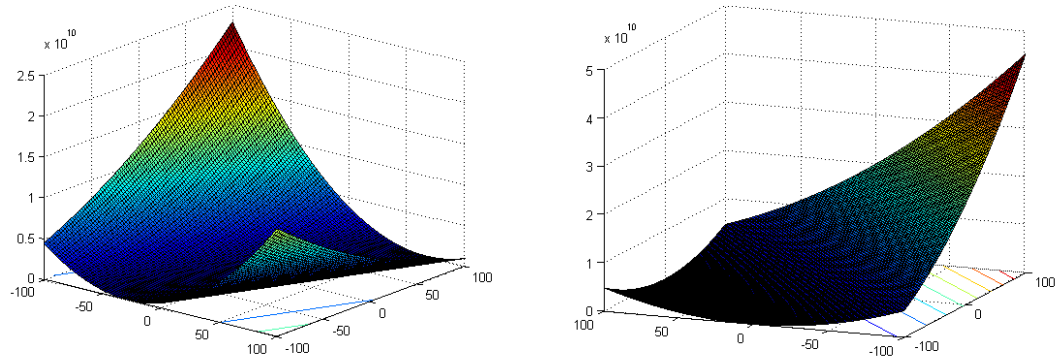
# Appendix B

# CEC'14 Test Functions

### ■ Unimodal Functions

1. Rotated High Conditioned Elliptic Function

$$F_1(x) = f_2(M(x - o_1)) + F_1^* \tag{B.1}$$




(a) F1 (b) F2

Figure B.1: 2-dimensional landscapes

2. Rotated Bent Cigar Function

$$F_2(x) = f_3(M(x - o_2)) + F_2^* \tag{B.2}$$

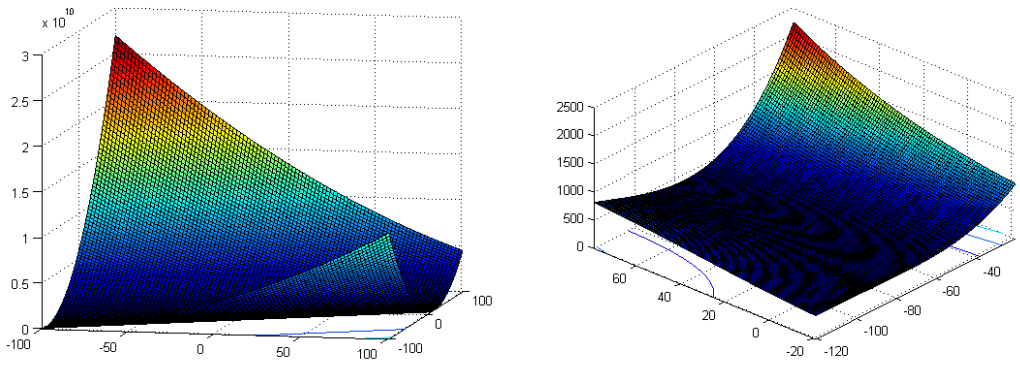
3. Rotated Discus Function

$$F_3(x) = f_4(M(x - o_3)) + F_3^* \tag{B.3}$$

■ **Multimodal Functions**

4. Shifted and Rotated Rosenbrock's Function

$$F_4(x) = f_6(M\frac{2.048(x_i - o_4)}{100}) + F_4^* \tag{B.4}$$




(a) F3 (b) F4

Figure B.2: 2-dimensional landscapes

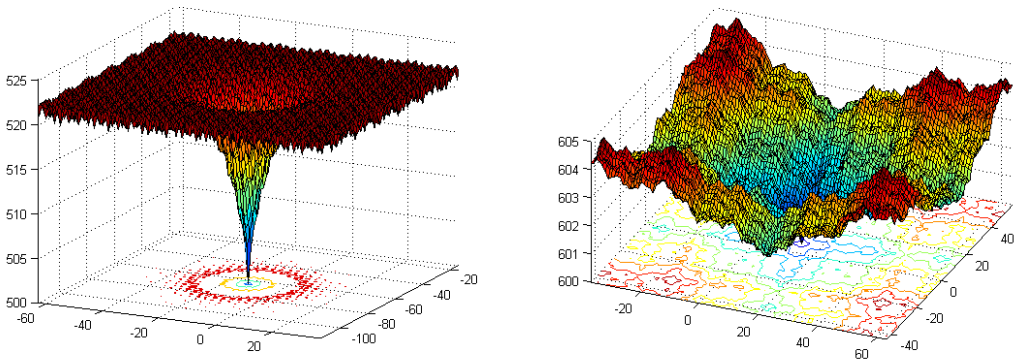
5. Shifted and Rotated Ackley's Function

$$F_5(x) = f_8(M(x - o_5)) + F_5^* \tag{B.5}$$

6. Shifted and Rotated Weierstrass Function

$$F_6(x) = f_9(M\frac{0.5(x - o_6)}{100}) + F_6^* \tag{B.6}$$

7. Shifted and Rotated Griewank's Function



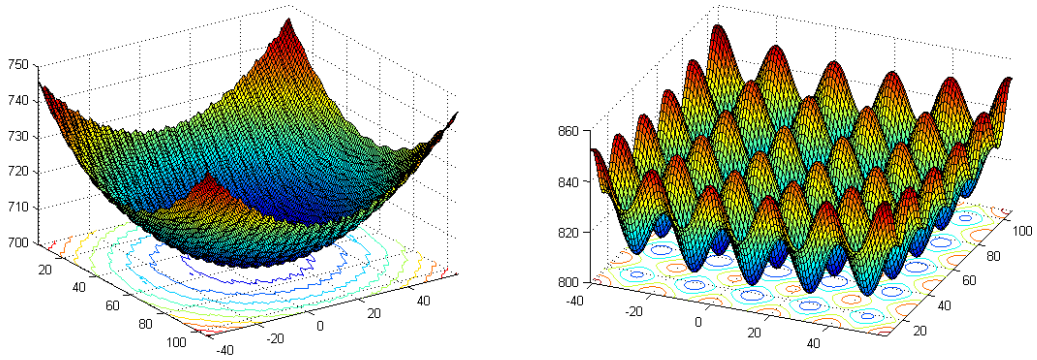

(a) F5 (b) F6

Figure B.3: 2-dimensional landscapes

$$F_7(x) = f_{10}(M(\frac{600(x - o_7)}{100})) + F_7^* \tag{B.7}$$

8. Shifted Rastrigin's Function

$$F_8(x) = f_{11}(\frac{5.12(x - o_8)}{100}) + F_8^* \tag{B.8}$$

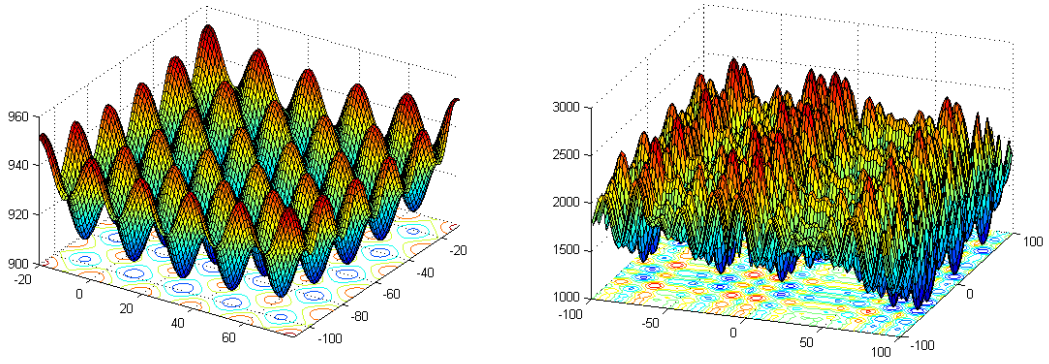



(a) F7 (b) F8

Figure B.4: 2-dimensional landscapes

9. Shifted and Rotated Rastrigin's Function

$$F_9(x) = f_{11}(M(\frac{5.12(x - o_8)}{100})) + F_9^* \tag{B.9}$$




(a) F9 (b) F10

Figure B.5: 2-dimensional landscapes

10. Shifted Schwefel's Function

$$F_{10}(x) = f_{12}(frac1000(x - o_{10})100) + F_{10}^* \tag{B.10}$$

11. Shifted and Rotated Schwefel's Function

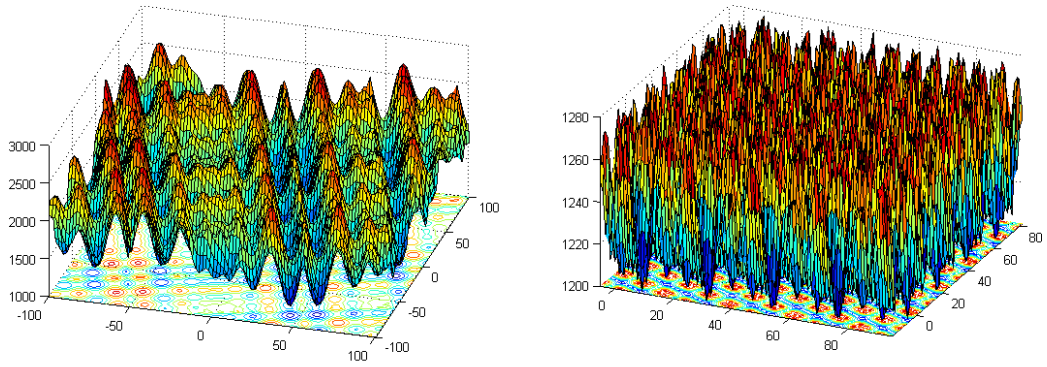
$$F_{11}(x) = f_{12}(M(\frac{1000(x - o_{11})}{100})) + F_{11}^* \tag{B.11}$$

12. Shifted and Rotated Katsuura Function

$$F_{12}(x) = f_{13}(M(\frac{5(x - o_{12})}{100})) + F_{12}^* \tag{B.12}$$

13. Shifted and Rotated HappyCat Function

$$F_{13}(x) = f_{16}(M(\frac{5(x - o_{13})}{100})) + F_{13}^* \tag{B.13}$$

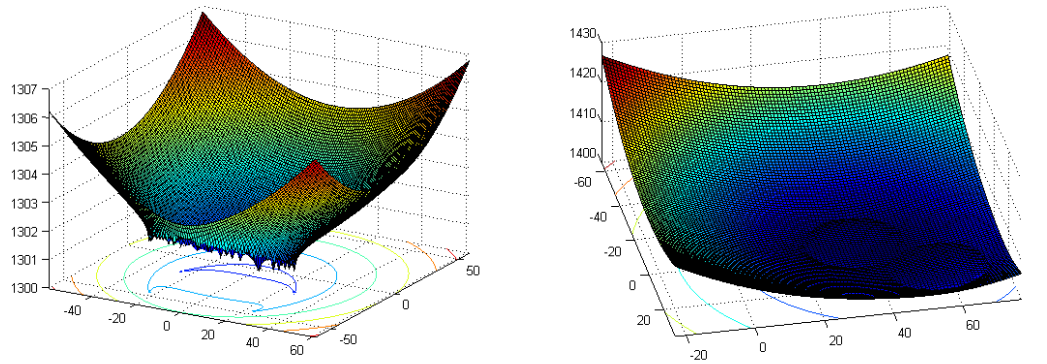



(a) F11 (b) F12

Figure B.6: 2-dimensional landscapes

14. Shifted and Rotated HGBat Function

$$F_{14}(x) = f_{17}(M(\frac{5(x - o_{14})}{100})) + F^*_{14} \tag{B.14}$$




(a) F13 (b) F14

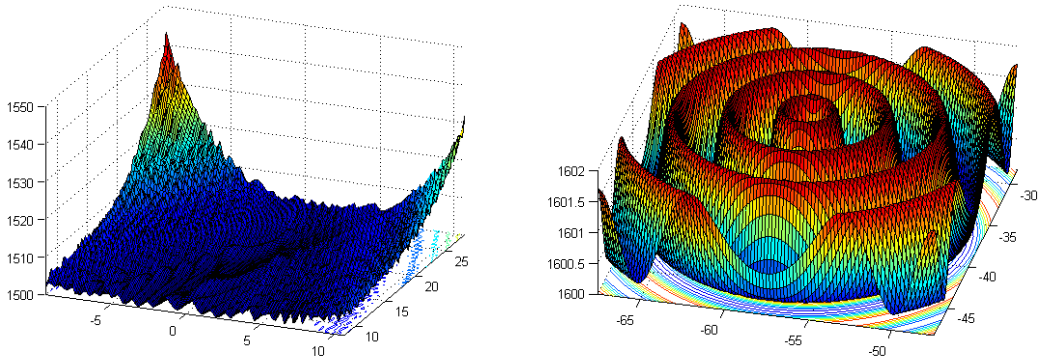
Figure B.7: 2-dimensional landscapes

15. Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function

$$F_{15}(x) = f_{14}(M(\frac{5(x - o_{15})}{100}) + F^*_{15} \tag{B.15}$$

16. Shifted and Rotated Expanded Scaffer's F6 Function

$$F_{16}(x) = f_{15}(M(x - o_{16})) + F^*_{16} \tag{B.16}$$




(a) F15 (b) F16

Figure B.8: 2-dimensional landscapes

■ **Hybrid Functions**

17. Hybrid Function 1

    N=3, p=[0.3, 0.3, 0.4], $g_1$: Modified Schwefel, $g_2$: Rastrigin, $g_3$: High Conditioned Elliptic

18. Hybrid Function 2

    N=3, p = [0.3, 0.3, 0.4], $g_1$ : Bent Cigar, $g_2$: HGBat, $g_3$: Rastrigin

19. Hybrid Function 3

    N = 4, p = [ 0.2, 0.2, 0.3, 0.3], $g_1$: Griewank, $g_2$: Weierstrass, $g_3$: Rosenbrock, $g_4$: Scaffer's F6

20. Hybrid Function 4

    N = 4, p = [0.2, 0.2, 0.3, 0.3], $g_1$: HGBat, $g_2$: Discus, $g_3$: Expanded Griewank plus Rosenbrock, $g_4$: Rastrigin

21. Hybrid Function 5

    N = 5, p = [0.1, 0.2, 0.2, 0.2, 0.3], $g_1$: Scaffer, $g_2$: HGBat, $g_3$: Rosenbrock, $g_4$: Modified Schwefel, $g_5$: High Conditioned Elliptic

22. Hybrid Function 6

    N = 5, p = [0.1, 0.2, 0.2, 0.2, 0.3], $g_1$: Katsuura, $g_2$: HappyCat, $g_3$: Expanded Griewank plus Rosenbrock, $g_4$: Modified Schwefel, $g_5$: Ackley

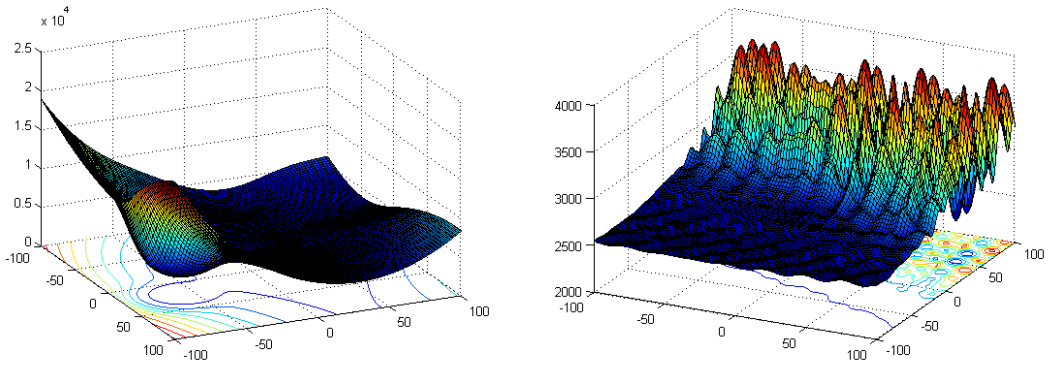
**■ Composition Functions**

23. Composition Function 1

    n=5, $\omega$=[10, 20, 30, 40, 50], bias=[0, 100, 200, 300, 400], $\lambda$=[1, 1e-6, 1e-26, 1e-6, 1e-6], $g_1$: Rotated Rosenbrock, $g_2$: High Conditioned Elliptic, $g_3$: Rotated Bent Cigar, $g_4$: Rotated Discus, $g_5$: High Conditioned Elliptic

24. Composition Function 2

    n=3, $\omega$=[20, 20, 20], $\lambda$=[1, 1, 1], bias=[0, 100, 200], $g_1$: Schwefel, $g_2$: Rotated Rastrigin, $g_3$: Rotated HGBat




(a) F23 (b) F24

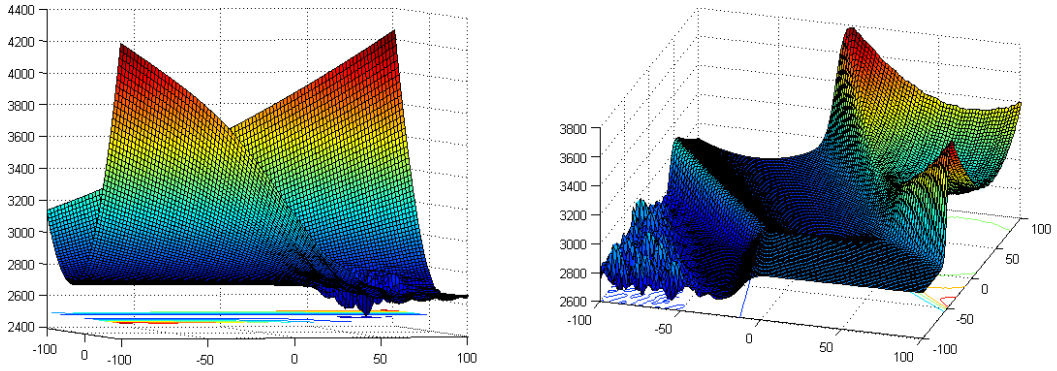
Figure B.9: 2-dimensional landscapes

25. Composition Function 3

n=3, $\omega$=[10, 30, 50], $\lambda$=[0.25, 1, 1e-7], bias=[0, 100, 200], $g_1$: Rotated Schwefel, $g_2$: Rotated Rastrigin, $g_3$: Rotated High Conditioned Elliptic

26. Composition Function 4

    n=5, $\omega$=[10, 10, 10, 10, 10], $\lambda$=[0.25, 1, 1e-7, 2.5, 10], bias=[0, 100, 200, 300, 400], $g_1$: Rotated Schwefel, $g_2$: Rotated HappyCat, $g_3$: Rotated High Conditioned Elliptic, $g_4$: Weierstrass, $g_5$: Rotated Griewank




(a) F25 (b) F26

Figure B.10: 2-dimensional landscapes

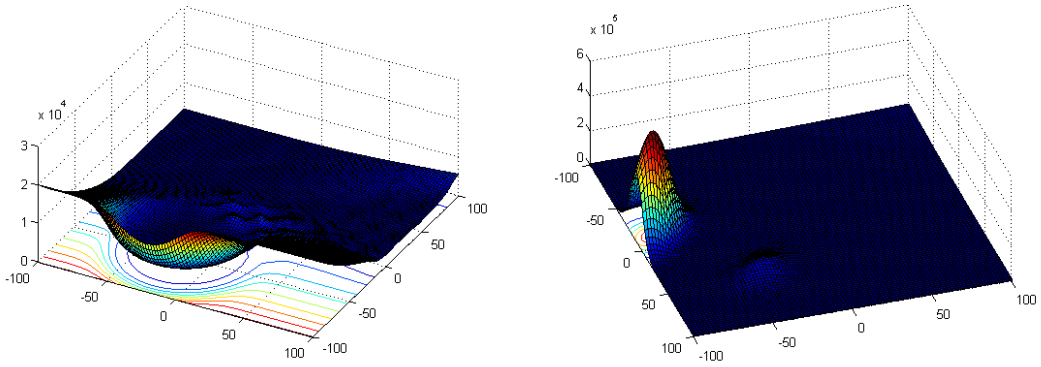
27. Composition Function 5

    n=5, $\sigma$=[10, 10, 10, 20, 20], $\lambda$=[10, 10, 2.5, 2.5, 1e-6], bias=[0, 100, 200, 300, 400], $g_1$: Rotated HGBat, $g_2$: Rotated Rastrigin, $g_3$: Rotated Schwefel, $g_4$: Rotated Weierstrass, $g_5$: Rotated High Conditioned Elliptic

28. Composition Function 6

    n=5, $\sigma$=[10, 20, 30, 40, 50], $\lambda$=[2.5, 10, 2.5, 5e-4, 1e-6], bias=[0, 100, 200, 300, 400], $g_1$: Rotated Expanded Griewank plus Rosenbrock, $g_2$: Rotated HappyCat, $g_3$: Rotated Schwefel, $g_4$: Rotated Expanded Scaffer, $g_5$: Rotated High Conditioned Elliptic

29. Composition Function 7




(a) F27 (b) F28

Figure B.11: 2-dimensional landscapes

n=3, $\omega$=[10, 30, 50], $\lambda$=[1, 1, 1], bias=[0, 100, 200], $g_1$: Hybrid Function 1, $g_2$: Hybrid Function 2, $g_3$: Hybrid Function 3

30. Composition Function 8

    n=3, $\omega$=[10, 30, 50], $\lambda$=[1, 1, 1], bias=[0, 100, 200], $g_1$: Hybrid Function 4, $g_2$: Hybrid Function 5, $g_3$: Hybrid Function 6

# Research Achievements

## Journal papers

1. Shan, H., Yasuda, T., Ohkura, K., "A Self Adaptive Hybrid Enhanced Artificial Bee Colony Algorithm for Continuous Optimization Problems", BioSystems, Vol. 132-133, pp. 43-53, 2015.

2. Shan, H., Yasuda, T., Ohkura, K., "An Improved Hybrid Artificial Bee Colony Algorithm for Solving Real Parameter Optimization Problems", International Journal of Engineering Research & Technology 4 (5), pp. 628-635, 2015.

## International Conferences

1. Shan, H., Yasuda, T., Ohkura, K., "A Self Adaptive Hybrid Artificial Bee Colony Algorithm for Solving CEC 2013 Real-parameter Optimization Problems", 2013 IEEE/SICE International Symposium on System Integration, pp. 706-711, 2013.

2. Shan, H., Yasuda, T., Ohkura, K., "A Levy Flight-Based Hybrid Artificial Bee Colony Algorithm for Solving Numerical Optimization Problems", 2014 IEEE Congress on Evolutionary Computation, pp. 2656-2663, 2014.