Nature Inspired Computing: Algorithms, Performance and Applications

Submitted in partial fulfillment of the requirements

of the degree of

Doctor of Philosophy

by

Rohit Salgotra

(UID: 901606022)

Supervisor

Dr. Urvinder Singh

(Associate Professor)

Department of Electronics & Communication Engineering

THAPAR INSTITUTE OF ENGG. & TECHNOLOGY

July 2020

Dedicated to

The Grand Weaver of my life,

MY PARENTS

Approval of Doctoral Committee

Certified that the thesis entitled **"Nature Inspired Computing: Algorithms, Performance and Applications"** submitted by **Mr. Rohit Salgotra** to the *Thapar Institute of Engineering & Technology, Patiala* for the award of **"Doctor of Philosophy"** has been accepted by the doctoral committee members.

Dr. Paramartha Dutta
External Examiner
Department of Computer & system Sciences
Visva Bharati University, West Bengal

Dr. Urvinder Singh
Supervisor
Dept. of Electronics & Communication
Thapar Institute of Engg. & Tech, Patiala

Dr. Alpana Aggarwal
Chairperson, Doctoral Committee
Dept. of Electronics & Communication
Thapar Institute of Engg. & Tech, Patiala

Dr. Kulbir Singh
Member, Doctoral Committee
Dept. of Electronics & Communication
Thapar Institute of Engg. & Tech, Patiala

Dr. Neeraj Kumar
Member, Doctoral Committee
Dept. of Computer Science & Engg.
Thapar Institute of Engg. & Tech, Patiala

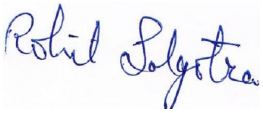
Dr. Vinay Kumar
Member, Doctoral Committee
Dept. of Electronics & Communication
Thapar Institute of Engg. & Tech, Patiala

Date: July 2020
Place: Patiala, India

Statement of Originality

This is to certify that:

- All the written material submitted in this thesis is my original work and any idea or words which have been added are adequately cited and referenced from the original source.
- The work presented in this thesis has never been used for any other degree or diploma course and I adhere to academic integrity and honesty and have not falsified or misrepresented or fabricated any data or fact or idea or source in my submission.
- I adhere to the norms and guidelines given in the Ethical Code of Conduct of the Institute and is liable for disciplinary actions from the sources which have thus not been properly cited or whose permission has not been taken.
- This this does not contain any studies with animal or human subjects.
- I declare that I have no conflict of interest.



Rohit Salgotra

UID: 901606022

Date: July 2020

Place: Patiala, India

Supervisor Declaration Statement

Certified that the thesis entitled **"Nature Inspired Computing: Algorithms, Performance and Applications"** submitted by **Mr. Rohit Salgotra** to the *Thapar Institute of Engineering & Technology, Patiala* for the award of **"Doctor of Philosophy"** has been thoroughly examined. I have reviewed the content and presentation style of this thesis and declare that it is free from plagiarism and grammatical errors. The research work stated in this thesis is the original work of the candidate. I confirm that the investigations have been conducted with the ethical policies of the institute and research is presented without prejudice.

Dr. Urvinder Singh
Associate Professor
Department of Electronics & Communication,
Thapar Institute of Engineering & Technology
Patiala - 147004
INDIA

Abstract

Nature Inspired algorithms have served as the backbone of modern computing technology and over the past three decades, the field has grown enormously. A large number of applications have been solved by these algorithms and are replacing the traditional classical optimization algorithms. In this thesis, some of these nature inspired algorithms such as cuckoo search algorithm (CS), flower pollination algorithm (FPA) and others have been studied. All these algorithms are state-of-the-art algorithms and have proven their worth in terms of competitiveness and application to various domains of research. The aim is to develop new improved algorithms through mitigating well-known problems that these algorithms suffer from, such as local optima stagnation, poor exploration, slow convergence and parametric complexity. Such improvements should help these new variants to solve highly challenging optimization problems in contrast to existing algorithms. Different ideas and logic are employed in designing such new versions such as hybridization that combine the strength of different mutation strategies to add diversity in the solution space, adaptive parameter adaptations to converge faster, improved global and local search strategy to enhance the exploration and exploitation respectively. Also self-adaptivity, population size reduction and lower computational complexity methods have been analysed to provide prospective algorithms for the next generation researchers. Apart from these, based on the mating patterns of naked mole-rat, a new algorithm namely naked mole-rat algorithm (NMR) was proposed. To validate the performance of all these developed algorithms, various challenging test suites from the IEEE-CEC benchmarks are used. Each of these benchmarks constitute problems of different characteristics such as ruggedness, multimodality, noise in fitness, ill-conditioning, non-separability and interdependence. Moreover, various real-world optimization problems from diversified fields such as Antenna arrays, frequency modulation, stirred tank reactor and others are also used. The results of comparative study and statistical tests affirm the superior and efficient performance of proposed algorithms.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Preface

This chapter presents the basic ideas behind the proposal of this thesis. First of all, motivation from the previous work are drawn and based on those motivations, new objectives are formulated. The second step is to define the major contribution of the thesis. This section details about the various proposed algorithms and their application to some real-world problems. In the final section a detailed organization of each of the chapter of the thesis is presented.

1.1 Motivation

In present world, there are numerous real-world applications of optimization. Nowadays, many scientific fields and industrial companies face a lot of hurdles in finding an appropriate optimization tool or more precisely an optimization algorithm to solve many real-world problems. The major reason for the use of these algorithms is that majority of the real-world applications can be formulated as potential domain optimization problems depending upon their nature and number of objectives to be optimized. This can be estimated from the fact that these algorithms find application in almost every domain of research such as economics, engineering, mathematics, signal processing, weather forecasting, operation research, management, production planning, scheduling, routing problems, machine learning and others. These optimization problems are highly complex and hence pose a challenge for researchers to solve them in an efficient manner. In recent times, nature inspired algorithms have dominated the various spheres of optimization research and are in use in almost every field.

Nature inspired algorithms are considered as most efficient algorithms for solving various optimization problems. Some of these algorithms include differential evolution (DE) (Storn and Price, 1997), cuckoo search (CS) (Yang and Deb, 2009), flower pollination algorithm (FPA) (Yang, 2012) and others. More details on these algorithms is given in Chapter 2. All these algorithms incorporate four major processes to simulate natural behaviour including initialization, global search, local search and selection. Although these algorithms have demonstrated exceptional performance and competitiveness, many theoretical and experimental analysis prove that they are still dependent on the choice of tuning parameters (Draa, 2015, Karaboga and Basturk, 2008). Such parameters include scaling factor, switch probability, crossover/mutation rate, local/global search, population size and standard randomization. Early investigation prove that these algorithms require a trial-and-error methodology in order to tune its parameters (Nelder and Mead, 1965). This method is tedious and hence is not efficient. Apart from that, these algorithms suffer from premature convergence, local optima stagnation, slow convergence and others (Draa, 2015). These drawbacks make the algorithms vulnerable to poor solution generation, thus adding to less diversity and hence are unable to find the desired solution. Overall, we can

say that nature inspired algorithms also suffer from problems and more work is required to design new algorithms for prospective researchers.

In this thesis, a substantial effort has been made to eradicate the aforementioned issues by introducing new enhanced versions of nature inspired algorithms by using new and novel strategies. Some major algorithms including CS, FPA and others have been used for designing new algorithms. The major reason for using these algorithms is because of their simplicity, yet powerful search strategy to solve high end optimization problems. Inspired by these observations, the focus is to improve the performance of these algorithms to solve diverse set of complex and challenging optimization problems including real-life problems. For performance evaluation, IEEE CEC benchmarks have been applied by many researchers to test their algorithms (Suganthan *et al.*, 2005, Awad *et al.*, 2016a, 2017, Price *et al.*, 2018). Many variants of algorithms used in this thesis have been successfully used by researchers to solve these problems, and this indicate the usefulness and applicability of nature inspired algorithms to solve different challenging optimization problems. This thesis uses different sets of CEC benchmark suites on single objective real-world numerical optimization to test the proposed algorithms. The major motivation is to design new and prospective algorithms which have the capability to produce better results than those reported in the literature.

1.2 Research Question

The main research problem that need to be addressed is linkage of nature with artificial systems. The rationale of using nature to derive algorithm is that by doing so we will get close to nature and can use the vast knowledge of nature to solve optimization problems at hand. This research would be significant for researchers to have a broader framework of algorithms to study and apply them on high end computational problems in almost every field of research.

1.2.1 Aim & Objectives

The aim of this work is to:

- Study natural species and derive single/multi-objective algorithms from them.
- Design a new algorithm for solving optimization problems.

- Test the algorithm on standard benchmark problems.
- Apply the algorithm to engineering design problems in the field of antenna array optimization.

1.2.2 Importance of proposed work

Limitations of other algorithms: Exploration and exploitation are the two important characteristics of the population (or swarm) based optimization algorithms. The exploration represents the ability to discover the global optimum by investigating the various unknown regions in the solution search space. While, the exploitation represents the ability to find better solutions by implementing the knowledge of the previous good solutions. The intrinsic weakness with most of the stochastic algorithms is premature convergence. For example, artificial bee colony shows poor performance and remains ineffective in exploring the search space. The solution search equation of artificial bee colony is considerably influenced by a random quantity which helps in exploration at the cost of exploitation of the search space (Karaboga and Basturk, 2008). DE has deficiency of premature convergence and stagnation. Also, DE sometimes stops proceeding toward the global optima even though the population has not converged to local optima or any other point (Qin *et al.*, 2008). Particle swarm optimization has the capability to get a good solution at a significantly faster rate but, when compared to other optimization techniques, it is weak to refine the optimum solution, mainly due to less diversity in later search (Xin *et al.*, 2009). Similarly, bio-geography based optimization is good at exploitation but poor at exploration because it involves sharing of features among the solutions (Singh and Kamal, 2012).

Improving the exploration and exploitation of nature inspired algorithms: In the literature there are number of algorithms available which are inspired from nature and many new are being proposed by the researchers. These algorithms mimic the behavior of single specie to solve the optimization problems. But there is little work available which is based on the improvement of two basic properties namely exploration and exploitation of any algorithm. The thesis will mainly focus on designing new algorithm with improved exploration and exploitation. The author aims to achieve this by using random generated solution inside the search space instead of moving all the random solutions toward the

best particles. Also some new mathematical operator can be added instead of the scaling factors to stabilize the exploration process. For exploitation, it is required that the random solutions should follow the local group leader or simply the current best solution. So here the general equation of both exploration and exploitation phases can be enhanced to achieve better results. Further one more parameter that defines the efficiency of an algorithm is the balance between exploration and exploitation. Different algorithms use different operators to define this condition. For example, in case of CS, switch probability is used to control this. For most of the algorithms, this probability is kept constant. So here adaptive probability can be used to increase the performance of any algorithm.

Fast and more accurate algorithm for optimization purposes: Already proposed algorithms have the disadvantage of getting trapped in local minima and due to this, they are slow and less reliable in their functioning. Since the algorithm will have a balanced exploration and exploitation phase, it will allow the algorithms to update their solutions without getting stuck in some local minima. This will make the algorithm fast, accurate and more reliable for higher dimensional problems.

Generalized algorithm for all engineering problems: Most of the already proposed algorithms work for one problem while no fit solution is provided for other. The algorithm though will be designed with a focus on electromagnetic community but due to its accuracy and fast response is expected to provide better results for almost all engineering design problems.

Significance/contribution to Discipline: We have already studied CS, FPA and others. All these state of art algorithms have proven their worth for solving real world problems and have been applied extensively to almost every branch of research. These algorithms though perform significantly better but have some inherent drawbacks as discussed above. A lot of hybrid and improved versions of these algorithms have also been proposed to eradicate the drawbacks of basic algorithms. We worked with enhancement of these algorithms and test their performance on numerical and real-world optimization problems. The citation details are given in List of Publications.

1.3 Major Contribution of the Thesis

The major interest is to enhance the performance of a variety of nature inspired algorithms to solve various optimization problem, including real-world problems. To achieve this, numerous different ideas and novel techniques are introduced and incorporated with basic algorithms under consideration. The first tactic is to enhance the performance of these algorithms by incorporating division of population and generations. By dividing the generations, half of the generations use a standard equation as per the original algorithms, having best exploration properties and for the second half of the generations, more intensive exploitation must be followed. This will help in maintaining a balance between exploration and exploitation. For population division, we allocate the population into sub-groups and multiple search equations are used in each of the sub-groups. It will help the members of population to change their positions abruptly in the initial stages and converge faster as the iterations proceed.

The second tactic is the addition of mutation operators in the algorithms. Adding mutation operators helps the algorithm in exploring the search space in a much better way. These mutation operators are added instead of simple random variable and constitute major adaptation in the global search phase of the proposed algorithms. The third tactic is the introduction of new adaptive strategies to control the parameters of the proposed algorithms. This adaptive mechanism if designed appropriately helps in updating the parameters with respect to iterations. They have shown better performance and are reliable in terms of convergence when compared to deterministic methods. In addition to that, adaptive parameters help the algorithm in getting out of local optima stagnation and improving the solution quality. Also hybridization of two different algorithms to design a new algorithm has also been carried out to effectively solve the problem under test.

Apart from that, a new swarm intelligent species namely naked mole-rat has been analysed and based on their mating patterns, a new nature inspired algorithm has been proposed. The algorithm is developed by using biological concepts of naked mole-rat species and formulation of the same to mathematical equations and models to design the new naked mole-rat algorithm. All the above said modifications are subjected to numerical and real-world optimization problems.

To summarize, the major contributions of the thesis are as follow:

- A new concept of division of population and generation along with Cauchy based mutation is proposed in Chapter 3. The CV 1.0 algorithm proposed in Chapter 3 is further enhanced by using adaptive probability to propose the new CVnew algorithm in chapter 4. The algorithm proposed in chapter 4 is further enhanced and CSsin algorithm is proposed in chapter 5 by employing the concepts of population size reduction. These algorithms constitute the first of their kind in which concept of division of population and generation is followed.

- The concept of self-adaptivity is added to CSsin algorithm in Chapter 6. This Chapter proposes the new SACS algorithm which is first of its kind that employs linearly decreasing population adaptation and is able to solve real-world optimization problems.

- Based on the concept of mutation operators namely Cauchy mutation, Gaussian mutation, Adaptive Lévy mutation, mean and combined mean mutation, five new algorithms are proposed in chapter 7.

- The concept of hybrid and adaptive algorithms is added in chapter 8 and chapter 9 to propose new self-adaptive variants. Further, it should be noted that algorithms introduced in Chapter 8 and chapter 9 are highly efficient and have proven their worth on real-world optimization of linear antenna arrays.

- The mating patterns of naked mole-rats is analysed in Chapter 10 and based on the same naked mole-rat algorithm is proposed in the same chapter.

In this thesis, to verify the proposed algorithms, the most recent studies and state-of-the-art algorithms have been used for performance evaluation. Different mathematical and statistical tests have been done to judge the superior performance and effectiveness of the new algorithms.

1.4 Organization of the Thesis

This section details about the organization of the thesis. Apart from the introduction in **Chapter 1**, this thesis consists of 11 chapters. These chapters are divided into three

parts: improved CS algorithms applied to numerical benchmarks, are proposed in the first part whereas in the second part new improved algorithms are tested on numerical as well as real-world optimization applications. Each chapter consists of four sections. In the first section, basic introduction and background of the chapter is discussed. In the second section, the new proposed algorithm is elaborated whereas in the third section, experimental results in terms of benchmark optimization are presented and in the final section, insightful implications and conclusions are drawn. In some chapters where real-world optimization problems are used, a new section is proposed. **Chapter 2** provides comprehensive literature review on the basics of optimization, the numerical benchmark test suites, real-world optimization problems and a review of recently introduced state-of-the-art algorithms. A detailed overview of rest of the chapters is enumerated as:

Chapter 3 presents three new algorithms namely CV 1.0, CV 2.0 and CV 3.0; and comparison has been performed for variable characteristics of the algorithm.

Chapter 4 deals with the enhancement of CV 1.0 algorithm and a new algorithm namely CVnew algorithm is proposed.

Chapter 5 enhances the performance of CVnew and proposes the CSsin algorithm. This chapter adds the concept of self-adaptivity and population size reduction to the proposed algorithm.

Chapter 6 the concept of self-adaptivity is analysed in a greater depth and is applied to CS algorithm. The new proposed algorithm is named as SACS and is tested on various real world scenarios.

Chapter 7 proposes five new algorithms based on the concepts of mutation operators to FPA. The chapter discusses the existing drawbacks of FPA and further provides viable solutions to mitigate those problems.

Chapter 8 extends the self-adaptive properties added to FPA to another level and EFPA algorithm is proposed. The algorithm is applied to real-world optimization of linear antenna arrays.

Chapter 9 deals with the hybridization of bat algorithm and FPAs. The new algorithm

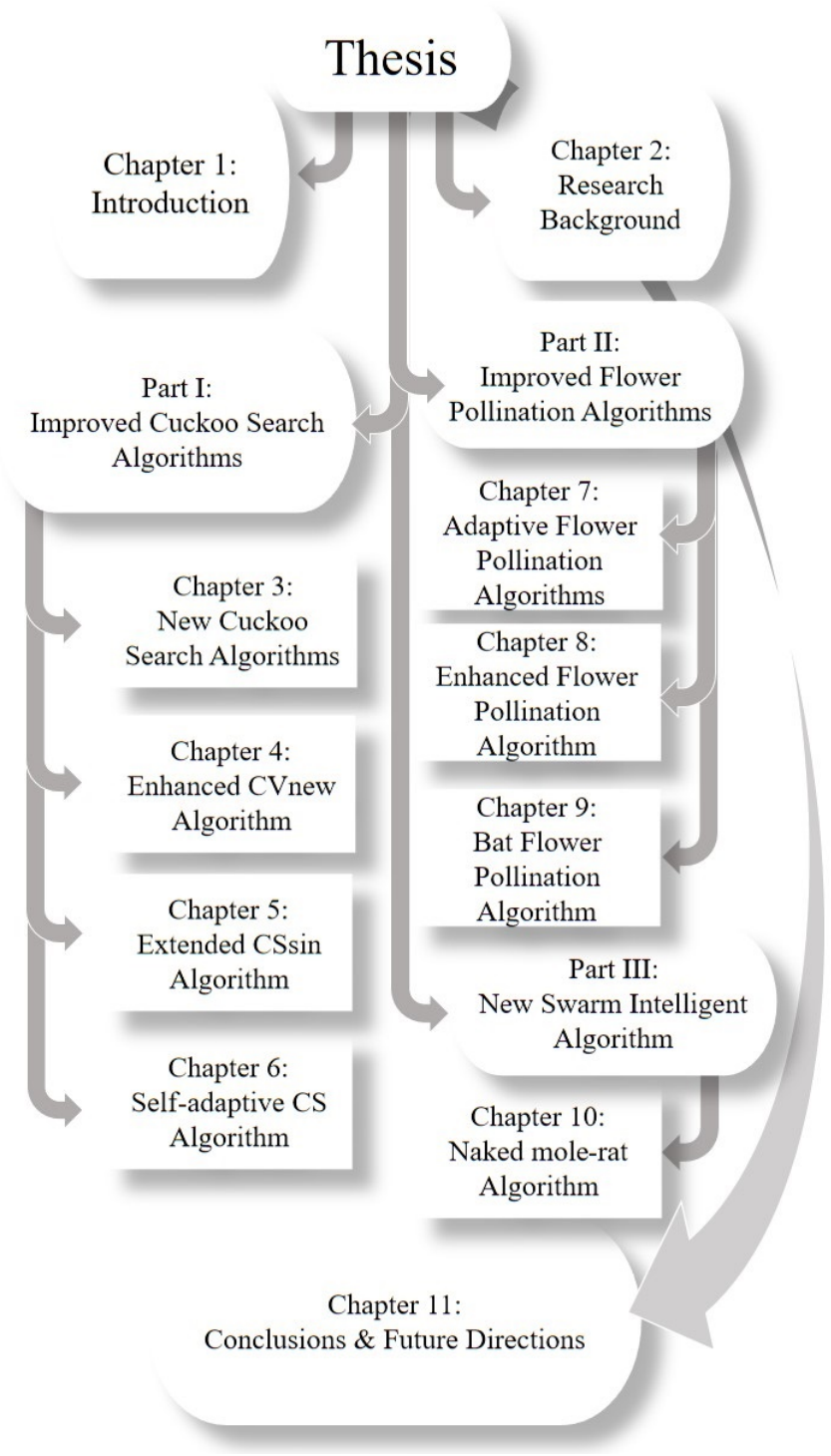


Fig. 1.1 Outline of the Thesis

proposed is BFP and uses the combined strategies of both the algorithms. It has been kept in mind that the computational complexity remains the same for both the algorithms.

Chapter 10 presents a new and novel optimization algorithm for the research community. The new algorithm is based on the mating patterns and eusoical behavior of naked mole-rats and has been named as naked mole-rat algorithm.

The conclusions of the proposed work and insightful implications for future work are given in **Chapter 11**. Detailed outline of the thesis is given in Fig. 1.1.

CHAPTER 2

Research Background and Literature Review

Preface

This chapter presents the basics of numerical optimization and motivation behind the proposal of new techniques. The major goal of this chapter is to provide an overview of the already existing optimization algorithms and their applications to various benchmark data sets and real-world scenarios. Basic details of major benchmark datasets such as CEC 2005, CEC 2015, CEC 2017 and CEC 2020 have also been discussed. All these discussed benchmark datasets have been used in the consecutive chapters for performance evaluation of the proposed techniques. The chapter also provides details on the problems of existing optimization algorithms such as CS and FPA and others. Apart from this, the basic inferences and motivation behind this thesis is also highlighted.

2.1 Numerical Optimization

2.1.1 Basics

Many real-world problems have been formulated as optimization problems to find the best possible solutions. Hence, a large number of researchers have been working in tandem to introduce new effective methods and tools to solve such domain specific optimization problems. In these optimization problems, the objective is to find the global optimal solution for an objective function under certain set of decision variables within a particular range. Without the loss of generality, these optimization problems are represented mathematically either as minimization or maximization problem. The generalized equation for the objective function is formulated as follows:

$$f(\vec{X}), (f : \Omega \subseteq \Re^D \rightarrow \Re) \tag{2.1}$$

where Ω is the problem domain which involves D decision variable or dimension set, each represented by a vector $\vec{X} = [x_1, x_2, x_3, ..., X_D]$. Thus it can be said that for the objective function $f(\vec{X})$, the task of the optimization technique is to search the potential solution $\vec{X}$ where $f(\vec{X}^*) < f(\vec{X}), \forall \vec{X} \in \Omega$ for a minimization problem.

Due to various optimization problems and real-world applications in many related fields, various optimization algorithms have been developed by researchers. Earlier classical methods were found to be effective (Gill and Murray, 1972, Nelder and Mead, 1965, Shor, 1985, Glover, 1986). These classical methods used differential mathematical calculus methods requiring two main features. The first feature is to differentiate the objective function of optimization problem with respect to $\vec{X}$ and secondly the derivative ought to be continuous. These requirements limit the use of classical optimization techniques to certain set of benchmark optimization problems and only particular set of real-world applications. Apart from that, these algorithms easily get trapped into local optima and require more computational time and cost with respect to other algorithms (Shor, 1985, Schramm and Zowe, 1992).

From millions of years, nature has been continuously solving problems and biological systems such as immune systems help to eliminate foreign bodies, perception is used to recognize different patterns, decision making for designing robots, learning process and

others to solve high end problems. These systems show high level computational intelligence present in nature and numerous optimization algorithms have been proposed in this context. The algorithms are known as nature inspired algorithms and are popular in various fields such as business management, stock market for profit or loss prediction, travelling optimal paths and others (Gutjahr, 2009). They are popular because of their flexibility, robustness and faster response when compared to classical optimization methods. One more reason is that they are population based algorithms and don't require an initial guess while solving any problem. Also because of the requirement of lesser number of parameters to tune, they are found suitable even for highly complex domain research problems.

In the past few decades, nature-inspired algorithms have dominated the sphere of optimization research and are being used in almost every field of research. Because of their simple structure, least requirement of parametric and gradient information and simple conceptual modelling, these algorithms are found to provide highly viable solutions. Broadly, these algorithms are classified into two categories namely evolutionary algorithms (EAs) based on the Darwin's theory of natural selection and swarm intelligent (SI) algorithms using the concept of social and swarming behavior of animal species. All EAs and SIs algorithms are stochastic and hence require a certain population for generating an initial solution. The new solutions are generated by operators, like the crossover, mutation, selection, probability, and others. The new solutions thus found are evaluated over subsequent iterations or until the stopping criteria is satisfied, and the final solution is obtained.

Generally, EAs were the first of their kind and use the concept of Darwinian theory of survival of the fittest. The genetic algorithm (GA) (Holland, 1992), proposed in late 1970s, can be called as the mother of all EAs. This algorithm is one among the best EAs and is still in use. The first variant was a binary algorithm and was extended to continuous domain lately. In present times, the algorithm is mostly used in optimization of multi-objective problems with non-dominated sorting based GA (NSGA) as one among the best variants. For single objective optimization, a large number of algorithms based on EAs structure have been proposed namely evolutionary strategy (ES) (Rechenberg, 1978), DE (Storn and Price, 1997), biogeography-based optimization (BBO) (Simon, 2008), genetic

programming (GP) (Koza and Koza, 1992), ant lion algorithm (ALO) (Mirjalili, 2015), probability base incremental learning (PBIL) (Michalewicz, 2013) and others.

The second group of nature-inspired algorithms is SI. Swarm in itself means the group of different organisms, such as ants, birds, bees, monkeys, and wolves, which work collectively towards a common goal. SIs use the learning ability and adaptive nature of these organisms to solve really complex problems at hand. The first work on SI was reported by Beni and Wang (Beni and Wang, 1993) in cellular robotics and in present times, it has been applied to almost every field including electrical load dispatch problems, business modelling, forecasting, search optimization, structural engineering design, and others. Major algorithms under this category include, artificial bee colony (ABC) (Karaboga, 2005), particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), firefly algorithm (FA) (Yang, 2010a), bacteria foraging optimization (BFO) (Passino, 2002), FPA (Yang, 2012), bat algorithm (BA) (Yang, 2010b), salp swarm algorithm (SSA) (Faris *et al.*, 2020), CS (Gandomi *et al.*, 2013), whale optimization algorithm (WOA) (Mirjalili and Lewis, 2016), grey wolf optimization (GWO) (Mirjalili *et al.*, 2014) and others. The concept of SI was put forth by Bonabeau and he defined it as "an attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insects colonies and other animal societies" (Bonabeau *et al.*, 1999). To depict whether a behavior is SI or not, Karaboga et al. coined that an algorithm is SI only if, it follows self-organization and division of labor (Karaboga, 2005). Self-organization is the process by which, low-level components interact with each other to produce a global result. The main feature is that there is no central control over the population, making all the elements to work and act at the same time. Hence a global leader or solution is produced purely on the basis of local interactions. Self-organization relies on four basic principles (Bonabeau *et al.*, 1999):

- **Positive feedback:** This phenomenon extracts information from the output of one system and reapplies it to input for producing convenient structures. It helps to provide diversity and accelerate the system. Examples include recruitment and reinforcement such as ant trails, bee dances, etc.

- **Negative feedback:** This phase is to counterbalance the effects of positive feed-

back. So, it ultimately helps to stabilize the collective patterns and avoid saturation caused due to foragers, crowding, competition or food exhaustion.

- **Fluctuations:** It refers to the randomness in a system, for innovation, creativity, and discovery of new solutions. This helps solution to get out of any stagnation problem.

- **Multiple interactions:** It provides different ways to learn based on interactions within a population. It thus enhances the intelligence by using combined results of all.

Division of labor: It means performing specialized tasks by individual groups and this is the second phase to depict SI behavior for any organism. All the tasks are simultaneously performed and hence the performance of such a system is better than the sequential tasks performed by unspecialized individuals (Jeanne, 1986, Oster and Wilson, 1978). Division of labor also enables swarm to produce a response as per the changes in the search space.

Nature-inspired algorithms do have certain drawbacks. They may be fit for one problem while unfit for others or simply we can say that no algorithm is perfect for solving all optimization problems. This has been validated by the No Free Lunch Theorem (NFL) (Wolpert and Macready, 1997). This theory motivates many researchers to investigate and derive new algorithms for solving optimization problems. Another reason to design a new algorithm is the inherent drawbacks of existing algorithms like inefficiency in exploring search space by ABC (Karaboga and Basturk, 2008), premature convergence and stagnation of population-based algorithms like DE (Lampinen *et al.*, 2000) and PSO (Shi and Eberhart, 1998). PSO is also found to be weaker in refining the solution due to less diversity in later stages and also problem-based tuning of parameters is required to get an optimal solution (Blouin and Blouin, 1988). Also, there is a major defect in one of the most recently introduced GWO algorithm (Niu *et al.*, 2019). This defect is the decrease in efficiency if the final solution tends to move away from zero. So, in order to eradicate these drawbacks, we need to evaluate, hybridize and design new powerful algorithms for solving optimization problems. This thesis deals with the investigation of some of the major algorithms namely CS (Gandomi *et al.*, 2013), FPA (Yang, 2012) and others by

developing new optimization techniques. The thesis also deals with the proposal of a new algorithm namely Naked-mole rat algorithm (NMR) (Salgotra and Singh, 2019) to solve diverse set of benchmark optimization problems and some real-world applications.

2.1.2 Test Suites

The test suite consists of various different functions and are used to validate the performance of algorithms under test. They are basically designed in order to validate and successfully test new and modified algorithms. These test functions have been formulated by various researchers and were first evaluated in the IEEE Congress on Evolutionary Computation (CEC) and hence got their name as CEC Benchmark functions. The first such functions which are widely in use were proposed by (Suganthan *et al.*, 2005) and have been named as CEC 2005 benchmark functions. Over the last one decade, these benchmarks have put forth the basis of optimization research and have been employed in almost every domain of research. The major reason for the popularity of these benchmark functions is the wide applicability, diversity, separability, high dimensionality, scalability and large number of local versus global minima, making them highly challenging to solve. Overall, these benchmark functions have been designed to depict real-world scenarios for highly challenging domain research problems. Other CEC benchmarks which have been used in this thesis are CEC 2015 (Awad *et al.*, 2016a), CEC 2017 (Awad *et al.*, 2017)and CEC 2020 (Yue *et al.*, 2019).

As far as characteristics of these test functions are concerned, they are uni-modal functions, multi-modal functions, fixed dimensional functions, hybrid functions and composite functions. All these functions are classified based on the number of local and global minimal solutions. Among all these test functions, uni-modal functions are the most simplest form of benchmark functions and have only one global minima. Here it should be noted that there is no difference between global and local minima in these test functions are are one and the same. These test functions are used to test the exploitation properties of an algorithm and hence provide details on local search capabilities of an algorithm. The second set of test functions include multi-modal functions and have one global minima and many local minima. Because of large number of local minima, these functions are highly challenging and as the number of local minima increases, the test complexity of these test functions also increases. These test functions are further used to judge the

exploratory power in any algorithm. The third set of functions which are fixed dimensional functions have fixed dimension size and these functions define the consistency of any algorithm in finding global minima or the final best solution. Thus it can be said that these test function provide details on the convergence properties of an algorithm. Hybrid and composite functions are other categories of test functions and these functions are designed by a combination of uni-modal and multi-modal functions. These test functions are highly challenging and make a significant contribution in providing highly complex environment to solve prospective algorithms under comparison. The complexity further increases with increase in the dimensinality.

Overall, the CEC 2005 benchmark set consists of simple uni-modal functions, multi-modal functions fixed dimension functions but all other benchmark sets that is CEC 2015, CEC 2017 and CEC 2020 benchmarks are the most challenging datasets with highly complex test problems. Any algorithm performing well on these test suites can be considered as a prospective candidate for becoming state-of-the-art algorithm. In the next section some of the recently introduced algorithms have been exploited and recent literature has been discussed.

2.2 Algorithms

Nature inspired algorithms as discussed above have proven their worth in terms of competitiveness. These algorithms preserve some of the previous information over subsequent iterations, require minimal set of parameters to be tuned and hence can be implemented easily. A list of major algorithms used in this thesis have been discussed in the consecutive subsections.

2.2.1 Cuckoo Search Algorithm

CS is a new heuristic algorithm inspired from the obligate brood parasitic behaviour of some cuckoo species as they lay their eggs in the nests of host birds. Some cuckoos have specialty of imitating colours and patterns of eggs of a few chosen host species. This reduces the probability of eggs being abandoned. If host bird discovers foreign eggs, they either abandon the eggs or throw them away. Parasitic cuckoos choose a nest where host bird just lays its eggs. Eggs of cuckoo hatch earlier than their host eggs and when it hatches, it propels the host eggs out of the nests. Hence cuckoo chicks get

good share of food and sometimes they even imitate call of host chicks to get more food (Yang, 2010b). Mostly cuckoos search food by a simple random walk, where random walk is a Markov chain whose next position is based on current position and transition probability of next position. Using Lévy flights instead of simple random walks improves the search capabilities. Lévy flight is a random walk in step-lengths following a heavy-tailed probability distribution (Gandomi *et al.*, 2013). Each cuckoo acts as a potential solution to the problem under consideration. The main aim is to generate a new and potentially better solution (cuckoo) to be replaced with a not so good solution. Each nest has one egg but as the problem complexity increases, multiple eggs can be used to represent a set of solutions. There are three basic idealized rules of CS as:

- Each cuckoo lays one egg and dumps it in a random nest;
- The nest with highest fitness will carry over to next generations;
- The number of available host nests is kept fixed and egg laid by cuckoo is discovered by host bird with a probability $p \in [0, 1]$.
- And depending on p, the host bird either throw the egg away or abandon the nest. It is assumed, that only a fraction p of nests is replaced by new nests.

Based on the three rules, CS has been implemented in the consecutive subsections. These are divided as global search phase, local search phase, switching pattern and greedy selection. Each of these are explained as follows:

Global search phase: In the global search phase, it is assumed that there is only one cuckoo egg which is laid in the host birds' nest. Based on this the new cuckoo or simply new solution x_i^{t+1} is generated by using Lévy flights and this new solution corresponds to the i^{th} cuckoo from the whole population of N cuckoos. This step corresponds to the exploration process and is given by Equation Eq. (2.2)

$$x_i^{t+1} = x_i^t + \alpha \otimes L(\lambda)(x_{best} - x_i^t) \tag{2.2}$$

where x_i^t corresponds to the previous solution, $x_i^t + 1$ corresponds to the current solution, $\otimes$ is the entry wise multiplication operator and $\alpha > 0$ is the fixed step size as per the problem dimension. The Lévy flight component here is the random step size and is used

to imitate the flight trajectory of the cuckoo birds. This step size is generated by using the following equation Eq. (2.3)

$$L(\lambda) \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}} \quad (s \gg s_0 \gg 0) \tag{2.3}$$

where $s = \frac{U}{|V|^{1/\lambda}}$ $U \sim N(0,\sigma^2)$, $V \sim N(0,1)$ and $\sigma^2 = \left\{\frac{\Gamma(1+\lambda)}{\lambda\Gamma[(1+\lambda)/2]}\cdot\frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right.$. Also $\Gamma(\lambda)$ is gamma function and the value of λ is equal to 1.5. The parameter N is taken from standard Gaussian distribution having mean 0 and variance, σ^2. This phase is the exploration phase and requires current best solution, x_{best}, for finding the new solution.

Local search phase: The second phase is the local random walk or the local search phase and corresponds to the exploitation process. This phase consists of two random solutions from the search pool and based on that the new solution x_i^{t+1} is generated. The general equation for local search phase is given by

$$x_i^{t+1} = x_i^t + \alpha \otimes (\epsilon) \otimes (x_j^t - x_k^t) \tag{2.4}$$

Here all the notations are similar to those of the global search phase except for x_j^t and x_k^t which correspond to two random solutions j and k from the whole population with respect to the t^th generation. The value of ϵ lies in the range of [0,1] and is a uniformly distributed random number.

Switching pattern: Though this parameter has already been discussed in the previous sections but a thorough study about this parameter is essential. The switching pattern in case of CS is decided by the probability p and it acts as the deciding factor for the extent of exploration and exploitation process in general. Various values of p have been stated but generally, a value of 0.8 for this parameter is found to provide better results. Here 0.8 corresponds to eighty percent exploration and twenty percent exploitation.

Greedy selection: This is the selection phase of CS. Here a greedy search is followed to determine whether the newly generated solution is better than the previous one or not. Based on that, the old solution is either replaced by the new solution or retained. For a minimization problem, the greedy selection mechanism is given by Eq. (2.5)

$$x_{new}^{t+1} = \begin{cases} x_{new} & iff(x_{new}) < f(x_i^t) \\ x_i^t & otherwise \end{cases} \tag{2.5}$$

where $f(x_i^t)$ represents the fitness of the x_i^t solution. The pseudo-code of CS is shown in Algorithm 1.

Algorithm 1 Pseudo-code of original CS algorithm

```
Begin
define initial parameters: stopping criteria and problem dimension
initialize cuckoo population (N)
for i= 1: t_max do
    global search using Eq. (2.2)
    evaluate fitness and perform greedy selection using Eq. (2.5)
    local search for fraction of worst nests using Eq. (2.4)
    evaluate fitness and perform greedy selection using Eq. (2.5)
    evaluate the current best
update final best
End
```

CS has proved its worth in various fields of research and since its inception, a large number of modifications have been proposed. Modified adaptive CS with division into subgroups was proposed by (Zhang *et al.*, 2012). It was seen that grouping strategy achieves a balance between exploration and exploitation tendencies of CS algorithm. As the algorithm was tested only on seven benchmark functions and that too from two different datasets, it can be said that only functions providing better results were used and others were neglected. CS with orthogonal learning was proposed in (Li *et al.*, 2014). The authors discussed the effect of population size and tested the performance of proposed variant on 23 benchmark problems. It was found that orthogonal based learning enhances the exploration capability of CS and also help in providing better convergence. The authors of the article didn't provide any discussion about the parameters of CS. Hybrid CS with initial crossover and mutation was proposed by (Kanagaraj *et al.*, 2013). This version aims at improving the premature convergence and was applied to reliability redundancy allocation problem. Though the proposed variant provided good results but no proper analytical study about CS parameters was presented. Other modifications include hybrid CS for knapsack problem (Feng *et al.*, 2014), hybrid CS with soils and wet local search

algorithm (Long *et al.*, 2014), parameter adjustment with chaotic map was introduced in (Wang and Zhong, 2015) and it was found that chaotic maps improve the performance of CS. Multi-objective CS was proposed by Yang and Deb for design optimization (Yang and Deb, 2013), discrete CS was proposed for solving travelling salesman problem (Ouaarab *et al.*, 2014), hybrid CS with wind driven algorithm was proposed for satellite image segmentation (Bhandari *et al.*, 2014), improved CS for hybrid flow shop scheduling was proposed in (Marichelvam *et al.*, 2014), hybrid CS and harmony search was proposed for numerical optimization in (Wang *et al.*, 2016) enhanced chaos based CS was proposed for global optimization in (Huang *et al.*, 2016).

In terms of applications, CS has been applied to a wide range of optimization problems such as CS for distributed networks (Nguyen *et al.*, 2016), feature extraction (Wang *et al.*, 2014), electrocardiogram signal watermarking (Dey *et al.*, 2013), multi-reservoir system (Ming *et al.*, 2015), CS for Sudoku problem (Soto *et al.*, 2014), web search clustering (Cobos *et al.*, 2014), PCB drill path problem (Lim *et al.*, 2014), finite impulse response fractional order differentiator design (Kumar and Rawat, 2015), designing 1-dimensional and 2-dimensional recursive filters (Sarangi *et al.*, 2014), video tracking system (Walia and Kapoor, 2014) and other applications such as economic load dispatch by (Bindu and Reddy, 2013), design of planer ebg structures (Pani *et al.*, 2013) and a brief review of CS was proposed by Fister et al. in (Fister *et al.*, 2014).Other applications include embedded system design (Kumar and Chakarverty, 2011), playing Mario game using CS (Speed, 2010), flood forecasting (Chaowanawatee and Heednacram, 2012), Levenberg Marquardt based using back propagation training using CS (Nawi *et al.*, 2013) and others. Most of these articles focus on the problem itself without proper configuring the parameters of CS algorithm.

From the above discussed literature, following inferences have been drawn:

- No proper discussion about the effect of parameters on performance of CS was presented in most of the cases.
- Most of the articles based on application domain didn't even mentioned about the parameters of CS.

- Effect of population size and dimension was exploited in very few cases and no proper justification of their analysis is presented.

- Grouping can help to improve the performance of CS algorithm.

The above said inferences have motivated the authors to work on the enhancement of CS algorithms. Various algorithms based on CS have been proposed in this thesis and details on each have been added in subsequent chapters.

2.2.2 Flower Pollination Algorithm

Almost 250,000 (80% approximately) of all plants species are flowering, hence dominating landscape aging from more than 125 million years that is Cretaceous period (Yang, 2012). From biological and evolutionary point of view, pollination is the basic function fulfilled by flowers in order to reproduce. Pollination is associated with pollinators such as insects, birds and the transfer of pollen takes place through them. This process in flowers ultimately pave way for optimal production of best fit flowers. Some flowers attract only specific species of insects or birds for pollination resulting into specialized flower-pollinator partnership often called flower constancy. The major forms of pollination are biotic and abiotic. The pollination in which pollinators such as insects, birds, bats, honey bees and other animals are associated is called biotic. It constitute about 90% of flowering plants. Abiotic pollination on the other hand does not require any pollinators. Pollination mainly occur either through wind or diffusion. At least 200,000 varieties of pollinators are present in nature and are found to develop flower constancy. By the process of constancy, pollinators tend to visit certain specific species and bypass other species. This maximizes the transfer of pollens to the same species of flower or conspecific plants and also increases reproduction rate of the same flower species. The pollinators in turn also gets benefited as in case of honey bee, they visit only certain species to collect nectar maintaining minimum cost and more intake of nectar .

FPA is derived from the biological pollination process of flowers. The algorithm basically simulates flower constancy and pollination behaviour of flowering plants in nature. Another phenomenon which adds to FPA is the flower constancy in which pollinators visit only particular species of flowers (Pavlyukevich, 2007). This process is advantageous for both the flower and the pollinator. The pollinator get sufficient amount of nectar whereas

the flower maintains pollination and hence increase the population of particular flower species. Depending upon the availability of pollinators, pollination can also be defined as self and cross pollination. In self-pollination, there is no reliable pollinator while in cross-pollination, pollinators such as bee, birds and bats fly over long distances and results in global pollination . The pollination process, pollinator behaviour and flower constancy together sum up to form following four rules (Yang, 2012):

- Global pollination occurs by movement of pollinators carrying pollens via Lévy flights, biotic and cross-pollination.
- Local pollination is performed by abiotic and self-pollination.
- Flower constancy which is proportional to the similarity of two flowers can be considered as the reproduction probability.
- Switch probability p ϵ [0, 1] is used to control the local and global pollination. Physical proximity and factors such as wind, bias the pollination activities towards local pollination.

As each plant has multiple flowers, and each flower patch release millions of pollen gametes. For present case, we assume only a single flower producing only a single pollen gamete. Thus a single pollen gamete, flower or plant acts as potential solution of the problem under consideration.

These four rules are idealized to form mathematical equations. The algorithm was further formulated by exploiting the concept of local and global pollination. In global pollination, pollinators carry flowers over long distances. This helps in pollination and hence reproduction of the fittest (T^*) flower. Rule 1 in combination with flower constancy is represented as:

$$x_i^{t+1} = x_i^t + L(\lambda)(T^* - x_i^t) \tag{2.6}$$

where x_i^t is the i^{th} solution in the t^{th} iteration, L corresponds to Lévy flights used to generate random step size drawn from a standard Lévy distribution as

$$L(\lambda) \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}} \quad (s \gg s_0 \gg 0) \tag{2.7}$$

where $s = \frac{U}{|V|^1/\lambda}$ $U \sim N(0, \sigma^2)$, $V \sim N(0, 1)$ and $\sigma^2 = \left\{ \frac{\Gamma(1+\lambda)}{\lambda\Gamma[(1+\lambda)/2]} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}} \right.$. Also $\Gamma(\lambda)$ is gamma function and the value of λ is equal to 1.5. The parameter N is taken from standard Gaussian distribution having mean 0 and variance, σ^2. The second phase is local pollination. This phase along with flower constancy are given by

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t) \tag{2.8}$$

where ϵ is a uniformly distributed random number, x_j^{t+1} is the current solution, x_j^t and x_k^t are pollen or solution of same plant with different flower, mimicking flower constancy behaviour in a limited search space. It is further said that flower pollination occurs at local as well as global scale. Here Rule 4 above controls the pollination at local and global level. The parameter which controls this process is called as the switching probability denoted by p. The pseudo-code of FPA is shown in Algorithm 2. It can be assumed from the

Algorithm 2 Pseudo-code of original FPA algorithm

Begin

Initialize flower population: $\boldsymbol{n}$

Define $\boldsymbol{d}-$dimensional objective function, $\boldsymbol{f(x)}$

for i= 1: t_{max} **do**

if **rand** $<$ **p**

A d$-$dimensional Gaussian-based step vector is drawn.

Perform global pollination: Eq. (2.6)

else

Perform local pollination: Eq. (2.8)

end if

if x_i^{t+1} better than x_i^t

$x_i^t = x_i^{t+1}$

end if

evaluate the current best

update final **best**

End

above discussion itself that local pollination corresponds to the exploitation phase whereas global pollination is for exploration phase. There is another parameter named as switch

probability (p) which decides the amount of exploration and exploitation. Here, if the value of p is 0.5, it means 50% exploration and 50% exploitation will take place. Likewise, if the value of p is $[0.6, 1]$, the rate of exploitation will increase and if it is decreased from 0.5, exploration will increase. In the proposed publication of FPA (Yang, 2012), it is said that one can use $p = 0.5$ as the initial value and further without any parametric study quoted that a value of 0.8 provides better results for most of the applications. Various other works on FPA in (Sharawi *et al.*, 2014), (Prathiba *et al.*, 2014), (Salgotra and Singh, 2018), (Yang *et al.*, 2013) and (Ochoa *et al.*, 2014) use same p value of 0.8 and no explanation regarding that has been given. In (Draa, 2015), an extensive survey about the values of p has been done qualitatively and it has been reported that $p = 0.2$ gives much better results as compared to $p = 0.8$. In (Nabil, 2016), a new modified FPA using clonal property has been proposed. The algorithm introduces a step-size scaling factor and it was said that a preliminary parametric study showed that a value of 3 works well for all test functions. But no such parametric study was given in the article. Also, the article uses $p = 0.8$ as standard and no detailed study about the selection of such value has been discussed.

For any generalized algorithm, we know that more global search should be done at the beginning as it will tend to explore the search space better and when enough exploration has been done, the process should move to local search that is exploitation (Zhou *et al.*, 2016b). Thus, use of dynamic switching will help to adjust the appropriate proportion of two searching processes. In FPA, local and global search are controlled by using switching probability and in (Wang and Zhou, 2014), the concept of dynamic switching was formulated. Some limited work has also been done on the analysis of value of p FPA in (Balasubramani and Marcus, 2014), (Sakib *et al.*, 2014), (Łukasik and Kowalski, 2015). It was concluded that no proper investigation about the value of p was given and no statistical testing of the results have been presented which further makes the analysis questionable. Apart from parameter settings, a lot of modifications have been proposed in the recent past. Hybrid chaotic-based FPA extension was proposed to solve large integer problem (El-henawy and Ismail, 2014), DE hybridization was proposed and validated on benchmark function in (Chakraborty *et al.*, 2014), flower pollination based maximum power point method for solar power tracking was proposed in (Ram and Rajasekar, 2016),

FPA with fuzzy approach proposed in (Valenzuela *et al.*, 2017), a bat flower pollinator (BFP) was proposed in (Salgotra and Singh, 2018). The authors in (Draa, 2015) provided a proper quantitative and qualitative analysis of the vagueness of these proposed techniques. Further, they proposed five opposition based variants of FPA and tested their performance on standard (Liang *et al.*, 2013) and real world benchmarks (Suganthan *et al.*, 2005). A deeper analysis of these algorithms has also been performed and it is shown that with respect to FPA they are better but overall, they have average performance as compared to state-of-the-art algorithms. Another recent modification includes elite opposition based FPA, which stressed on the global elite opposition, local self-adaptive greedy search and dynamic switch probability (Zhou *et al.*, 2016b). The dynamic switching helped to balance the exploration and exploitation while other helped to enhance the local and global pollination. From the above literature, it is evident that FPA is a good algorithm and based on that following inferences have been drawn.

- Proper switch probability p should be identified. It can be dynamic in nature in order to control the exploration and exploitation capability of FPA.
- In global pollination, the Lévy distribution based random number has not been explored in any of the work. It can be treated same as mutation operator and must be exploited.
- Defining a better global and local search equation is required to make FPA a better state-of-the-art algorithm.
- In local pollination, the epsilon (ϵ) variable is to be analysed for some standard value to optimize the local search.

These inferences have motivated the authors to estimate that there is requirement of new enhanced versions of FPA. Based on this, new modified algorithms have been proposed and details of the same is provided in subsequent chapters.

2.3 Inferences and Summary

In this chapter, the numerical optimization technique with a generalized mathematical explanation for any optimization problem is discussed. It provides details on the application

and importance of optimization in almost every domain of research, hence the requirement of highly efficient optimization techniques have attracted the attention of numerous researchers. In the recent years, nature inspired algorithms have been used to solve high end optimization problems. These algorithms include meta-heuristic population based techniques using different global and local search strategies. CS, FPA, DE and others are such algorithms which have proven their worth and are considered as most effective and powerful algorithms for solving competitive optimization problems.

Although a lot of optimization algorithms have been proposed in the literature, but many experimental and theoretical studies show that they are highly sensitive to the choice of control parameters. Also, they fail to solve complex optimization problems which is caused by the premature convergence and stagnation. Major drawbacks include poor exploration, poor exploitation, slow run-time complexity, higher computational cost and others. Thus in order to mitigate these problems, several improvements in the basic structure of these algorithms have been introduced.

It has also been shown in the literature that plenty of improved versions of basic algorithms have been developed in the last era. These algorithms used various different training parameters, adaptive approaches, reduction phenomena and synthetic methods. Although these hybridization succeed in improving the performance of these algorithms, but while solving diversified optimization problems, finding the optimal solution, these algorithms find little application for complex challenging optimization problems. Finding the global optimal solution under certain constraints such an pre-defiend number of iterations, lower computational time and others, is still a challenging task for researchers. Furthermore, it still remains an open challenge to improve the performance of hybrid algorithms with the development of new optimization problems and complex real-world applications.

These above cited considerations define a clear direction for further estimation and investigation of our proposed research in the following chapters. The overall goal is to develop new prospective algorithms by enhancing the performance of existing algorithms using new and novel ideas to solve recently developed benchmark test suits and real-world applications. In our research, the focus is mainly on the development of new efficient techniques by enhancing the performance of CS, FPA, FA and DE algorithms. Apart

from that, the research also focused on analysing the basic properties of naked-mole rat species and based on that a new meta-heuristic technique namely Naked-mole rat algorithm (NMR) has been proposed. All the new proposed algorithms aim at producing better results with respect to the other algorithms reported in the literature.

Part I : Cuckoo Search Algorithm

CHAPTER 3

New Cuckoo Search Algorithms

Preface

In this chapter, three new variants of CS have been proposed to improve the exploration and exploitation. The new variants employ a new concept of division of population and generations. Division of generations makes the algorithm perform exploration for the first half of population and exploitation for second half of the population. Also division of population has been done to help the members of the population to change their positions abruptly and converge faster toward the end of iterations. For improving the exploration, instead of using Lévy flight based random walks, Cauchy based walks are employed and for exploitation, multiple searching strategies are used to evaluate the final solution.

Publication: **Rohit Salgotra**, Urvinder Singh, and Sriparna Saha. "New cuckoo search algorithms with enhanced exploration and exploitation properties." Expert Systems with Applications 95 (2018): 384-420.

3.1 Introduction

CS as discussed in literature, is a swarm intelligent algorithm introduced in the recent past (Yang and Deb, 2009). The algorithm is based on the obligate brood parasitic behaviour of cuckoos. The search process in this algorithm is divided into two phases that is global and local phase. In the global phase, formation of new nests takes place while in the local phase, removal of a fraction of worst nests is followed. The global phase is governed by Lévy flight based random walks rather than simple Brownian or Gaussian walks. Lévy flight because of their heavy tail, infinite mean and variance, helps in exploring the search space in potentially more efficient way. The CS algorithm, because of its searching capabilities, has gained attraction from the research community in the recent years. It has been applied to a large set of real world optimization problems (Gandomi *et al.*, 2013, Fister *et al.*, 2014).

Over the past few decades, the complexity of problems to be optimized is increasing day by day and so do the evolutionary algorithms. Wolpert and Macready proved that no algorithm can be generalized for all optimization problems (Wolpert and Macready, 1997). So, it becomes necessary to design new algorithms as per particular set of problems. The algorithm though is very efficient in exploration, much is required to improve its exploitative tendencies. So, in order to improve the exploitation of CS, three new variants have been proposed in this chapter. The proposed variants aim at maintaining a balance between exploration and exploitation. Two new concepts namely division of population and division of generations have been employed to design new variants. Also for more extensive exploration Lévy flight based random walks are replaced by Cauchy random walks. In this chapter, local search and global search phase of CS are modified to achieve better position of new cuckoo. The basic ideas of proposal are discussed as:

- Two new concepts based on the division of generations and population are proposed.
- Based on these two concepts, three new versions of CS have been proposed.
- The first version uses enhanced global and dual division in local search phase.
- The second version uses dual division in global search only.

- In the final version, four-fold division of population is followed in global search phase.
- Division of generations is applied in all the three versions.
- Division of population is applied to local and global search phases.

Why division of generations and population is used: There are two factors namely exploration and exploitation which decide whether an algorithm is efficient or not. Exploration decides how good an algorithm is in escaping local minima and exploitation decide its convergence properties. So, an algorithm which can balance both these phenomenon, is able to achieve better results (Wolpert and Macready, 1997). For a generalized algorithm, extensive global search should be followed at the beginning of search process and as the search progresses, more intensive exploitation should be done. So, to maintain a balance in exploration and exploitation, division of generations and population has been implemented. By division of generations, we mean to say that for half of the iterations a standard equation having best explorative tendencies should be used and for rest of the iterations more intensive exploitation should be there. This will maintaining a balance between exploration and exploitation. For division of population, we allocate the population to multiple locations or simply into smaller groups and this process is carried out by use of different search equation in each group. It will help the members of population to change their positions abruptly in the initial stages and converge faster as the iterations proceed. Thus, both the modifications will add up to maintain a balance between the exploration and exploitation.

How division of generations is applied: The generations have been divided into two halves. For first half of the generations, Cauchy based global search is adopted and the reason for its use has been explained in the subsequent sub-section. The local search phase follows the original equation of CS. For second half of the generations, division of generations is followed. Also, to avoid repetitiveness, the first half of the generations which is same in all the three cases has not been discussed explicitly in every modification proposed. Only modifications based on the second half of generations are presented.

How division of population is applied: The division of population has been employed using

multiple search equations. Using different division strategies, three modified versions have been proposed. In the first version, modified global search and dual local search is followed. By dual local search we mean to say that population is divided into two parts and two search equations are used to implement this phenomenon. In the second version, the dual search equations are employed in global search phase. For the third version, population is divided into four parts and four search equations are employed to update the solution.

Cauchy based mutation operator: Though Lévy mutated step size is good in exploring the search space, still there is more room for a much better exploration operator. So, in this context, Cauchy based mutation operator is applied to generate step size instead of Lévy mutation operator. The Cauchy based mutation operator is used to generate a Cauchy distributed random numberδ. This random number is then integrated in the global search equation to generate a new solution. The first concept for such mutation was formulated in (Yao *et al.*, 1999).The Cauchy density function is given by

$$f_{Cauchy(0,g)}(\delta) = \frac{1}{\pi}\frac{g}{g^2+\delta^2} \tag{3.1}$$

The Cauchy distribution function is given by:

$$y = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{\delta}{g}\right) \tag{3.2}$$

where g is the scale parameter and its value is taken as 1 and$y \in [0,\ 1]$. Solving Eq. (3.2) for the value of δ, we get

$$\delta = \tan\left(\pi(y - \frac{1}{2})\right) \tag{3.3}$$

The above equation will generate a Cauchy distributed random number in the range of 0 to 1. Based on this the general Eq. (2.1) of the global search phase changes to

$$x_i^{t+1} = x_i^t + \alpha \otimes Cauchy(\delta)(x_i^t - x_j^t) \tag{3.4}$$

The use of Cauchy distribution because of its fatter tail, generate larger steps which helps the algorithm in exploring the search space in a much better way. This will help the algorithm in escaping local minima. This has been done because the best solution guided difference prevents the algorithm form exploring the search space and this may lead to premature convergence. The Cauchy based mutation operator has been employed in all the proposed versions for first half of the iterations and has not been discussed explicitly in

every subsection. Also in the pseudo-code of proposed versions only the second half of the generations are presented. This is because for first half of the generations, the proposed versions are same as that of the basic CS algorithm except for change in Eq. (2.1) which is replaced by Eq. (3.4).

Based on these facts, three new variants of CS are proposed in the next section. Note that the above discussed parameters are same in all the variants along with the other proposed modifications.

3.2 The Proposed Approach

3.2.1 Cuckoo Version 1.0

In this version, two modifications are proposed in the standard CS and it has been named as Cuckoo Version 1.0 (CV 1.0). Firstly, the global pollination phase has been enhanced by taking the mean of first three solutions from the pool of search agents. These three solutions are generated based upon the current best solution. The equations proposed in this regard are given by

$$x_1 = x_i - A_1(C_1.x_{best} - x_i^t);\ x_2 = x_i - A_2(C_2.x_{best} - x_i^t); x_3 = x_i - A_3(C_3.x_{best} - x_i^t) \quad (3.5)$$

$$x_{new} = \frac{x_1 + x_2 + x_3}{3} \quad (3.6)$$

Here the current best solution x_{best} is moved to a new location by multiplying it with a random variable and the random solution x_i is subtracted from it to generate the new solution x_{new}. The basic concept for this modification has been inspired from the search equations of GWO (Mirjalili *et al.*, 2014). In GWO, the solution is generated by adjusting the parameters with respect to the position of best search agent. The same concept has been employed to obtain new solutionx_{new}. This new solution is then subjected to Eq. (3.4) of the global search phase which leads to

$$x_i^{t+1} = x_{new}^t + \alpha \otimes Cauchy(\delta)(x_{best} - x_{new}^t) \quad (3.7)$$

where $A_1,\ A_2,\ A_3$ *and* $C_1,\ C_2,\ C_3$ belongs to A and$C$ respectively.A and C are given by

$$A = 2a.r_1 - a; C = 2.r_2 \quad (3.8)$$

where a is a linearly decreasing random number in the range of [0, 2] with respect to iterations, r_1 *and* r_2 are two uniformly distributed random numbers in the range of [0,1]. These random numbers r_1*and* $r_2$allow the search agents to reach any position within the search space and hence increase the explorative tendencies of CS algorithm.

Algorithm 3 Pseudo-code of CV 1.0 algorithm

1: **Begin**
2: define initial parameters: stopping criteria and problem dimension
3: initialize cuckoo population (N)
4: **for** i= 1: maximum number of iterations **do**
5: global search using Eq. (3.7)
6: evaluate fitness and perform greedy selection using Eq. (2.5)
7: local search for fraction of worst nests using Eq. (2.4) & Eq. (3.9)
8: evaluate fitness and perform greedy selection using Eq. (2.5)
9: evaluate the current best
10: update final **best**
11: **End**

Secondly, the local search phase is enhanced to improve the performance of basic CS. The population in this case is divided into halves and two different searching strategies are used to evaluate the final solution. For first half of the population, the general equation Eq. (2.1) of the standard CS is used and for the other half a new search equation is introduced. This new search equation is given by

$$x_i^t = x_i^t + F.\left(\left(x_j^t - x_k^t\right) + \left(x_l^t - x_m^t\right)\right) \tag{3.9}$$

wherex_j^t, x_k^t, x_l^t *and* $x_m^t$are four random solutions and the new solution will be updated based on them. Here because of the use of same random number F(in the range of 0 to 1), the equation becomes efficient for smaller search space. So, using the same for second half of the population will definitely provide better results. On a whole, the combination of Cauchy based global search and dual local search can provide an exact balance between exploration and exploitation of the proposed variants. The pseudo-code of CV 1.0 is shown in Algorithm 3.

3.2.2 Cuckoo Version 2.0

For the second proposed version, division of population has been applied for both local and global phase. Here two-fold division has been followed in both the phases. For global search, half of the population is updated based on the Eq. (3.10) and the next half is updated based on the Eq. (3.6). As towards the later stages, more exploitation is the requirement so here the Eq. (3.4) used has been modified by using the difference of best solution with a random solution. This will create copies of the candidate solution around the current best solution and hence will add to the exploitative tendencies. The modified equation is given by

$$x_i^{t+1} = x_i^t + \alpha \otimes Cauchy(\delta)(x_{best} - x_i^t) \tag{3.10}$$

Both the solutions from Eq. (3.6) and Eq. (3.10) are then combined to generate a new solution. For local search phase, Eq. (2.2) and Eq. (3.9) have been used to evaluate the new solution. Also for the global search phase, Cauchy based equation will ensure extensive exploration for the initial set of population whereas Eq. (3.6) will ensure intensive exploitation. The justification for local search equation has been elaborated in subsection 3.1. The pseudo-code of CV 2.0 is shown in Algorithm 4.

Algorithm 4 Pseudo-code of CV 2.0 algorithm

1: **Begin**
2: define initial parameters: stopping criteria and problem dimension
3: initialize cuckoo population (N)
4: **for** i= 1: maximum number of iterations **do**
5: global search using Eq. (3.6) & Eq. (3.10)
6: evaluate fitness and perform greedy selection using Eq. (2.5)
7: local search for fraction of worst nests using Eq. (2.4) & Eq. (3.9)
8: evaluate fitness and perform greedy selection using Eq. (2.5)
9: evaluate the current best
10: update final **best**
11: **End**

3.2.3 Cuckoo Version 3.0

The use of Cauchy based mutation operator in the first half of the iterations have added to the explorative capabilities of basic CS and it is required to maintain a balanced exploitation also. So, for this case, in order to balance the exploitative tendencies of CS, four-fold division of population in the global search phase is followed. By four-fold division, we mean that the population is divided into four parts and each part is evaluated by a different search equation. The first part is evaluated using Eq. (3.6), second part by using Eq. (3.9), third by using Eq. (3.10) and the fourth equation is given by

$$x_i^{t+1} = x_i^t + b.(x_i^t - x_j^t) \tag{3.11}$$

where b is a uniformly distributed random number in the range of [0, 1]. The local search phase is kept same as that of the original CS. The four-fold division strategy will help the population to be selected from four different smaller groups. This will add diversity in each group and will enhance the exploitation capabilities. So here again a balance of exploration and exploitation can be achieved. The pseudo-code of CV 3.0 is shown in Algorithm 5.

Algorithm 5 Pseudo-code of CV 3.0 algorithm

Begin
define initial parameters: stopping criteria and problem dimension
initialize cuckoo population (N)
for i= 1: maximum number of iterations **do**
  global search using Eq. (3.6), Eq. (3.9), Eq. (3.10) & Eq. (3.11)
  evaluate fitness and perform greedy selection using Eq. (2.5)
  local search for fraction of worst nests using Eq. (2.4)
  evaluate fitness and perform greedy selection using Eq. (2.5)
  evaluate the current best
update final **best**
End

3.2.4 Complexity analysis of proposed versions

The computational complexity of the original CS algorithm can be given by O (n. D.$\boldsymbol{t_{max}}$) where n is the population size, D is the dimension size and $\boldsymbol{t_{max}}$ is the maximum num-

ber of iterations. The main aim of providing this complexity analysis is to analyse the basic operations of CS and the proposed versions, and to summarize their worst case complexities. The equation is derived based on the fact that the algorithm is initialized with a population of size n and the dimension factor (D) decides the complexity of the algorithm. For each population member, we need to perform O(D) number of operations, resulting in O($n.D$) complexity. The above mentioned complexity is for one iteration, but in general CS runs for a number of iterations. So the total complexity depends on the maximum number of iterations. This procedure gives the total complexity of CS as O ($n.\ D.\boldsymbol{t_{max}}$). Compared to the original CS algorithm, all the proposed versions have no extra computational burden. Also, all the new parameters added to the proposed version choose very small random values and have no effect on the space or time complexity. Therefore, the overall time complexity of all the proposed versions is almost same as the original CS algorithm.

3.3 Result and Discussion

In this section, a detailed analysis of proposed versions of CS is presented to check whether they improve the performance of basic CS or not. First of all, an experimental study of proposed versions with respect to standard CS is presented for benchmark test problems and then the best among the proposed versions is compared with state-of-the-art algorithms. The effect of parameters such as population size, dimension and switch probability has been discussed explicitly. For performance testing, 50 runs have been performed and results are presented in terms of best, worst, mean, standard deviation and statistical wilcoxon ranksum test. All the algorithms are experimentally tested on Dell Inspiron 1464 with Intel core i3 processor having 4.00 GB RAM, windows10, x64-bit operating system and Matlab R2010a, version 7.10.0. For performance evaluation, twenty-six test functions with eight uni-modal, ten mulit-modal and seven fixed dimension functions from CEC 2005 benchmark set have been used Suganthan *et al.* (2005).

3.3.1 Effect of switch probability

In the literature, it can be seen that some detailed works have been done on evaluation of proper switch probability but no proper justification about how the study was done is present. In the present work, the effect of switch probability (p) has been analysed using different values of p. For analysis of these values 13 benchmark functions

from Table A.2 are used. Here four uni-modal (f_1, f_3, f_6 *and* f_8), five mulit-modal (f_{14}, f_{15}, f_{16}, f_{17} *and* f_{18}) and four fixed dimensional functions (f_{20}, f_{22}, f_{23} *and* f_{24}) are used. The population size of 20 and total number of iterations of 500 are taken for comparison. In Table C.1, Table C.2, Table C.3, Table C.4 and Table C.5 comparison results for $p = 0.05,\ 0.25,\ 0.50,\ 0.75$ *and* 0.95 respectively. The results are discussed as below:

Comparison of Best values: From the best values of all the tables, it can be seen that for f_1, CS was not able to reach global optimum while all other proposed versions provided very near optimum solution with CV 2.0 providing the best among all at $p = 0.75$. In f_2 and f_{18} CS was not able to reach global optimum whereas all the proposed versions provided exact global optimum. For f_6, again CV 2.0 was found to be the best for $p = 0.75$. For f_8 all the algorithms provided competitive results. For f_{14} and f_{17}, all the algorithms provided exact global optima for $p = 0.05,\ 0.25$ *and* 0.50 and for $p = 0.75$ *and* 0.95, CS was not able to achieve exact minima while for others it was very difficult to analyse which one is the best. For f_{15} and f_{16} CS is found to be the worst among all while all others provided very competitive results with CV 3.0 providing best results for f_{15} at $p = 0.75$ and for f_{16} at $p = 0.50$. For f_{20}, f_{22}, f_{23} and f_{24} all the algorithms provided the same optimums.

Comparison of Worst values: From worst values of all the tables, it can be said that for f_1 and f_6 CS was found to be the worst and all the proposed algorithms were competitive with CV 2.0 providing the best among all at $p = 0.75$ for both the functions. In f_2 and f_{18} all the proposed versions provided exact global optimum except CS. For f_8 all the algorithms provided competitive results. For f_{14} and f_{17}, all the algorithms were very competitive for $p = 0.05,\ 0.25$ *and* 0.50 and for $p = 0.95$, CS was not able to achieve exact minima while for others it was very difficult to analyse which one is the best. For f_{15} and f_{16} CS provided the worst results and all others provided very competitive results with CV 1.0 providing best results for f_{15} at $p = 0.95$ and CV 3.0 for f_{16} at $p = 0.50$. For f_{20} and f_{23}, the results of CS and CV 3.0 deviated to some extent from global optima at $p = 0.95$ and for f_{22} and f_{23} all the algorithms provided the same optimums.

Comparison of Mean values: It can be seen from the tables that means values gives almost the same results as given by the best and worst cases. Also for mean values of CS

only, it gives worst mean value at $p = 0.05$ while for at $p = 0.25,\ 0.50$ *and* 0.75 the results are best and again at $p = 0.95$, the results become worse except for f_{14} and f_{17},where best values at $p = 0.05$. For fixed dimension functions, f_{20} and f_{23}, the mean value is not exactly the global optima for $p = 0.95$ and for other cases all fixed dimension functions provide exact global optima. The results for all the proposed variants don't deviate much for any value of p.

Comparison of standard deviation values: From the standard deviation section in the tables, it can be seen that for $f_1$and f_6 CV 2.0 provides the best results at $p = 0.95$. For f_2, f_{14}, f_{17} and f_{18} CS provides the worst results while from the proposed variants, all the proposed variants provide the exact zero standard deviation.For f_8 the results are competitive. For f_{15} and f_{16}, CV 1.0 provides best result at $p = 0.95$ and CV 3.0 at $p = 0.50$ respectively. For all the fixed dimension problems, that is for f_{20}, f_{22}, f_{23} and f_{24} CS is found to be highly competitive for all the cases but still CV 1.0 was found to be the best among all. Also for these functions, it should be noted here that CV 1.0 provided best results for all the values of p.

Inference drawn: Overall it can be seen that for most of the cases, the performance of CS is worst for $p = 0.05$ while for $p = 0.25,\ 0.50$ *and* 0.75 its performance is better and for $p = 0.95$, the results degrade. This proves that performance of CS depends upon the value of p but when we compare it with the proposed versions, there is either no change in performance or is modest. Hence, we can say that the proposed techniques have no effect of p on their performance.

3.3.2 Effect of population size

In meta-heuristic algorithms, it has been seen that the number of function evaluations depend on the population size and iterations. For a fixed number of iterations and smaller population size, less are the number of function evaluations required to achieve the global optima and greater the population size more are the function evaluations. Here three set of population sizes namely 20, 40 and 60 are used to analyse the effect of population on the performance of CS and the proposed approaches. The total number of iterations is kept fixed at 500. As proved in the previous section, the value of $p = 0.25,\ 0.50$ *and* 0.75 provides comparable results for CS algorithm. For the proposed version, the value of

$p$has no effect on their performance. So, in this section we have used $p = 0.50$ as the standard value for analysing the effect of population. Fourteen test function from CEC 2005 benchmark set (Suganthan *et al.*, 2005) are selected to check the effect of population size on the performance of proposed variants and the standard CS. These functions are uni-modal (f_1, f_3, f_4 *and* f_6), mulit-modal (f_9, f_{13}, f_{15}, f_{16} *and* f_{18}) and fixed dimension functions (f_{20}, f_{21}, f_{22}, f_{23} *and* f_{24}).

Population size 40: In Table C.6, the simulation results for population size 40 are presented and it can be seen that for f_1, CS is not able to attain global minima while all the proposed versions provided very good results and among them CV 2.0 is the best. In f_3CV 1.0 provides exact optimum results. For rest of the variants, the results are competitive. For f_4, CV 2.0 gives the best results while all other variants give quite competitive results. For f_6, CS gives the worst results while other are still competitive. The best algorithm in this case is the CV 2.0. For f_9, CV 1.0, CV 2.0 and CV 3.0 provide the same best and worst and the comparison is in terms of mean and standard deviation where CV 1.0 is found to be the best. For f_{13}, CV 2.0 provides the best results. For f_{15} and f_{16} it is found that CV 1.0 and CV 3.0 gave best results respectively. For f_{18}, the values of best, worst and mean are same for all the proposed variants. Here it is difficult to comment which one is better. For f_{20}, f_{22} and f_{23}, CV 1.0 gives the best standard deviation. For f_{21} and f_{24}, CV 1.0 and CV 3.0 are the best variants respectively. In this case, CS is found to be better for no function, CV 1.0 for seven functions, CV 2.0 for six functions and CV 3.0 for four functions. So, overall CV 1.0 is found to be the best among all the proposed variants for population size of 40.

Population size 60: The simulation results for population size 60 are found in Table C.7 and it can be seen that for f_1, f_4, f_6 and f_{13} CV 2.0 provides the best results in terms of best, worst, mean and even standard deviation. In f_3 CV 1.0 provides exact optimum results. For rest of the variants, the results are competitive. For f_9, CV 1.0, CV 2.0 and CV 3.0 provide the same best and worst and the comparison is in terms of mean and standard deviation. In this case, CV 1.0 is found to be the best. For f_{15} and f_{16} it is found that CV 1.0 gave best results. For f_{18}, the values of best, worst and mean are exactly zero for all the proposed variants. Here it is difficult to comment which version is better.

For f_{20}, f_{22} and f_{23}, CV 1.0 gives the best standard deviation. For f_{21}, CV 1.0 is better and f_{24}, CS gives the best results. In this case, CV 1.0 for eight functions, CV 2.0 for five functions, CV 3.0 and CS for one function each. So, overall CV 1.0 is found to be the best among all the proposed variants for population size of 60.

Population size 80: The results for this case are presented in Table C.8. For most of the functions, the results are similar to that of the previous case of population size 60. The only exception with respect to previous case is for f_{24}, where CV 3.0 s better than CS algorithm. Overall for this case, CV 1.0 is better for eight functions, CV 2.0 for five functions, CV 3.0 for two function and CS for none. Hence again CV 1.0 is better for all this population size also.

Population size 100: Simulation results for population size 100 are presented in Table C.9. In most of the functions, the results are similar to population size 60, except for f_3 where best obtained by CV 3.0 is better among all and for worst, mean and standard deviation, CV 1.0 provides better results. So, for this case CV 1.0 can be said to have better results. For rest of the function, the results are same as that of population size 60. Overall, we find that CS is found to be better for one function, CV 1.0 for eight functions, CV 2.0 for five function and CV 3.0 for one function. This case also shows that CV 1.0 is better for 100 population size.

Inference drawn: It can be seen from the Table C.3, Table C.6, Table C.7, Table C.8 and Table C.9 that for variable population size, the change in the results is large for 20 population size and as we move towards higher population sizes, the change is only modest. This change is seen in only some of the uni-modal and multi-modal functions namely f_1, f_4, f_6 and f_{13}. But for rest of the test functions, the change in results is only modest. Overall, we can say that the results don't vary too much and thus any population size can be taken as standard population size. Here for a fixed number of iterations, if we double the population size, the number of function evaluations also doubles. So, we can say that if iterations are 500 and population size is 20, the number of function evaluations is 10,000, for population size of 40, maximum number of function evaluations become 20,000 and for the other three cases of 60, 80 and 100 population sizes, the maximum number of function evaluations become 30,000, 40,000 and 50,000 respectively. But an

algorithm providing best results by using minimum number of function evaluations is always considered as the best one. For present case, since all the test functions provide same comparative results, so we can choose a population size of 20 as the best population size.

3.3.3 Comparative study

From the above discussion, it can be found that CV 1.0 is found to be the best among all the proposed variants. It can be seen that the optimum probability switch is taken to be 0.5 and population size of 20 is the best. The reason for no change in performance at higher populations is because of the division of iterations and population which helps in proper exploration and exploitation of the search space, helping the algorithm in achieving better results by employing minimum number of function evaluations. Now to prove CV 1.0 to be state-of-the-art, it is compared with well-known algorithms. The major algorithms used in this chapter are GWO, CS, DE, BA, FA, FPA and parameter setting for these algorithms is given in Table B.1.

The simulation results for comparison are presented in Table C.10. It can be seen from the table that for functions f_1, FA, GWO and CV 1.0 are able to reach near optimum results while other algorithms got stuck in some local minima, for this function CV 1.0 is found to be the best. For f_2 *and* f_7 GWO and CV 1.0 have equal performance and it is not evident which algorithm is best among both. For f_3, f_4, f_5, f_6, f_9 *and* f_{10}, CV 1.0 is the best and only GWO is slightly competitive to it. No other algorithm is able to match the performance of CV 1.0. For f_8, FA and CV 1.0 are very competitive where as others provide highly deviated solutions. For f_{11}, the best, worst and mean of CV 1.0 and GWO are similar and difference lies only in the standard deviation part. Here CV 1.0 has slightly lower standard deviation as compared with GWO. For f_{12}, GWO is the best algorithm and for f_{13} *and* f_{14} again CV 1.0 provides better results. For f_{15} and f_{16}, CV 1.0 is competitive but for these functions, FA provides the best results. For f_{17}, f_{18} *and* f_{19} CV 1.0 provide the best results. For the fixed dimension function, CV 1.0 has the best performance except for function $f_{25}$where results of DE are highly competitive. So, from the twenty six function used in this comparison, CV 1.0 is found to be better for 19 functions, GWO for four functions, FA for two functions whereas DE, BA and FPA were found to be best for none.

For a generalized algorithm, because of extensive exploration in the initial stages, the convergence is slow due to smaller steps at the start but as the algorithm approaches towards the final stages, intensive exploitation start, making the algorithm to move faster. This property makes the algorithm to converge faster towards the later stages. Both extensive exploration and exploitation in any generalized algorithm ensures that the algorithm is efficient in achieving a balanced exploration and exploitation. From the convergence profiles in the Fig. 3.1, it is evident that CV 1.0 is a very good algorithm with the ability of extensive exploration, intensive exploitation and maintaining a balance between the two. Thus, both comparative results from Table C.10 and convergence plots form Fig. 3.1 validate the hypothesis of CV 1.0 to be considered in the category of state-of-the-art algorithms.

3.3.4 Statistical testing

To test the performance of the proposed CV 1.0 version, Wilcoxon rank-sum test is performed (Derrac *et al.*, 2011). This test uses two different samples of population and returns a p-value corresponding to the two. This p value determines the significance of the two population sets. In terms of algorithmic studies, two algorithms are tested using the same method and corresponding p-value is checked at 5% level of significance. If two algorithms satisfy these conditions, they are said to be statistically significant. For present case, the significance of CV 1.0 is to be calculated with respect to others to prove its efficiency statistically. This is done by comparing CV 1.0 with BA, CV1.0/DE, CV 1.0/FA, CV 1.0/FPA and CV 1.0/GWO. Since we cannot compare the best algorithm with itself, so NA (Not applicable) has been incorporated in its place. The '$\sim$' sign has been added where ever two algorithms provide same statistical results. These algorithm are either identical or don't have any statistical relevance so can't be compared (Salgotra and Singh, 2017). The statistical results are presented in Table D.1. It can be seen from the table that CV 1.0 provides better statistical significance of more than 80% of the cases and hence making it potential candidate for becoming a state-of-the-art algorithm.

3.3.5 Effect of dimension size

From the comparative analysis, it is imperative that CV 1.0 is best fit candidate for becoming a state-of-the-art algorithm. But for any algorithm to be called as a generic problem solver, it should be able to provide good results on higher dimension problems.

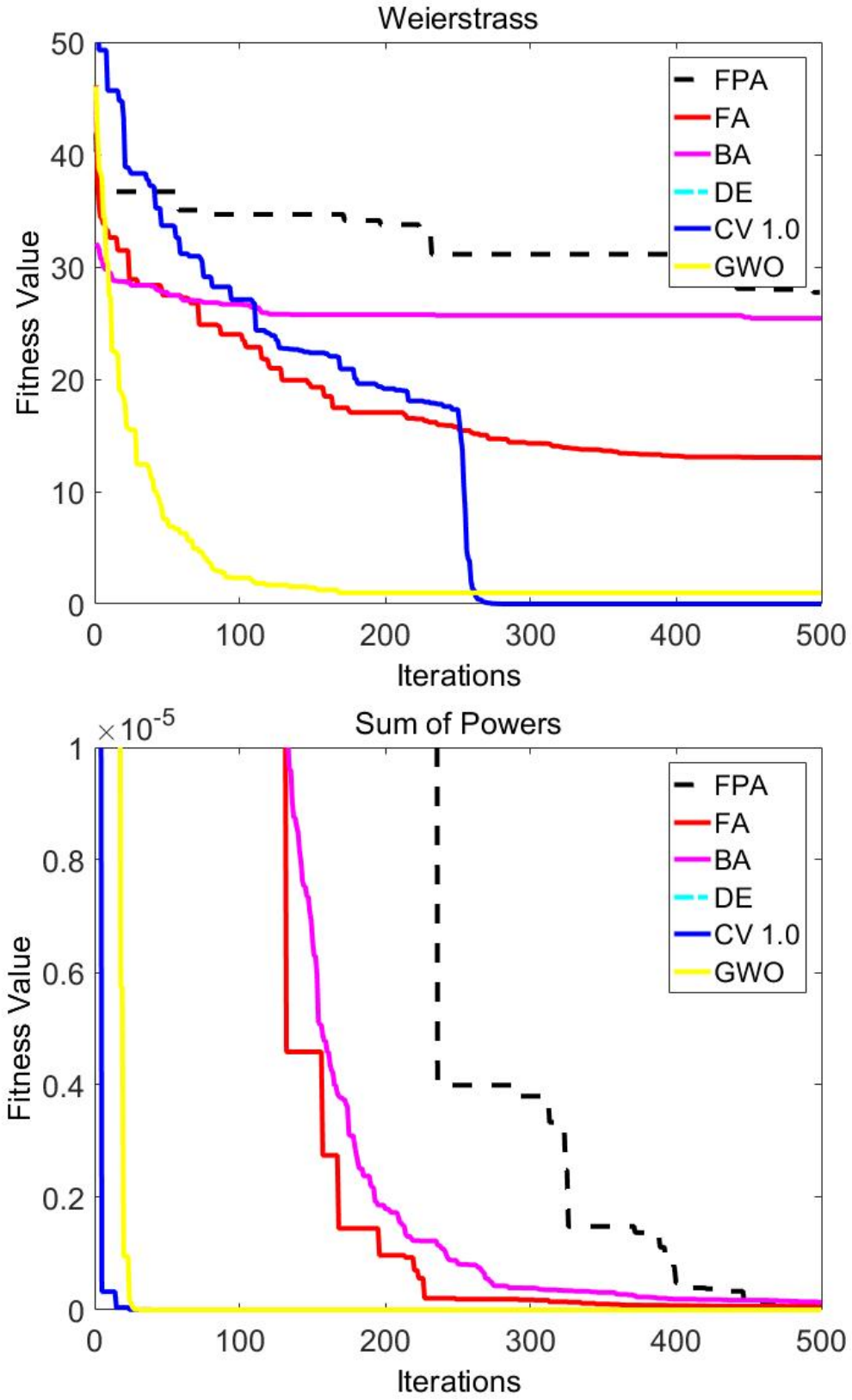


Fig. 3.1 CV1.0 algorithm: Convergence curves with respect to other algorithms

So, to test this hypothesis, CV 1.0 is tested on higher dimension functions. Here seventeen functions from CEC 2005 benchmarks of consisting of uni-modal and multi-modal sets from Table A.2 are used and four different dimension size is used to validate the results. The dimension set include 30 dimension functions whose comparison is shown above. For analysing the effect of dimension, dimension size 50, 100, 500 and 1000 is used. Table C.11, Table C.12, Table C.13 and Table C.14 present the results of dimensional analysis. It is evident from the tables that though the performance of CV 1.0 reduces to some extent with increase in dimension but with respect to the basic dimension size of 30, the decrease in results in marginal. Also, the solution quality with respect to other algorithms doesn't deviates from standard results in Table C.10. This shows that even with change in dimension size, the algorithm is able to provide global optimal results. This is because of the presence of enhanced global and local search phases. On a whole, we can say that CV 1.0 is a very competitive algorithm and performs very well for higher dimension functions.

3.3.6 Summary of results

The main findings of results are discussed as

- The basic CS algorithm, though is competitive, but is limited in scope. Also, the already available works on either applications or modifications, don't provide proper justification.
- Three versions of CS algorithms namely CV 1.0, CV 2.0 and CV 3.0 are proposed in this work and CV 1.0 is found to be best among them.
- The performance of CV 1.0 is better because of the presence of Cauchy based exploration which because of its heavy tail, searches the search space in a much better way and helps to improve the exploration of basic CS algorithm. Further, use of dual local search makes the algorithm efficient in exploiting the search to the best of its potential. Both these phases add up to improve the performance of CS algorithm.
- Another reason for the improved performance is the division of generations into two halves, which helps to infuse a proper balance between exploration and exploitation. Thus, combination of these two modifications makes CV 1.0 an efficient tool for

optimization problems.

- Apart from finding a new state-of-the-art algorithm, this chapter also finds that the switch probability is an important factor in deciding the solution quality of basic CS algorithm. Too low and too high values of switch probability degrade the performance of CS whereas for mid-range of values, the change in results is marginal.

- From the results showing the effect of varying population and dimension sizes, it is evident that the proposed version is a very efficient tool in finding global optima even with lower population and higher dimensional problems.

- Though CV 1.0 is very efficient in performance but due to its stochastic nature, it is not guaranteed that it will find the global optima for all the cases. It can be seen from the comparison table that it didn't perform well on all the benchmark functions. So, more work is to be done to design a standard problem solver for all optimization problems.

- A balanced exploration and exploitation is the basic for this proposal, but the algorithm can get stuck in local minima at later stages due to less extensive global search. Also for a balanced exploration and exploitation, the algorithm should be able to explore and exploit in every iteration. This can be considered as a drawback and more work should be done to improve the balance between exploration and exploitation phase.

- Though the algorithm has some limitations but it has proven its worth with respect to other state-of-the-art algorithms. The algorithm because of its simple structure is best fit candidate for inclusion in hybrid intelligent and expert systems.

- Exploration and exploitation can be improved to make the algorithm fit for all engineering design problems.

- Designing binary version of the proposed work can extend its use to the field of medical science, wireless sensor networks and others. In the field of medical science, binary CV 1.0 can be used in electroencephalogram (EEG) for reducing the total number of sensors required for its operation while maintaining the accuracy. In

wireless sensor networks, it can be used to find the energy efficiency of different nodes in the system.

- The proposed versions can also be extended to solve some multiobjective optimization problems. The multiobjective CS algorithms are very limited in number. In future some efficient multiobjective CS algorithms will be developed which can be further applied for solving various engineering design problems.

- In future the convergence properties of CS algorithms will also be studied. Some theoretical analysis will be done in this direction.

3.4 Concluding Remarks

This chapter dealt with enhancing the performance of CS by proposing three new variants of the original algorithm. One of the problems with CS is that it may converge prematurely due to low exploration capability in some conditions. To avoid this problem, Cauchy operator has been used in the proposed modified versions of the CS. A new notion of dividing the population and generations has been done so as to balance diversification and intensification. These modifications have resulted in significant enhancement of performance of CS. The enhanced versions of CS namely CV 1.0, CV 2.0 and CV 3.0 have been tested on various benchmark functions. A study of increase in effect of population has been carried out and it is observed that the improvement is modest with respect to the increase in the computational cost. The new algorithms have been also subjected to same benchmark functions having high dimensions and again there was little effect in their performance showing their robustness to optimize functions with large variables. Apart from this, different experiments were carried out to see the effect of probability switch on the performance of basic CS and the improved algorithms. The basic CS is found to be very sensitive to probability switch with its best performance at mid values of switch. But in case of all the three new algorithms, there was hardly any change in the performance of algorithms in finding the global optima. This is due to the fact that due to division in population and generations the exploitation and exploration are balanced and switch does make any difference, hence the requirement of tuning switch for different problems is eliminated. Non-parametric statistical tests have been carried out to prove the superiority of the modified version and it is found that CV 1.0 algorithm outperforms

state-of-the-art algorithms.

For future work, a balanced exploration and exploitation is one of the cause of poor performance of population based algorithms and this can be removed by employing the concept of division of population and division of generations to other algorithms. Also, instead of using the operators of GWO, more search operators available in literature can be analysed. Also to improve the global search combined mutation operators can be applied. The proposed algorithms can be extended for solving multi-objective optimization problems and also can be applied for developing some clustering approaches for application in some real-life problems like satellite image segmentation, gene expression data clustering, cancer classification etc.

CHAPTER 4

Enhanced CV_{new} Algorithm

Preface

This chapter presents an improved version of CS namely CVnew in which three modifications are proposed. The first modification is the introduction of two new search equations to improve the global search while the second one deals with the incorporation of four search equations to improve the local search. As a third modification, a balance between global and local search has been increased by exponentially decreasing the switch probability. The proposed algorithm has been applied to solve single objective real-parameter problems of CEC2017. The numerical results prove the better performance of CVnew in comparison with respect to others.

Publication: **Rohit Salgotra**, Urvinder Singh, and Sriparna Saha. "Improved Cuckoo Search with Better Search Capabilities for Solving CEC2017 Benchmark Problems." In 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-7. IEEE, 2018.

4.1 Introduction

In CS, global search and local search are the two important phases and both these phases signifies the exploration and exploitation phase respectively. Here exploration means exploring the search space or simply searching every corner of the whole search space whereas exploitation corresponds to exploring a particular area of search space. The global phase or the exploration phase in case of CS is controlled by the use of Lévy flights based component whereas local search is followed by using random solutions. Both local and global phases are required in proper balance for an algorithm to be efficient (Salgotra and Singh, 2018). In case of CS, this balance is achieved by the use of switch probability. A higher value of switch probability may lead the algorithm toward more intensive local search and a lower value may lead to more extensive global search (Salgotra and Singh, 2017). So a proper value of probability needs to be identified in order to maintain a balance between the two phases. So we can say that in CS, there are three important parameters or conditions which needs to be met for its proper functioning. The first such parameter is the Lévy flights component which decides the extent of exploration. This component is same as the scaling factor used by the other algorithms. The second parameter is the uniformly distributed random number of the local search phase and this parameter decides the extent of exploitation. The third parameter is the probability which controls both exploration and exploitation. In CS, it has been found that it satisfies all these requirements and hence has the ability to converge towards the global solution without getting trapped in the local minima.

One of the recent introduction to CS is the cuckoo version 1.0 (CV1.0) (Salgotra *et al.*, 2018b). The algorithm has been inspired from the Cauchy based global random walk and dual division based local search. The algorithm for its validity was checked on CEC 2005 and CEC 2015 benchmark problems and it was found to provide near optimal results. The algorithm has proved it worth with the likes of DE (Storn and Price, 1997), FA (Yang, 2010a), GWO (Mirjalili *et al.*, 2014) and others. As far as CEC benchmarks are concerned, they are the most widely used benchmark problems and have been used by a large number of researchers to efficiently test their algorithms. In present work, a new extension of CV1.0 algorithm has been proposed and tested on CEC 2017 benchmark

problems. Before the introduction of new algorithm, let us add a brief overview of CV1.0 algorithm. The CV1.0 algorithm is based on the Cauchy distribution based global search, normal distribution based local search and a standard value of probability for controlling the exploration and exploitation. The algorithm also uses the concept of division of population and iterations in order to add a balance between the exploration and exploitation capabilities. Though the algorithm is able to achieve better results for most of the cases but from the inferences it has been found that there is still scope of some modification. So in present work, a new CVnew has been proposed by modifying the original CV1.0 algorithm. The proposed approach adds modification in global search phase and local search phase by employing division of population and iterations. In global search phase, this has been achieved by using Cauchy based operations for certain half of the population and using GWO based operations for the other half (Salgotra *et al.*, 2018b). In local search phase, dual division of population is followed by employing new search equations. The modification is also carried out in the probability parameter by exponentially decreasing the same with respect to iterations. Moreover, division of iteration has also been added to further improve the searching capabilities of the proposed approach.

The remainder of chapter is organized as follows. The section 2 deals with the proposed approach, section 3 gives the simulation results and the experimental setup and in the final section 4 the conclusions are summarized.

4.2 The Proposed Approach

Since CV1.0 is found to a powerful algorithm, a new modification based on CV1.0 is expected to provide better results. So in this section, CVnew is proposed. Before starting the proposed algorithm, let us highlight the major difference between CV1.0 and CVnew algorithms. Firstly in the global search phase, CV1.0 uses two search equations to find a new solution and this new solution is then fed to a third equation to find the global solution. By using such a complex pattern of equations, the complexity of algorithm increases many folds. But in case of CVnew only two search equations are updated in order to find the global solution, the first equation used is the Cauchy search equation whereas the second search equation is the equation inspired from the GWO algorithm. Both the equations are independently used to find the global solution and hence aim at

exploring the search space. Secondly, for the local search phase, the CV1.0 algorithm uses two search equations to exploit the solution whereas CVnew algorithm uses four search equation, along with addition of some new parameters, so as to improve the exploitation capabilities of CV1.0 algorithm. The third factor is the probability switch which helps the algorithm in achieving a proper balance between the global and local search. In CV1.0 this parameter is chosen randomly with 0.5 as optimal value for most of the cases. A value of 0.5 means that the algorithm aims at achieving 50% local and 50% global search but in broader sense, an optimization algorithm should have higher exploration rate towards the starting iterations and better exploitation towards the end of iterations. Thus in order to achieve better exploration and exploitative balance, CVnew employs exponential decreasing distribution. The reason for the use of exponential decreasing function is explained later in the text.

The proposed CVnew algorithm starts by initializing a set of N cuckoos within the searching capabilities of problem under test.

$$N_{i,j} = N_{min,j} + rand(0,1).(N_{min,j} - N_{max,j}) \tag{4.1}$$

where $i \epsilon \{1,....N\}$, $j \epsilon \{1,.....D\}$, $N_{i,j}$ is the i^{th} solution in the j^{th} dimension; $rand(0,1)$ is randomly generated number between [0,1]; $N_{min,j}$, $N_{max,j}$ are lower and upper bounds of the problem under test respectively and D is the dimension of the problem under consideration.

The second step in this algorithm is to define the global search phase and local search phase. Both these phases uses the concept of division of population and generation in order to find the global solution. The basic idea for this modification has been inspired from the concepts discussed in [18]. The concept of division of population has been added to diversify the solution during the initial set of population and later converge them towards a global solution in the later stages. Also, division of iterations on the other hand is followed to maintain a balance between the global and local search phase. The concept of division of population and iterations is followed in both local and global search phase and details about the modification are followed as.

In the global search phase, during the first half of the iterations, the basic concepts of CS

are used. Here Lévy flight based random search is changed to Cauchy based search. The general search equation of global search phase for CS is given by

$$x_i^{t+1} = x_i^t + \alpha \otimes L(\lambda)(x_{best} - x_i^t) \tag{4.2}$$

where x_i^t is the previous solution, x_{best} is the current best solution, $\alpha > 0$ is the step size and λ is the Lévy distributed random number.

In order to modify the search equation, Lévy flights component is changed to Cauchy distributed component by using Cauchy distributed random number instead of Lévy flight based random number. The new equation is given by

$$x_i^{t+1} = x_i^t + \alpha \otimes Cauchy(\delta)(x_i^t - x_r^t) \tag{4.3}$$

This Cauchy distributed number is generated by using Cauchy density function given by

$$f_{Cauchy(0,g)}(\delta) = \frac{1}{\pi}\frac{g}{g^2 + \delta^2} \tag{4.4}$$

The Cauchy distribution function is given by:

$$y = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{\delta}{g}\right) \tag{4.5}$$

where g is the scaling factor and its value is 1 and $y \in [0,\ 1]$. For finding the value of δ, Eq. (4.3) is solved and we get

$$\delta = \tan\left(\pi(y - \frac{1}{2})\right) \tag{4.6}$$

Eq. (4.6) will generate Cauchy based random number in the range of 0 to 1. The idea for this adaptation has been followed from the concepts used in (Yao *et al.*, 1999). The reason for the better performance of Cauchy based global search is because of the fatter tailed structure of Cauchy distribution, which helps the operators to generate larger steps and hence improves the explorative tendencies of the algorithm.

For the local search phase, the local random walk search equation of basic CS is used and is given by:

$$x_i^{t+1} = x_i^t + \alpha \otimes H(p - \epsilon) \otimes (x_g^t - x_h^t) \tag{4.7}$$

where and $\otimes$ is entry wise multiplication, p is the probability, ϵ is a simple random variable and x_g^t *and* x_h^t are random solutions selected from the population.

For second half of the iterations, dual division of population is followed in the global search phase whereas four fold division of population is followed in the local search phase. By dual division in the global search phase, we mean that two search equations are used to find the new solution. For first half of the population, Eq. (4.3) is used as such but for the second half, this equation is modified by using the concepts of GWO. In GWO new solution is generated by using three best positions of search agents. This concept is modified slightly in case of CS by using the best solution instead of best position to find the new solution and equations regarding the same are given by

$$x_1 = x_i - A_1(C_1.x_{best} - x_i^t);\ x_2 = x_i - A_2(C_2.x_{best} - x_i^t)$$

$$x_3 = x_i - A_3(C_3.x_{best} - x_i^t)$$

$$x_{new}^{t+1} = \frac{x_1 + x_2 + x_3}{3} \tag{4.8}$$

Here x_{new} is the new solution and A_1, A_2, A_3 *and* C_1, C_2, C_3 belongs to A and C respectively. A and C are given by

$$A = 2a.r_1 - a; C = 2.r_2 \tag{4.9}$$

where a is a linearly decreasing random number in the range of [0, 2] and it changes with respect to iterations, r_1 *and* r_2 are uniformly distributed random numbers in the range of [0,1]. This search equation helps the second half of the population in finding more intensive solutions and ultimately reach the global minima.

In the local search phase, four fold division of population is followed that is four search equations are used to formulate the new solution. The first search equation is same as that used by CS algorithm that is equation Eq. (4.7) whereas the second one is from CV1.0 algorithm and is given by

$$\mathrm{x_i^t}=\mathrm{x_i^t}+ F.(\left(\mathrm{x_j^t}-\mathrm{x_k^t}\right) + (\mathrm{x_l^t}-\mathrm{x_m^t})) \tag{4.10}$$

where x_j^t, x_k^t, x_l^t *and* $x_m^t$are four random solutions and F is the scaling factor. This scaling factor is generated by using an ensemble of parameter adaptation using sinusoidal decreasing adjustments and has been inspired from LSHADE-EpSin algorithm (Awad *et al.*, 2017). The first configuration is sinusoidal decreasing function and is given by

$$F_i^{t+1} = \frac{1}{2} \times \left(\sin\left(2\pi \times freq \times t + \pi\right) \times \frac{t_{max} - t}{t_{max}} + 1 \right) \tag{4.11}$$

The second configuration is sinusoidal increasing adjustment and is given as

$$F_i^{t+1} = \frac{1}{2} \times \left(\sin\left(2\pi \times freq \times t + \pi\right) \times \frac{t_{max} - t}{t_{max}} + 1 \right) \tag{4.12}$$

where *freq* is the frequency of sinusoidal function and is set to be a fixed function, t is the current iteration or generation and t_{max} is the maximum number of iterations.

Apart from these search equations, for third half of population, a new search equation inspired from the concepts of FPA (Yao *et al.*, 1997) is used and is given by

$$x_i^{t+1} = x_i^t + \epsilon.(x_i^t - x_j^t) \tag{4.13}$$

where ϵ is a uniformly distributed random number in the range of [0, 1]. The final search equation for the fourth part of population is given by

$$x_i^{t+1} = x_i^t + a.\left(x_{best}^t - x_j^t\right) + b.(x_k^t - x_l^t) \tag{4.14}$$

where a *and* b are uniformly distributed random numbers in the range of [0, 1]. The main reason for the use of this search equation is that one of the random solution will be guided by the best solution whereas other two solutions help them in moving toward the global solution.

Overall, two search equations in the global search phase will add diversity among the search agents and hence perform more extensive global search whereas four search equations of the local search phase help the algorithm in exploiting all the major areas of the search space. Thus a balance combination of both global and local search phases helps the algorithm in moving towards the final solution and hence improves the searching capabilities of CS. Apart from these modifications, in order to add a balance between the

exploration and exploitation components, the probability parameter is selected by using an exponentially decreasing function and is given by (Chen *et al.*, 2006):

$$p(t) = p_{min} + (p_{max} - p_{min}) .e^{-(\frac{t}{t_{max}/10})} \tag{4.15}$$

where p_{min} and p_{max} are taken in the range of [0,1], t is the current iteration, e is the exponential function and t_{max} is the maximum number of iteration or generations. The reason for the use of exponentially decreasing function is that it helps the algorithm in converging slowly during the initial iterations and change faster towards the final stages. This further helps the algorithm in maintaining a balance between the global and local search phase. Apart from this if the total number of iteration size becomes greater than three-fourth of the total iteration size, the population size is reduced by one fourth. This has been done to reduce the total number of function evaluations required to evaluate the final solution. The pseudo-code of CVnew is shown in Algorithm 6.

4.3 Result and Discussion

This section deals with the simulation results of the proposed approach. In this section first of all, numerical benchmark are presented. These benchmark include the recently introduced CEC2017 benchmarks (Awad *et al.*, 2016b). The second subsection presents the parameters for various algorithm, third subsection gives the algorithm complexity, fourth subsection deals with the statistical results and finally in the fifth subsection the comparison with other popular algorithms is presented.

4.3.1 Numerical Benchmarks and Parameter Settings

The performance of the proposed CVnew algorithm is tested by using the single objective bound constraint CEC2017 real parameter benchmark problems. This test suite consists of 30 benchmark problems with 1-3 as unimodal function, 4-10 as multimodal functions, 11-20 as hybrid functions and 21-30 as the composition functions. These are highly complex functions and more details about these functions can be had from (Awad *et al.*, 2016b). The initial population for the proposed variant is set to 50 where asp_{min} and p_{max} are taken to be 0.8 and 0.25 respectively. The *freq* parameter is set to 0.5, ϵ, *a and b* are chosen randomly using uniform distribution. Apart from this, the benchmark set is tested for 10, 30 and 50 dimension size. The stopping criteria for the algorithm is the total number of function evaluations whereas the total number of runs is 51. Both the

Algorithm 6 Pseudo-code of CVnew algorithm

Begin

define initial parameters: stopping criteria and problem dimension

initialize cuckoo population (N)

Define d-dimensional objective function, *f(x)*

for i= 1: $\frac{t_{max}}{2}$ **do**

global search using Eq. (4.2)

evaluate fitness and perform greedy selection using Eq. (2.5)

Chose a nest j randomly from n initial population

if fitness of x_i^{t+1} *better than that of* x_i^t

replace j by x_i^{t+1}

local search for fraction of worst nests using Eq. (2.4) & Eq. (4.8)

evaluate fitness and perform greedy selection using Eq. (2.5)

evaluate the current best

for i= $\frac{t_{max}}{2}$+1: t_{max} **do**

global search using Eq. (4.3)

evaluate fitness and perform greedy selection using Eq. (2.5)

Chose a nest j randomly from n initial population

if fitness of x_i^{t+1} *better than that of* x_i^t

replace j by x_i^{t+1}

local search using Eq. (4.7), Eq. (4.10), Eq. (4.13) & Eq. (4.14)

evaluate fitness and perform greedy selection using Eq. (2.5)

Update probability using Eq. (4.15)

update current **best**

update final **best**

End

Table 4.1 CVnew Algorithm: Computational complexity

	T_0	T_1	$\widehat{T_2}$	$(\widehat{T_2} - T_1)/T_0$
$D = 10$	0.156244	2.8396	2.8533	0.8711
$D = 30$	0.156244	12.4593	12.6296	1.0899
$D = 50$	0.156244	29.3607	29.3184	0.2701

total number of function evaluations and number of runs is kept same as given in (Awad *et al.*, 2016b). Another reason for the use of this set of parameters is that the algorithm used for comparison, employs the same set of parameters.

4.3.2 Algorithm Complexity

The algorithm complexity of CVnew algorithm is shown in Table 4.1. The algorithm was simulated on Matlab version 2017a on a windows 10 system with Intel Xeon Processor (E5-2630), 2.20GHz with 32 GB RAM. The computation complexity of the algorithm is calculated in the same was as done in (Awad *et al.*, 2017). In Table 4.1, T_0 denotes the running time of the code given as;

for $i = 1 : 1000000$

$$x = 0.55 + double\,(i)\,; x = x + x; x = x/2; x = x \times x;$$

$$x = sqrt\,(x)\,; x = \log\,(x)\;; x = \exp\,(x)\;; x = x/(x + 2);$$

end

The other notations in the Table I are T_1 as the computing time for f_{18} for 200,000 number of function evaluations, $\widehat{T_2}$ is the mean running time for f_{18} for a total of five runs, with a total budget of 200,000 function evaluations per run. The final column representing $(\widehat{T_2} - T_1)/T_0$, gives the total complexity of the proposed approach for $D = 10,\ 30\ and\ 50$, where D corresponds to the problem dimension.

4.3.3 CEC 2017: Statistical results for 10D, 30D and 50D

The algorithm was run 51 times for each test problem with total computational burden given by $10,000 \times D$. The results are calculated by computing the difference between the best found solutions with the optimal solution and when this difference becomes less than

10^{-8}, the error is treated as zero. The statistical results of the proposed CVnew algorithm are computed for $D = 10,\ 30\ and\ 50$, and are presented in Table C.15, Table C.16 and Table C.17 respectively. Each Table gives the best, worst, mean and standard deviation over 51 runs of the error values. These values are found by computing the fitness values between the desired value and the true value obtained after each run. For unimodal functions, the algorithm was successfully able to obtain the optimal solution for $D = 10$. For multimodal functions, the algorithm was able to obtain good solutions except for F_{12} in which the performance was reduced with increase in D. With regards to hybrid and composite functions, performance of CVnew was worse than other functions with increase in dimension size. The main reason for the lower performance of these functions is because of more number of local optimum solutions prevalent in these solutions.

4.3.4 CEC 2017: Comparison results with respect to other algorithms

In this subsection, the performance of the CVnew algorithm is compared with the performance of other state-of-the-art algorithms as CV1.0 (Salgotra *et al.*, 2018b), Self-adaptive differential evolution algorithm for numerical optimization (SaDE) (Qin *et al.*, 2008), adaptive differential evolution with optimal external archive (JADE) (Zhang and Sanderson, 2009), differential evolution with success-history based parameter adaptation (SHADE) (Tanabe and Fukunaga, 2014) and mean-variance mapping optimization (MVMO) (Erlich *et al.*, 2014). The results pertaining to SaDE, JADE, SHADE and MVMO are taken from LSHADE-cnEpSin (Awad *et al.*, 2017) algorithm and are for 50D and 51 independent runs. The best error values of CVnew when compared to SaDE, JADE and SHADE are better and for MVMO the results are comparable. The Wilcoxon's rank-sum test has also been performed in the last row of Table V to judge the significance of CVnew statistically (Derrac *et al.*, 2011). The results of this test are performed at a level of significance of 0.05 and comparison is done with respect to $(w/l/t)$ where $w,\ l\ and\ t$ signifies the win, loss and tie condition for a particular algorithm and is denoted as $w(+$:win$)/\ l(-$: loss$)/\ t(=$:tie$)$. Here in Table C.18, "+" signifies the condition where performance of the particular algorithm is better than the proposed algorithm, "−" signifies the condition where performance of an algorithm under comparison is worse than the proposed algorithm and "=" signifies the condition where both the proposed and the

algorithm in comparison are having same results. Also from the comparison results, it can be seen that CVnew performs better than SaDE for 18 functions whereas for 12 functions SaDE performs better. With respect to JADE and SHADE, CVnew is better for 17 functions each and for MVMO it has an equivalent performance. Overall we can say that CVnew is better than SaDE, JADE and SHADE but is equally competitive with respect to MVMO. With respect to CV1.0, the means and standard deviation of CVnew is competitive and comparatively better. Apart from these results, the main aim of the chapter was to prove the worthiness of other meta-heuristic in line with the DE variants. It should be noted that if serious effort is given for the improvement of other algorithms such as CS, they can also prove their competitiveness both qualitatively and quantitatively.

4.4 Concluding Remarks

This chapter introduces CVnew algorithm in which division of iteration and dual division of population is followed in the local and global search phase to improve the exploration and exploitation capabilities. The algorithm also enhances the switch probability by using linearly decreasing function in order to ensure a balance between the global and local search phases. Moreover, the concepts of Cauchy based scaling factor has also been introduced to further enhance the performance. The proposed algorithm was tested on benchmark problems of IEEE CEC2017 bound constraint single objective real parameter numerical optimization problems. When compared with other state-of-art algorithms, the CVnew algorithm showed highly competitive results. For future directions, the algorithm can be extended to real world problems in various fields of research.

CHAPTER 5

Extended CSsin Algorithm

Preface

In this chapter, a new enhanced version of CVnew algorithm is proposed, named as CSsin and it employs four major modifications, *i*) new techniques for global and local search, *ii*) dual search strategy, *iii*) a linearly decreasing switch probability, and *iv*) linearly decreasing population size. Apart from these modifications, the division of iterations has been employed as a further modification. The CSsin algorithm has been tested on IEEE CEC 2017 and CEC 2020 benchmark test problems. The numerical results prove that the newly proposed CSsin algorithm performs better with respect to other major algorithms.

Publication: **Rohit Salgotra**, Urvinder Singh, Sriparna Saha, and Amir H Gandomi. "Improving Cuckoo Search: Incorporating Changes for CEC 2017 and CEC 2020 Benchmark Problems." Accepted In 2019 IEEE Congress on Evolutionary Computation (CEC).

5.1 Introduction

CS is very much affected by the choice of parameters used and is highly dependent on them (Yang and Deb, 2009). The major parameters used are the Lévy flight component or scaling factor (**L**), a uniform random number and the switching probability (**p**). Based on this fact, a large number of CS algorithms have been proposed in the recent past to enhance the performance of CS and improve its working capability (Fister *et al.*, 2014). Most of the studies have improved either the Lévy flight component or switching probability but less work has been done to enhance local search phase. Cuckoo version 1.0 (CV 1.0) is one such recently introduced algorithm which has been applied to CEC 2005 and CEC 2015 benchmarks for real parameter optimization (Salgotra and Singh, 2017). CV 1.0 is proven to be very powerful algorithm and has proven its worth in line with differential evolution (DE) (Storn and Price, 1997), grey wolf optimization (GWO) (Mirjalili *et al.*, 2014), and others. The algorithm used Cauchy and Normal distribution instead of simple Lévy flight based component of exploration and uniformly distributed exploitation random numbers respectively. The algorithm though used a standard value for p but still was capable of providing better results. This algorithm also used concepts of division of population and generations to achieve a proper balance between the extent of exploration and exploitation. But from the inference of CV 1.0, it can be found that there is still scope of modification. So in (Salgotra *et al.*, 2018a), the authors have modified the original CV 1.0 algorithm by employing parameter adaptation and population reduction with respect to generations, to design a new algorithm namely CVnew. The proposed approach uses Cauchy based scaling factor along with equation modification using GWO in the exploration phase and dual division strategy along with sinusoidal adaptive decreasing adjustment (Awad *et al.*, 2017) for enhancing the exploitation phase. The third parameter that is probability was also adapted by using exponentially decreasing distribution (Salgotra *et al.*, 2018a). The Cauchy based scaling and GWO equations because of their heavy tailed nature, tend to potentially explore the search space whereas dual division strategy and ensemble of sinusoidal waves is used to adapt the parameters in an efficient manner. Both Cauchy based random number for global phase and dual division strategy for local phase is followed for first half of the iterations whereas for second half Cauchy based adaptation along with GWO equations for exploration and sinusoidal based adaptation for exploitation is

followed (Salgotra *et al.*, 2018a).

In present work, the CS algorithm has been modified to test its performance on the CEC 2017 (Awad *et al.*, 2016a) and CEC 2020 (Yue *et al.*, 2019) problems. The newly proposed algorithm has been named as CSsin algorithm and uses dual population division in both local and the global searching phases. The major reason for the use of dual division of population is because of the extra computational burden on the algorithm in solving two more equations for finding the final solutions. Since most of the work aims at providing the better performance at minimal computational cost, so here CSsin algorithm aims at providing the same when compared to its counterpart CS. The same point can be validated from, where three versions of CS were designed and it was found that there were marginal variation of results in all the proposed algorithm for different population divisions. So a slightly better solution can be compensated if the computational burden can reduce significantly. Another modification which has been proposed in the CSsin algorithm, is the use of linear decreasing probability instead of constant probability. The reason for the use of linearly decreasing probability is that it decrease the probability marginally with respect to iterations and there is not much variation in the probability over the course of iterations and hence limit growth of algorithm from exploration to exploitation gradually with time. But in case of exponentially decreasing function, the variation is slow at the start of iterations but as the solution approaches final stages it converges at a faster pace, making the algorithm getting stuck in some local minima. The third modification which has been added in the proposed version is the change in the total population with respect to iterations. That is, with increase in iterations or generations, the population size is reduced in order to restrict the maximum function evaluations.

The rest of the chapter is organized into three sections, including proposed algorithm in the next section. The third section deals with experimental results and discussion where as in the final section 4, conclusion and future work are presented.

5.2 The Proposed Approach

In this section, the proposed CSsin algorithm is elaborated in detail. The algorithm firstly starts with initialization of a random population of N cuckoo birds with respect to the search range of the problem under consideration. After the initialization step, two

concepts based on population division and iterations are employed. The basic idea for these concepts have been motivated from (Salgotra *et al.*, 2018b) and it has been found that both these conditions have helped the algorithm in successfully finding the global optimal solution. The population division has been followed to add diversity among the positions of search agents during initial steps and converge towards the final steps. On the other hand, iterative division is added for achieving a balanced exploration and exploitation. This modification is followed in both local and global search phase and detailed discussion has been presented as follows.

For the first iterative half, dual population division is employed for both global and local search. Here Cauchy distributed global search is followed during the first half of the population and for the second half is modified using concepts of GWO algorithm (Mirjalili *et al.*, 2014) to generate the new solution. The Cauchy based solution is generated by using Eq. (5.1)

$$f_{Cauchy(0,g)}(\delta) = \frac{1}{\pi}\frac{g}{(g^2+\delta^2)} \tag{5.1}$$

$$y = \frac{1}{2} + \frac{1}{\pi}arctan(\frac{\delta}{g}) \tag{5.2}$$

$$\delta = tan(\pi(y - \frac{1}{2})) \tag{5.3}$$

$$x_i^{t+1} = x_i^t + \alpha \otimes Cauchy(\delta)(x_{best} - x_i^t) \tag{5.4}$$

where scaling factor g is equal to 1, δ is the Cauchy random number in the range of [0, 1].The main reason for using Cauchy based distribution is its fatter tail generating larger steps for exploring search space in a much better way. Also in CS, the final solution is guided by the best solution of the current iteration, so there are chances that it may fall in some local minima, ultimately leading to premature convergence. Thus Cauchy based random number will help the algorithm in performing extensive exploration. For second half of the population, three different solutions from the search pool are taken to find the new solution. These three different solutions are generated by using current solution and equations regarding the same are given by Eq. (5.6)

$$x_1 = x_i - A_1(C_1.x_{new} - x_i^t); x_2 = x_i - A_2(C_2.x_{new} - x_i^t); x_3 = x_i - A_3(C_3.x_{new} - x_i^t); \tag{5.5}$$

$$x_i^{t+1} = \frac{x_1 + x_2 + x_3}{3} \tag{5.6}$$

where A_1, A_2, A_3 and C_1, C_2, C_3 are given by $A = 2a.r_1 - a$; $C = 2.r_2$, a is linearly decreasing whereas $r_1 and r_2$ are uniform distributed random numbers.This search equation because of the presence of three different solution helps in providing more intensive global search and ultimately paves way for better performance of the proposed algorithm.

In the local search phase, population division is controlled by using a new parameter namley F and this search equation has been derived from the search equation used by CV1.0 (Salgotra *et al.*, 2018b) and is given by Eq. (5.9)

$$F_i^{t+1} = \frac{1}{2} \times (sin(2\pi \times freq \times t + \pi) \times \frac{t_{max-t}}{t_{max}} + 1); \; if \; r_1 > 0.5 \tag{5.7}$$

$$F_i^{t+1} = \frac{1}{2} \times (sin(2\pi \times freq \times t) \times \frac{t_{max-t}}{t_{max}} + 1); \; if \; r_1 < 0.5 \tag{5.8}$$

$$x_i^{t+1} = x_i^t + F.((x_j^t - x_k^t) + (x_l^t - x_m^t)) \tag{5.9}$$

where F is the scaling factor, $x_j^t, x_k^t, x_l^t, x_m^t$ are four random solutions. The scaling factor F has been derived from (Awad *et al.*, 2017). Here because of the use of a single scaling factor, the search equation becomes efficient for smaller sections of search space and hence make the algorithm perform better exploitation. Also, instead of using a simple random F, the local search phase uses a new ensemble of parameters for adaptation by using sinusoidal decreasing F as used by LSHADE-cnEpSin (Awad *et al.*, 2017). The frequency in case of LSHADE-cnEpSin is generated using Cauchy distribution whereas for present case uniform distribution is followed to generate the same. All these modifications have been added for the second iterative half, whereas for the first half, basic equations of CS have been used.

Apart from these modifications, the probability parameter p which controls the extent of exploration and exploitation is also adapted by linearly decreasing its value from a high value of $p_{max} = 0.75$ to $p_{min} = 0.25$ by using Eq. (5.10)

$$p = p_{init} - \frac{(p_{max} - p_{min})}{t_{max}} \tag{5.10}$$

Here p_{init} is the initial value of p and is taken to be 0.75, whereas t_{max} is the maximum iteration size. This equation has been inspired form the perturbation rate equation used by modified spider monkey optimization algorithm [24]. The reason for the use of this adaptation is because of the requirement of less extensive exploitation towards the start of the iterations and more intensive local search towards the end. A higher value of p allows the algorithm in performing more extensive local search whereas lower values leads to lesser intensive local search. But an adapted value of p may help in balancing the two search phases and hence provide proper exploitation operation toward the end of the iterations. Furthermore, adaptive linear population reduction (Tanabe and Fukunaga, 2014) is also followed to reduce the total computational burden. Note that this population is firstly sorted and only the worst members of the population are eliminated. The general equation for population reduction is given by Eq. (5.11)

$$N(g+1) = round[(\frac{N_{min} - N_{max}}{FEs_{max}}).FEs + N_{max}] \tag{5.11}$$

where values for N_{max} and N_{min} are set to $18 \times D$ and 4, respectively, FEs_{max} are maximum number of function evaluations. The minimum population size is kept 4 because only 4 individuals are required at the minimum to perform global and local search operations. This modification helps in creating a self adaptive population step and hence reducing the computational complexity of the algorithm. In the next section, detailed discussion about the experimental results is presented.

5.3 Result and Discussion

This section presents the simulation results of CSsin algorithm with respect to other state of the art algorithms for CEC 2017 and CEC 2020 benchmark test problems. The first subsection deals with the numerical benchmarks used and the PC configurations whereas in the second subsection, parametric details of the proposed algorithm in comparison to other algorithms are presented. The third subsection details about the statistical results of different dimension sizes for CEC 2017 benchmark problems and in the fourth subsection a comparison of the proposed algorithm with respect to other state of the art algorithms is presented. The fifth subsection provides the results for CEC 2020 benchmark problems and an analysis of results is also presented in that section.

5.3.1 Numerical Benchmarks and Parameter Settings

In this section, detailed discussion related to the simulation results of newly proposed CSsin algorithm are presented. For performance evaluation, simulations are performed on Windows 10 having Matlab 2017a, E5-2630 2.20GHz Intel Xeon Processor with 32GB RAM. Here CEC2017 numerical benchmark problems are used and comparison has been performed with respect to SaDE (Qin *et al.*, 2009), JADE (Zhang and Sanderson, 2009), SHADE (Tanabe and Fukunaga, 2014), mean-variance mapping optimization (MVMO) (Erlich *et al.*, 2014), CV1.0 (Salgotra *et al.*, 2018b) and CVnew (Salgotra *et al.*, 2018a). The results for SaDE, JADE, MVMO and SHADE are taken from (Awad *et al.*, 2017) whereas for CV1.0 and CVnew are taken from their respective papers. The CEC2017 test suite on the other hand consists of 30 real challenging benchmark problems with 1-3 unimodal, 4-10 multimodal, 11-20 hybrid and 21-30 composite functions. This data set is the most recent and highly complex one, consisting of all major types of optimization problems. A general discussion about these benchmark problems and their definitions can be had from (Awad *et al.*, 2016a). As far as parameter settings are concerned, the proposed CSsin algorithm consists of very few parameters when compared to its counter parts. The first parameter is the initial population and is set to 50 where as other major parameters consist of the upper and lower limits of probability that is p_{min} and p_{max} and are taken as 0.75 and 0.25. The final parameter is the *freq* which is set to 0.5. Apart from this, all other parameters are chosen as such as used by original CS algorithm. The stopping criteria was taken as $10,000 \times D$ total number of function evaluations with 51 runs performed for each test problem. The results are calculated as error values and are presented in terms of best, worst, mean and standard deviation. The error values are calculated by finding the difference expected and the desired solution and if the difference becomes less than 10^{-8}, the error is considered as zero. The results in this section are divided into two subsections. In the first subsection, the results of CSsin algorithm are presented for $D = 10$, $D = 30$ and $D = 50$ where D is the dimension size.In the second subsection, results with respect to other state-of-the-art algorithms is presented. Further Wilcoxon's rank-sum test (Derrac *et al.*, 2011) has also been done to prove the significance of CSsin algorithm statistically.

5.3.2 Algorithm Complexity

The run time complexity of the CSsin algorithm in terms of run time t_0 is calculated by using the code given as:

$for\ i = 1 : 1000000$

$x = 0.55 + double(i);\ \ x = x + x;\ \ x = x/2;\ \ x = x \times x$

$x = sqrt(x);\ \ x = log(x);\ \ x = exp(x);\ \ x = x/(x + 2)$

end

The complexity of algorithm is also shown in Table 5.1. Here the notations T_0 corresponds to the computing time for the code given above and T_1 is the computing time for F_18 function from the Test suite given in the next section, with a total number of function evaluations of 200,000. T_2 is the mean run time for the same function with same number of function evaluations for a total number of five runs.

Table 5.1 CSsin algorithm: Computational complexity

D	T_0	T_1	T_2	$(T_2 - T_1)/T_0$
10	0.1562	4.094	3.964	0.834
30		20.76	20.148	4.08
50		55.581	58.134	15.54

5.3.3 CEC 2017: Results for $10D$, $30D$, $50D$ and $100D$

The performance of the proposed CSsin algorithm is tested for $10D$, $30D$ and $50D$ for 51 runs and the total function evaluations is set to $10,000 \times D$ where D is the dimension size. The results are computed in terms of error values and this error value is calculated by evaluating the difference between the optimal solution and a standard value of 10^{-8} and if it becomes less than this value, the error is considered as zero. Table C.19, Table C.20 and Table C.21 present the results for each dimension size, respectively. The results are taken in terms of best, worst, mean and standard deviation of 51 error values.

5.3.4 CEC 2017: Results with respect to other algorithms

In this section, SaDE, JADE, SHADE, MVMO, CV1.0 and CVnew algorithms have been used for comparison with the newly proposed CSsin algorithm. The last row of the table gives the values of Wilcoxon rank-sum test (Derrac *et al.*, 2011). These results

are calculated at 0.5 significance level with comparison performed in terms of $w(+win)$ $/l(-loss)$ and $/t(= tie)$. It should be noted that " + " signifies that the algorithm under consideration is better than the proposed algorithm, " − " means the opposite and " = " means that there is either no relevance or are having same statistical results and are equivalent to each other. From the results of last row of Table C.31, it is imperative that here also the proposed algorithm is better than CV1.0, SaDE, CVnew, JADE and SHADE and is highly competitive when compared to MVMO. Thus we can say that the proposed algorithm is highly competitive and future modification in the same approach may lead to much better results. Also here CS can be considered as equally competitive in line with DE and exceptional performance can also be had from this algorithm also if serious efforts are put forth for its modifications.

5.3.5 CEC 2020: Benchmark Results

This section deals with the performance evaluation based on CEC 2020 benchmark problems. The CEC 2020 benchmark set is the most recently introduced dataset in the field of numerical optimization (Yue *et al.*, 2019). This dataset consists of ten highly challenging optimization problems with one unimodal function, three multi-modal functions, three hybrid functions and three composite functions. The experimental setting used for testing the proposed CSsin algorithm is similar as given by (Yue *et al.*, 2019). For the 10 minimization problems, the experiments are performed 30 times for a variable dimension (D) size of 5, 10, 15 and 20. The maximum number of function evaluations for $D = 5$ is $50,000$, for $D = 10$ is $1,000,000$, for $D = 15$ is $3,000,000$ and for $D = 20$ is $10,000,000$ with a search range of $[-100, 100]^D$ for all the test functions under consideration (Yue *et al.*, 2019). This is the first termination criteria while the second being the error value smaller than 10^{-8}. The authors have used the same parameter setting has been used for evaluating the performance of proposed CSsin algorithm. From the experimental results, it has been found that CSsin algorithm performs better for $D = 5$, 10 whereas provides competitive results for $D = 15$ and 20.

5.4 Concluding Remarks

This chapter presented a new CSsin algorithm aimed at improving the performance of CVnew algorithm proposed in the previous chapter. The new algorithm employs adaptive parameters including dual search strategy to enhance exploration and exploitation proper-

ties, linearly decreasing switch probability to balance between local and global search, and linearly decreasing population size to reduce the computational burden. The algorithm is tested on CEC 2017 benchmark problems and it is found that the proposed algorithm attains very competitive results in line with SaDe, JADE, SHADE, LSHADE, CV1.0, CVnew and other algorithms. Also for CEC 2020 numerical benchmark problems, the algorithm is able to achieve competitive results. For future directions, more works can be done to enhance the performance of CSsin for finding good results on all the CEC 2020 benchmark problems and even the algorithm can be applied to real world problems in various fields of research.

CHAPTER 6

Self-adaptive Cuckoo Search Algorithm

Pcreface

In this chapter, in order to improve the performance of CS algorithm, a new version namely self-adaptive CS (SACS) is proposed. Here proportional population reduction based on fitness of current best and previous best solution, is followed. Secondly, the Gaussian sampling mechanism known as bare bones variant is added. The concept of Weibull distributed probability switching is also added to increase a balance between the exploration and exploitation processes. Further, CS as an extension of DE, GA and stability analysis with respect to Von Neumann's criteria is also presented. Experimental and statistical results shows the superior performance of SACS algorithm.

Publication: **Rohit Salgotra**, Urvinder Singh, Sriparna Saha and Amir H. Gandomi. "Self-adaptive cuckoo search algorithm: Analysis and experimentation." Swarm and Evolutionary Computation (Under Revision).

6.1 Introduction

Most of the works done till date focused on the parameter modifications or improving the exploration or exploitation processes. But no generalized study has been carried out to improve all the sections of CS algorithm till date to the best knowledge of the authors. One such study was conducted by (Salgotra *et al.*, 2018b), in which the basic parameters such as probability switch, global search and local search were adapted but still the other parameters such as the scale factor for global search, population size and others need to be investigated and more work is required to be done to improve the overall nature of CS algorithm. This work was further improved and a new variant was proposed in (Salgotra *et al.*, 2018a) where modification with respect to probability switch was performed and population was adapted to some extent. The proposed algorithm though is adaptive in nature but we need to define some initial and final values for the parameters to be adapted. These additional parameters, make the algorithm more complex and challenging to execute. Thus there is a need of parameter adaptation for CS where all the parameters are self-adaptive and no tuning of basic parameters is required.

In the current work, a similar adaptation of the basic CS algorithm has been incorporated. Though the work is mainly an extension of the existing CVnew algorithm (Salgotra *et al.*, 2018a) but the new proposed algorithm has various major differences. Here three new modifications have been added in addition to the modifications proposed in CVnew algorithm. The first modification includes adaptive population reduction based on the fitness of the current best and the previous best solutions. The modification has been inspired from the work proposed on genetic algorithm (GA) (Hallam *et al.*, 2010) and helps in proportional population reduction. The reason why this modification is incorporated is to make the algorithm parameter free. Second modification which has been proposed is the introduction of Weibull distributed random number (Weihull, 1951) instead of a simple random number for deciding the switching probability. The major reason for the use of Weibull distribution is that it has wide applicability and can be reduced to normal or exponential distribution depending upon the application. The distribution also has long tails and this feature is required to generate larger step sizes in any optimization algorithm. The distribution reduces in such a way that during the initial phase, it helps

the algorithm in performing extensive exploration and intensive exploitation towards the final stages. The third modification which has been added is the introduction of Gaussian sampling mechanism commonly called as bare bones variant. The bare bones algorithm variant is inspired from the bare bones particle swarm optimization (BBPSO) (Kennedy, 2003) and bare bones DE (BBDE) (Wang *et al.*, 2013) and is meant for improving the premature convergence problem of the CS algorithm. The bare bones variation helps the algorithm in achieving cooperative search by hybridizing the original search equation to reinforce the complementary search parameters and overcome the limitations of single search equation. The algorithm also uses Gaussian distributed parameters for adapting the global and local search operators. The bare bones approach has been successfully implemented in artificial bee colony algorithm (Zhou *et al.*, 2016a), water cycle algorithm (Heidari *et al.*, 2017), DE (Omran *et al.*, 2009) and various others. More details regarding the addition of these modifications have been discussed in the subsequent subsections.

The organization of the chapter includes four sections where first section is the introduction section, detailing about the basics of CS, its extensive literature and generalized motivation behind the newly proposed approach. In the second section, theoretical analysis of CS has been carried out. This analysis is carried out with respect to the well known GA (Holland, 1992) and differential evolution (DE) (Storn and Price, 1997). Computational complexities of both of these algorithms have also been analyzed. The reason for this analysis is that not much work has been done in this context and also CS is a clear adaptation of GA and DE with the only addition of exploitation phase. Even the parameters for all the three algorithms are almost similar. The third section deals with the proposed algorithm, in which concepts related to why and how the modifications have been applied are elaborated. Here first of all, the details about the drawbacks of previous work and motivation for the present work are discussed and then extensive concepts related to the proposal are elaborated and finally complexity analysis of the proposed approach is presented. The next section deals with the result and discussion and in the final section, conclusion and future directions are presented.

6.2 Theoretical Analysis of CS

CS though is a very efficient problem solving algorithm in various fields of research but it doesn't comply with the likes of genetic algorithm (GA) (Holland, 1992) and differential evolution (DE) (Storn and Price, 1997). Both of these algorithms have proven their worth and numerous modified versions of these algorithms have been proposed till date. Any algorithm which is found to provide better or competitive results with respect to these algorithms is considered as a challenging algorithm and can be added to the list of state-of-the-art algorithms. So here it is imperative to study CS algorithm with the likes of DE and GA as minimal work has been done till date in this context. Here in this section, we will be dealing with the analysis of CS algorithm with respect to other state of art algorithms and also computational complexity will also be discussed.

6.2.1 CS as an extension of GA

GA since its inception has achieved greater heights and is considered as the benchmark algorithm in the field of global optimization. The algorithm has been widely accepted and has been explored by researchers of almost every field. A typical GA consists of five different phases namely initialization, fitness function evaluation, crossover, mutation and selection. When compared with that of CS, these phases are exactly same except for crossover and mutation. Here crossover is meant for convergence operation and is meant for pulling the population towards a particular section of search space. Mutation on the other hand is meant for breaking some members of a population out of a local minimum space and hence discover a better minimum. The end goal is to have more convergence and less divergence. Hence crossover operator should be applied for more times and mutation should be applied for less time (Man *et al.*, 1996). Overall we can say that, crossover is the exploration operation which is meant for finding local minimum whereas mutation is for exploitation required to find the end solution. When compared with CS, the crossover process can be taken as the global search whereas the mutation operation is considered as the local search. The only difference is that in GA search processes, two general equations are used to find the new solution where as in CS only one equation is there to find the new solution. As far as parameters of GA are concerned, there are two parameters one is the crossover probability and other is the mutation probability. When compared with CS, the crossover probability is equivalent to the standard exploration operator that is Lévy

flights and mutation probability is equivalent to the exploitation operator ϵ of the local search phase. Apart from these, the generalized equations of both the algorithms remain almost similar. Overall, we can say that CS is an adaptation of GA with the addition of randomized probability and deletion of second search equation.

6.2.2 CS as an extension of DE

DE algorithm was proposed in (Storn and Price, 1997) and has become one of the most challenging and competitive algorithms of the present times. The algorithm though is not as widely accepted as GA but still finds application in almost all the fields of research.The DE algorithm performs exploration and exploitation using two important factors namely scaling factor (F) and the crossover rate (CR). Numerous research articles have been proposed claiming different values of F and CR for efficient performance of DE algorithm. One such study was conducted by (Ronkkonen *et al.*, 2005) where a value of $0 < CR < 0.2$ and $0.4 < F < 0.95$ were found to be efficient, while in (Omran *et al.*, 2005) values of $0.15 < F < 0.5$ and $0.1 < CR < 1.0$ were used. Some used Gaussian distributed values for F and self-adaptive values of CR (Abbass, 2002) while in (Liu and Lampinen, 2005) fuzzy based parameter adaptation has been performed. There were other studies (Qin *et al.*, 2009) where self-adaptive values of F and CR were found to be better. Overall, a general conclusion regarding the values of CR and F is that their values remain in the range of $[0, 1]$ and standard Gaussian, fuzzy or Lévy L distributed values respectively. Now when we compare the above discussed DE algorithm with the likes of CS, it can be said that the exploration phase for both the algorithms are same with parameters CR of DE similar to the p of CS and F of DE equivalent to L of CS. This conclusion is based on the fact that the exploration equation of DE is similar to the general equation of CS algorithm. Apart from that, the only addition is the exploitation phase in CS algorithm. This phase is missing in the generalized DE algorithm and overall we can say that CS is clear adaptation of DE algorithm with proper exploitation phase added to it. Here with respect to parameters, the scaling factor is equivalent to the Lévy distributed random parameter and crossover rate is equivalent to the probability values of CS. Hence it can be concluded that CS algorithm is a clear adaptation of DE algorithm.

6.2.3 Inferences drawn and motivation behind present work

For most of the works done till date on CS and its application to various domains, such as distributed networks (Nguyen *et al.*, 2016), feature selection (Wang *et al.*, 2014), video tracking system (Walia and Kapoor, 2014), flood forecasting (Chaowanawatee and Heednacram, 2012) and others, it has been found that most of the works done lack proper justification about the parameters of CS (Salgotra *et al.*, 2018b). Though in the application domain also, a large number of articles have been published (Shehab *et al.*, 2017) but when it comes to discussion related to parameters and the modifications, not much work has been done in that context. Most of the works have been done on modification part and those deal only with changes in the exploration or exploitation part or just the probability switching. But not much work has been done on population adaptation, parameters and theoretical analysis (Salgotra *et al.*, 2018b). Also all the modifications have never been added in a single version of CS. Further, it has been found that division of generations increases diversity in population and provides better results (Salgotra *et al.*, 2018a). So the above said inferences have motivated the authors to estimate that there is still scope of future modifications in the CS algorithm. So in order to address the issues of CS and to make it a state-of-the-art algorithm, a new version of CS has been proposed and is discussed in detail in the next section.

6.3 The Proposed Approach

CS is a highly competitive algorithm added to the field of optimization in the recent years. The algorithm because of its intensive local search capabilities and extensive global search properties, has been found to be very efficient. But as the problem complexity increases, the algorithm needs to be crefined. Also it is a well-known fact that no algorithm can be generalized for all optimization problems (Wolpert and Macready, 1997). Apart from that, when it comes to higher dimensional problems where the algorithm requires minimum computational burden and maximum output, the algorithm is found to provide less efficient results. Thirdly, parameter tuning can be a burden when the application is to be remotely managed. So there is a requirement of new algorithm to be designed. Here in the present work, all these points have been kept in mind and SACS (self adaptive Cuckoo search) has been proposed. The main ideas on which the SACS have been based are discussed below:

- A new concept namely self-adaptive population has been proposed.
- The concept of division of generation has been exploited for both global and local search phases.
- Adaptive probability switching based on Weibull distribution has been used.
- Finally, bare bones mechanism has been added in the global and local search phase for searching new solutions.

Analysis on each of the proposed modifications have been added in the subsequent subsections.

6.3.1 Component wise comparison of each modification proposed

In this section we have discussed the importance of individual component added in the proposed modified algorithm. First of all, details on why these modifications were proposed have been added and then each individual adaptation is thoroughly studied in consecutive subsections.

Why each modification has been proposed: Four major modifications have been added in the proposed algorithm. These include division of generations, population size reduction, Weibull distributed probability and introduction of bare bones component in the basic CS algorithm. This section details about what are the properties of these modifications and why these modifications are important.

Division of generations is a new concept used in evolutionary computing and employs multiple different sets of equations for multiple generation sets. Here for SACS, dual division of generations is followed and two different sets of equations are employed. The procedure is followed in such a way that for first half of the generations, extensive exploration and intensive exploitation are followed whereas for the second half of iterations, extensive exploitation and intensive exploration are performed. This parameter is used in conjugation with other parameters and provides better balance between the exploration and local searching capabilities of CS.

The bare bones variant is not a new term in heuristic algorithms and has been exploited

by researchers for major algorithms such as DE and PSO. In PSO, personal best and the global best solutions are used to find the mean and variances whereas in case of DE, personal best is replaced by the previous solution and global best is meant for current best solution. Similar properties have been used in SACS where mean and variance are calculated based on the previous best solution and the current best solutions obtained from the general equations of basic CS algorithm. It has been found that adding the bare bones component or more precisely the Gaussian bare bones component enhances the working capability of an algorithm. The bare bones component because of large deviation in step size, facilitates initial exploration whereas with the increase in the number of generations, the deviation approaches to zero by focusing on the exploitation of the initial solutions. The overall goal is to increase the exploration during initial stages and exploitation towards the end. Thus helping the proposed algorithm in coming out of local minima.

It has been found from the literature that the algorithm switches from exploration to exploitation, if the switch probability is decreased exponentially, linearly or sigmoidally. Any algorithm can perform better on simple unimodal and simple multimodal problems but when the problem complexity increases and the number of local minima increase many fold, the true essence of an algorithm is calculated. That is how good an algorithm performs for such complex functions. Most of the studies suggest the use of decreasing density functions rather than increasing or constant functions. Let us further evaluate this point by using the range of probability parameter. The probability parameter is usually in the range of $[0, 1]$. Generally, a larger value of probability means more exploration and smaller value corresponds to more exploitation. Also for any optimization algorithm, the extensive exploration and intensive exploitation are to be followed during the initial stages whereas extensive exploitation and intensive exploration are followed towards the end. As the value of probability generally lies in the range of 0 and 1, a probability of 1 means 100 percent exploration and probability of 0 means 100 percent exploitation. Thus gradually decreasing the probability value from 1 to 0 will make the algorithm shift from exploration to exploitation. In present case, Weibull distribution has been adapted in such a way that it follows a gradual transition from exploration towards exploitation with increase in population size and generations. The Weibull distribution is a random

distribution and has found little application in evolutionary computing. This distribution is highly effective, widely applicable and it can be derived from the fact that Weibull distribution can be reduced to any general distribution such as exponential or Gaussian distribution. The distribution has long and fatty tailed structure helping in generating larger step sizes which is the salient feature of exploration process in any optimization algorithm. The distribution reduces in such a way that it follows extensive exploration during the initial generation and extensive exploitation towards the end of generations. The overall gain from this distribution is balanced exploration and exploitation operations and hence reduced search capabilities.

Reducing the population size over the course of iterations is also an important characteristic of evolutionary algorithms. The major reason for such adaptation is that during the initial stages, more exploration is required and hence more number of search agents must be employed to perform the operation whereas towards the end exploitation is the major requirement and only a few number of search agents can be used to perform such operations. In the present case, proportional population reduction based on the current best and the previous best solution is proposed. Thus using the knowledge from existing solution to reduce the population, mainly aiming at larger population with proper exploration and smaller population without loss of best solution for exploitation operation.

Overall all these major properties have been incorporated in SACS to make it highly effective. The next four subsections detail about the inherent properties of each of these modifications.

Population adaptation: The population size in case of any optimization algorithm is a very basic and important parameter. This parameter decides the total number of solutions to be generated along with the computational complexity of any algorithm. With a standard population size, the computational burden remains static but if there is a continuous adaptation in terms of decreasing population size, the computational complexity will decrease accordingly. The concept of population adaptation has been formulated in (Eiben *et al.*, 2004) and was further extended in (Hallam *et al.*, 2010). In (Eiben *et al.*, 2004) the population adaptation was done in such a way that with increasing fitness, the population size was increased and with decreasing fitness, it was reduced. The

disadvantage of this method was that the new individuals generated are cloned images of the old ones in the population and hence there is no improvement in the population diversity. Since it is also known that the fitness increases during the initial stages, so there will be increase in the number of population clones as well which ultimately reduces the total diversity of the population. To over come this problem, authors of cref. (Hallam *et al.*, 2010) proposed the opposite approach of decreasing population with increasing fitness and obtained better results. The only point to be kept in mind is that population is reduced only when the best fitness increases. The major reason for this can be justified from the point that suppose we are dealing with a multimodal problem, the new solution generated should be capable of optimizing large landscapes. If the initial population size is very large, the solutions will explore whole of the search space and as the time continues, the solutions will start moving toward some random direction and we can reduce the population size. Thus, it can be said that with increase in fitness, the population size can be reduced. Now the problem is how to perform population reduction? Here shrinking population strategy can be applied. A smaller population size with elitism, allows for better genetic drift to find new solutions, without the loss of best solution. Here if there is very small change in the solution quality, i.e., if the solutions are aggregated in a small portion of search space, it would be economical to use a smaller population size. This is because for a smaller population, each member will have a better chance of getting selected as the best and ultimately the global best solution can be found. The general equation for population reduction has been taken from (Hallam *et al.*, 2010) and is given by:

$$N_{t+1} = \begin{cases} (1 - \Delta f_t^{best}) N_t, & if \Delta f_t^{best} \leq \Delta f_{max}^{best} \\ (1 - \Delta f_{max}^{best}) N_t, & if \Delta f_t^{best} > \Delta f_{max}^{best} \\ min_{popsize}, & if N_{t+1} < min_{popsize} \end{cases} \tag{6.1}$$

Here N_{t+1} is the population size at particular generation t, Δf_t^{best} is given by $(\frac{f_{t-1}^{best} - f_{t-2}^{best}}{|f_{t-2}^{best}|})$ is the change in best fitness, Δf_{max}^{best} is the threshold value. Here a minimum population size needs to be defined in order to avoid the negative effects of extremely small population sizes. Significantly if the shape of this population is defined in terms of a standard curve, the curve is firstly exponential, then steady and finally an exponential decrease and the pattern continues.

Division of generation: For any algorithm, the performance is based upon the phenomenon of exploration (diversification) and exploitation (intensification). Here exploration is meant for exploring the unexplored areas of search space whereas exploitation is meant for performing intensive exploration of smaller search groups inside the whole population. Also for an optimization algorithm, there is requirement of extensive local search in the beginning and as the generation progresses, the algorithm should gradually move toward the exploitation phase. Though some studies suggest that exploration should be done for major part of the generations whereas exploitation should be done towards the end only (Črepinšek *et al.*, 2013). But it has been found that a gradual change in both exploration and exploitation operations provides much better results (Salgotra and Singh, 2017). The study in (Salgotra and Singh, 2017) also suggests that 50 percent exploration and 50 percent exploitation can be a better adaptation. Thus in present scenario, a generalized division of generations is followed. The concept of division of generations or iterations is inspired from (Salgotra *et al.*, 2018b) and is followed in two folds. For the first half of the iterations, the global search is followed by the combination of general equation of CS along with the modification proposed in (Salgotra *et al.*, 2018a). The modified equations are inspired from the grey wolf optimization (GWO) algorithm (Mirjalili *et al.*, 2014) and are some adaptations of the basic CS search equation. The new solution generated using CS is modified and is given by

$$x_1 = x_i - A_1(C_1.x_{new} - x_i^t);\;\; x_2 = x_i - A_2(C_2.x_{new} - x_i^t);\;\; x_3 = x_i - A_3(C_3.x_{new} - x_i^t) \quad (6.2)$$

$$x_{new}^{t+1} = \frac{x_1 + x_2 + x_3}{3} \quad (6.3)$$

where x_{new} is the new solution and A_1, A_2, A_3 and C_1, C_2, C_3 are generated from A and C and are given by $A = 2a.r_1 - a;\;\; C = 2.r_2$. The parameter a is a linearly decreasing random number while r_1 and r_2 are uniformly distributed random numbers. This generalized equation is used because of the better exploration properties of the GWO. As far as local search is concerned, it is followed by using the standard equation of CS with the introduction of a new parameter namely scaling factor F. This parameter has been used instead of ϵ parameter and is generated by (Awad *et al.*, 2017):

$$F_i^{t+1} = \frac{1}{2} \times (sin(2\pi \times freq \times t + \pi) \times \frac{t_{max-t}}{t_{max}} + 1); \quad if r_1 > 0.5 \quad (6.4)$$

$$F_i^{t+1} = \frac{1}{2} \times (sin(2\pi \times freq \times t) \times \frac{t_{max-t}}{t_{max}} + 1); \quad if r_1 < 0.5 \tag{6.5}$$

Here the sinusoidal frequency $freq$ is a fixed function, t is the current generation or iteration and t_{max} is the maximum iteration count.

This process is executed for first half of the iterations only while in the second half of the iterations, the standard equations of CS are changed as per the bare bones variant. The reason for the use of bare bones algorithm is that it helps the algorithm in gradual transition from exploration to exploitation rather than a random step (Kennedy, 2003). More discussion about how and why bare bones algorithm has been used is discussed in the consecutive subsection.

Weibull distributed probability switching: It is already a well-known fact that a good balance between exploration and exploitation has to be maintained for an algorithm to perform well (Črepinšek *et al.*, 2013). In case of CS, this property is maintained by the use of switching probability. Though the concepts of exponential or Gaussian distributed switch probability have been explored extensively in the literature (Salgotra *et al.*, 2018a) but more thorough study in the literature (Shehab *et al.*, 2017) shows that there is requirement of such a distribution which helps in reducing the probability gradually from global to local search and that too in an extensive manner. In the present work, this process is achieved by the use of Weibull distribution (Weihull, 1951). This distribution is one among the best distributions proposed till date and is used to model the lifetime of any problem under consideration. The distribution has mainly three parameters namely shape parameter (β), scale parameter (η) and location parameter (γ). The general probability distribution for this function with respect to iteration t is given by:

$$f(t) = \frac{\beta}{\eta}(\frac{t-\gamma}{\eta})^{\beta-1} \; e^{-(\frac{t-\gamma}{\eta})^{\beta}} \tag{6.6}$$

where $f(t) \geq 0$, $t \geq 0$ *or* $\gamma, \beta > 0$, $\eta > 0$, $-\infty < \gamma > \infty$. The location parameter (γ) is not used frequently and is set to zero. The probability distribution in this case reduces to two parameter distribution. The most important parameter in this case is the shape parameter which helps the distribution to reduce to another distribution. The most common examples include reduction to normal distribution if the value is $\beta = 3.602$, L-shaped distribution if $\beta \leq 1$, bell-shaped if $\beta > 1$ and for large values of β the distribution

has sharp peak. Further, if we model the exponential life time of this distribution, for $\beta < 1$, failure rate with respect to a particular application increases with time, for $\beta > 1$, the failure rate decreases and for $\beta = 1$ it is constant. For present case, the Weibull distribution has been reduced to two parameters and values of third parameter η has been adapted as per the total number of iterations required whereas all other parameters are exactly the same. The value of β is taken as 2. This value is taken after cacreful investigation and thorough study of literature concerning Weibull distribution (Scholz, 2008).

Bare bones based local and global search: It is a commonly known fact that majority of the optimization algorithms suffer from premature convergence and local optima stagnation based upon the problem complexity. The major reason for this drawback is the lack of proper balance between exploitation and exploration phases. CS also suffers from this problem and requires new solutions to be generated by using a single differential learning mechanism either in local or the global search phase. Moreover, due to the presence of current best solution, the algorithm suffers from the problem of local optima stagnation and requires more powerful search mechanism for finding high quality solutions. Thus bare bones mechanism has been added in the CS algorithm for adapting the search strategies. Here two generalized phenomenons namely cooperative search and Gaussian mutation are followed (Kennedy, 2003), (Wang *et al.*, 2013). The generalized adapted equations for both the search phases are given by:

- The improved global search:

$$gx_{i,d} = rw \times x_{new}^{t+1} + (1 - rw) \times bx_{i,d} \tag{6.7}$$

  for $bx_{i,d} = N(\frac{x_{j,d}^t + x_{k,d}^t}{2}, |s_{j,d}^t - x_{k,d}^t|)$

- The improved local search phase:

$$lx_{i,d} = rw \times x_{new}^{t+1} + (1 - rw) \times bx_{i,d} \tag{6.8}$$

  for $bx_{i,d} = N(\frac{x_{j,d}^t + x_{k,d}^t}{2}, |s_{j,d}^t - x_{k,d}^t|)$

where $gx_{i,d}$ and $lx_{i,d}$ are the evolved solutions for both global and local search, respectively, with dimension d and stochastic weight coefficient rw in the range of $[0, 1]$. All other

notations here have the same significance as given by original CS algorithm. This search mechanism is referred to as cooperative search. This method uses the complementary advantages of various search operators and overcomes the problems arising because of single operator. The second part of this modification includes the Gaussian mutation strategy which is introduced to focus on the current global best solution. This strategy is used only when the current best solution is not able to improve in a single iteration. Here m individual solutions are generated with respect to the current best solution, x_{best}, and then a comparison is performed. If the x_{best} solution is not better than best individual of m solutions, the x_{best} solution is replaced by the m best solution. The general formula for this strategy is given by:

$$mutx_{t,d} = x_d \times (1 - N(0,1)) \quad i = 1, 2, ..., m \tag{6.9}$$

where $N(0,1)$ is normalized random number. The pseudo-code of SACS is given in Algorithm algorithm 7.

6.3.2 Computational complexity of SACS

Complexity analysis for any algorithm is performed in order to summarize the worst case complexities and analyze the basic operations of any algorithm. Before calculating the complexity of proposed SACS, let us discuss the complexity of original CS algorithm. When estimating the computational complexity of original CS algorithm, for n population size of a D dimensional problem and having t_{max} as the maximum number of iterations, it has been found to be $O(n.D.t_{max})$. This estimation is derived from the fact that for a population size of n, the dimension size D decides the computational complexity and for each population, we need to perform $O(D)$ operations. Thus for the total population, the complexity is given by $O(n.D)$. This complexity is provided for only a single iteration but when the iteration size is increased to t_{max}, the computational burden increases and is given by $O(n.D.t_{max})$. This is the computational complexity of basic CS algorithm.

Now when we compare the steps of original CS algorithm with respect to the proposed SACS algorithm, it is evident that there is no extra computational burden as all the basic parameters of the new SACS remain same. But here the computational cost will reduce with reduction in population size. Suppose we denote this reduced population size as $n_{reduced}$, the total reduced computational burden will be given by $O(n_{reduced}.D.t_{max})$.

Algorithm 7 Pseudo-code of proposed SACS algorithm

Begin

Define: population size ($\boldsymbol{N}$); switch probability ($\boldsymbol{p}$);

stopping criteria; problem dimension (D)

if i= 1: $\frac{t_{max}}{2}$ **then**

global search using (6.2) & (6.3)

local search using (2.4)

evaluate fitness and perform greedy selection using (2.5)

update N using (6.1)

update p using (6.6)

else

global search using (6.3) & (6.7)

local search using (6.8)

evaluate fitness and perform greedy selection using (2.5)

update N using (6.1)

update p using (6.6)

close;

update final **best**

End

Apart from the population size, there is no other parameter which has any effect on the computational complexity. Thecrefore, the time and space complexities of proposed SACS are less than the original CS algorithm.

6.4 Results and Discussion

In this section, a detailed analysis on the improvement of basic CS has been added in terms of experimental results and statistical testing. The experimental tests have been performed on a x64-bit windows 10 having Intel Xeon Processor ($E5-2630$) with 4.00 GB ram and MATLAB version 7.10.0, $R2010a$. For performance evaluation, the test suite used is CEC2017 consisting of 30 highly challenging benchmark problems. The section is divided into eight subsections including test suite and parameter settings in the first subsection, subsection two, three, four and five discuss about the effect of switch probability,

population adaptation, dimension size and bare bones algorithms, respectively. In the next subsections, comparative study and statistical testing with respect to state-of-the-art algorithms including self-adaptive DE (SaDE), JADE, SHADE, LSHADE, MVMO, CVnew and others are provided. The next subsection details about the algorithm complexity and in the final subsection, the summary of results in terms of main findings, drawbacks and insightful implications are presented. The proposed SACS has been evaluated for all the test functions in the CEC2017 test suite and each function is evaluated 51 times with a total computational burden of $10,000 \times D$ and the results are calculated by computing the difference of best solution and optimal solution. This difference is called the error and when it becomes less than 10^{-8}, it is treated as zero.

6.4.1 Test Suite and Parameter Settings

In this section, a detailed study about the test suite and the parameters of various algorithms used for evaluating the test functions has been presented. The test suite used in CEC2017 benchmark problems, consist of 30 benchmark problems. The test suite has all the set of problems including unimodal $(1-3)$, multimodal $(4-10)$, hybrid $(11-20)$ and composite $(21-30)$ functions. These functions are really challenging and are detailed in Table A.3. The performance has been evaluated in terms of best, worst, mean and standard deviation values of error of 50 dimension size and 51 runs. Here the error values are calculated by using the difference between the final solution and the optimal solution and when this difference is less than 10^{-8}, the error is zero. The total computational cost for each test problem is kept $10,000 \times D$ where D is the problem dimension. It should be noted that all the algorithms used for comparison are state-of-the-art and have been found to be highly competitive.

6.4.2 Effect of switch probability

From the literature, it has been evident that switch probability p_a is a very important parameter of CS algorithm and little variations in this parameter changes the extent of exploration and exploitation capabilities of CS. As discussed in the above sections, the introduction of Weibull's distribution has been experimentally tested in this subsection. The results of new proposed adaptive switch probability of SACS in comparison with CS are given in Table C.24 In Table C.24 the results for $p_a = 0.25, 0.50$ and 0.80 are presented for CS and SACS only. For SACS, the adaptive probability $p_a = adaptive$

values are also presented. If we see the comparative values for SACS only, it can be seen the results are best for $p_a = 0.80$. But when we compare the results with respect to $p_a = adaptive$, the results for adaptive probability are better for most of the test functions under comparison. It should be noted that the results for adaptive p_a are not presented for simple CS. This is because no such adaptation has been done by the original CS article. So a comparison can be done in terms of adaptive values of SACS versus $p_a = 0.25, 0.50$ and 0.80 values for CS. It can be seen from the results that for functions F_1, F_2, F_{12} and F_{13} both the algorithms were not able to achieve an optimal solution. The results for these functions were significantly equivalent for all of them. For rest of the unimodal functions, the results were significantly similar for both the algorithms. But when we see the results in comparison to multimodal, hybrid and composite functions, the results of SACS are significantly better than its counterpart. Thus we can say that adding adaptive properties to the probability parameter makes the algorithm more competitive and also provides significant improvement in the results. This is due to the fact that the algorithm is able to make a balance between the exploration and exploitation processes and hence provides competitive performance.

6.4.3 Effect of population adaptation

The population size is the basic and the most important parameter of any optimization algorithm. This parameter decides the total number of function evaluations for an algorithm and in turn decides the computational complexity of an algorithm in solving a particular set of problems. Here it should be noted that less is the population size, lesser will be the total number of function evaluations required. For present scenario, adaptive population strategy is taken into consideration, or more precisely decreasing adaptive population. Thus the algorithms will require lesser number of function evaluations and hence lower computational cost in solving the problem under test. The performance evaluation for this case is shown by using different sets of population sizes for SACS in comparison with the proposed adaptive population strategy. The results with respect to population size of $20, 40, 80, 100$ and adaptive population sizes are shown in Table C.25.

From the results in Table C.25, it can be seen that results for the algorithms are highly significant and there is little variation for functions F_1, F_2, F_{12} and F_{13}. For these functions we can say that the results are either insignificant or are almost similar. For most of the

functions, the results for CS are better for all the population sizes with $pop_{size} = 100$ giving the best results. For rest of the cases, the results of SACS algorithm are better and are highly significant. The major reason for this is the presence of adaptive properties in the SACS algorithm. Another advantage of this adaptation is that reduced number of function evaluations are required to solve the problem under test. As suggested by (Awad *et al.*, 2016a), the maximum number of function evaluations for computation are $10,000 \times D$. Since there is population reduction, so lesser number of search agents towards the end of the iterations and hence more number of iterations can be performed before the actual termination. Overall we can say that adding the adaptive population parameter not only achieves better results but also allows SACS in exploiting the problem under test for more number of function evaluations.

6.4.4 Effect of dimension size

This section deals with the dimensional analysis of CS and results with respect to dimension size of 10, 30 and 50 are shown in Table C.26. The results in this section are taken with 100 population size for CS and adaptive values of the same for SACS. This test is done to evaluate the capability of the proposed SACS for higher computational complex problems. Since the test suite is highly challenging, so better performance of the proposed algorithm on these test functions will make it suitable for real world problems as well.

Here the results of proposed algorithm with respect to CS are presented in Table C.26. It can be interpreted that for first case of 10 dimension size, that for almost all the unimodal functions, the SACS algorithm was able to find the near optimal solution and attains the best results. For the same case, the results for CS were very limited and it was not even able to reach the sub optimal solution. For the second and third cases of 30 and 50 dimension sizes, respectively, the SACS algorithm was found to be better than CS for almost all the cases; thus proving its worth on higher dimensional problems as well. Though with increase in dimension size, the results also varied significantly. But keeping the complexity of the problem in context, it can be considered as significant. Also more details regarding the same are presented in the comparison section where other popular algorithms are used for comparison.

6.4.5 Effect of each modification proposed

In this section, results with respect to each modification are presented. The results are referenced in Table C.27. Here the results are taken at 50 dimension size and all other parameters are adaptive. That is all the parameters of proposed SACS are incorporated and comparative results are presented. According to Table C.27, the first section deals with results of adaptive population and barebones modification. In the second section, the results for adaptive probability and barebones is presented whereas in the final section all the modifications are clubbed together into one and hence the proposed variant. From the results we can say that, with each modification added, there is significant improvement in the results of CS algorithm. If we compare the results, adding adaptive population decreases the number of function evaluations required. The search agents are also reduced and hence diversity in the population decreases toward the later stages and hence corresponds to better exploitation operation. By adding barebones variant, the explorative properties of algorithm are enhanced. This is because of the presence of cooperative searching mechanism which keep the diversity of the algorithm intact. The second thing which helps is the addition of adaptive probability. Here Weibull distributed solution makes the searching capabilities of SACS better by properly maintaining a balance between both the exploration and exploitation processes. Now if we see each modification independently, the barebones variant helps to maintain diversity among the search agents, hence improving the exploration rate. Weibull distributed probability helps in maintaining the proper exploration and exploitative balance and population adaptation adding to diversity reduction, hence improving the exploitation properties. Note that population adaptation is followed only after the second half of the generations so that maximum diversity can be maintained among the search agents. The next sub-section details about the proposed variant and its comparison with respect to other state-of-the-art algorithms.

6.4.6 Algorithm run-time complexity

The complexity of the proposed SACS in terms of run time t_0 is calculated by using the code given as:

$for\ i = 1 : 1000000$

$$x = 0.55 + double\,(i)\,; x = x + x; x = x/2; x = x \times x;$$

$$x = sqrt\,(x)\,; x = \log\,(x)\;\; ; x = \exp\,(x)\;\; ; x = x/(x+2);$$

end

The complexity of algorithm is also shown in Table 6.1. Here the notations T_0 corresponds to the computing time for the code given above and T_1 is the computing time for F_{18} function from the Test suite given in the next section, with a total number of function evaluations of 200,000. T_2 is the mean run time for the same function with same number of function evaluations for a total number of five runs.

Table 6.1 SACS algorithm: Computational complexity

Dimension size (D)	T_0	T_1	T_2	$(T_2 - T_1)/T_0$
D=10	0.1562	2.839	2.853	0.871
D=30		12.459	12.6296	1.089
D=50		29.306	31.318	2.012

6.4.7 Comparison with respect to state of the art algorithms

In order to have a fair comparison of the proposed SACS with respect to state-of-the-art algorithms, SaDE, JADE, SHADE, LSHADE, MVMO, CV1.0 and CV_{new} algorithms are used. All these algorithms are highly competitive and have proved their worth in various CEC competitions and other real world optimization problems. A rank-sum test for checking the statistical significance of SACS with respect to other algorithms has also been performed. The results are presented in Table C.28. The results have been presented in terms of mean error and standard deviation values. The rank is also provided in the third row. This rank is calculated by using Wilcoxon's rank-sum test (Derrac *et al.*, 2011). The results for different cases are given in terms of $w(win)/l(loss)/t(tie)$. Here $w(win)$ corresponds to the scenario where the algorithm under test is better than the proposed algorithm, $l(loss)$ corresponds to the situation in which algorithm under test is worse as compared to SACS. In the final scenario $t(tie)$, it can be said that both the algorithms under test are either statistically significant or have no relevance with respect to each other. From the results in the comparison Table C.28 we observe the following:

- For unimodal functions, F_1, F_2 and F_3, SaDE, JADE, SHADE, LSHADE and MVMO are highly competitive and provide good results where as CV1.0, CV_{new},

CS and SACS were not able to converge. Here SHADE was found to be the best performing algorithm for function F_1 and F_3 and SaDE for function F_2.

- For most of the multimodal functions, F_4 to F_{10}, SaDE, JADE, MVMO and SHADE perform better and LSHADE was found to be the best among others.
- For the third set of functions, that is hybrid functions F_{11} to F_{20}, the variants of CS were found to be better and among them the newly proposed SACS is the best performing algorithm.
- For the final set of composite functions, F_{21} to F_{30}, the SACS algorithm was again the best performing algorithm among all the algorithms under test.
- As far as the statistical ranksum p-values are concerned with respect to SACS, SaDE is better for ten functions, JADE for twelve functions, SHADE for fourteen, LSHADE for sixteen, MVMO for fifteen, CV1.0 for one function, CV_{new} for six and CS for none. So statistically, SACS is better that SaDE, JADE, SHADE, CV1.0, CV_{new}, CS and highly competitive with respect to MVMO and LSHADE algorithm.
- If we compare the overall performance for all the algorithms, LSHADE was found to be better for eight functions, SHADE for two functions, CV_{new} for six function, CV1.0 for one function, JADE for one function, MVMO for two functions and SACS for twelve functions. Thus overall the proposed SACS is better among all other algorithms under comparison.

Apart from these results, the convergence profiles have also been shown in Figure Fig. 6.1. The convergence curves are drawn between the original CS and SACS algorithm. The parameter settings for these are same as used for the statistical testing done in Table C.28. The only difference is that here we have used a defined set of 2500 iterations for both the algorithms. For performance evaluation, the results for functions F_1, F_2, F_{12} and F_{13} are almost similar for both the algorithms and not much significant changes are visible in the convergence profiles. From the results in Fig. 6.1, it can be seen that for almost all the functions, SACS converges much faster in comparison to the CS algorithm. For functions, F_4, F_8, F_{25}, F_{27}, F_{28} and F_{29}, the convergence for both the algorithm is quite similar in

the initial stages but towards the later stages, the convergence of SACS is better. Overall we can say that of all the state-of-the-art algorithms under comparison, the proposed SACS algorithm is highly competitive and can be considered as potential candidate for becoming a state of the art algorithm.

6.5 Testing on Real world Optimization Problems

Here in present case, four problems from the standard CEC 2011 benchmark function are used for parameter evaluation (Das and Suganthan, 2010). The results of SACS for each of the test function are compared with other algorithms namely flower pollination algorithm (FPA) (Yang, 2012), differential evolution (DE) (Storn and Price, 1997), bat algorithm (BA) (Yang, 2010b), GWO (Mirjalili *et al.*, 2014), self-adaptive MODE (SAMODE) (Elsayed *et al.*, 2011), evolutionary algorithm- differential evolution memetic algorithm (EADAMA) (Singh and Ray, 2011) and firefly algorithm (FA) (Yang, 2010a). The parameter settings for all these algorithms have been taken from their original work and not much modifications have been done in adaptation or changing the parameters. All the algorithms used the same set of basic parameters such as population size, problem dimension, number of function evaluations and same stopping criteria. Here for this set of optimization problems, we have used a population size of 50, dimension size as defined by the problem under test and $150,000$ total number of function evaluations as given in (Das and Suganthan, 2010). The results are presented in terms of best, worst, mean and standard deviation (std) values as given by the original works (Das and Suganthan, 2010). The convergence profiles are drawn from the median values of the maximum number of runs. The four optimization problems used for evaluations are presented in the next four sub-subsections.

6.5.1 Parameter Estimation for Frequency-Modulated Sound Waves (FM)

Frequency-modulated (FM) sound system is an integral part of modern music systems. This section details about the parametric optimization of FM synthesizer using SACS algorithm. The objective here is to automatically generate sound similar to the desired sound and the features are extracted using dissimilarities of features between synthesized and target sounds. This process of feature extraction is followed unless and until both

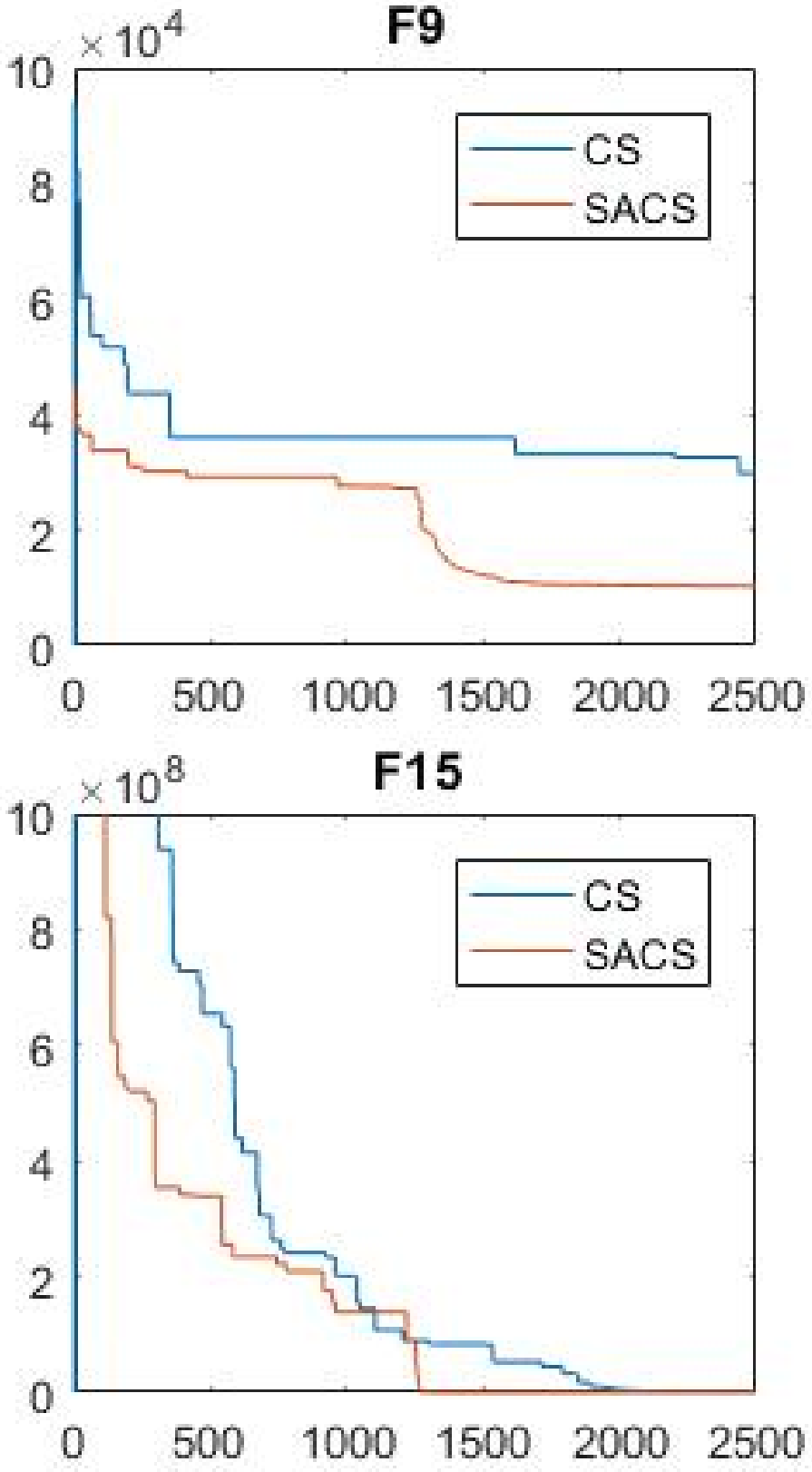


Fig. 6.1 SACS algorithm: Convergence profiles versus CS

synthesized and target sounds become similar. The problem is highly complicated and is a multi-modal function having epistasis. The global minimal for this problem lies at $f(X_{sol}) = 0$. The problem has been investigated by large number of researchers with major application to GA (Horner *et al.*, 1993) and DE (Dukic and Dobrosavljevic, 1990). The general expression for synthesized sound $y(t)$ and desired sound $y_0(t)$ waves are given by:

$$y(t) = a_1.sin(\omega_1.t.\theta + a_2.sin(\omega_2.t.\theta + a_3.sin(\omega_3.t.\theta))) \tag{6.10}$$

$$y_0(t) = 1.0.sin(5.0.t.\theta - 1.5.sin(4.8.t.\theta + 2.0.sin(4.9.t.\theta))) \tag{6.11}$$

Here $\theta = 2\pi/100$ and the range of the FM function lies between $[-6.4, 6.35]$. The goal here is to minimize the sum of square of error values between the synthesized and desired sound and is given by equation Eq. (6.12).

$$f(X) = \sum_{t=0}^{100}(y(t) - y_0(t))^2 \tag{6.12}$$

The requirement here is to find the best optimal solution of the problem under test and compare it with other algorithms form literature. The results for this problem are presented in Table E.1. The objective function has been named as FM and from the experimental results, it is evident that for all the algorithms under comparison only FA, GWO and SACS are able to converge towards the final solution. Though from the convergence profiles we can say that FA converges better than SACS but overall the final result of SACS is much better. Thus we can say that in case of SACS, the probability of getting the desired FM sound signal is very high when compared to other algorithm such as BA, FPA, DE, FA, SAMODE, SADEMA and GWO algorithms.

6.5.2 Lennard-Jones Potential Problem (LJ)

Lennard-Jones (LJ) is a multi-modal potential energy problem dealing with the minimization of potential energy of a pure LJ cluster and having an exponential number of local minimal (Hoare, 1979). The LJ cluster here consists of icosahedral core with a combination of surface lattice points. The task here is to confirm molecular structure and atoms are aligned such that the molecules have minimum energy. In Cartesian coordinates

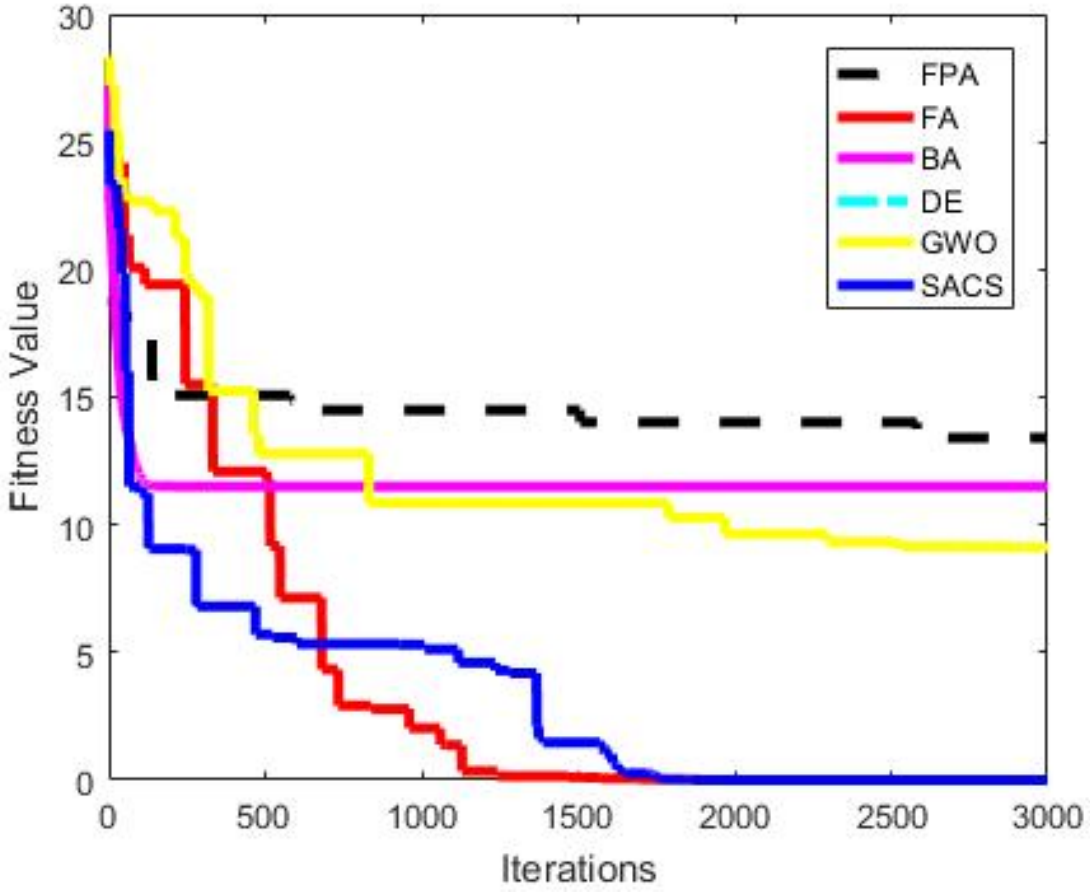


Fig. 6.2 SACS algorithm: Convergence of Frequency Modulated sound waves

$p_i = x_i, y_i, z_i$, for $i = 1, ..., N$, the LJ pair potential for N atoms is given by

$$V_N(p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (r_{ij}^{-12} - 2.r_{ij}^{-6}) \tag{6.13}$$

where $r_{ij} = ||p_j - p_i||_2$ with gradient

$$\Delta_j V_N(p) = -12 \sum_{i=1, i \neq j}^{N} (r_{ij}^{-14} - r_{ij}^{-8})(p_j - p_i), \;\; j = 1, ..., N \tag{6.14}$$

The LJ potential has a standard minimum value at some pair distance r between the two points and is given by $V(r) = r^{-12} - 2.r^{-6}$. Now if we decrease the value of r with respect to optimum, the value of $V(r)$ increases rapidly and reaches infinity for $r = 0$. This aspect of the LJ pair makes it a hard optimization problem. Further, a reduction in the problem dimension can be followed as given by (Moloi and Ali, 2005). The first step is to fix one atom at the origin, choose the second one on the positive side of X-axis

and the third atom in the upper half of the X-axis. For adding the fourth atom, three variables (Cartesian co-ordinates of fourth atom) are required to design the minimization problem with six independent variable. Here it should be noted that for each new atom added, three variables are added to find the potential energy associated with the clusters. Suppose a variable x has three components for first three atoms, six for four atoms and so on. The first, second and third variable due to their corresponding second, third and fourth variable respectively are given by $x_1 \in [0, 4]$, $x_2 \in [0, 4]$ and $x_3 \in [0, pi]$. On a whole the coordinates for any other value i of x is given by $x_i \in [-4 - \frac{1}{4}[\frac{i-4}{3}], 4 + \frac{1}{4}[\frac{i-4}{3}]]$.

From the results in Table E.1, it can be seen that all the algorithms except DE were able to reach near optimal for the best of 50 runs where as BA and SACS were able to do the same for almost all the runs. Here the result are compared with respect to standard deviation (std) values. For present case, we can see that std of SACS is much better in comparison to other algorithms. Also from the convergence curves in Figure Fig. 6.3 we can conclude that the SACS algorithm is better.

6.5.3 Optimal Control of a Non-Linear Stirred Tank Reactor (STR)

The first order irreversible chemical reaction performed in a continuous stirred tank reactor (CSTR) is a multi-modal optimization control problem. This problem is a benchmark problem taken from the Handbook of Test Problems in Local and Global Optimization (Floudas *et al.*, 2013) and has been investigated in the recent past by a large number of researchers using stochastic global (Ali *et al.*, 1997) and dynamic (Luus, 2000) optimization algorithms. The modelling of the chemical process for this application is given in the form of two general equations as:

$$x_1 = -(2 - u) + (x_1 + 0.25) + (x_2 + 0.5)exp(\frac{25x_1}{x_1 + 2}) \tag{6.15}$$

$$x_2 = 0.5 - x_2 - (x_2 + 0.5)exp(\frac{25x_1}{x_1 + 2}) \tag{6.16}$$

Here $u(t)$ is the flow rate of cooling fluid, x_1 and x_2 are dimensionless and derivative of dimensionless steady state temperature respectively. The objective of this problem is to find suitable value of u in order to minimize the performance index for initial conditions

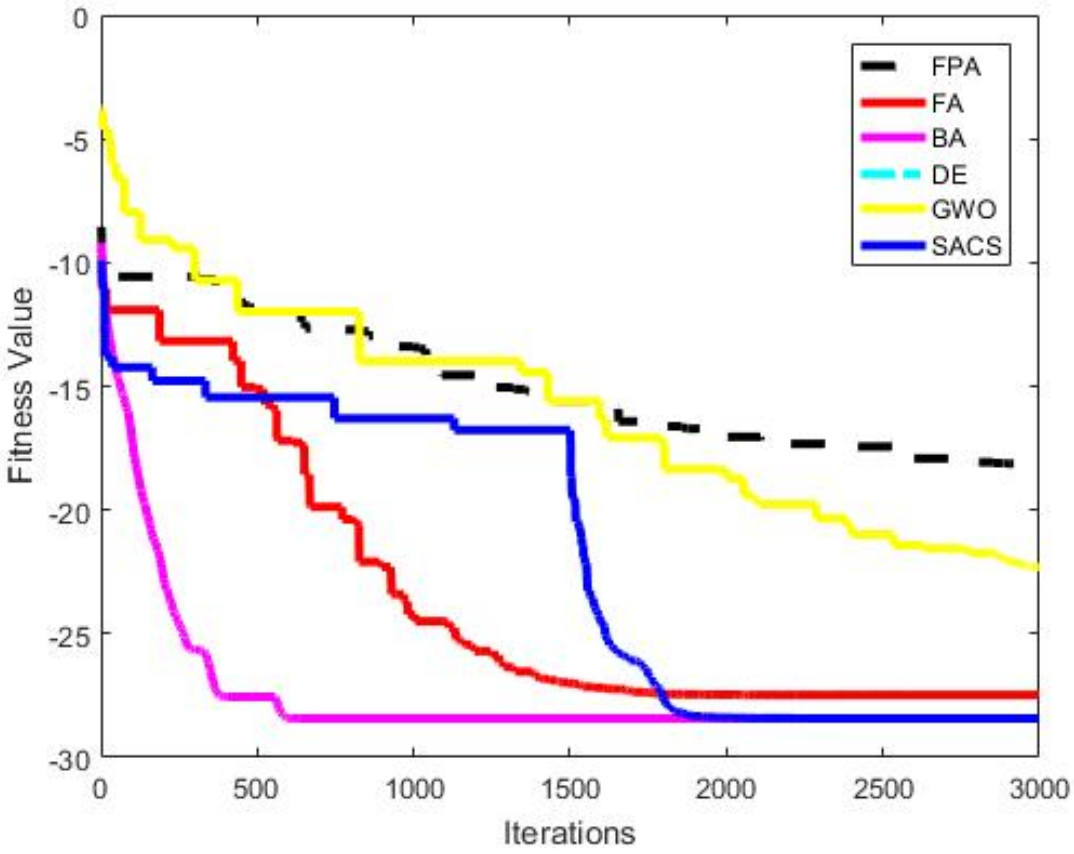


Fig. 6.3 SACS algorithm: Convergence of Lennard-Jones potential problem

of $x(0) = [0.9\ 0.09]^T$ as given by

$$J = \int_0^{t_f=0.72} (x_1^2 + x_2^2 + 0.1.u^2)dt \tag{6.17}$$

Here the initial guess for the value of $u(t)$ for this unconstrained problem lies with in the range of $[0.0\ 5.0]$ with tolerance level 1×10^{-1}.

The major goal of the problem is to minimize the performance index for initial conditions as given by Eq. (6.13). The problem is unconstrained one and the results are reported in Table E.1. From the comparative results for all the algorithms under test, it can be said that SACS algorithm provide highly progressive results and there is not much variation in the final end solution for majority of the times. Also from the convergence profiles, it can be seen that though BA is faster during initial stages but as the iterations progresses, SACS is found to be more effective.

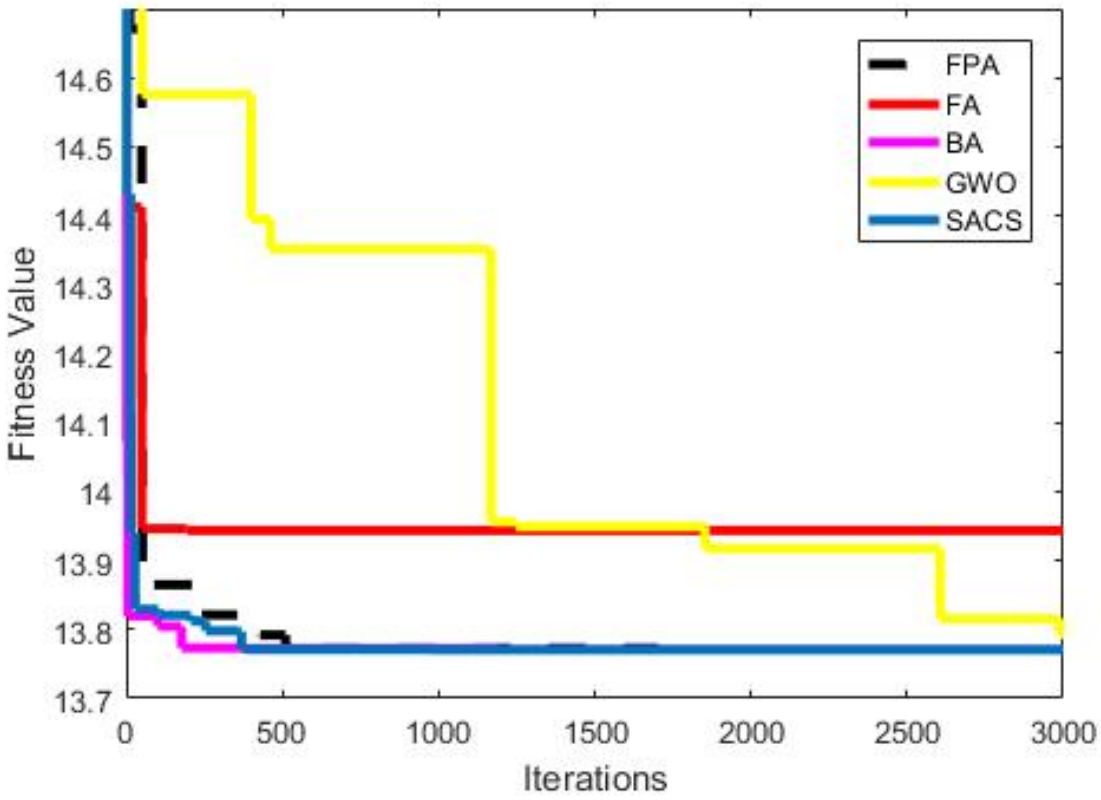


Fig. 6.4 SACS algorithm: Convergence of stirred tank reactor problem

6.5.4 Spread Spectrum Radar Polly Phase Code Design (SPRP)

Designing a radar system with pulse compression is a NP-hard problem and requires great attention for choosing the proper waveform. There are various methods of radar pulse modulation that help in pulse compression. One among them is the polyphase codes, and is one among the best because of lower side-lobe compression. Aperiodic auto-correlation and coherent radar pulse processing based polyphase pulse compression is a recent introduction in the radar system compression (Dukic and Dobrosavljevic, 1990). This problem is modeled as min-max nonlinear non-convex optimization problem with large number of local optima. It is given by

$$Global\ min\ f(x) = max\phi_1(x), ..., \phi_{2m}(x) \tag{6.18}$$

$$X = (x_1, ..., X_n) \in R^n | 0 \leq x_j \leq 2\pi; \qquad j = 1, ..., n \tag{6.19}$$

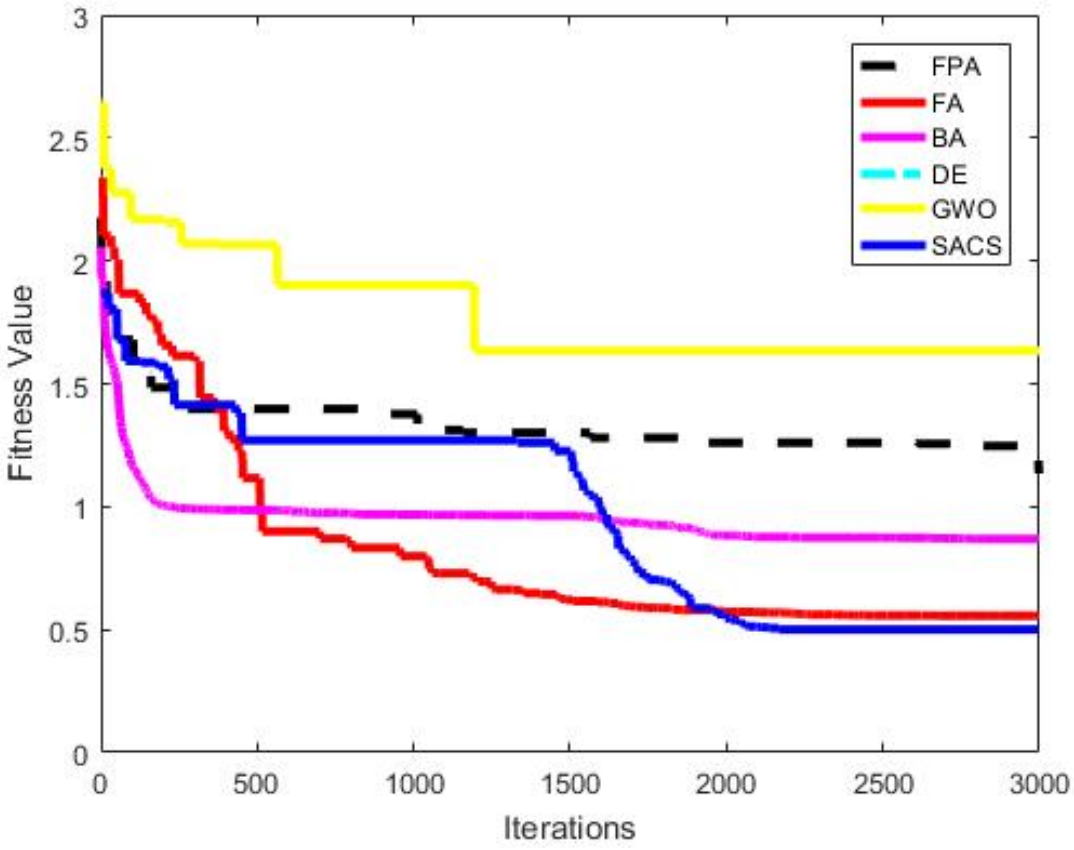


Fig. 6.5 SACS algorithm: Convergence of spread spectrum radar polly phase code design

where $m = 2n - 1$;

$$\phi_{2i-1}(x) = \sum_{j=i}^{n} cos(\sum_{k=|2i-j-1|+1}^{j} x_k); \quad i = 1, ..., n$$

$$\phi_{2i}(x) = \sum_{j=i+1}^{n} cos(\sum_{k=|2i-j|+1}^{j} x_k); \quad i = 1, ..., n-1$$

$$\phi_{m+i}(x) = -\phi_i(x); \quad i = 1, ..., m$$

Here it should be noted that variables in equation Eq. (6.18) are symmetric phase differences.

From the results in comparison Table E.1, it can be seen that the proposed SACS performs better in comparison to BA, DE, FA, FPA, SAMODE, EADEMA and GWO algorithms. The results are further validated from the convergence profiles given by Fig. 6.5.

6.5.5 Summary of results

The original CS algorithm is a highly challenging algorithm and because of its simple structure has found application in almost every field of research. But when we compare it with respect to its counter parts, the basic algorithm doesnot provide much of the flexibility and has limited performance. Also the algorithm suffers from the problem of poor exploration, local optima stagnation and even the existing literature doesn't provide enough justification on the modification or applications proposed. So in the present work, a new self adaptive variant of CS is proposed. The major outcomes of the current work are highlighted as below:

- A new self adaptive variant of CS is proposed. The proposed algorithm employs linear population reduction, Weibull distributed probability and finally barebones based exploration to enhance the performance of CS algorithm.

- The newly proposed CS algorithm has been named as SACS and also employs the concepts of division of population and generation as inspired from CV1.0 algorithm. This division of population and generation plays very important part and adds to the enhanced exploration and exploitation processes.

- Linearly decreasing population helps to attain an adaptive population and reduce the total number of function evaluations required. This property further helps the algorithm in achieving better exploitation among the search agents towards the end of iterations.

- Weibull distribution and barebones addition make the algorithm more flexible in explorative and exploitative capabilities. These features further help in maintaining a balance between the global and local search phases.

- Further, theoretical analysis with respect to already existing algorithms mainly GA and DE is presented. Here it can be seen that the CS algorithm in its generalization is an extension of GA and DE. As far as parametric knowledge is concerned, both the algorithms have almost similar set of basic parameters.

- Moreover a theoretical analysis on the stability of the CS algorithm has also been

presented. This stability analysis uses von Neumann criteria and based on that it can be interpreted that parameters of CS algorithms play very important part in its application.

- The experimental results based on varying population sizes, dimension sizes and probability switching further validate that the proposed algorithm is an efficient tool in finding near global optimal solution.

- Now as far as the experimental results are concerned, it has been found that SACS is able to achieve better results than algorithms such as SaDE, JADE, SHADE, MVMO and others.

- The results for 4 real world optimization problems from CEC 2011 benchmarks also show significant performance improvement and make SACS as one among the most competitive algorithm.

Overall we can summarize that CS algorithm is highly challenging algorithm and in order to improve its working capabilities, SACS is proposed. But, the improved results for SACS are limited to hybrid and composite functions only. If we see these results in comparison to unimodal and multimodal functions, more work is still required to be done.

6.6 Concluding Remarks

This paper proposes a self-adaptive variant SACS of CS for enhancing its performance. The major problem with CS algorithm is that it converges prematurely due to the poor exploration properties. To avoid this problem, the bare bones mechanism is adopted in the modified SACS algorithm. It has also been found that CS algorithm is not able to make a proper balance between the global and local search phases. So here Weibull distributed switch probability helps in maintaining a balance between the two phases. The enhanced version also employs population reduction mechanism to reduce the computational burden of the proposed SACS algorithm. The enhanced SACS algorithm is tested on CEC 2017 benchmark problems and analysis with respect to variable dimension size, variable probability and variable population size is also presented. The study of variable population size shows that with the increase in population size, the change in results is modest but the total number of function evaluations increases many folds. But if we see the results

of adaptive population, it can be seen that the results are lot better and also the number of function evaluations required is minimal. The new algorithm has also been subjected to variable dimension size to test its performance for higher dimensional problems. Here also it has been found that there is little variation in results showing its capability to solve problems with higher number of variables. In order to balance the exploration and exploitation properties, Weibull distributed probability was used and compared with variable switch probabilities. It has been found that the basic CS algorithm is very sensitive to changes in switch probability whereas the proposed SACS switches the probability as per the upper and lower limits. Finally, the incorporation of division of population and generations further makes the algorithm more flexible in balancing the exploration and exploitation operations. Non-parametric Wilcoxon's ranksum test has also been done to prove the significance of proposed SACS statistically and it has been found that SACS performs better than other stat-of-the-art algorithms under comparison.

Apart from this, a theoretical study based on CS as an extension of DE and GA is also presented. For evaluating the stability criteria, Von Neumann's stability criteria is used and stability analysis of CS with respect to that is also presented. From these theoretical analyses, it has been found that CS algorithm is a very challenging algorithm and more work can be done in order to improve its overall capabilities. As a future direction, more balanced exploration and exploitation operations can be performed by using different statistical operators. An experimental analysis on the performance of CS algorithm can also be presented. As far the proposed SACS algorithm is concerned, it can be used for solving real world problems such as satellite imaging, data clustering, feature selection, antenna design, cancer detection and others.

Part II : Flower Pollinzation Algorithm

CHAPTER 7

Adaptive Flower Pollination Algorithms

Preface

This chapter presents modifications to basic FPA in order to enhance its performance. A dynamic switch probability, an improved mutated global and local search has been used in the enhanced versions of FPA. The mutation techniques give five variants of FPA in which Cauchy, Gaussian, adaptive Lévy, mean mutated and adaptive mean mutated distributions have been used. The enhanced performance of variants can be attributed to dynamic switching which balances effectively local and global search. The global mutations also help in improving the exploration capabilities of the basic FPA which is required to escape local minima. In addition to these changes, in ALFPA the local search equation has been changed to improve the exploitation operation.

Publication: **Rohit Salgotra**, and Urvinder Singh. "Application of mutation operators to flower pollination algorithm." Expert Systems with Applications 79 (2017): 112-129.

7.1 Introduction

The performance of any algorithm is based upon the phenomenon of exploration and exploitation. These two processes are also called as diversification and intensification respectively. The exploration process helps to perform global search that is to explore the unexplored areas of the search space where as exploitation is to perform local search in order to achieve more intensive results. It has been found that a good balance between both these phenomena is required to achieve better results. However, it is still a matter of concern and more research is to be done to interpret which part of meta-heuristic algorithm does exploration and which one is meant for exploitation (Draa, 2015). Another important factor is the local optima stagnation and slow convergence associated with these algorithms. It has been found that FPA also suffers from the problem of local optima stagnation and slow convergence and hence requirement of new enhanced version of FPA is the need of the time. Also, the problems being optimized are becoming more and more complex and so do the evolutionary algorithms. It has been proved by Wolpert and Macready that no algorithm can be generic for all optimization problems (Wolpert and Macready, 1997). So, it becomes a necessity to optimize the algorithm by using appropriate mutation or selection operators.

In this chapter, the concept of mutation is exploited and applied to standard FPA algorithm. Different variants are proposed and these variants are based upon the following basic ideas:

- A new concept based on mutation operators is applied to the global pollination phase and five new variants of FPA are proposed.
- The first and the second variants exploit the concept of Cauchy and Gaussian distribution.
- The third and the fourth variants use a combination of Cauchy and Gaussian distribution in two different ways: firstly, by using mean of both and then by using an adaptive component of both respectively.
- The fifth one uses adaptive in both global as well as in local search.

- Dynamic switch probability is used in all the proposed variants.
- Apart from this, analysis of FPA as an extension of DE is also presented.

Dynamic switch probability: As elaborated in the literature, switch probability plays a very important role on the performance of FPA and using it dynamically makes the algorithm more reliable in exploring and exploiting the search space. The general equation for dynamic switch probability p used here is given by:

$$p = p - \frac{t_{max} - t}{t_{max}} \times (0.1) \tag{7.1}$$

Here t is the current iteration and t_{max} is the total number of iterations. The dynamic p value is used in all the proposed variants of FPA and so is not explicitly discussed in every subsection. The initial value of p is taken as 0.8.

The effect of mutation operators and details on new variants of FPA are proposed in section 6.2. These variants use mutation operators like Gaussian, Cauchy, mean mutation, adaptive mean mutation and adaptive Lévy to add a modification to the basic FPA. Further the FPA variants are also compared with respect to the mutation operators of DE and an explicit discussion about the similarity between the operators of DE and FPA is analysed in section 6.2.5. The variants are analysed in terms of different population size, dynamic switch probability, statistical testing and are applied to standard benchmark problems. A detailed comparative study with respect to other state of art is also discussed. The result and discussion part is in section 6.3. The concluding section is section 6.4 and it provides a brief review of the analysis of results.

7.2 The Proposed Algorithms

7.2.1 Gaussain FPA

The Gaussian mutation operation has been derived from the Gaussian normal distribution and its application to evolutionary search was formulated by Rechenberg and Schwefel (Bäck and Schwefel, 1993). This theory was referred to as classical evolutionary programming (CEP) (Yao *et al.*, 1999), (Yao *et al.*, 1997). The Gaussian mutations increase the searching capabilities of an algorithm and has been used to exploit the desirable properties (Yao *et al.*, 1997). Also, Gaussian mutation is more likely to create a new offspring near the original parent because of its narrow tail. Due to this, the search equation will

take smaller steps allowing for every corner of the search space to be explored in a much better way. Hence it is expected to provide relatively faster convergence. The Gaussian density function is given by:

$$f_{gaussian}(0, \sigma^2)(\alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{\alpha^2}{2\sigma^2}} \tag{7.2}$$

where σ^2 is the variance for each member of the population. This function is further reduced to generate a single D dimensional random variable by setting the value of mean to zero and standard deviation to 1. The random variable generated is applied to the general equation of global pollination phase of standard FPA algorithm and is given as

$$x_i^{t+1} = x_i^t + G(\alpha)(T^* - x_i^t) \tag{7.3}$$

Here all the notations are same as that of FPA except for G(α), which corresponds to a D dimensional Gaussian step vector generated using Gaussian density function with α as a Gaussian random number between [0, 1]. The pseudo-code for Gaussian-FPA (GFPA) is given in Algorithm 8.

7.2.2 Cauchy FPA

Premature convergence can occur if relative minima of fitness functions are far away from small mutation of CEP. To overcome this effect, Yao et al. used a Cauchy distributed random number to generate a new solution (Yao *et al.*, 1997). In FPA, the Lévy based step size mutation operator is switched to Cauchy based operator C(δ) to control smaller steps and the algorithm is named as Cauchy FPA (CFPA). This operator is simply a Cauchy based random number generated from Cauchy distribution function given by:

$$y = \frac{1}{2} + \frac{1}{\pi} arctan(\frac{\delta}{g}) \tag{7.4}$$

The Cauchy density function is given by

$$f_{Cauchy}(0, g)(\delta) = \frac{1}{\pi} \frac{g}{g^2 + \delta^2} \tag{7.5}$$

The value of g is set to 1 and is the scale parameter, y is a uniformly distributed random number between [0, 1]. Solving Eq. (7.4) for the value of δ, we get

$$\delta = tan(\pi(y - \frac{1}{2})) \tag{7.6}$$

Algorithm 8 Pseudo-code of original GFPA algorithm

Begin

Initialize flower population: ***n***

Define dynamic switch probability

Define ***d***−dimensional objective function, ***f(x)***

for i= 1: t_{max} **do**

if **rand** < **p**

A d−dimensional Gaussian based step vector is drawn.

Perform global pollination: Eq. (7.3)

else

Perform local pollination: Eq. (2.4)

end if

if x_i^{t+1} better than x_i^t

$x_i^t = x_i^{t+1}$

end if

Update probability using Eq. (7.1)

evaluate the current best

update final **best**

End

This operator modifies the general equation in global pollination of the standard FPA to

$$x_i^{t+1} = x_i^t + C(\delta)(T^* - x_i^t) \tag{7.7}$$

Here use of Cauchy based mutation operator allows for larger mutation and hence helps in exploring the search space. This is because Cauchy distribution has a fatter tail and this tail helps to generate a larger step, which can help in escaping local minima and this is the reason for its enhanced performance. It further helps to avoid premature convergence. The exploitation in this case is handled by the local pollination Eq. (2.8) of the standard FPA. Due to the presence of Cauchy mutation operator, the algorithm is able to search more areas inside the search space which further makes it possible for the local pollination phase to exploit the available space in a much better way. The pseudo-code for CFPA is same as that of GFPA except for change in the global pollination phase where Eq. (2.6)

is changed to Eq. (7.7).

7.2.3 Combined mutated FPA

Combined mutation refers to use of combination of two or more mutation operators. The concept of combined mutation was first formulated in (Chellapilla, 1998). Here they used a combination of Cauchy and Gaussian distributions, convolve them into a single distribution and based on that generate a new solution. The two combined mutation operators formulated by this process are known as mean mutated (MM) operator and adaptive mean mutation (AMM) operator (Koenig, 2002). The general equation for MM and AMM are given by

For MM:

$$x_{t+1} = x_i^t + 0.5\sigma_1(G(\alpha) + C(\delta)) \tag{7.8}$$

For AMM:

$$x_{t+1} = x_i^t + \sigma_2 G(\alpha) + \sigma_3 C(\delta) \tag{7.9}$$

where all the notations are same as that used in previous subsections, σ_2 is a part of population and is identical to σ_1 making the mutations as:

$$\sigma_i = \sigma_i e^{r_1.G(\alpha)+r.G_t(\alpha)} \tag{7.10}$$

where $i = 1, 2, \ldots N$, for N population size, $G(\alpha)$ and $G_t(\alpha)$ are random numbers generated form Gaussian distribution with $G_t(\alpha)$ generated only once. The value of r_1 and r are given by $r \infty (\sqrt{2\sqrt{n}})^{-1}$ and $r_1 \infty (\sqrt{2n})^{-1}$ respectively. Here the mean mutation (MM) and adaptive mean mutation (AMM) equations are not used as such. A simplified modification as per FPA is applied to the general equations and two new versions namely MMFPA and AMMFPA are proposed. The σ value generated here is a random solution for each member of the population group. The general equation of FPA uses the difference of two random solution, one generated from the best solution and other from any random solution in the population. So, for present case the σ value is changed as per FPA. The modification is applied in the global pollination phase and Eq. (7.8) and Eq. (7.9) changes to:

For MMFPA:

$$x_i^{t+1} = x_i^t + 0.5(T^* - x_i^t).(G(\alpha) + C(\delta)) \tag{7.11}$$

For AMMFPA:

$$x_i^{t+1} = x_i^t + G(\alpha).(T^* - x_i^t) + C(\delta).(T^* - x_i^t) \tag{7.12}$$

The above procedure shows that MM uses sum of two distributions to generate new solution or offspring whereas AMM weighs each distribution before adding the two and hence allows the adaptation of not only distribution parameters, but also the shape. The pseudo-code for combined mutated FPAs is same as that given in Algorithm 2 except for the changes in global pollination phase where Eq. (2.6) is replaced by Eq. (7.8) for MMFPA and Eq. (7.9) for AMMFPA.

7.2.4 Adaptive Lévy FPA

The concept of adaptive Lévy was introduced by Yao and Lee (Yao *et al.*, 1997) in order to balance CEP with fine local and faster global search. This is done by deriving the fittest distribution among the candidate pool of four different distributions. The probability density function for Lévy distribution is given by

$$f_L(\lambda, \gamma) = \frac{1}{\pi} \int_0^\infty e^{-\gamma q^\lambda} cos(q\lambda) dq \tag{7.13}$$

where $1 < \lambda < 2$, $\gamma > 0$ and is set to 1. On the basis of λ values, four different equations are proposed. When $\lambda = 1$, the distribution reduces to Cauchy and when $\lambda = 2$, it generates Gaussian distribution. Two other values of 1.3 and 1.7 are used to generate two Lévy random 1 and Lévy random 2 distributions respectively. The four equations are formulated using the values of λ are given as:

$$x_{i,1}^{t+1} = x_i^t + G(\lambda)(T^* - x_i^t) \tag{7.14}$$

$$x_{i,2}^{t+1} = x_i^t + C(\lambda)(T^* - x_i^t) \tag{7.15}$$

$$x_{i,3}^{t+1} = x_i^t + L_1(\lambda)(T^* - x_i^t) \tag{7.16}$$

$$x_{i,4}^{t+1} = x_i^t + L_2(\lambda)(T^* - x_i^t) \tag{7.17}$$

where $G(\lambda)$, $C(\lambda)$, $L_1(\lambda)$ and $L_2(\lambda)$ corresponds to values of Lévy distributed random numbers with respect to Gaussian distribution, Cauchy distribution, Lévy random 1 and

Lévy random 2 respectively. All the four equations are used simultaneously to generate new solutions and based upon the fitness one among them is selected. This increases the computational intensity of basic algorithm. By using the same phenomenon in the global pollination phase, adaptive Lévy (AL) FPA is proposed in this section. A similar mechanism is followed in local pollination phase to counter balance the maximum number of search equations. The four search equations used in local pollination use two neighbouring solutions each and every solution is different. The general equations for this phase are given by

$$x_{i,5}^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t) \tag{7.18}$$

$$x_{i,6}^{t+1} = x_i^t + \epsilon(x_l^t - x_m^t) \tag{7.19}$$

$$x_{i,7}^{t+1} = x_i^t + \epsilon(x_n^t - x_o^t) \tag{7.20}$$

$$x_{i,8}^{t+1} = x_i^t + \epsilon(x_p^t - x_q^t) \tag{7.21}$$

Here x_j^t, x_k^t, x_l^t, x_m^t, x_n^t, x_o^t, x_p^t and x_q^t corresponds to random solutions in the search pool with $j \neq k \neq l \neq m \neq n \neq o \neq p \neq q \neq i$ and each solution is generated from the combined population of flower pollinators. The value of ϵ is chosen from a uniform distribution in the range of [0, 1]. These four equation are evaluated based on their fitness and best among them is selected and used to generate the final output. Since the solution is updated four times, the population size is reduced to $\frac{1}{4}$ so that the number of function evaluations remain same with respect to standard algorithm. he pseudo-code for adaptive Lévy-FPA (ALFPA) is given in Algorithm 9.

7.2.5 FPA as an extension of DE

Since all the above proposed variants are based on the mutation operators so it is imperative to study the mutation operators of state-of-the-art algorithms with respect to mutation operators of FPA. The work here focuses on DE algorithm. The general discussion about basic DE is out of the scope of this paper and a better understanding about their basics can be have from publication (Storn and Price, 1997). First of all, let us discuss about the operators of DE with respect to FPA. DE has two main operators that is

Algorithm 9 Pseudo-code of original ALFPA algorithm

Begin

Initialize flower population: $\boldsymbol{n}$

Define dynamic switch probability

Define $\boldsymbol{d}-$dimensional objective function, $\boldsymbol{f(x)}$

for i= 1: t_{max} **do**

if **rand** $<$ **p**

A d$-$dimensional Gaussian-based step vector is drawn.

Perform global pollination: Eq. (7.14), Eq. (7.15), Eq. (7.16) & Eq. (7.17)

else

Perform local pollination: Eq. (7.18), Eq. (7.19), Eq. (7.20) & Eq. (7.21)

end if

if x_i^{t+1} better than x_i^t

$x_i^t = x_i^{t+1}$

end if

Update probability using Eq. (7.1)

evaluate the current best

update final **best**

End

crossover rate (CR) and scaling factor (F). Various theories have been proposed claiming different values of CR and F to be efficient. Some quote it to be between $0.4 < F < 0.95$ and $0 < CR < 0.2$ (Ronkkonen *et al.*, 2005), some as $0.15 < F < 0.5$ and $0.1 < CR < 1.0$ (Cheng *et al.*, 2019), some used fuzzy logics to define a value (Liu and Lampinen, 2005), some use self-adaptive values (Yang *et al.*, 2008) and others use Gaussian distribution for F and self-adaptive values of CR (Abbass, 2002). So generally, the value of F and CR lies between $[0, 1]$. Now when we compare it with FPA, we see that CR and probability switch p both are basically the controlling parameters of exploration and exploitation for different algorithms whereas F and L are the scaling factors. Also in both the algorithms, it has been explicitly proven that the values of CR and p either lies between $[0, 1]$ or is self-adaptive or dynamic in nature and the values of F and L can be from any standard distribution function or it can also be a random variable. So, we can say CR is equivalent

to p of FPA and F is simply the Lévy (L) distributed step size. Since the mutation parameters in both the algorithms are comparatively similar with values either chosen randomly or by using any specific distribution, so it is concluded that FPA is similar to DE. This conclusion is based on the fact that the general equation for DE is similar to Eq. (2.6) of FPA with only difference of L and F. F in DE is a random variable in the range of $[0, 1]$ or a standard distribution. So, using F as Lévy distribution makes the general equation same as that of FPA. A proper exploitation phase is missing in the DE algorithm whereas in FPA improvement is added in terms of local pollination where the exploitation takes place. Overall, we can say that FPA is an extension of DE having scaling factor produced by a standard Lévy distribution and addition of one local pollination phase to improve the exploitation tendencies of DE.

7.3 Result and Discussion

In this section, we provide details about the proposed variants of FPA, analyse them and see whether they improve the performance of basic FPA or not. First of all, we aim to study the proposed variants with respect to FPA for test functions and secondly compare the best variant among them with the state-of-the-art algorithms. The influence of population and mutation operators is also discussed explicitly. For testing the variants, 50 runs are performed and results are computed in terms of best, worst, mean and standard deviation values. Wilcoxon ranksum test has also been performed for statistical analysis. The algorithms are run on Dell Inspiron 1464 having Intel core i3, 2.13 GHZ 2.13GHz processor, 4 GB ram and Matlab version 7.10.0 R2010a. The performance of all the proposed variants have been tested on 17 standard benchmark problems from Table A.2 (Suganthan *et al.*, 2005). These benchmark problems are unimodal, multimodal and fixed dimension functions as given by Table A.3. The dimensionality of unimodal and multimodal function depends on the user requirement and for present case we have taken 30 as the standard dimension for these functions. Third set of functions corresponds to the fixed dimension function, these functions mainly aim to show the consistency of an algorithm and are found to have fixed number of variables. The parameter settings of various algorithms used in comparison are given in Table B.1

7.3.1 Influence of Population Size

In the literature, it can be seen that work has been done to identify the influence of population on the performance of FPA algorithm (Draa, 2015) but the authors of that publication have changed the dimension as well. For example, if they are using the dimension of 10, the population size is also changed to 10 and if dimension is 20, the population size is also changed to 20. This makes it difficult to analyse the effect of population on the performance of FPA. In present work, the dimension of the problem is fixed and population size is varied. Three different population sizes (n) of 40, 60 and 80 are used to test the performance of FPA and the proposed variants. Table C.29 shows comparison for n=40, Table C.30 for n=60 and Table C.31 for n= 80 for seventeen benchmark functions.

Case I: Population size 40: In Table C.29, the simulation results for population size 40 are presented and it can be seen that for f_1, only ALFPA was able to reach global optimum while all other proposed versions stuck at some non-optimal solution. In f_2 ALFPA provides exact optimum results. For rest of the variants, the results are competitive but ALFPA is far better. For f_3, standard deviation for ALFPA is better in comparison to others. For f_4, ALFPA gives the worst results while all other variants fives quite competitive results. The best algorithm in this case is the CFPA. For f_5, AMMFPA gives the best results while other are still competitive. For f_6 and f_7 respectively, FPA and CFPA has the best standard deviation and are found to provide better results for these fixed dimensions function. For f_8 and f_9, AMMFPA, CFPA, GFPA, MMFPA and standard FPA are very competitive and among all AMMFPA is found to be best. For f_{10}, all the algorithms are competitive but ALFPA provides the best results. For f_{11}, AMMFPA, CFPA and MMFPA are found to be highly competitive while rest are comparable. For f_{12}, ALFPA provides exact global optimum solutions. AMMFPA and CFPA also provide the best solution but for worst, mean and standard deviation ALFPA is far better. For f_{13}, f_{14} and f_{15} CFPA is found to be better among all in terms of best, worst, mean as well as standard deviation. For f_{16}, the values of best, worst and mean are same for all the variants. Here ALFPA gives the best standard deviation. For f_{17}, ALFPA, AMMFPA, CFPA, MMFPA and FPA gives the exact zero standard deviation giving no clue on

which should be the best for this function. In this case, GFPA is found to be better for no function, MMFPA for two, FPA for three, AMMFPA for five, ALFPA for 7 and CFPA for eight functions. So, overall CFPA is found to be best among all the proposed variants for population size of 40.

Case II: Population size 60: Table C.30 shows the comparison of various proposed variant of FPA with the standard one. Here for f_1, only ALFPA gives a near optimum solution. In f_2 ALFPA gives an exact zero optimal solution. The results are comparable among rest of the variants but ALFPA is far better. For f_3, standard deviation for ALFPA is best among all. For f_4 and f_5, all the variant gives competitive results but ALFPA provides the best results among all. For f_6 and f_7 respectively, ALFPA has the best standard deviation and is found to provide better results for these fixed dimensions function. For f_8 *and* f_9, AMMFPA and CFPA are quite competitive while rest are comparable. For f_{10}, ALFPA provides the best results. For f_{11}, AMMFPA and CFPA are found to be highly competitive. For f_{12}, ALFPA, AMMFPA, CFPA and FPA gives zero best but for worst, mean and standard deviation ALFPA is much better. For f_{13}, f_{14}, f_{15} *and* f_{16}, ALFPA is found to be better among all in terms of standard deviation for these fixed dimension functions. For f_{17}, ALFPA, AMMFPA, CFPA, MMFPA and FPA gives the exact zero standard deviation and no algorithm can be stated as the best one. In this case, MMFPA for one, FPA for two, AMMFPA for five, CFPA for eight and ALFPA for fourteen functions. So, overall ALFPA is found to be best among all the proposed variants for population size of 60.

Case III: Population size 80: The results for this case are given in Table C.31. For f_1 and f_2 ALFPA is able to attain global optimum while others are not. For f_3, ALFPA has the best standard deviation among all. For f_4 *and* f_5, all the variant gives competitive results but best values are obtained by ALFPA. For f_6, all variants have very competitive standard deviation and for f_7 CFPA gives the best results. For f_8, f_9, f_{10}, f_{11} and f_{12}, ALFPA gives the best results. For fixed dimension functions f_{13}, f_{14}, f_{15} again ALFPA gives the best results and for f_{16} AMMFPA and ALFPA both have almost same standard deviation. For f_{17}, ALFPA, AMMFPA, CFPA, MMFPA and FPA gives the exact zero standard deviation and no algorithm can be stated as the best one. In this case, GFPA for

one, FPA and MMFPA for two, AMMFPA and CFPA for three and ALFPA for fifteen functions. So, overall ALFPA is found to be best among all the proposed variants for population size of 80.

Inferences drawn: The first thing to see here is that five new variants are proposed and apart from GFPA, all other variants are found to be better than FPA algorithm. For population size of 40, FPA is able to provide better results for only three functions while for rest of the fourteen functions, the proposed variants are better. For population size of 60, FPA is better for only two and for 80 population size, it is better for two. So overall, among 51 functions, FPA gives better results for only seven functions while for rest of the function the proposed variants provide better results.

Among all the proposed variants, the worst algorithm is GFPA which is better for only one function, MMFPA is the second last and is better only for five functions, AMMFPA for thirteen functions, CFPA for sixteen functions and ALFPA for thirty-six functions. The total number of functions are only 51 for all the three cases combined but the results are for 78 functions. This is because there are some cases in which two or more variants gives exact result for a particular function. That is, for the function in which two algorithms give same results can be counted two times. The discussion shows that ALFPA gives the best performance among all the proposed variants.

Now as far as population size is concerned, it can be seen that as the population size increases from 40 to 60, the performance of ALFPA improves and it almost gets stagnated on further increase in population size. It can be said from the fact that for first case, ALFPA was better for seven functions while CFPA was better for eight but as the population size was increased in second case, ALFPA became better for fourteen test functions while others were far less comparable. In the final case of 80 population size, ALFPA became better for fifteen functions but the increase from previous case was only modest whereas the total number of function evaluations increased. Overall, we can assume a population size of 60 to be the standard population size for proposed variants of FPA.

7.3.2 Comparative Study

From the above discussion, it has been found that ALFPA is better than other FPA variants. From the above discussed inferences, it can be said that the perfect population

size for comparison should be 60 where the best algorithm is ALFPA. The reason for better performance of ALFPA at a population size of 60 is that for higher population size, the diversity in the population increases. Due to the increased diversity, the exploration increases helping the algorithm to search for better solutions and avoid local minima. This concept is also valid for exploitation or the local pollination phase. The same algorithm doesn't work well for lower population sizes because of less diversity in the solution space. The population is divided into four parts and there are four equations working together to find global minima, so more is the population size more are the chances to get the required results. For higher population sizes, the algorithm gives good results, but it can be seen that ALFPA results don't vary too much with further increase in the population size. Now to prove it to be state-of-the-art, it is compared with well-known algorithms. The major algorithms used in this paper are ABC, DE, BA, FA and GWO and these algorithms are compared with the best proposed variant of FPA. Here all the popular algorithms are compared with ALFPA for a population size of 60. The parameters settings for these algorithms are given in Table B.1.

The results are presented in Table C.32 as: For function f_1, only ALFPA was able to reach global optimum. In f_2 ABC and ALFPA provides exact optimum results. DE is also able to obtain a global best of zero but for worst, mean and standard deviation the results are less significant. For rest of the algorithms, the results are competitive but ALFPA is still better. For f_3, ABC is not able to attain the required optimum, while all other algorithms provide competitive results. ALFPA in this case has the best standard deviation. For f_4, ABC, FA and ALFPA are competitive and better than GWO, BA and DE. Here it is difficult to comment whether ABC or ALFPA is better. For f_5, ALFPA is better than ABC, DE and BA whereas FA is competitive with respect to ALFPA. GWO is found to be the best. For f_6 *and* f_7, ALFPA has the best standard deviation are is found to provide better results for these fixed dimensions function. For f_8 *and* f_9, ALFPA is better than DE and BA. GWO provides the best optimum but the overall results of FA are better. For f_{10}, ALFPA is very competitive when compared to ABC, DE, FA and BA but in this case GWO provides the best results. For f_{11}, ALFPA is better than DE and BA, very competitive with ABC and GWO is found to be the best. For f_{12}, ABC, GWO and ALFPA provides exact global optimum solutions. For f_{13}, f_{14} and f_{15} ALFPA is found to

be better among all in terms of best, worst, mean as well as standard deviation. For f_{16}, the values of best, worst and mean are same for almost every algorithm. Here standard deviation gives the relative comparison among the algorithms. DE and ALFPA are found to provide the best competitive results. For f_{17}, also the standard deviation acts as the deciding factor. Here ALFPA gives the exact zero standard deviation and hence proving to be best among all. Overall, GWO and FA are found to be better for four functions and two functions each, DE for only one function, ABC for two functions while ALFPA gives best results for twelve functions. The convergence profiles are given in Fig. 7.1. These convergence curves again show the superior performance of AFLPA as compared to other algorithms. This conclusion proves the competitiveness of ALFPA and it can be said that ALFPA is best fit candidate for becoming state of art algorithm.

7.3.3 Statistical Testing

The Wilcoxon's rank-sum test (Derrac *et al.*, 2011) is performed to test the performance of ALFPA with other popular algorithms. This test is done to evaluate the performance of two different algorithms or samples. It is a non−parametric test, involving design of two samples and is analogous to paired t-test. So, it is a pairwise test used to identify significant differences among two algorithms. The test gives a p-value as the end result and this p-value determines the significance level of two algorithms under test. An algorithm is said to be statistically significant if the p-value is < 0.05. For comparison, two algorithms are taken for each test function and comparison is performed. This comparison is done with respect to the best algorithm that is if ALFPA is best, comparison is performed between ALFPA/ABC, ALFPA/DE, ALFPA/BA, ALFPA/GWO and so on. Since best algorithm can't be compared with itself so NA has been incorporated for each algorithm and it stands for Not Applicable (Draa, 2015) while '$\sim$' means that both the algorithms have same performance. From the Table D.2, it can be seen that ABC shows better statistical analysis for two functions, DE and BA for none, GWO and FA for three each and ALFPA for twelve functions. So even from the statistical tests, it can be said that ALFPA is much better and is best fit for becoming a standard algorithm.

7.3.4 Summary of Results

The main findings can be summarized as follows.

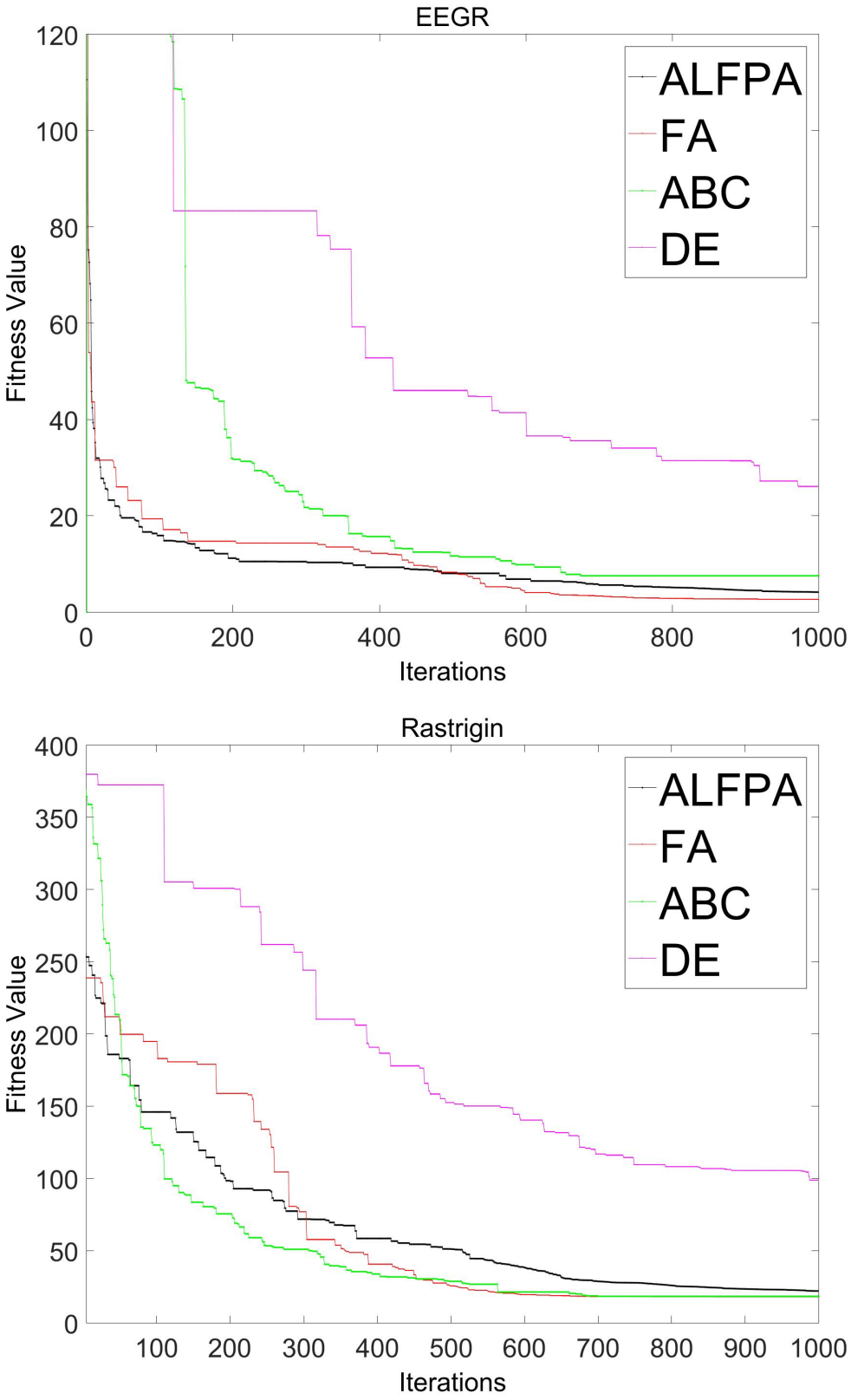


Fig. 7.1 ALFPA algorithm: Convergence curves with respect to other algorithms

- The basic FPA is limited in performance and the already proposed work does not provide proper justification. The variants proposed in present work are competitive and provide good results. ALFPA is found to be the best among the proposed variants.

- The reason for better performance of ALFPA is because of the good exploration capabilities of adaptive component in the global pollination phase, and, at the same time, it inherits the good exploitative capabilities of enhanced local search. Also, the dynamic switching helps to maintain a balance between exploration and exploitation. Overall, ALFPA combines three efficient tools for searching the global solution.

- Although ALFPA is found to be the best algorithm and acts as potential candidate for becoming a state-of-the art, it has some limitations too. The algorithm doesn't work well for lower population and this is because at lower population the algorithm diversity is limited. As the population size required for this algorithm is large, more is the number of function evaluations required to obtain a particular solution for the problem under test. So, more work is to be done to design FPA variant which is fit for lower population size.

- Due to stochastic nature of ALFPA and all other nature inspired algorithms, there is no guarantee for finding 100% optimal solution. Also, the proposed variant is not fit for all the optimization problems. That is ALFPA didn't performed well on all benchmark functions. So, to overcome this problem more work is to be done to design a generic problem solver.

- Though, ALFPA has some limitation but still it fares better than the already proposed work in the literature. Also, the algorithm is quite simple in implementation and because of this it can be included in hybrid intelligent and expert systems.

Further, insightful implications extracted from the experiments are

- Parallelism of ALFPA can be done by dividing the search space into smaller parts using corresponding equations, then each equation under each search space can be

used independently and these spaces can be made to interact with each other after certain set of iterations.

- Extending the algorithm to binary version can be used in electroencephalogram (EEG). In EEG, signals are measured by placing sensors in different positions on a person's head. A binary version of ALFPA can help to reduce total number of sensors required for the operation while maintaining the accuracy of the system.

7.4 Concluding Remarks

This chapter presented modifications to basic FPA in order to enhance its performance. Three modifications to FPA have been proposed which make it competitive with respect to popular algorithms. A dynamic switch probability, an improved mutated global and local search has been used in the enhanced versions of FPA. The mutation techniques give five variants of FPA in which Cauchy, Gaussian, adaptive Lévy, mean mutated and adaptive mean mutated distributions have been used. It has been found that all of these variants except GFPA are better than the basic FPA and ALFPA is best among all the variants. The performance of ALFPA has been also compared with that of well-known algorithms such as DE, BA, FA, ABC and GWO. Experimentally, best, worst, mean and standard deviation has been used to measure the results. Due to stochastic nature of optimization algorithms, statistical tests have been conducted. To measure the speed of attaining the global optima, convergence profiles have also been drawn. Numerical results show the strength of ALFPA in solving the standard benchmark problems.

The enhanced performance of variants can be attributed to dynamic switching which balances effectively local and global search. The global mutations also help in improving the exploration capabilities of the basic FPA which is required to escape local minima. In addition to these changes, in ALFPA the local search equation has been changed to improve the exploitation. The proposed methods of FPA have been tested for different population sizes and it has been observed that for lower population size CFPA gives better results and for medium population size ($n = 60$) ALFPA is better. Further if the population is increased, there is no significant improvement in the results. The CFPA has sufficient number of candidate solutions to explore the search space. Hence it is good at both exploration and exploitation. But in case of ALFPA for lower population size the

diversity of possible solutions is less and hence its exploration is poor. Overall it has been seen that ALFPA fares better than all the variants and other popular algorithms. ALFPA has proved its significance for solving benchmark problems but for it to be state-of -the-art, it has to be applied to various engineering optimization problems like antenna, wireless sensor networks, digital filter design problems, economic load dispatch problems, knapsack problem. It is also possible to study the extension of ALFPA for discrete combinatorial optimization problems. The switching probability which acts as the controlling factor for exploration and exploitation is very delicate. The dynamic switching probability used in proposed approaches is linear in nature. One can use sigmoidal or exponential functions to properly identify the best switching probability. A balanced exploration and exploitation can also be introduced. Further, one can extend the work by modifying the search equations of local and global pollination phase. More research can be done by exploiting the epsilon parameter in the local search part of FPA. The inertia weight factors can also be introduced in local and global pollination phase of standard FPA. Besides these implementations, step sizes can be analysed to propose new FPA variants.

CHAPTER 8

Enhanced Flower Pollination Algorithm for Antenna Applications

Preface

In this chapter, a new variant of FPA namely enhanced flower pollination algorithm (EFPA) has been proposed for the pattern synthesis of non-uniform linear antenna arrays (LAA). The proposed algorithm uses the concept of Cauchy mutation in global pollination and enhanced local search to improve the exploration and exploitation tendencies of FPA. It also uses dynamic switching to control the rate of exploration and exploitation. Experimental and statistical results show that EFPA algorithm provides enhanced performance in terms of side lobe suppression and null control than FPA and other popular algorithms.

Publication: Urvinder Singh, and **Rohit Salgotra**. "Pattern synthesis of linear antenna arrays using enhanced flower pollination algorithm." International Journal of Antennas and Propagation 2017 (2017).

8.1 Introduction

FPA as discussed in the above chapters, is based on the flowering patterns and pollination process in flowers. They are the most fascinating plant species and have been dominating earth from the cretaceous period, partly from about 125 million years. Almost 80% of plant species are flowering and it has been made possible by the process of pollination. Four basic rules were proposed for randomization patterns, local searching capabilities and global search and finally controlling the extent of both of these patterns. The algorithm uses a single flower to be the solution for the problem under test.

A large number of articles have been published in the recent past on the performance evaluation and applicability of FPA. The design of antenna arrays with desired radiation pattern has found great interest in the literature. Antenna arrays design is intricate and non-linear problem. Hence number of optimization techniques such as GA (Holland, 1992), DE (Storn and Price, 1997), PSO (Kennedy and Eberhart, 1995), BBO (Simon, 2008) and many others have been used to synthesize antenna arrays (Singh *et al.*, 2010, Rattan *et al.*, 2007, Pappula and Ghosh, 2017). It is required that antenna arrays radiate in desired directions so that these do no add to electromagnetic pollution. This can be achieved if the energy is maximum in the main lobe and minimum in side lobes. Moreover, to avoid interference from undesired directions and to circumvent jamming, null placement in radiation pattern has also gained importance. So, overall the main issues in designing of the antenna arrays are reducing the side lobe level (SLL) and placing nulls in the desired directions of radiation pattern.

LAA is one such problem where FPA finds applications and various enhanced versions of the same have been proposed. Recently, a novel algorithm which mimics the behavior flowers known as bat-flower pollination algorithm (FPA) has been used to synthesize LAA (Salgotra and Singh, 2018). Though number of algorithms have been proposed for synthesis of LAA (Khodier and Christodoulou, 2005, Saxena and Kothari, 2016a, Chowdhury *et al.*, 2010, Cengiz and Tokat, 2008), these suffer from certain shortcomings like getting stuck in local minima, slow convergence speeds, precise parameter tuning. So, in this work, the authors propose an enhanced version of FPA known as enhanced FPA (EFPA). The newly proposed algorithm uses the concept of Cauchy distribution to follow

large steps in global pollination, enhanced local search and dynamic switch probability to control the rate of local and global pollination. It has better exploration and exploitation capability and also is less likely to stuck in local minima.

The rest of the organized as: Section 2 proposes a new EFPA algorithm, section 3 gives result and discussion whereas the final section 4 provides an extensive conclusion.

8.2 The Proposed Approach

In the recent past, a large number of researchers have focused on enhancing the basic capabilities of FPA. The algorithm due to its linear nature, make it suitable for deeper analysis. But it has been proved qualitatively and quantitatively in (Salgotra and Singh, 2017) that the FPA algorithm has a very limited scope for optimization problems at hand. Also the performance of FPA has not been analysed to a deeper level and the algorithm is still to prove its worthiness for becoming a state of art algorithm. Keeping in view the above analysis, a new version of FPA namely EFPA has been proposed. The newly proposed EFPA aims at providing three different modifications to the basic FPA. These include, Cauchy based global pollination, enhanced local pollination based upon experience of current best flower pollinator as well as local flowers in proximity and thirdly, by using dynamic switching probabilities.

Cauchy based global pollination: In the global pollination phase, instead of using a standard Lévy distribution, a Cauchy based operator $C(\delta)$ is used. This operator is basically a Cauchy random variable with distribution given by (Yao *et al.*, 1999):

$$\delta = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x}{g}\right) \tag{8.1}$$

The Cauchy density function is given by

$$f_{Cauchy(0,g)}(x) = \frac{1}{\pi}\frac{g}{g^2 + x^2} \tag{8.2}$$

The general equation of global pollination becomes:

$$x_i^{t+1} = x_i^t + \ C(\delta)(R_* - x_i^t) \tag{8.3}$$

where g is a scale parameter and its value is generally set to 1. The use of Cauchy operator allows for larger mutation by searching the search space at a faster pace and further accounts for avoiding premature convergence.

Enhanced local pollination: The second modification is added in the local pollination phase, here based upon the experience of local and current best pollinators, the position of new pollinators is updated. Further, if the fitness of new position is greater than the old one, each pollinator updates its position with respect to the previous one. The general equation is given by:

$$x_i^{t+1} = x_i^t + \ a\left(R_* - x_i^t\right) + b(x_j^t - x_k^t) \tag{8.4}$$

where a and b are uniformly distributed random numbers in the range of [0,1]. Also, the solutions x_j^t *and* x_k^t, corresponds to j^{th} and k^{th} flower pollinator in the group with $j \neq k$. This phase, enhances the local search capabilities of FPA algorithm.

Dynamic switch probability: In FPA, local and global pollinations are controlled by switching probability $p \ \epsilon \ [0, 1]$. For a standard state of art algorithm, it is imperative to follow more global search at the start and as the algorithm progresses more intensive local search is followed. Using this basic concept, the value of switching probability is selected dynamically. The switch probability is updated by following a general formula as:

$$p = p - \frac{t_{max} - t}{t_{max}} \times (0.1) \tag{8.5}$$

The above general equation decreases the value of p linearly with iterations and hence adds to intensive global search at the beginning and local search towards the end. Here t_{max} corresponds to the maximum number of iterations and t is the current iteration. The pseudo-code of EFPA is shown in Algorithm algorithm 10.

8.3 Results and Discussion

The performance of EFPA is evaluated by applying it to two set of optimization problems. One set includes test functions and other is the real world application of antenna array design. The algorithm is also compared with other state of art to prove its competitiveness. The simulations are performed on Intel Core i3 personal computer with windows 10 operating system using MATLAB version 7.10.0 (R2010a). EFPA is tested on 7 well known benchmark problems (Suganthan *et al.*, 2005) as shown in Table A.2 with search range and optimal solution. For comparison purposes, ABC (Karaboga and Basturk, 2008), DE (Storn and Price, 1997), BA (Yang, 2010b) and FPA (Yang, 2012) are used.

Algorithm 10 Pseudo-code of original EFPA algorithm

Begin

Initialize flower population: $\boldsymbol{n}$

Define dynamic switch probability

Define $\boldsymbol{d}-$dimensional objective function, $\boldsymbol{f(x)}$

for i= 1: t_{max} **do**

if **rand** $<$ **p**

A d$-$dimensional Gaussian-based step vector is drawn.

Perform global pollination: Eq. (8.3)

else

Perform local pollination: Eq. (8.4)

end if

if x_i^{t+1} better than x_i^t

$x_i^t = x_i^{t+1}$

end if

Update probability using Eq. (8.5)

evaluate the current best

update final **best**

End

The maximum number of function evaluations for each algorithm is set to 30 $\times$ 1000, with 30 as the population size and 1000 as the maximum number of iterations. The parameter settings of all algorithms are given in Table B.1. All the algorithms are run 50 times and results in terms of best, worst, mean and standard deviation are reported. Since because of the stochastic nature of algorithm, Wilcoxon rank-sum test (Derrac *et al.*, 2011) has also been performed to significantly prove the better performance of EFPA. The performance in terms of time-complexity has also been shown in Table 8.1 to prove its competitiveness. Time-complexity is basically taken for each of the algorithms in the benchmark problems and it has been calculated for one run for each algorithm.

8.3.1 CEC 2005: Benchmark Results

The result of comparison for various test functions is given in Table C.33 . The best and worst values give the most optimal and least optimal solution among the 50 solutions

Table 8.1 EFPA algorithm: Time-complexity analysis

Function	DE	ABC	BA	FPA	EFPA
$f_1(x)$	1.284464	2.527247	1.190053	2.328157	1.965927
$f_2(x)$	15.98023	15.62962	19.02841	15.22211	14.80168
$f_3(x)$	1.185347	2.470673	1.047533	2.183295	1.900515
$f_4(x)$	3.022755	5.167423	4.165333	2.183153	2.030502
$f_5(x)$	2.848487	5.988428	2.974728	3.755994	3.498411
$f_6(x)$	1.667705	3.186360	2.261759	2.346810	2.383599
$f_7(x)$	1.693011	3.265792	2.087382	3.384049	2.999886

from the maximum number of runs. Mean gives the average values of results and is used when the best and worst values don't give comparable results. But mean may give same results for widely different results. So standard deviation is also used to calculate results. For function f_1, f_4, f_5, f_6 and f_7 it can be seen that EFPA performs much better than other algorithms except for f_2 where ABC gives better results. For function f_3 ABC and EFPA gives same results. Further statistical analysis from the Table D.3 verify the results. In Table 8.1 , time-complexity for each algorithm is shown. Here time-complexity show that EFPA takes comparatively less time for solving the benchmark problem for most of the benchmark functions in comparison to FPA, BA and ABC. For DE, only four functions gives better time complexity while for rest three, again EFPA is better. Overall, we can say that EFPA is superior to ABC, DE, FPA and BA. The convergence graohs for FEPA are shown in Fig. 8.1. In the next sub-section, EFPA is applied for synthesis of LAA.

8.3.2 Non-uniform Linear Antenna Array Design

The design of non-uniform LAA has gained importance in the past and several researchers have synthesized it using different optimization techniques (Khodier and Christodoulou, 2005), (Rattan *et al.*, 2007), (Dib *et al.*, 2010), (Pal *et al.*, 2010). The array factor of a non-uniform array is aperiodic by virtue of non-uniform spacing between the elements. This aperiodic nature of arrays helps in obtaining lower SLL with lesser number of elements for a given antenna size. Furthermore, the uniform element excitations help in reducing the feed network cost and complexity. However, the relationship between the array factor and

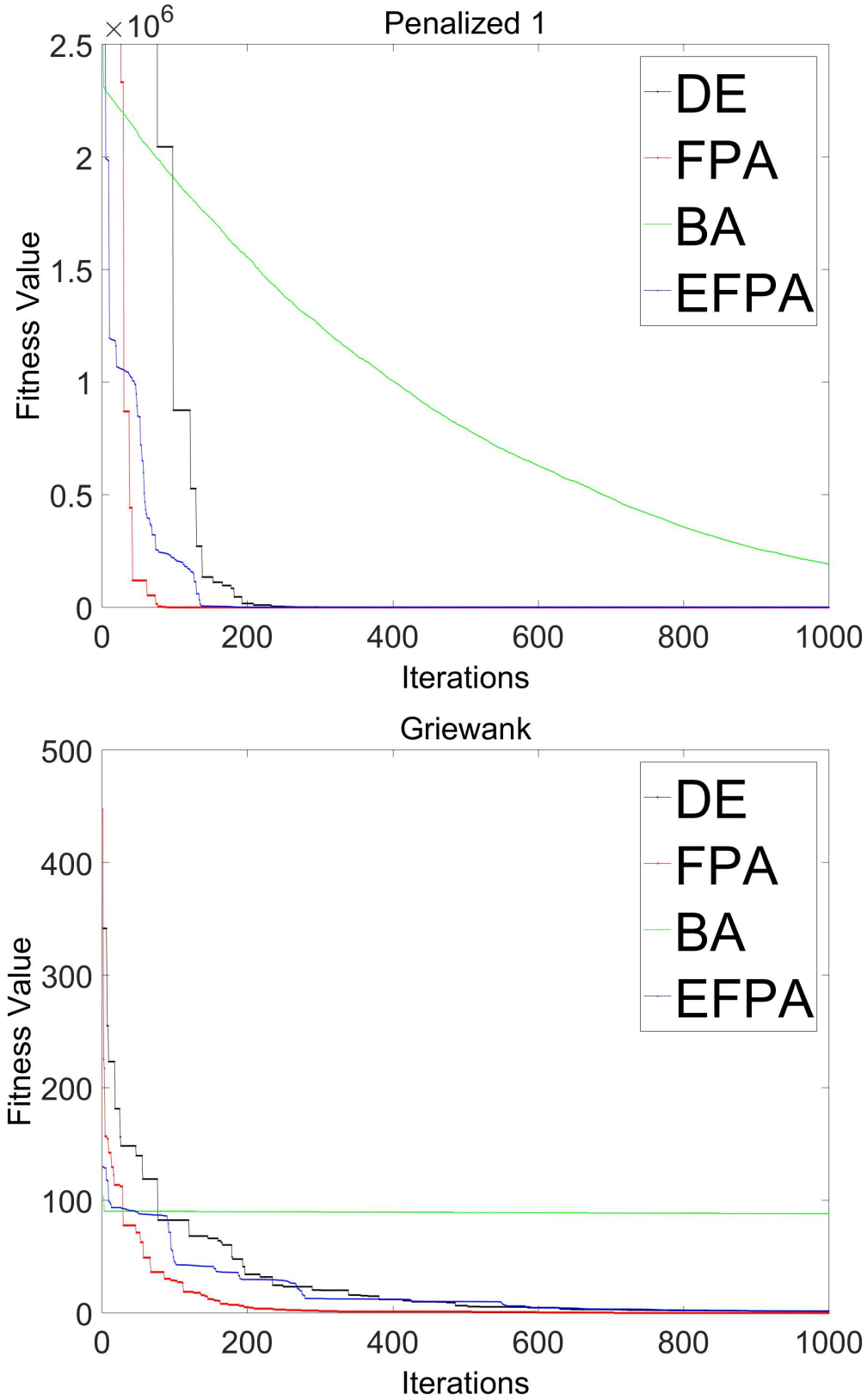


Fig. 8.1 EFPA algorithm: Convergence curves for CEC 2005 benchmarks

element spacing is non-convex and non-linear. So, this makes the design of non-uniform arrays difficult.

A 2N LAA geometry is shown in Fig. 8.2. It consists of antenna elements arranged linearly in a straight line. The array factor (AF) defines the radiation pattern of antenna array varying with element amplitude excitation I_n, element spacing x_n and element phase excitation $\emptyset_n$. The AF is given by (Balanis, 2016, Salgotra and Singh, 2018):

$$AF\left(\varphi\right) = \sum_{n=1}^{2N} I_n \exp(j([kx_n \cos\left(\phi\right) \ + \varphi]) \tag{8.6}$$

where k is the wave number given by $\frac{2\pi}{\lambda}$. Assuming the antenna elements are placed in

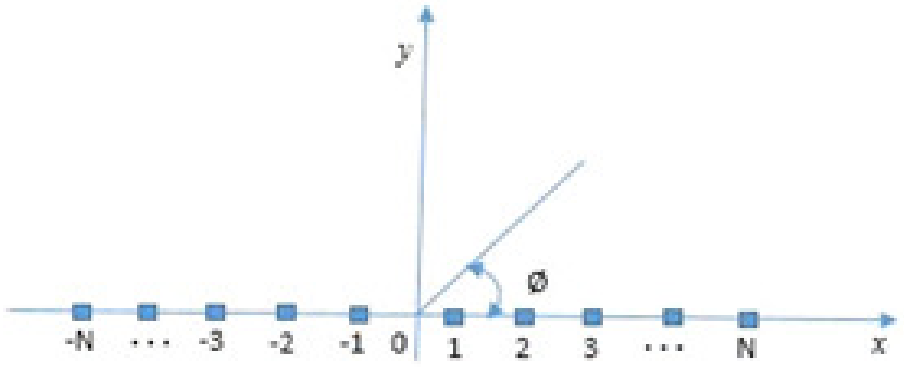


Fig. 8.2 A 2N-Symmetric Linear Antenna Array

a symmetric fashion along the x-axis, the AF can be simplified as

$$AF\left(\varphi\right) = 2\sum_{n=1}^{N} I_n \exp(j([kx_n \cos\left(\phi\right) \ + \varphi]) \tag{8.7}$$

Due the geometrical symmetry of the array, the radiation pattern is symmetric and it is useful in many applications. Moreover, while designing, only N elements need to be optimized instead of $2N$ which reduces the computational cost. LAA can be further classified into two types of arrays namely, equally spaced or uniform arrays and unequally spaced arrays or non-uniform arrays. As the name goes, equally spaced arrays have equal or same spacing between the antenna elements. Unequally spaced arrays have non-uniform spacing but uniform element excitation. Antenna arrays find their applications mainly in radar, sonar, mobile and satellite systems. These provide high gain, aid in enhancing

signal quality, extend the coverage range and also increase the spectrum efficiency in case of wireless systems. Hence, the performance index of wireless systems rely heavily on the design of these arrays. Antenna array design aims to obtain a physical structure consisting of antenna elements that will have desired radiation pattern. Based on their geometry, the arrays can be grouped as linear, circular, rectangular and concentric. In case of LAA, the elements are placed along a straight line. These arrays are very popular because of their simple geometry. However, the design of LAA is complex and non-linear. Hence, it has prompted number of researchers to take up this challenging problem in the past (Rattan *et al.*, 2007)- Pappula and Ghosh (2017). Rattan *et al.* have applied GA for the synthesis of aperiodic LAAs. The authors designed two antennas with 28 and 32 elements with the help of GA. The results were better than PSO and quadrature programming method (QPM) (Rattan *et al.*, 2007). Dib *et al.* have contrasted the performance of Tagacuhi's method (TM) and self-adaptive DE (SADE) for the design of linear antennas. Number of antenna designs of different sizes including uniform and non-uniform antennas were taken up. The authors also compared their results with other algorithms and showed that these were better than other popular algorithms (Dib *et al.*, 2010). Unequally spaced antenna arrays have been synthesised by (Lin *et al.*, 2010). DE using best of random differential mutations has been employed for 32-element and 37-element linear arrays. The best of random differential offers better balance between exploration and exploitation. In this work, symmetric aperiodic linear antennas for both position-only and phase-only synthesis have been studied. The authors also investigated the angle resolution effect on the antenna radiation pattern and it was found that 0.2^0 of resolution gave consistent results with the previous reported results. In (Merad *et al.*, 2008), tabu search (TS) has been used to find the optimized amplitudes of elements of linear arrays. Zaman *et al.* have designed non-uniform spaced antennas using firefly algorithm (Zaman and Abdul Matin, 2012). Khodier has used CS for the optimization antenna arrays (Khodier, 2013). Different design examples have been taken up to illustrate the use of the CS algorithm. Results have been compared with results obtained using other popular methods and it was found that CS algorithm outperforms other evolutionary algorithms (EAs). A new optimization algorithm namely cat swarm optimization (CSO) was applied for the design of linear antennas (Pappula and Ghosh, 2014). CSO has been used to get the

optimum element positions so as to supress the side lobe level and also to introduce nulls in the required directions. A number of examples have been considered and results have been compared with results of PSO to validate the results of CSO. Khodier and Christodoulou have presented the design of non-uniform antennas using PSO for reducing the SLL and null control (Khodier and Christodoulou, 2005). An improved version of PSO namely comprehensive learning PSO (CLSPSO) has been employed for the design of LAAs (Khodier and Al-Aqeel, 2009). Authors have used a fitness function which employs penalty method to achieve optimum antenna design. The results of CLSPSO are better than those of PSO (Goudos *et al.*, 2010). Recently, FPA has been used for the design of both uniform and non-uniform antennas (Singh and Salgotra, 2018). FPA is a simple and robust technique based on the pollination of flowers. The authors have compared the FPA results extensively with well-known techniques available in the literature. Subhashini and Satapathy have applied a modified version of ALO namely enhanced ALO (e-ALO) to synthesise the linear antennas (Subhashini and Satapathy, 2017). There are number of other publications related to LAA optimization and are listed as GWO used for LAA in (Saxena and Kothari, 2016b), adaptive DE in (Chowdhury *et al.*, 2010), dipole element optimization using FPA in (Ram *et al.*, 2019), enhanced FPA in (Singh and Salgotra, 2017), mutual coupling using BA in (Das *et al.*, 2017) and others.

For directing antenna arrays energy in a specific direction, it is desired that its radiation pattern should have low side lobes. Moreover, nulls are also required in the direction of interference. The desired radiation pattern can be achieved by suitably adjusting the positions of antenna elements for a non-uniform LAA. The adjustment of positions is a tricky and non-linear problem. So, optimization algorithms are well suited for this type of problems. For applying, optimization to antenna problem, it is imperative to define a fitness function which is given as:

$$fit = SLL + \alpha \times max\left\{0, |BW - BW_d| - 1\right\} + \alpha * \left\{\sum_{k=1}^{K} max\left\{0,\ AF_{dB}\left(\phi_{\mathrm{k}}\right) - \mathrm{Nu}_{dB}\right\}\right\} \tag{8.8}$$

In the above fitness function, SLL is the side lobe level, AF_{dB} is the array factor in

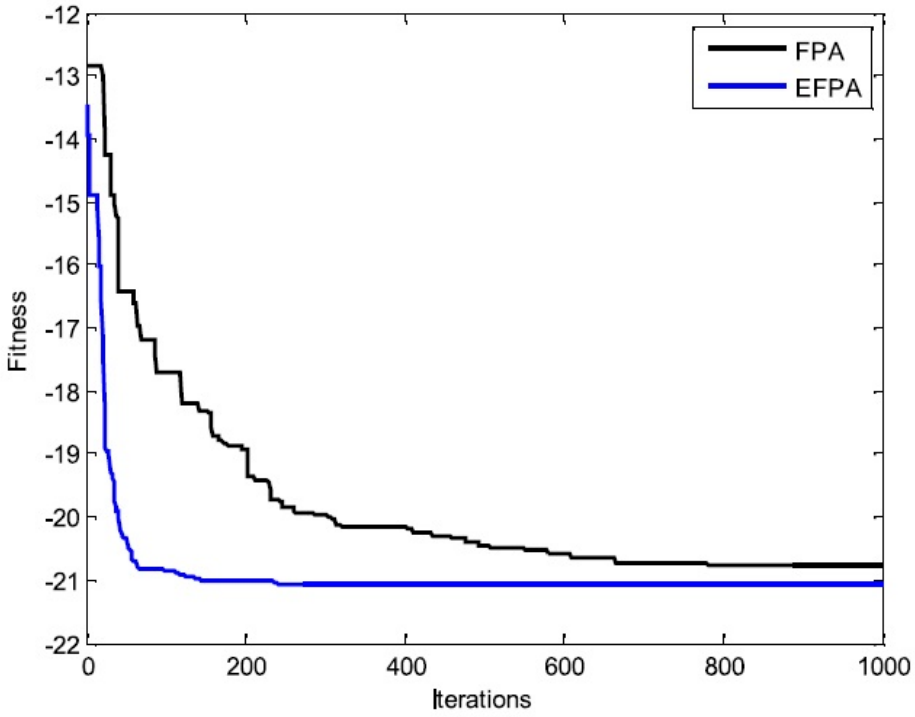


Fig. 8.3 EFPA algorithm: Convergence profile for 12-element non-uniform LAA

decibels, BW and BW_d are the calculated and desired beamwidth of the main lobe, ϕ_{k} gives the direction of the k^{th} null, Nu_{dB} is the required null depth in dB, K is the number of the required nulls, and α is a very large number. The fitness function given in Eq. (8.8) uses a penalty method which penalizes any antenna design which does not meet the required constraints of beamwidth and nulls (if desired). This is achieved by taking value α to be very large. The BW here is obtained computationally from the radiation pattern data.

12-element non-uniform LAA: In the first example, a 12-element non-uniform antenna is taken for optimization. The goal of optimization is to minimize the SLL in the region $\phi \in [98^0, 180^0]$. Because of symmetry there are only six element positions which are to be optimized. The number of generations for EFPA are taken as 1000. The population size taken as 20. As the aim of the optimization is to suppress the SLL only, so the value of α is taken as 0 in the fitness equation of Eq. (8.8). The EFPA algorithm is run for 30 times and best results are listed in this work. The optimum positions obtained using EFPA are shown in Table E.2 . The performance of EFPA is contrasted against other

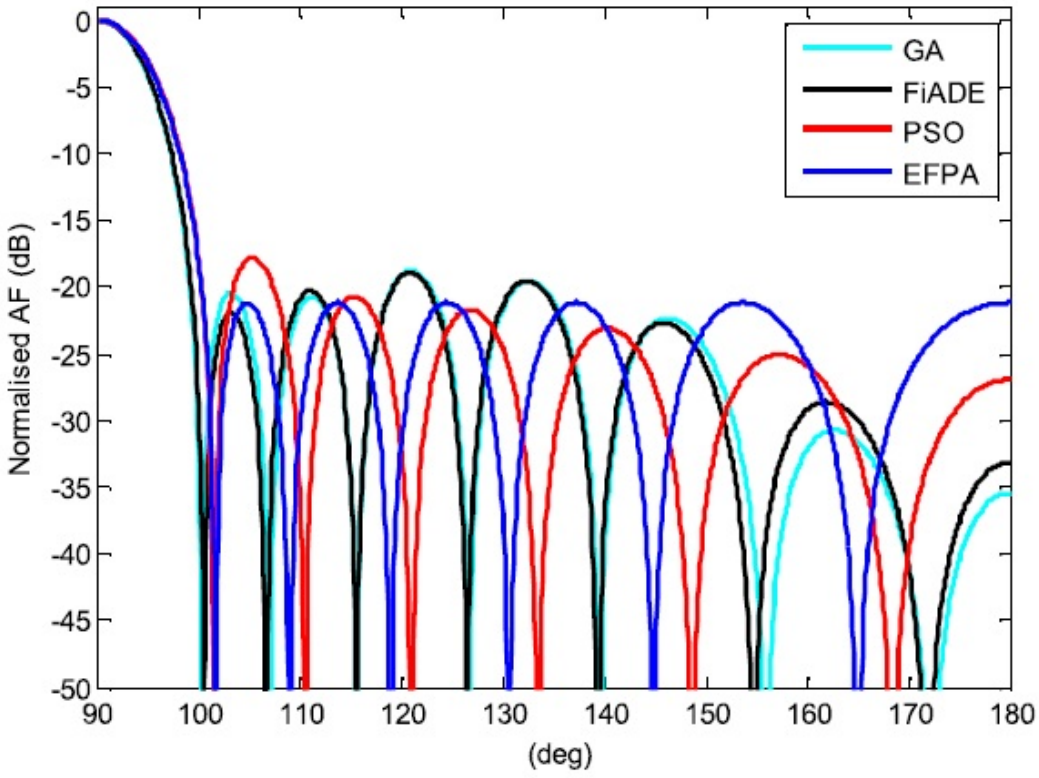


Fig. 8.4 EFPA algorithm: Radiation Pattern for 12-element non-uniform LAA

popular algorithms in terms of SLL in Table E.3 . The minimum SLL obtained by EFPA is -21.07 dB and is better than GA (Cengiz and Tokat, 2008), FiADE (Chowdhury *et al.*, 2010), TS (Cengiz and Tokat, 2008), MA (Cengiz and Tokat, 2008), FPA (Salgotra and Singh, 2018) and PSO (Chowdhury *et al.*, 2010). The comparison of convergence curve between EFPA and FPA is shown in Fig. 8.3 which clearly shows faster convergence of EFPA than the FPA (Salgotra and Singh, 2018). The FPA converged in about 700 iterations whereas EFPA took only 225 iterations to converge to the final solution. This is due to enhanced exploration due to Cauchy operator and better exploitation in EFPA. For comparison the radiation patterns of GA (Cengiz and Tokat, 2008), FiADE (Chowdhury *et al.*, 2010) and PSO (Chowdhury *et al.*, 2010) are also shown in Fig. 8.4.

22-element non-uniform LAA: In the next example, a 22-element non-uniform LAA is optimized for reducing SLL in the region $\phi \in [98^0, 180^0]$ and also imposing a null in the direction of 99^0. For achieving the required pattern, the value of Nu_{dB} is set to -60 dB. It is established fact that as SLL is decreased the width of the main lobe in radiation pattern

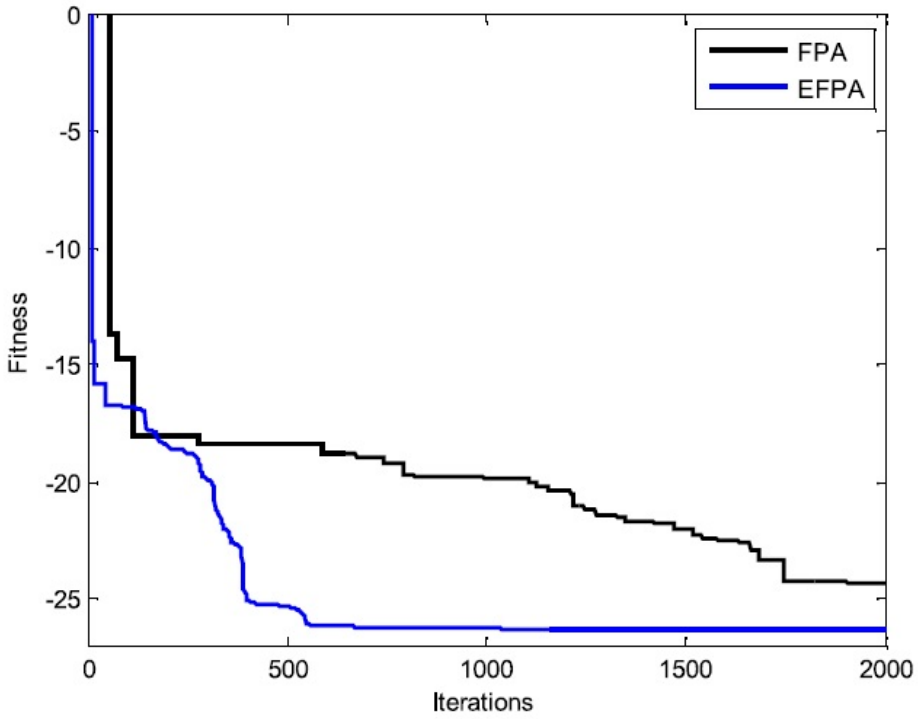


Fig. 8.5 EFPA algorithm: Convergence profile for 22-element non-uniform LAA

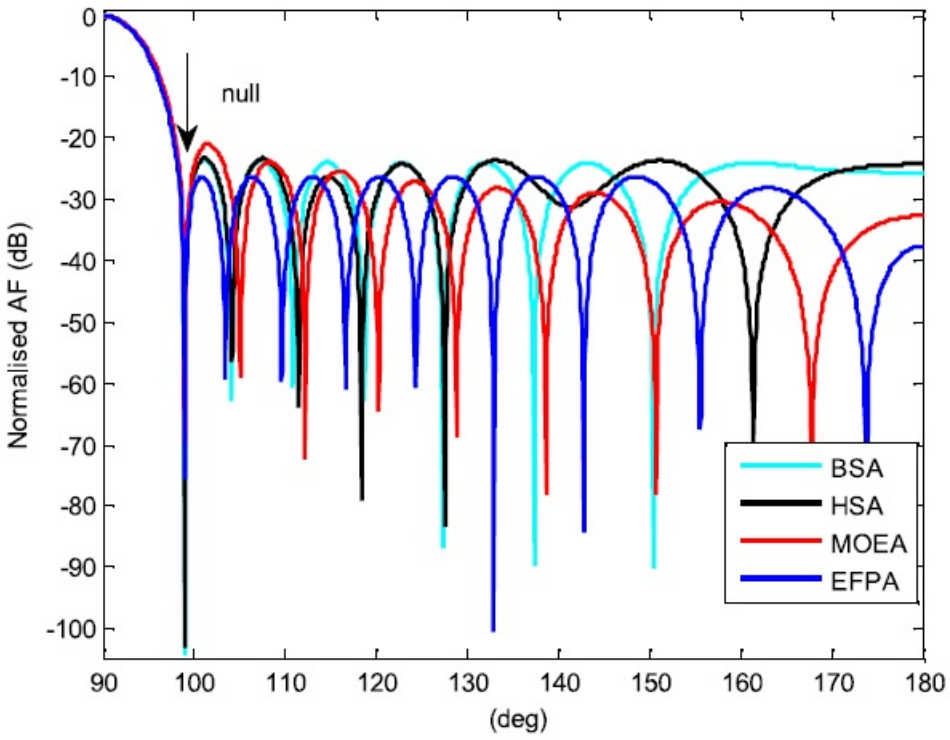


Fig. 8.6 EFPA algorithm: Radiation Pattern for 22-element non-uniform LAA

increases. So, for this design, the desired beamwidth BW_d is taken as 18^0. The value of α is taken as 10^6 in fitness function. The optimum positions are shown in Table E.4 . The results are compared in Table E.5 which clearly indicate the better performance of EFPA as against BSA (Guney and Durmus, 2015), MOEA/DE (Pal *et al.*, 2010), HSA (Guney and Onay, 2011), PSO (Chowdhury *et al.*, 2010), GA (Cengiz and Tokat, 2008), MA (Cengiz and Tokat, 2008), TS (Cengiz and Tokat, 2008) and FPA (Salgotra and Singh, 2018). The SLL of EFPA antenna array is -26.31 dB which is much lower than obtained using other algorithms. Furthermore, the null in the direction of 99^0 is -75.58 dB which is less than the desired value of -60 dB and deeper than MOEA/DE (Pal *et al.*, 2010), GA (Rattan *et al.*, 2007), PSO (Khodier and Christodoulou, 2005) and MA (Cengiz and Tokat, 2008). In order to compare the performance of EFPA and FPA in terms of convergence speed, FPA is also run for the same objective with same population size. The convergence graphs of EFPA and FPA are plotted in the Fig. 8.5. The convergence characteristics clearly substantiates the superior performance of EPFA in terms of faster convergence speed. The radiation plots of EFPA, HSA (Guney and Onay, 2011), BSA (Guney and Durmus, 2015) and MOEA/DE (Pal *et al.*, 2010) optimized arrays are shown in Fig. 8.6.

28-element non-uniform LAA: In order to show the capability of EFPA in synthesizing the radiation pattern with multiple nulls, a 28-element non-uniform LAA is optimized for SLL reduction in the region $\phi \in [100^0, 180^0]$ and having nulls in the directions of 120^0, 122.5^0 and 125^0. The desired beamwidth is set to the value of 8.35^0 with tolerance of $\pm\ 1^0$. The optimum geometry obtained using EFPA is shown in Table E.6 . The comparison of convergence curves between EFPA and BBO (Singh and Rattan, 2014), CS (Singh and Rattan, 2014), DE (Singh and Rattan, 2014), FA (Singh and Rattan, 2014), COA (Singh and Rattan, 2014) is given in Fig. 8.7 which shows that EFPA reaches the optimum solution in less number of iterations. The comparison of results in terms of SLL and null values achieved is given in Table E.7 which again clearly indicates the superior performance of EFPA as against other methods. The SLL of EFPA is -22.90 dB which is lower by 1 dB, 9.24 dB, 9.3 dB, 1.04 dB, 8.26 dB, 9.24 dB, 1.27 dB and 0.30 dB than BSA (Guney and Durmus, 2015), QPM (Khodier and Christodoulou, 2005), COA (Singh and Rattan, 2014), ACO (Raja *et al.*, 2015), GA (Rattan *et al.*, 2007), CLPSO (Gou-

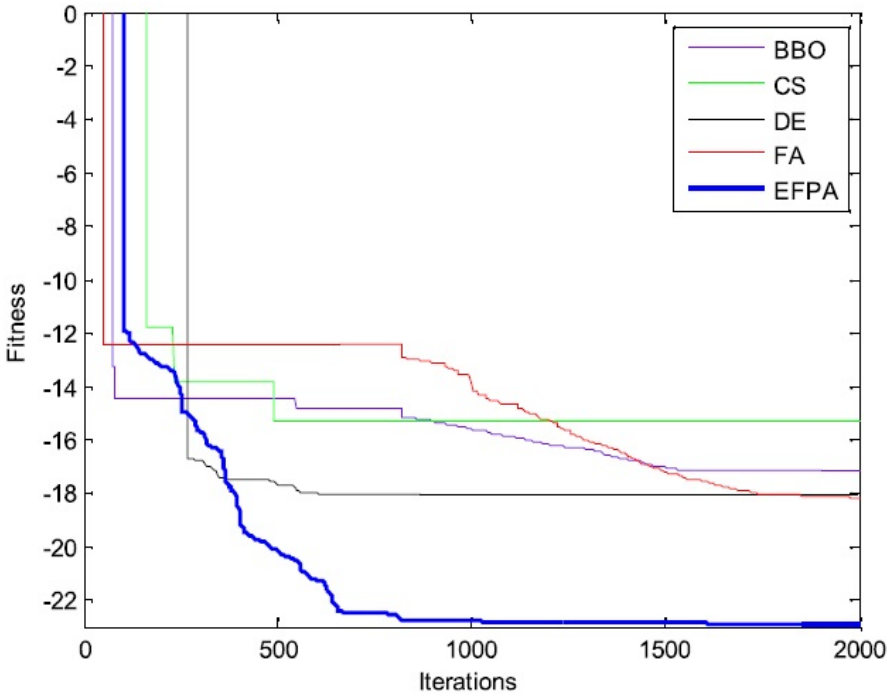


Fig. 8.7 EFPA algorithm: Convergence profile for 28-element non-uniform LAA

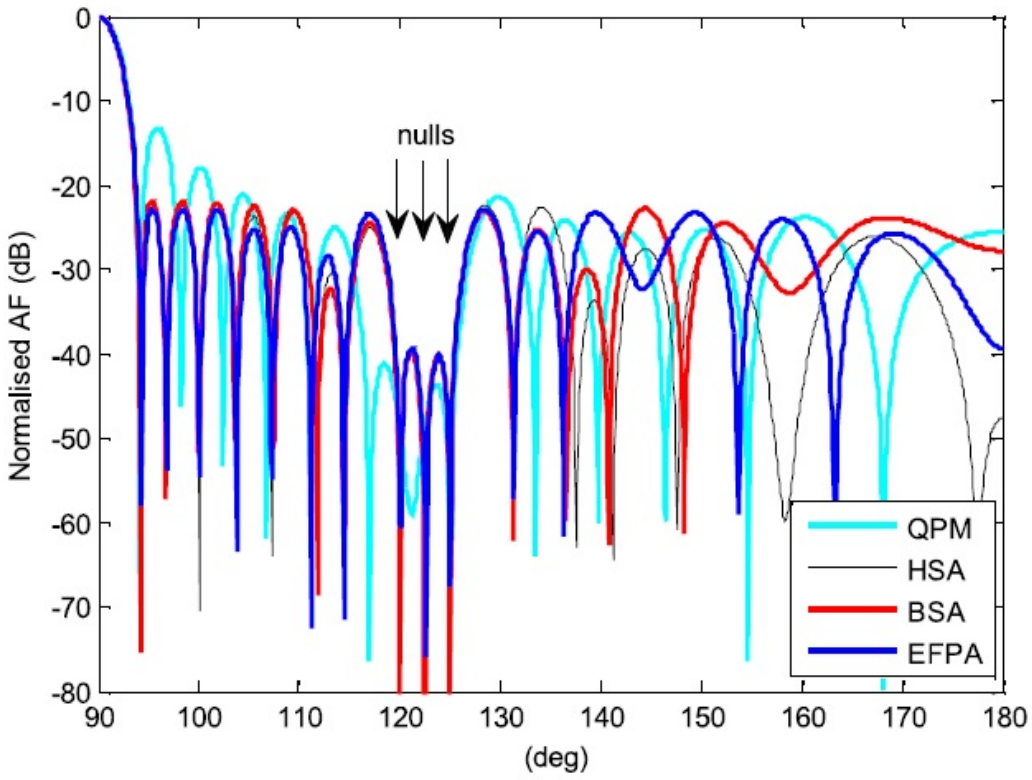


Fig. 8.8 EFPA algorithm: Radiation Pattern for 28-element non-uniform LAA

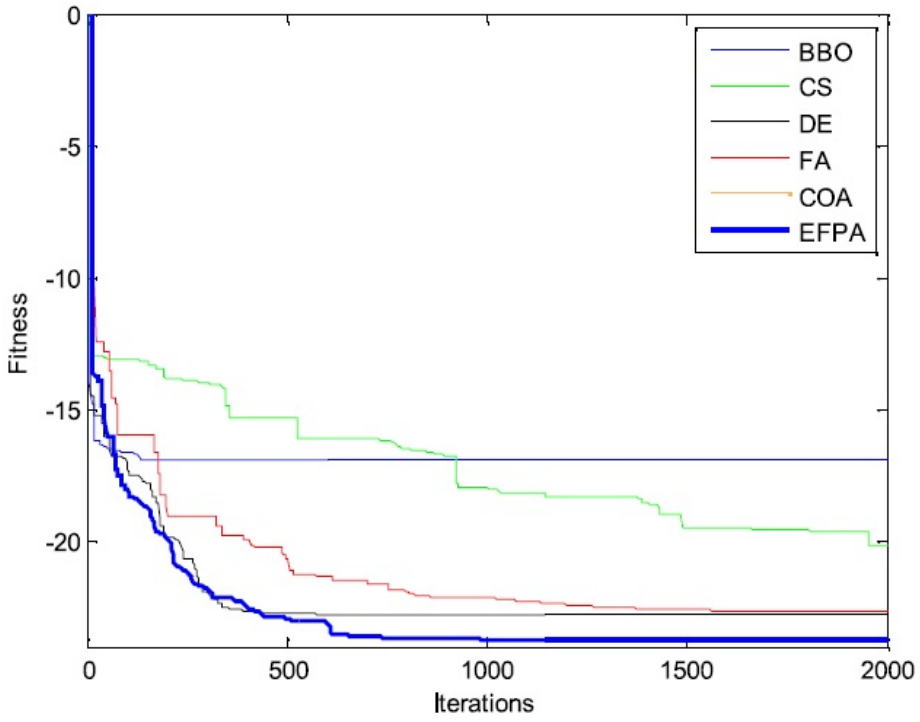


Fig. 8.9 EFPA algorithm: Convergence profile for 32-element non-uniform LAA

dos *et al.*, 2010) and FPA (Singh and Salgotra, 2018) respectively. The null values are also equal to or less than -60 dB. Fig. 8.8 shows the comparison of QPM (Khodier and Christodoulou, 2005), HSA (Guney and Onay, 2011), BSA (Guney and Durmus, 2015) and EFPA optimized array radiation plots.

32-element non-uniform LAA: The last design example illustrates the use of EFPA to optimize the element positions for minimization of SLL and null placement for 32-element LAA. The SLL is to be minimized in the region $\phi \in [93^0,180^0]$ and null is to be placed in the direction of 99^0. The required beamwidth BW_d is 7.1^0. The plot of convergence for EFPA is shown in Fig. 8.9. Also for the sake of comparison, the convergence plots of BBO (Singh and Rattan, 2014), CS (Singh and Rattan, 2014), DE (Singh and Rattan, 2014), FA (Singh and Rattan, 2014) and COA (Singh and Rattan, 2014) are shown in the same Fig. 8.9. The element positions for EFPA optimized array are shown in Table E.8 . Table E.9 compares the performance of different optimization methods for design of 32-element LAA. The SLL of proposed EFPA antenna array is -23.73 dB which is least in the Table E.9 . The reduction is considerable as compared to BSA

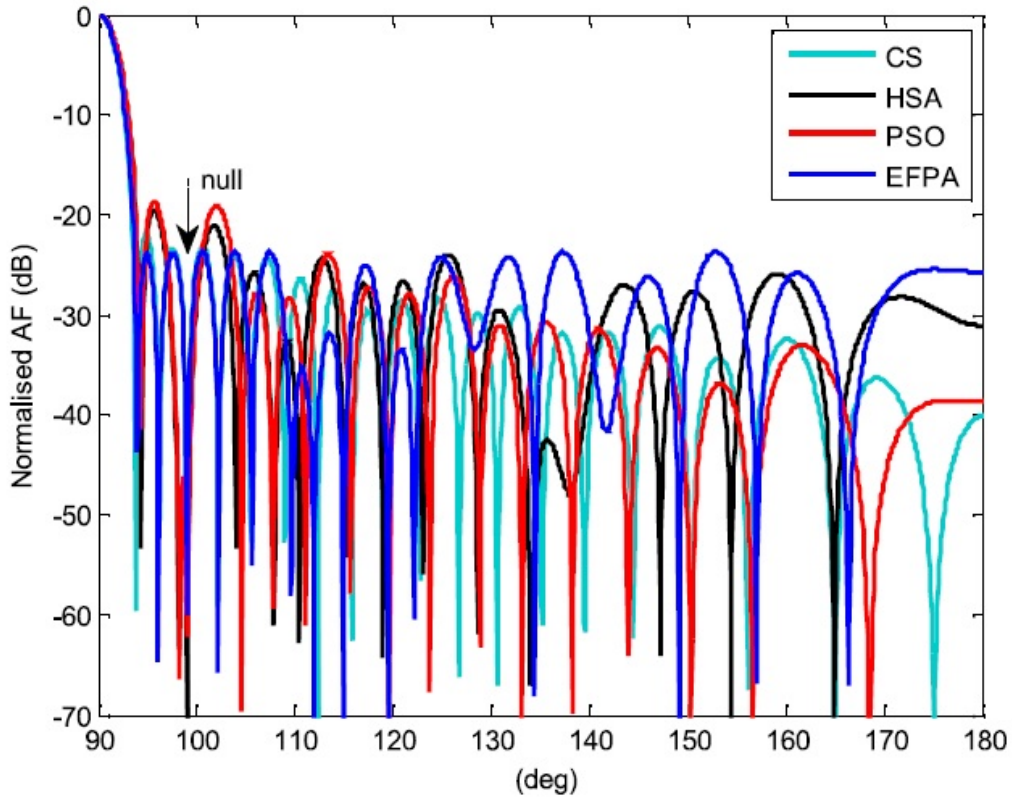


Fig. 8.10 EFPA algorithm: Radiation Pattern for 32-element non-uniform LAA

(Guney and Durmus, 2015), QPM (Khodier and Christodoulou, 2005), PSO (Khodier and Christodoulou, 2005), HSA (Guney and Onay, 2011), and GA (Rattan *et al.*, 2007) methods. The radiation patterns for this antenn array are shown in Fig. 8.10.

8.4 Concluding Remarks

FPA is a novel nature inspired algorithm which mimics the pollination of flowers. But it has some weaknesses like poor exploitation capability and slow convergence speed. So, to overcome these problems an improved version of FPA called EFPA has been proposed in this paper. Three modifications in the basic FPA have been proposed in the enhanced version. The modifications in EFPA help in providing better search capability and helps it to escape local minima. The proposed algorithm has been tested on seven benchmark functions. The results show that EFPA is able to find global optima of most of the benchmark functions. Moreover, the EFPA is utilized for pattern synthesis of non-uniform LAA. Four different antenna arrays of different sizes and different design constraints have been taken. When EFPA is used for SLL minimization only, it reaches the optimum result

in less number of iterations than FPA. For multi-objective design with SLL suppression, null and beamwidth control, EFPA obtains better performance than the results of well-known algorithms reported in literature. The results indicate the potential ability of EFPA to be used for optimization for other antenna designs.

CHAPTER 9

Bat Flower Pollination Algorithm for Antenna Applications

Preface

In this chapter, a novel parallel algorithm namely, bat flower pollination (BFP) is proposed for synthesis of unequally spaced LAA. The new method is a combination of BA and FPA. In BFP, both BA and FPA interact with each other to escape from local minima. The results of BFP for solving a set of thirteen benchmark functions demonstrate its superior performance as compared to variety of well-known algorithms available in literature. The novel proposed method is also used for the synthesis of unequally spaced LAA for single and multi-objective design. Simulation results show that BFP is able to provide better synthesis results than wide range of popular techniques.

Publication: **Rohit Salgotra**, and Urvinder Singh. "A novel bat flower pollination algorithm for synthesis of linear antenna arrays." Neural Computing and Applications 30, no. 7 (2018): 2269-2282.

9.1 Introduction

In the recent past, unequally spaced antenna also known as aperiodic array, have been synthesized using different methods (Balanis, 2016). In contrast to equally spaced or periodic array, aperiodic array lack periodicity which helps in obtaining lower side lobe level (SLL) using fewer elements. Moreover, the reduced SLL can be obtained using uniform element excitations which help in reducing intricacies related to the design of feed network and hence lowers the system cost. But the synthesis of unequally spaced array is complex and a big challenge for antenna engineers because of non-linear and non-convex relationship between array factor and positions of elements. Number of analytical methods, including Taylor and the Chebyshev method, have been employed for solving these highly nonlinear optimization problem. However, the synthesis problem is complex and cannot be solved with analytical methods. Therefore, global optimization methods are a good option to overcome these problems.

In the recent years, many evolutionary techniques have been used for the design of antenna arrays such as GA (Rattan *et al.*, 2007), DE (Lin *et al.*, 2010), CS (Khodier and Al-Aqeel, 2009), MSMO (Singh and Salgotra, 2016), PSO (Khodier and Christodoulou, 2005, Goudos *et al.*, 2010), BBO (Sharaqa and Dib, 2014), and others. Apart from application to antenna arrays, a large number of optimization algorithms have been proposed for other engineering optimization problems. These include BA , FPA etc. These algorithms are competitive but are some found to be inefficient in solving various problems. They suffer from certain weaknesses like slow convergence speed stagnation and premature convergence (Singh and Salgotra, 2016).

In this chapter, a new parallel algorithm based on BA and FPA namely bat flower pollination (BFP) has been proposed. The algorithm is mainly designed to overcome the problems of local optima stagnation among pre-existing BA (Yang, 2010b). BFP algorithm basically enhances the working capability of basic BA with added advantage of finding better optimal solution provided by FPA. Though BFP algorithm enhances BA but sometimes it helps FPA in getting out of local optima. Hence, on a whole both algorithms help each other in getting out of local optima and to achieve better convergence profiles. To check the performance of BFP algorithm, it has been applied to standard

benchmark problems and compared with well-known algorithms for establishing its competitiveness on unconstrained optimization problems. The algorithm has also been tested statistically to prove its superiority over other state-of-the-art algorithms. Further as an application to real-world problem, BFP is applied to classical electromagnetic LAA design problem. Three different unequally spaced linear arrays have been optimized using the novel method and a comprehensive comparison of BFP results has been done with the results available in literature.

The rest of the chapter is organized as: Section 2 deals with the proposed BFP algorithm and section 3 is concerned with the result and discussion part. Finally, in section 4 the chapter is concluded.

9.2 The Proposed Approach

The BFP algorithm proposed in this chapter is a parallel version of BA and FPA. Both these algorithms are highly competitive and have proved their worthiness for a number of optimization problems. Since the two algorithms act as global problem solvers, so proposing an enhanced integrated version of both will definitely provide better results. The basic idea is to find a current best solution by adjusting frequencies, updating velocities and locations for bats and for flower pollinators, switch probability is used. After finding these solutions, both are compared and the best one is retained and this procedure is continued unless termination criteria is satisfied. The solution at the final stage is the best fit solution of the problem under consideration. The proposed BFP algorithm is divided into three phases:

The first phase of BFP algorithm is the initialization phase. In this phase, bats and flower pollinators are initialized with N as the combined population of both. The initial random population is generated by using equation **??** and fitness of objective function of the problem under consideration is calculated. The current best solution based on fitness is evaluated and saved as the initial best solution. This initial best solution is assigned to both bats (B_i) and pollinators (FP_i). Note that the initial random population for both bats and pollinators is generated using the same equation.

In the second phase, two new solutions are generated one from the bats and other from

flower pollinators. One solution x_i^t is generated using equation **??**. Finally, based on the pulse rate and loudness (A_i^t), the new best fit solution (B_i^{best}) for the problem under consideration is evaluated. Other solution (x_i^{t+1}) is generated by flower pollinators using equations Eq. (2.6) and Eq. (2.8) based on a switch probability. This solution is further used to evaluate optimization problem under test and the fittest solution (FP_i^{best}) among all flower pollinators is saved. So at the end of this stage there are two solutions, one from the bats (B_i^{best}) and others from the flower pollinators (FP_i^{best}).

In the final phase, the two solutions B_i^{best} and FP_i^{best} are compared and best among them is the final best solution. The solutions thus generated at final stage is again set as initial best B_i and FP_i for both bats and flower pollinators respectively. The same procedure is followed until either global best fit individual is found or termination criteria is fulfilled. Here it should be noted that since bats and flower pollinators are searching for best fit solutions in parallel. So when solution produced by bats gets stuck in local optimum and is not able to provide the global optimum, the flower pollinators helps it in getting out of the local search loop to achieve global optimum and vice versa. This property provides the BFP algorithm with better chances of achieving global best solution.

The main focus is to improve the performance of BA. BA is a recently introduced swarm intelligent algorithm based on the echolocation of bats. This algorithm is able to obtain good results for lower dimension problems but while dealing with higher dimension problems it becomes problematic because of its premature convergence rate (Yang, 2010b). This makes the algorithm to stuck in local optima. So there is a need of another better solution to make the algorithm move towards global optimum. This move is provided by FPA and if sometime the FPA algorithm stuck, it is taken towards global optimum by BA. Hence analytically both bats and flower pollinators are working in collaboration to achieve global optimum solution. The flow code for BFP algorithm is shown in Fig. 2.

9.3 Result and Discussion

The performance of proposed BFP algorithm is tested by applying it to standard benchmark functions and LAA design problem. BFP algorithm is firstly used to find optimum solution of the benchmarks and compared with firefly algorithm (FA), FPA and BA algorithms to prove its superiority. Then, secondly it is applied for synthesis of LAA with

reduced side lobe level and again compared with state-of-the-art algorithms. The simulations are performed on Windows 7, Intel core i5 processor with MATLAB version 7.10.0 (R2010a).

BFP is used to optimize unconstrained benchmark problems (Suganthan *et al.*, 2005). Table A.1 shows test functions general equations, their dimension with particular search ranges and final optimal solution. For the purpose of comparison FA, BA and FPA are also applied to the same benchmark functions. The maximum number of iterations are set to 1000 with population size of 20 and each algorithm is run 20 times. Finally, best results are drawn for test functions to show the better performance of BFP algorithm over others.

These stochastic algorithms are further tested statistically. The Wilcoxon non-parametric rank-sum test (Derrac *et al.*, 2011) is conducted to detect the static significance of BFP algorithm as compared to others. In this test two pairs of populations are compared and difference among them is analyzed. This tests ultimately gives a p-value to determine the static significance level of two algorithms under comparison and should be less than 0.05 for a result to be significant.

The parameter settings for all the algorithms are shown in Table B.1. The experimental results are shown in Table C.34. When compared to other algorithms, BFP algorithm is found to perform better in finding optimal solution for most of the test functions. For f_5, f_6 (BFP is competitive for f_6), f_7 and f_9 FPA performs better than proposed algorithm but for rest of the algorithms the results for BFP are much better. For f_{10} and f_{13} BFP algorithm gives an exact zero standard deviation. FPA is found to be better than proposed algorithm for f_3 while BA is found to be the poorest of all. It can be seen here that though standard deviation for most of the functions is not good enough but priority of work is finding global optimum solution and it has been achieved by BFP algorithm for most of the functions proving its superiority. Also, the proposed algorithm is found to perform much better than BA hence further adding to its competitiveness. The statistical performance in terms of p-value for BFP is shown in Table D.4. It can be seen that BFP is found to perform better for almost every function except f_3, f_5, f_6, f_7 and f_9. For f_{13} BFP and FPA have the same statistical significance level. Overall, the BFP algorithm can be said to

have achieved significant p-value for it to be competitive. This happens due to the benefits of higher exploitation assisting BFP to converge rapidly towards the optimum solution. Thus from the above comparison, it has been found that BFP algorithm performs better than FA, BA and FPA algorithm.

9.4 Real World Applications: Antenna Array Synthesis

The aim of non-uniform antenna synthesis is to obtain set of optimum element positons which yield a radiation pattern having suppressed SLL, desired beamwidth with tolerance of $\pm 1^0$ and null control in specified directions. The reduced SLL and narrow beamwidth helps to increase the directivity of antenna and also helps in reducing electromagnetic pollution by radiating in the desired direction. Nulls are imposed in radiation pattern so as to avoid jamming and inference in certain directions. To achieve the desired antenna array, the fitness function to be minimized is given as:

$$fit = SLL + \alpha * max\left\{0, |BW - BW_d| - 1\right\} + \alpha * \left\{\sum_{k=1}^{K} max\left\{0,\ AF_{dB}\left(\phi \mid \mathrm{k}\right)\right\} - \mathrm{Nu}_{dB}\right\} \tag{9.1}$$

where SLL is the peak side lobe level calculated in the region excluding mainlobe, BW, BW_d are the calculated and desired beamwidth, respectively, K is the number of the required nulls directions, Nu_{dB} is the desired null level in dB,ϕk is the direction of the k^{th} null and α is a very large number. The fitness function employs penalty method. Any antenna design which does not meet the constraint either of in terms of beamwidth or null values is given a very high value.

In order to evaluate the strength and effectiveness of BFP, it is imperative to apply it for real-world problems and compare its performance with well-known optimizers like GA, PSO, CS, BBO, BSA and so on. So, in this work, BFP is used for classical electromagnetic problem relating to non-uniform LAA design. The same LAA has been synthesized extensively in the past using state of the art algorithms like PSO (Khodier and Al-Aqeel, 2009), GA (Rattan *et al.*, 2007), CLPSO (Khodier and Christodoulou, 2005), CS (Khodier and Al-Aqeel, 2009), HSA (Guney and Onay, 2011), ACO (Raja *et al.*, 2015), cuckoo optimization algorithm (COA (Singh and Rattan, 2014), back scattering algorithm (BSA)

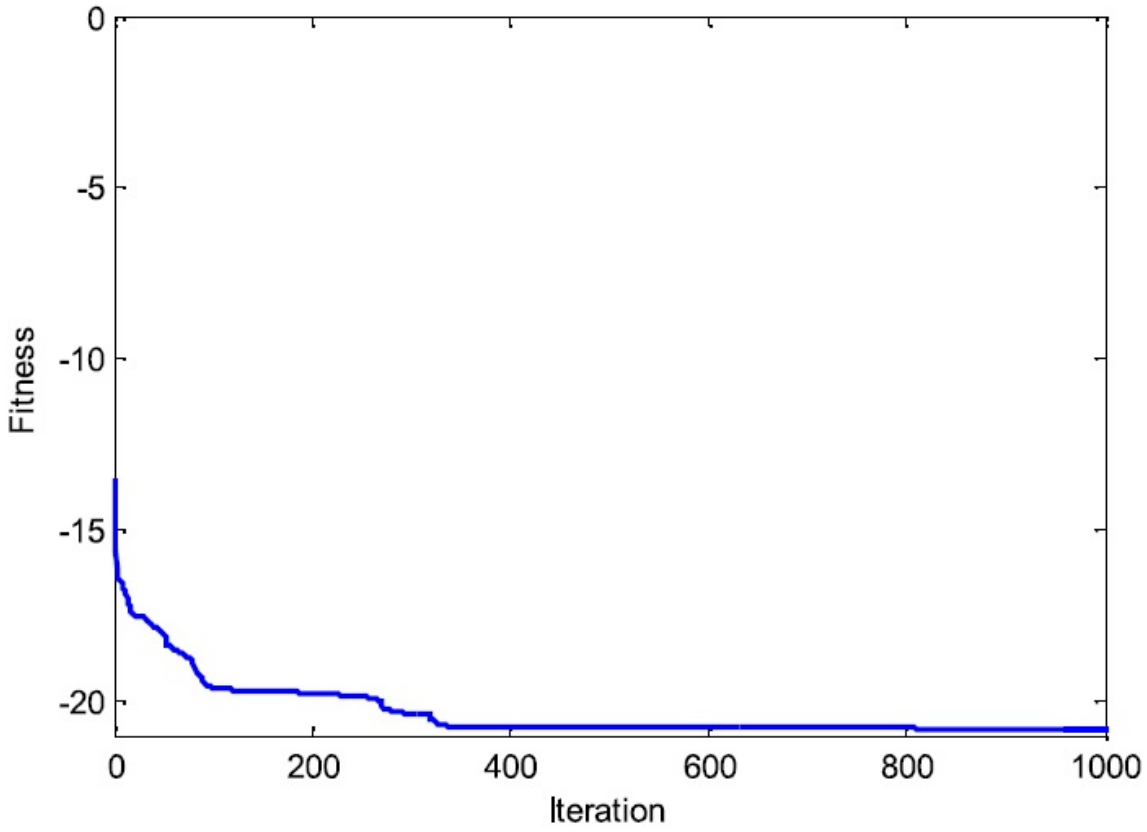


Fig. 9.1 BFP algorithm: Convergence profile for 12-element non-uniform LAA

(Guney and Durmus, 2015) and MSMO (Singh and Salgotra, 2016). So it will be easy to make comparison of BFP results and measure its performance against with those available in the literature.

In this section, BFP is used to optimize three different unequally spaced arrays. For each example of antenna optimization, 20 independent runs of BFP have been done with the population size of 20. For BA, both loudness factor and pulse rate is set to be equal to 0.5.

12-element LAA: In the first example for non-uniform linear arrays, a 12-element LAA is taken for BFP optimization. The aim is to reduce SLL only in the region $\emptyset \in [98^0, 180^0]$ by optimizing the element positions. So the value of α is taken as 0 in the fitness Eq. (9.1). For this optimization, stopping criteria is number of iterations and is set equal to 1000. The BFP optimized positions for the 12-element LAA are given in Table E.10. The convergence graph for BFP optimization is shown in Fig. 9.1. The

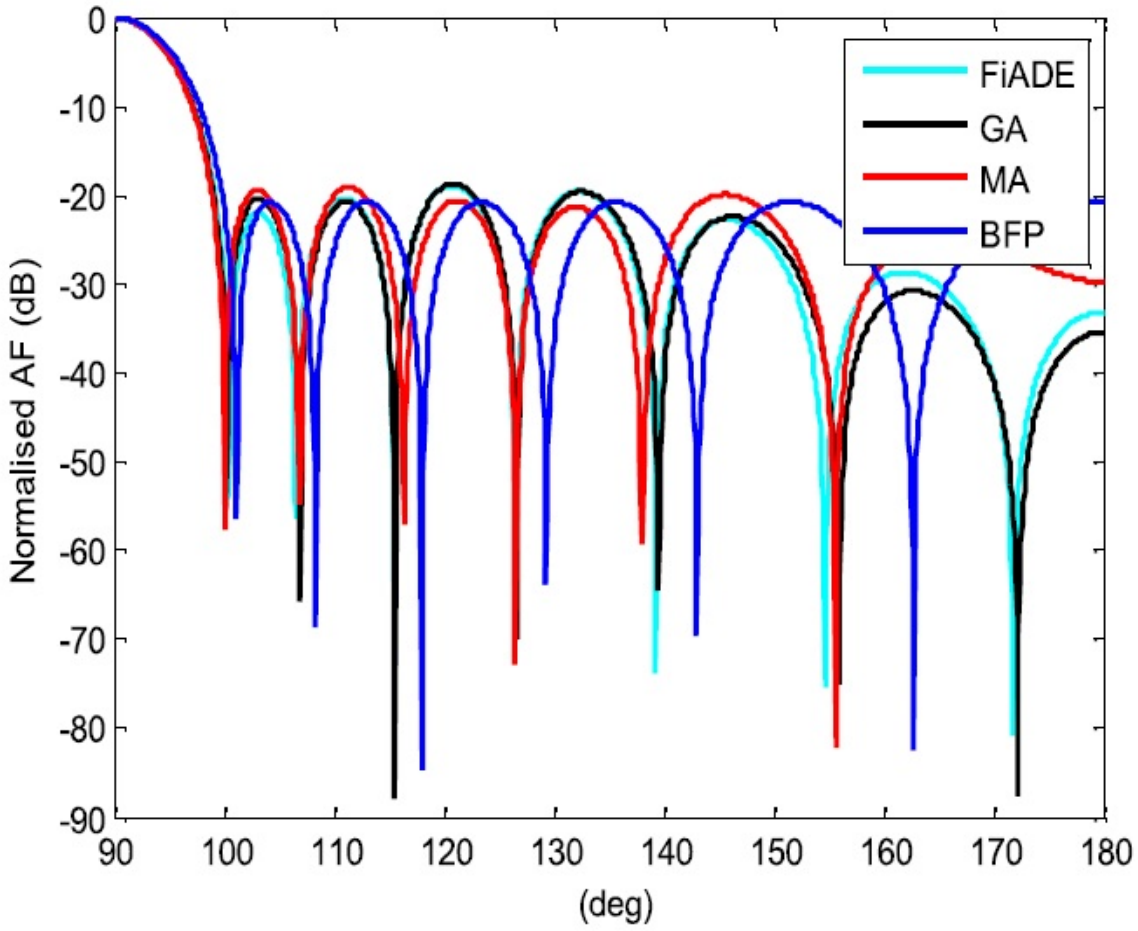


Fig. 9.2 BFP algorithm: Radiation Pattern for 12-element non-uniform LAA

performance of BFP in terms of SLL attained is compared with that of fitness adaptive differential evolution (FADE) (Chowdhury *et al.*, 2010), TS (Cengiz and Tokat, 2008), memetic algorithm (MA) (Cengiz and Tokat, 2008), GA (Cengiz and Tokat, 2008) and PSO (Chowdhury *et al.*, 2010). The maximum SLL achieved by BFP is -20.77 dB which is least in Table E.11. For the same problem, the SLL obtained by FADE (Chowdhury *et al.*, 2010), TS (Cengiz and Tokat, 2008), MA (Cengiz and Tokat, 2008), GA (Cengiz and Tokat, 2008) and PSO (Chowdhury *et al.*, 2010) is 18.96 dB, -18.40 dB, -19.10 dB, -18.79 dB and -17.90 dB. The optimized radiation pattern for BFP optimized LAA is given in Fig. 9.2. Also in the same figure, for the purpose of comparison, radiation patterns of GA (Cengiz and Tokat, 2008), PSO (Chowdhury *et al.*, 2010), FADE (Chowdhury *et al.*, 2010) are also plotted.

28-element LAA: The second example is optimization of 28-element array for reducing

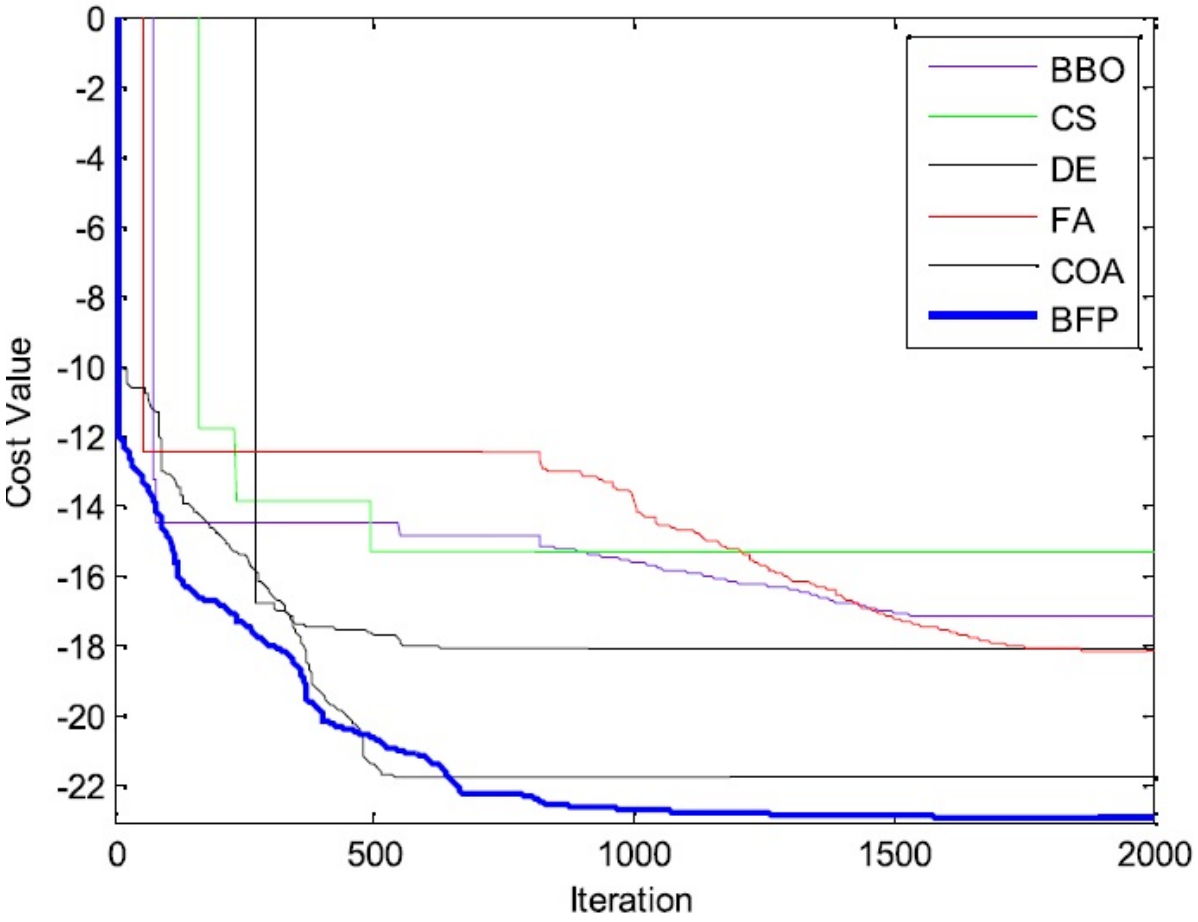


Fig. 9.3 BFP algorithm: Convergence profile for 28-element non-uniform LAA

SLL, controlling beamwidth and placing nulls in desired nulls. The aim is to find a set of optimum element positions of 28-element antenna that will have reduced SLL in the region $\emptyset \in [100^0, 180^0]$, beamwidth equal to $8.35^0 \pm 12\%$ and nulls deeper than -60 dB in the directions 120^0, 122.5^0 and 125^0. This problem is complex than the previous example as it entails achieving multiple objectives. To achieve the required design, the value of α in the fitness function Eq. (9.1) is taken to be very large and equal to 10^6. The BW_d is set to 8.35^0 and Nu_{dB} equal to -60 dB in the fitness function. For this problem, the BFP algorithm is run for 2000 iterations. The convergence graph of BFP algorithm for this optimization is given in Fig. 9.3 and contrasted with the convergence of COA (Singh and Rattan, 2014), FA (Singh and Rattan, 2014), BBO (Singh and Rattan, 2014), DE (Singh and Rattan, 2014). The optimized array geometry is given in Table E.12. The optimized radiation pattern is shown in Fig. 9.4. For sake of comparison, radiation plots of quadratic programming method (QPM) (Khodier and Al-Aqeel, 2009), HSA (Guney

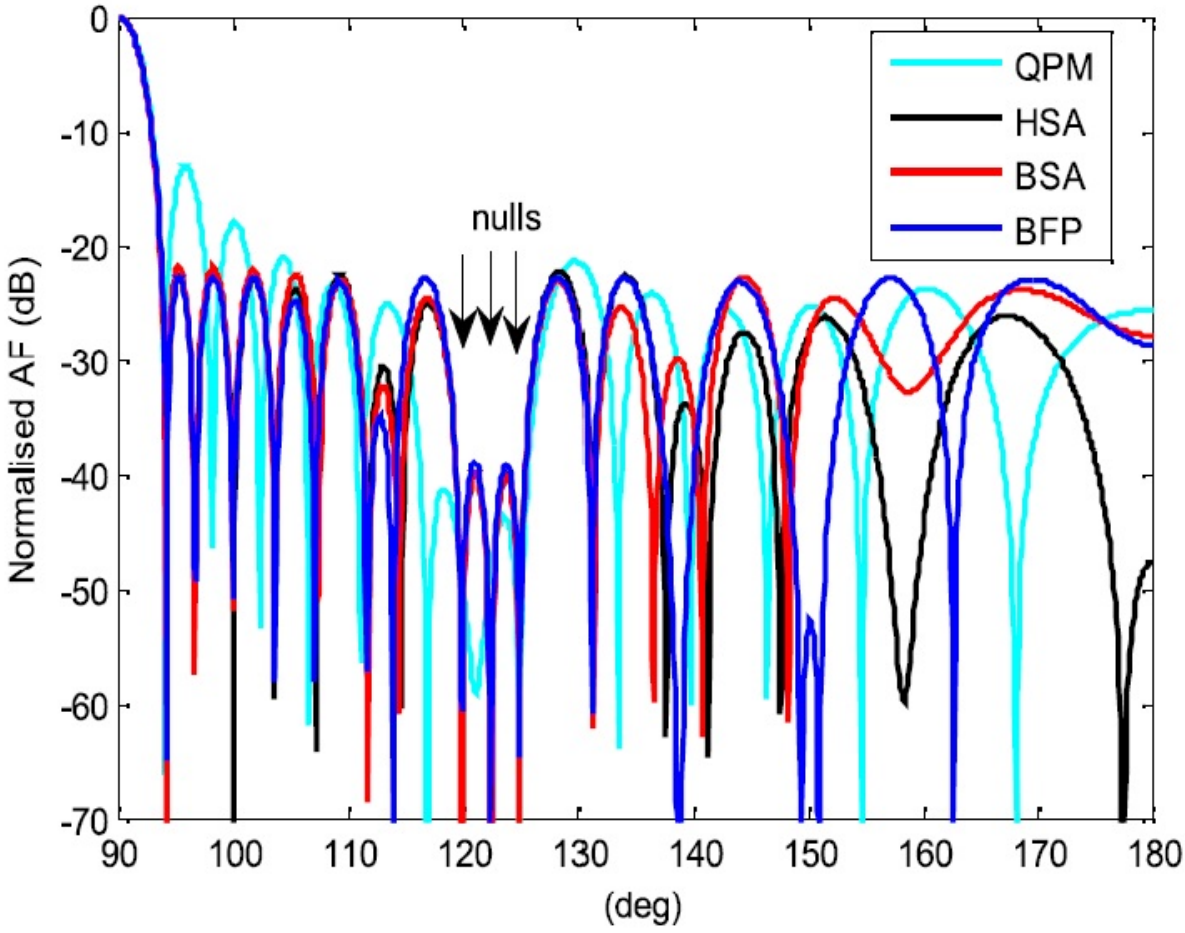


Fig. 9.4 BFP algorithm: Radiation Pattern for 28-element non-uniform LAA

and Onay, 2011), and BSA (Guney and Durmus, 2015) are also shown in the same figure. The comparison of performance of different algorithms is done in Table E.13. Clearly, it is evident from the comparison that the SLL attained by BFP is least in comparison to other methods like GA (Rattan *et al.*, 2007), BSA (Guney and Durmus, 2015), HSA (Guney and Onay, 2011), PSO (Khodier and Al-Aqeel, 2009), CLPSO (Khodier and Christodoulou, 2005), ACO (Raja *et al.*, 2015), COA (Singh and Rattan, 2014), QPM (Khodier and Al-Aqeel, 2009) listed in Table E.13. The SLL of BFP array is -22.68 dB and the null values in directions 120^0, 122.5^0 and 125^0 are -60.72 dB, -72.85 dB and -64.55 dB.

32-element LAA: In the final example, to compare the performance of BFP with the other popular optimization methods, a well-known problem of optimizing 32-element LAA for SLL reduction, beamwidth control and null placement is taken up. The null is to be placed in the direction of 99^0. The required beamwidth is 7.1^0 with tolerance

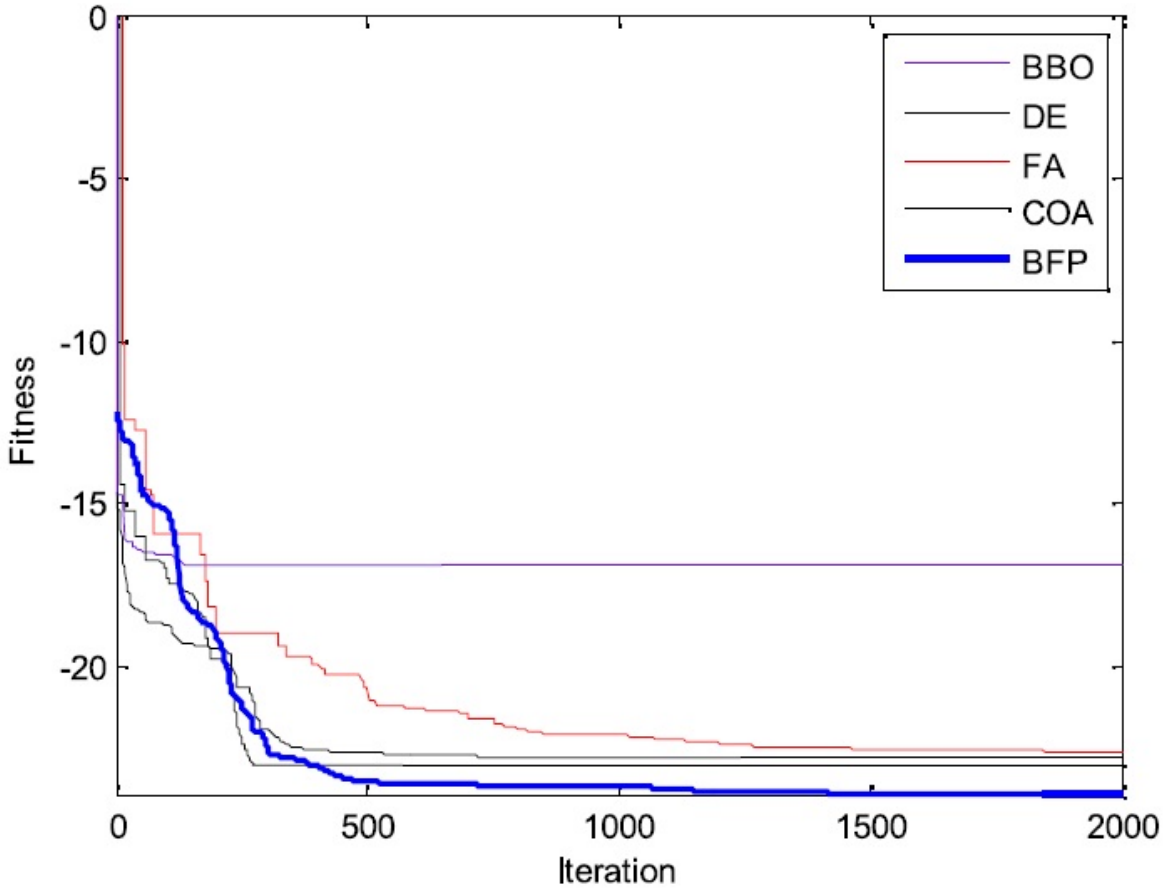


Fig. 9.5 BFP algorithm: Convergence profile for 32-element non-uniform LAA

of set ±14%. This problem has been optimized by popular optimization methods like PSO (Khodier and Al-Aqeel, 2009), CLPSO (Khodier and Christodoulou, 2005), GA (Rattan *et al.*, 2007), HSA (Guney and Onay, 2011), BSA (Guney and Durmus, 2015), CS (Khodier and Al-Aqeel, 2009), COA (Singh and Rattan, 2014), inheritance learning PSO (ILPSO) (Goudos *et al.*, 2010), and chaotic PSO (CPSO) (Wang *et al.*, 2011) in the recent past. The convergence plot of BFP along with convergence graphs of COA (Singh and Rattan, 2014), FA (Singh and Rattan, 2014), BBO (Singh and Rattan, 2014), DE (Singh and Rattan, 2014) is shown in Fig. 9.5. The comparison of convergence plots clearly shows the faster convergence of BFP as compared to other methods. The optimum array geometry achieved after BFP optimization are given in Table E.14. Table E.14 presents the comparison of results of different algorithms that have been used to optimize the same antenna. Clearly, BFP has achieved best results in terms of SLL reduction and also null value lesser than -60 dB as desired. The maximum SLL of BFP antenna is -23.99 dB which

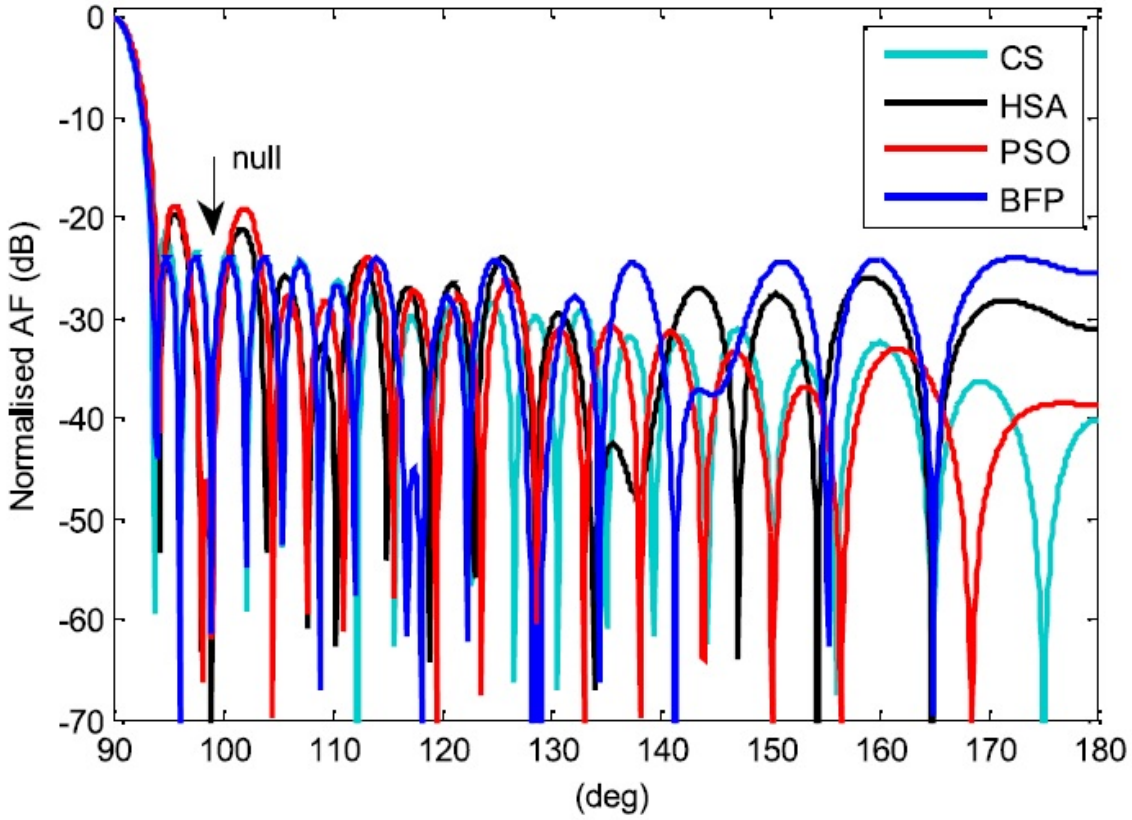


Fig. 9.6 BFP algorithm: Radiation Pattern for 32-element non-uniform LAA

is least when compared with BSA (Guney and Durmus, 2015), PSO (Khodier and Al-Aqeel, 2009), CPSO (Wang *et al.*, 2011), ILPSO (Goudos *et al.*, 2010), CS (Khodier and Al-Aqeel, 2009), GA (Rattan *et al.*, 2007), CLPSO (Khodier and Christodoulou, 2005) and COA (Singh and Rattan, 2014) optimized antennas. The array patterns of BFP, HSA (Guney and Onay, 2011), PSO (Khodier and Al-Aqeel, 2009), and CS (Khodier and Al-Aqeel, 2009) optimized antennas are shown in Fig. 9.6.

9.5 Concluding Remarks

This chapter proposed and employed a new parallel algorithm BFP for the pattern synthesis of LAA. BFP is combination of BA and FPA in which the new solution is influenced by global solution of the two algorithms. This helps in escaping local minima as if either algorithm is stuck in local minima the other will pull it off with the help of global best solution. To substantiate the effectiveness of the new algorithm, it has been used to find solutions for thirteen benchmark functions. The results show that the BFP works more

effectively and adaptively as compared to other state of the art algorithms like FA, BA, and FPA. Moreover, as a real-world application, it has been used to find optimum element positions of three unequally spaced LAA. Numerical results show that BFP gives best performance for single objective SLL reduction for the first case in contrast to other popular optimization methods available in literature. For multi-objective synthesis of LAA involving SLL suppression, null control and beamwidth, BFP again gives superior performance in comparison to results reported in the recent past. The results of BFP demonstrate its ability to solve difficult antenna design problems and it has potential to be a good optimizer for solving other real-world problems.

Part II : New Swarm Intelligent Algorithm

CHAPTER 10

Naked mole-rat Algorithm

Preface

This chapter proposes a new swarm intelligent nature-inspired algorithm called Naked Mole Rat (NMR) algorithm. This NMR algorithm mimics the mating patterns of NMRs present in nature. Two types of NMRs called workers and breeders are found to depict these patterns. Workers work continuously in the endeavour to become breeders while breeders compete among themselves to mate with the queen. Those breeders who become sterile are pushed back to the worker's group and the fittest worker become a new breeder. This phenomenon has been adapted to develop the NMR algorithm. The experimental results and statistical analysis prove that NMR algorithm is very competitive as compared to other algorithms.

Publication: **Rohit Salgotra** and Urvinder Singh "The naked mole-rat algorithm" Neural Computing and Applications 31, no. 12 (2019):8837-8857.

10.1 Introduction

Sexually reproducing animals are evolving mainly due to inbreeding, this is due to the fact that sexual recombination helps to increase the variability of offspring's (Jeanne, 1986). This helps to increase efficiency, remove deleterious alleles from different genes and provide resistance from parasites. This also helps species in adapting to local conditions. There is a type of inbreeding called close inbreeding which occurs when alternative males are not available. This type of phenomenon is very rare in natural species. Fallow deer is one such animal showing a father-daughter mating and has been confirmed by the genetic model. But from the behavioral and paternity analysis, there is no conformity of such behavior (Thornhill, 1993). The only other species showing inbreeding patterns is the NMR.

NMRs occur in semi-arid eastern Africa and are fossorial, showing large subterranean tunnel systems while foraging for tubers Fig. 10.1c. They have evolved and adapted to this hostile habitat and are found to be present on this earth from the Miocene era (24 million years ago) (Thornhill, 1993). NMRs were firstly described in African mammals by Eduard Ruppell in 1842. He named them 'different headed' (*Heterocephalus*) and smooth-skinned/hairless (*grabber*) because of their odd shaped skull and absence of flurry pelage. They belong to the *kingdom:* Animalia, *phylum:* chordate, *order:* Rodentia, *family:* Bathergidae and was the only species classified in *genus* Heterocephalus until the 1980s. But till date, 15 species have been included in this genus. NMRs as discussed, are close inbreeding animals and they do so in a belief that new colonies can be formed through fission, helping in sealing the tunnels (Buffenstein *et al.*, 1994). By sealing the tunnels, the danger due to predation, extreme temperatures, and starvation, are minimized. The subsequent subsections provide a brief overview of NMR behavior.

10.1.1 Mating Patterns of NMR

They are eusocial animals living in a group of up to 295 individuals. A detailed discussion about eusociality in NMR is given in the next subsection. The average population of a single colony is through close to 75 (Brett, 1991). The colony is headed by a single female, that is, there is only a single breeding female called queen. This queen breeds with a limited number of males, while the remaining members of the colony perform necessary tasks like construction, provisioning, maintenance, and defense. Workers are sterile and

(a) Breeding mates

(b) Breeders and workers

(c) Burrowing patterns

Fig. 10.1 Naked mole-rats

become sexually active only when separated from the colony. They separate from the colony only to act in the assistance of offspring produced by the queen (Buffenstein *et al.*, 1994). So, based upon the mating patterns, we can define two types of NMRs that are breeders (assist in mating) and workers (helping to perform other tasks). Breeders, live close to the queen and hence are at no risk or at lower risks of predation while workers are not. This is the reason why breeders live up to 17 years, 4 times longer than their nonbreeding counterparts or workers (Brett, 1991). So, is there only a single female in the colony that grows to a queen, if so, how will the new queen be selected if present one die? The answer is that there are a large number of females in the colony and they form the worker class. Any female who is older than 6 months is capable of becoming a queen. This further adds a question mark. Where will the present queen go, if any female can be queen? All the females in the group may fight until death to become the queen and establish their dominance. When a female becomes the queen, it becomes reproductively active and starts producing estrogen-dependent 'pubescent' growth surge resulting in an increase in their body lengths. The other females in the colony remain anovulatory with a lower level of sex steroids. Males in the worker groups exhibit the same patterns and have low testosterone, low luteinizing hormone, abnormal sperms and infertile (Brett, 1991). Hence, it can be said that only the best males among the worker class form the breeder class and one among the best females get the status of a queen. The breeding mates, breeders and workers and burrowing patterns of NMRs are shown in Fig. 10.1.

10.1.2 Eusociality in NMR

Eusociality is defined as the colonial lifestyle of a species, following a strict division of labor and having a single female for breeding. This concept was laid down by Richard Alexander

in the mid 1970s. He identified that such systems are found in the subterranean population living in soil and relying on others to dig burrows and forage food for underground tubers. This theory was explored by Jarvis, who found that like termites, ants and wasps carry out tasks in a single community of about 295 individuals, with 1 to 3 breeding males and only a single breeding female (Alexander *et al.*, 1991). So, it is evident from the study that NMRs are eusocial animals following the division of labor to perform various tasks and having only a single female called queen for mating. Division of labor is followed only by the workers whereas breeders and queen are only meant for reproduction. This eusocial lifestyle of NMRs improve their inclusive fitness and ultimately contribute to the prolonged lifespan.

10.1.3 NMR and SI

Self-organization: It means a species should produce a global level response from low-level interactions without any central authority. In NMR, the workers act as a low-level component and they produce high-level breeder. These high-level breeders further choose among themselves the best breeder, who mates with the queen. So, a hierarchical pattern is followed without any central authority. Hence, we can say that NMRs follow self-organization. This can be further validated by proving the four steps of self-organization as:

- Positive feedback: In NMR, the queen is assisted by a few sets of breeding males and only they contribute to the reproduction process. These breeding males are selected from the pool of worker population. This process from workers to breeders and ultimately to queen follows positive feedback. This helps in providing diversity and accelerating the system to move to a new stable state.

- Negative feedback: This part focuses on compensation through positive feedback. The breeder who is not able to cooperate or whose fitness is not up to the mark for mating, either dies or is sent back to the worker's pool, hence forming a negative feedback path from breeders to workers. This process stabilizes the system and hence maintain a proper level of breeder population.

- Fluctuations: It means randomness in swarms. In NMR, randomness is in finding the best possible breeders and queen among the workers. This process helps in

getting rid of the stagnation problem.

- Multiple Interactions: It is a learning process either by interactions or experience. Here, the worker pool of NMRs interact with each other and cooperate or assist each other in completing various tasks like maintenance, construction, defense and provisioning. The breeders also follow this pattern by cooperating with each other during mating, as only a single male with a mate with the queen at a particular time and hence enhance the combined intelligence of all NMRs.

Division of labor: It has been already discussed that NMRs are eusocial animals and every eusocial animal follows the strict division of labor. Here the workers basically do all sorts of work and breeders do the mating.

From the above discussion, it is evident that NMRs are SI animals. In the next section, a new SI based algorithm called NMRA has been proposed.

10.2 The Proposed Approach

The social behavior of NMR has inspired authors to propose a stochastic optimization algorithm. The algorithm mimics the mating patterns in NMR and has the following key features:

- NMRs are eusocial animals living in a group of 295 members with an average number of members to be 70-80. For experimental analysis, the present work uses a group of 50 NMRs.

- A female queen leads the group and divides the population among breeders and workers. The breeders are the best performing NMR among the working group and are meant for mating only whereas workers perform other tasks.

- The workers perform necessary tasks and best among them are replaced by breeders. In simple words, high performing workers, become breeders and low performing breeders are again sent to the worker's pool.

- The best breeder among the breeder pool mates with the queen.

The above four rules have been idealized to propose a naked mole rat algorithm (NMR).

The algorithm is divided into three phases. In the first phase, the population of NMRs is initialized, second is the worker phase and third is the breeder phase. The breeder phase is selected based upon the breeding probability. The details of the algorithm are explained as:

Initialization: Initially, it generates a uniformly distributed random population of n NMR where each NMR in the range [1, 2 ... n] is a D-dimensional vector. Here D represents the number of variables or parameters to be tested in the problem. Each NMR is initialized as

$$NMR_{i,j} = NMR_{min,j} + U(0,1) \times (NMR_{min,j} - NMR_{max,j}) \tag{10.1}$$

where $i \in [1,\ 2,\ \ldots n]$, $j \in [1,\ 2,\ \ldots D]$, $NMR_{i,j}$ is the i^{th} solution in the j^{th} dimension, $NMR_{min,j}, NMR_{max,j}$ are the lower and upper bounds of the problem function respectively and $U(0,1)$ is uniformly distributed random number. After initialization, the objective function is evaluated and its fitness is calculated. Based upon the fitness, B breeders and W workers are identified and overall initial best solution d is calculated. After initialization, the population of NMR is subjected to repeated cycles or iterations of the search process of worker and breeder phase.

Worker phase: In this phase, the workers tend to improve their fitness so that they get a chance to become a breeder and eventually mate with the queen. So here, the new solution of worker NMR is generated based upon its own experience and local information. Here the fitness of new NMR is evaluated and if the new mating fitness is better, the old solution is discarded and the new solution is memorized. Otherwise, the older solution is retained. After all the worker rats complete the search process, the final fitness of all of them is remembered. In order to produce a new solution from the old one, the NMR uses the following equation:

$$w_i^{t+1} = w_i^t + \ \lambda \left(w_j^t - w_k^t \right) \tag{10.2}$$

where w_i^t corresponds to the i^{th} worker in the t^{th} iteration, w_i^{t+1} is the new solution or worker, λ is the mating factor and w_j^t and w_k^t are two random solutions chosen from the worker's pool. The value of λ is obtained from a uniform distribution in the range of [0, 1].

Breeder phase: The breeder NMR also update themselves in order to be selected for mating and also to stay as a breeder. The breeder NMRs are updated based upon a breeding probability (bp) with respect to the overall best d. This bp is a random number in the range of [0, 1]. Some of the breeders may not be able to update their fitness and hence may be pushed back to the workers' category. The breeders modify their positions according to the equation given below:

$$b_i^{t+1} = (1 - \lambda)\, b_i^t + \lambda(d - b_i^t) \tag{10.3}$$

Here b_i^t corresponds to the breeder i in the iteration t, λ factor controls the mating frequency of breeders and helps in identifying a new breeder b_i^{t+1} in the next iteration. To start with, the value of bp has been set to 0.5 as the initial value.

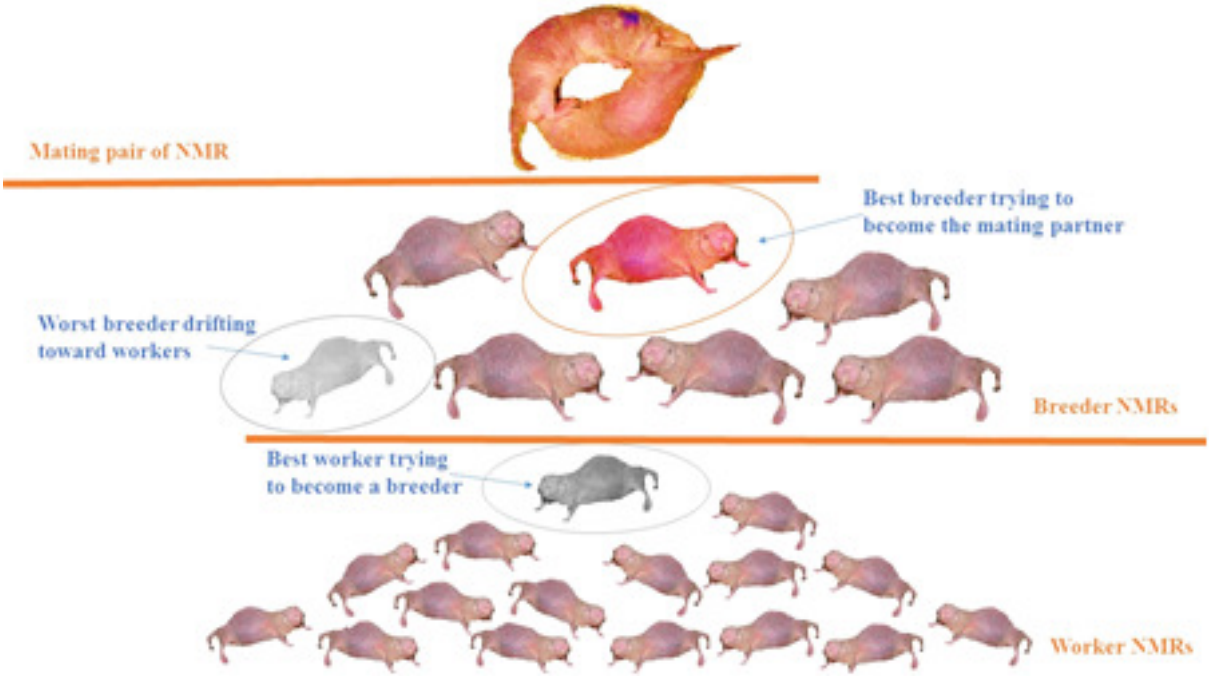


Fig. 10.2 Worker-breeder relationship and their mating pair

For simplicity, we have assumed that there is only a single queen and best among the breeder mates with the queen. So here we find only the best breeding male who will breed with the female. The algorithm works by differentiating or identifying the breeders and workers among the pool of NMRs. After an initial evaluation, the best breeder, and the best worker is selected. The fitness of workers is updated so that their fitness improves and they may get a chance to become breeders. On the other hand, breeders also

update their fitness based upon breeding probability so that they remain breeders. The breeder which becomes sterile will be pushed into the workers' category. The best breeder among the population serves as the potential solution to the problem under test. The above-mentioned worker-breeder relationship and their mating with the queen are best elaborated in Fig. 10.2. The pseudo-code for NMRA is given in Algorithm algorithm 11.

Algorithm 11 Pseudo-code of NMR algorithm

Begin

Inputs:

Initialize mole-rat population: $\boldsymbol{n}$

Define breeders *B: n/5*

Define workers *W: n-B*

Define $\boldsymbol{d}-$dimensional objective function, $\boldsymbol{f(x)}$

Outputs:

Find overall best

do Until $t < t_{max}$

for i= 1: W **do**

Perform worker phase using Eq. (10.2)

evaluate fitness of workers

for i=1: B **do**

Perform breeder phase using Eq. (10.3)

if x_i^{t+1} better than x_i^t

evaluate fitness of breeders

Combine worker and breeder population

Evaluate whole NMR population

Find the current best

end Until

update final **best**

End

10.3 Result and Discussion

A two-fold strategy is followed for experimental analysis. Firstly, we aim to show that the proposed approach gives better results for the test suite when compared to standard algorithms and then perform the rank-sum test to see if the proposed results are significant or not. The experimental study is divided into benchmarks (Suganthan *et al.*, 2005) given by Table A.2 and statistical testing (Derrac *et al.*, 2011). The results are taken in terms of best, worst, mean and standard deviation values of 30 runs.

10.4 Comparative Study

10.4.1 Results with respect to basic algorithms

The results taken by authors are presented in this sub-section. Here the proposed algorithm has been compared with the standard versions of DE (Storn and Price, 1997), FA (Yang, 2010a), FPA (Yang, 2012) and BA (Yang, 2010b). Table B.1 gives the parameter description of all the algorithms and in Table C.35, the simulation results are presented. It can be seen that for f_1, only NMR has given optimum results and no other algorithm was close to the results of NMR. In f_2, DE provides the best results while NMR was still comparable and with respect to other algorithms, NMR is far better. For f_3, the best achieved by each algorithm is the same and every algorithm provides almost the same results with FPA having the best standard deviation. For f_4, FA gives a better best solution but the complication is in the worst, mean and standard deviation where the results obtained by NMR are far better. All other algorithms, in this case, provide no competitiveness. The best algorithm, in this case, is the NMR. For f_5, all algorithms give comparable results and comparison is done in terms of standard deviation. It can be seen that FPA and NMR are better than others. For f_6, f_7, f_8, and f_9, NMR is able to achieve the minimum best solution, even in terms of worst, mean and standard deviation the results are much better than the others. For f_{10}, NMR and FA provide exact global optimum results while others were not able to achieve any good results. Here the FA was able to attain only the global best solution whereas NMR provided exact zero optimum for all the results. For f_{11}, NMR is the best while all others are found to be highly competitive. For f_{12}, and f_{13} NMR is the best while others get stuck in some local optima. For f_{14}, BA, FA, FPA, and NMR are highly competitive but here DE gives the best results. For f_{15}, NMR is found to be better

among all in terms of best, worst, mean as well as standard deviation. Functions f_{16}, f_{18}, f_{20}, f_{21}, and f_{26} all are fixed dimension and results are judged on the basis of standard deviation. Here the algorithm with a minimum standard deviation is the best. Here for f_{16}, f_{18}, f_{20}, DE provide the best standard deviation, for f_{21}, all are comparable but NMR has the minimum among all and for f_{26}, FA and NMR show almost same results while others are comparable. For f_{17} f_{19} and f_{22} NMR is better in every aspect and for f_{23} FA is comparable to NMR but still, NMR is better. For f_{24} and f_{25}, DE and NMR provide the exact zero global optimum. Here in terms of worst, mean and standard deviation, NMR is better. For f_{27}, NMR gives the best results. Overall, DE is found to be better for five functions, FPA for two functions, FA for one and for rest of the nineteen functions the proposed NMR provides better results.

For an algorithm to converge to a point in search space, search agents should change abruptly at the start or initial stages of optimization and converge gradually as the iterations proceed. The convergence curve for each algorithm is drawn by using the best fitness values over 500 iterations and 50 runs. Fig. 10.3 shows that for most of the functions, NMR algorithm is superior to other algorithms under comparison. Wilcoxon's rank-sum test (Derrac *et al.*, 2011) is performed to significantly test the performance of NMR with other popular algorithms. It is a non-parametric test, involving the design of two samples and is analogous to paired t-test. So, it is a pairwise test used to identify significant differences between the two algorithms. The test gives a p-value as the end result and this p-value determines the significance level of two algorithms under test. An algorithm is said to be statistically significant if the p-value is < 0.05. In Table D.5, the places where NA is written correspond to the best algorithm with respect to others. Again, from the Table D.5, it can be seen that DE gives best results for five functions, FA for one, FPA for two, BA for none and for rest of the functions NMR is more statistically significant.

10.4.2 Results with respect to State-of-the-art algorithms

For comparing the results of the literature, eleven test functions have been used. The algorithms used for comparison are WOA (Mirjalili and Lewis, 2016), GWO (Mirjalili *et al.*, 2014), PSO (Kennedy and Eberhart, 1995), gravitational search algorithm (GSA) (Rashedi *et al.*, 2009) and fast evolutionary programming (FEP) (Yao *et al.*, 1999). The results of WOA, PSO, and FEP have been taken from (Mirjalili *et al.*, 2014). The eval-

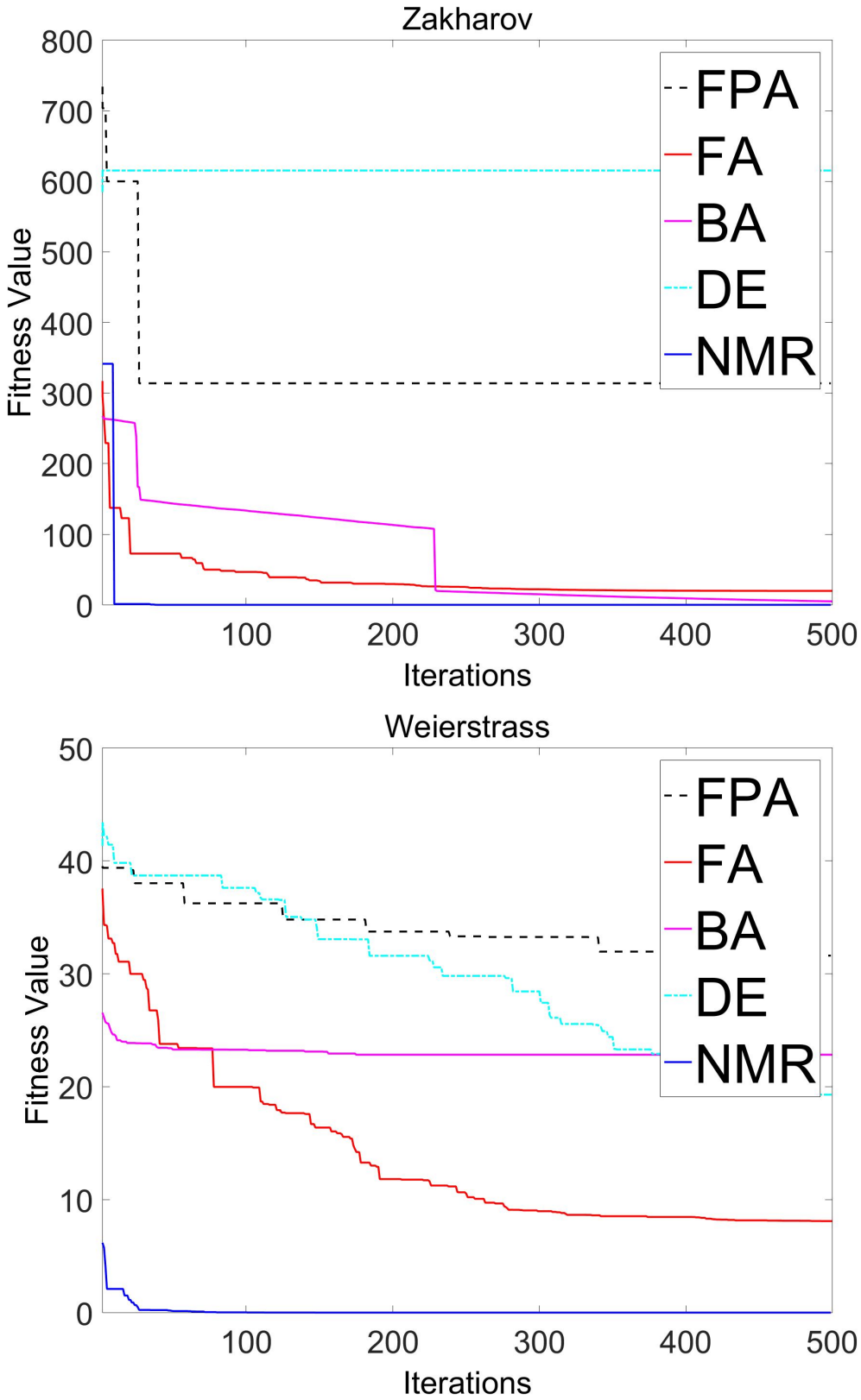


Fig. 10.3 NMR algorithm: Convergence curves with respect to other algorithms

uation criteria for these set of results is in terms of the maximum number of function evaluations and the maximum number of function evaluations is given by population size multiplied by the total number of iterations. For NMR, the population size is taken as 30 and the maximum number of iterations is 500. All other algorithms use the same set of population size and iterations as reported in the literature.

In Table C.36, the simulation results are in comparison to literature are presented. Here all the functions have been taken from Table A.1. The functions used are f_1, f_3, f_9, f_{10}, f_{19}, f_{20}, f_{21}, f_{22}, f_{23}, f_{24}, and f_{26}. It can be seen that for f_1 NMR is far better than WOA, PSO, GSA, and FEP. Here only GWO is found to be the best algorithm. For f_3, the best achieved by each algorithm is similar and every algorithm provides almost the same results with FEP having the best standard deviation. For f_9, NMR is better than PSO and FEP and given comparable results with respect to others. For f_{10}, NMR gives exact global optimum results while others were not able to achieve any good results. Here the standard deviation is also found to be exactly zero. For f_{19}, WOA is superior to NMR while all other algorithms, NMR is found to be highly competitive. Functions f_{20}, f_{21}, and f_{26} all are fixed dimension and results are judged on the basis of standard deviation. In these functions, GWO is found to provide the worst standard deviation while all others are comparable. Also, the standard deviation presented by GWO is not exactly the standard deviation and maybe the results reported by the author are vague. These results have been marked with an asterisk (*) sign. Here GSA is found to provide the best results. For f_{22} and f_{23} FEP is better in every aspect and among others, NMR is found to be comparable. For f_{24} NMR is found to provide the best solutions. Overall, we can say that NMR is highly competitive even when compared to other recently proposed algorithms.

10.4.3 Summary of Results

Exploration and exploitation are the two-necessary phenomena of evolutionary computing. These two processes decide whether an algorithm is performing better or not. Exploration means global search and is meant for exploring the whole of the search space whereas exploitation is concerned with the local search or simply intensive search to achieve global results. It has been found in literature, that both these phenomena should be balanced to achieve better results. However, it is still to be explored that to what extent local and global search is to be performed.

In NMR algorithm, exploration and exploitation are balanced by the worker and breeder phase. Here workers because of their large population size perform exploration. The workers continuously update themselves in order to become part of the breeder's group and in turn tend to explore different regions of the search space. The breeders belong to a population with higher fitness than the workers and perform more intensive exploitation as they tend to improve their fitness in order to mate with the queen. The breeders tend to follow the best-known breeder and hence take small steps to improve the good solutions. Moreover, if a breeder does not improve its fitness it is pushed into the worker's group, so it avoids solutions getting stuck in local minima and of stagnation of population of breeders. Further, exploration and exploitation of NMR have been tested by applying it to uni-modal and multi-modal functions. Here uni-modal functions have only one global optimum and hence helps to evaluate the exploitation tendencies of an algorithm. For multi-modal functions, there are more than one local optima and their number increases exponentially with population size. So, these functions are capable of evaluating the explorative tendencies. From the results in Table C.35 and Table D.5, it can be seen that the NMR algorithm is able to outperform other algorithms and is mostly at the first or second place in the majority of the test functions. This is due to the integrated mechanism of exploration and exploitation in NMR.

In this chapter, the parameters such as population size, dimension, breeding probability, and mating factor are not exploited because the article is concerned with the proposal of a new optimization technique and so the exploitation of parameters is left for future works.

10.5 Concluding Remarks

A new swarm intelligent optimization algorithm based on the mating behavior of naked mole rats has been presented in this paper. The algorithm is inspired by the naked moles and divides the population into two groups namely, workers and breeders. Workers are responsible for exploring the search space and breeders tend to exploit good solutions. The algorithm has been tested on 27 benchmark functions to verify its explorative and exploitative behavior, avoidance of local minima and convergence speed. Its performance has been compared with popular and state-of-the-art algorithms. It was found that NMR

algorithm shows good exploitation and exploration for unimodal and multimodal functions respectively. Moreover, its performance was better than FA, DE, BA, FPA, GWO, WOA, PSO, GSA and FEP algorithms for most of the benchmark functions in terms of solution quality and convergence speed. The Wilcoxon rank-sum test also proved that NMR is able to give optimum solutions consistently with respect to other algorithms.

In the future, this algorithm may be tested for real-world problems and see how it performs. Extending the algorithm to the binary version can be used in an electroencephalogram (EEG). In EEG, signals are measured by placing sensors in different positions on a person's head. A binary version of NMR can help to reduce the total number of sensors required for the operation while maintaining the accuracy of the system. The binary version can also be used for thinning of antennas and placement of nodes in wireless sensor networks. Moreover, its multi-objective version can be developed for optimization problems having many conflicting objectives.

CHAPTER 11

Conclusions and Recommendations

Preface

This chapter provides the concluding remarks on the all the proposed algorithms in literature. Here discussion based on each of the enhancements has been added and research direction on how the newly proposed algorithms aim at mitigating the problems of existing algorithms is presented. Further some future research directions on the applicability and modification of these approaches has also been discussed. The chapter also discusses the major drawbacks and requirement of hybridization, mutations, adaptivity and other techniques for designing new and novel algorithms. Finally, new hypothesis and important taxonomies have also been presented to prospective researchers for achieving scientific and technical soundness in this field.

11.1 Conclusions

In the last two decades, nature inspired algorithms have dominated every sphere of optimization research in engineering and science for solving high end problems. These algorithms are aimed at providing optimal solution when classical mathematical models fail to do so or when classical methods pose extra computational burden. Many real world problems are categorized as optimization problems, this is because of their nature, number of objective functions to be optimized, computational complexity, scalability, higher dimensionality and large number of local opitmal solutions. Such problems pose a challenge to design new efficient algorithms having the capability of providing high quality solutions. So in order to overcome such problems, many improved nature inspired algorithms have been developed in the literature. This thesis has covered some of the most powerful nature inspired algorithms namely CS and FPA. Apart from these existing algorithms, a new animal species namely naked-mole rat is analyzed and based on it, naked mole-rat algorithm is proposed. The primary focus is to introduce new hybrid variants to solve diverse set of optimization problems in an efficient manner in comparison to other algorithms from the literature. Efforts have been made to propose new and novel ideas which help CS, FPA algorithm and others to produce high quality and better solutions than other major algorithms under comparison. The thesis consists of three parts, in the first part CS algorithm has been enhanced and different variants of the same have been proposed whereas in the second part new algorithms based on FPA are proposed and have been applied to various real world applications. The third part deals with the proposal of a new NMR algorithm based on the mating patterns of naked mole-rats. Apart from basic introduction, literature and conclusions, eight new algorithms have been proposed in each of those eight chapters. In Part I, Chapter 3 deals with the enhancement of CS algorithm and aimed at improving the searching capabilities of the same. Three new algorithms namely CV 1.0, CV 2.0 and CV 3.0 were proposed by employing the concepts of division of population and generation, improved exploration and exploitation operations by equation modification and adaptive parametric settings. All of these added properties aim at improving the overall performance of CS algorithm. The next chapter 4 deals with the investigation and enhancement of the best algorithm from Chapter 3 that is CV 1.0 algorithm is improved by adding adaptive properties. The newly proposed algorithm

has been named as CVnew algorithm and the properties added include enhancements in terms of exponentially decreasing switch probability, sinusoidal decreasing local search randomization and linearly decreasing global searching properties. Adding all these enhancements aim at improving the overall performance of CV 1.0 algorithm. In Chapter 5, CVnew algorithm is enhanced by using dual division of population and linear population size reduction. The new algorithm has been named as CSsin and is found to be highly competitive with respect to the earlier proposed versions of CS algorithm. Chapter 6 deals with the introduction of SACS algorithm in which self-adaptive properties have been added to the CSsin algorithm. Here proportional population reduction based on fitness of current best and previous best solution, is followed. Secondly, the Gaussian sampling mechanism known as bare bones variant is added to improve the exploration and exploitation tendencies. The concept of Weibull distributed probability switching is also added to increase a balance between the exploration and exploitation processes. The concepts of division of population and generations to improve the exploration and exploitation properties. This new SACS algorithm is introduced to solve diverse set of real-life optimization problems from various diversified fields such as frequency modulation, stirred tank reactor, spread spectrum poly phase code design and Lennard-Jones potential problem. The experimental results show that SACS is best among all the algorithms under comparison.

The effort is continued in Part II which focused on the FPA algorithm. The section consists of three chapters, proposing high In chapter 7, FPA algorithm is exploited to the best of its potential. Here enhanced version of FPA employs the concepts of mutation operators and adaptive parameters to define new algorithms. Five new variants based on five different mutation operators namely Gaussian, Cauchy, mean-mutation, adaptive mean mutation and adaptive Lévy mutation were proposed. All these mutated strategies aimed at improving the exploration properties and local optima stagnation problem of basic FPA. Another important feature is the addition of dynamic switching in all the proposed variants which helps in balancing the local and global search properties of basic FPA. Chapter 8 introduces a new version of FPA by employing adaptive properties and in chapter 9, a hybrid variant of FPA with BA has been proposed. This algorithm is named as BFP and employs exploration and exploitation properties of both BA and FPA to solve

the problem under test. Both the algorithms proposed in chapter 9 and chapter 10 have been applied for synthesis of LAA design problem. And on experimental investigation, it has been found that both of these proposed algorithm fares better than other algorithms under comparison.

In Part III, there is only one chapter and it deals with the proposal of a new swarm intelligent algorithm. Chapter 10 deals with the analysis of a new swarm intelligent species namely naked mole-rats. The social behavior of naked mole-rats is then analysed and mathematical formulations are drawn. Based on that, a new algorithm is introduced by using the concept of worker breeder relationship of naked mole-rats. The newly proposed algorithm has been named as NMR algorithm and aimed at providing an altogether a new meta-heuristic algorithm for prospective researchers to exploit and apply the same to their research domains.

Apart from the applications used for comparison in Part II, performance evaluation of all the proposed algorithms from Part I, Part II and Part III has been done using the well known CEC benchmark datasets. The most commonly used dataset is CEC 2005 which has been used in chapter 3, chapter 5, chapter 6, chapter 10, chapter 11 and chapter 12. All other chapter use either CEC 2015 or CEC 2017 or a combination of both of these datasets. Apart from these, CEC 2019 dataset has also been used in chapter 7. A summary of comparison of all the proposed techniques with other algorithms using the above said optimization benchmarks is presented in every chapter. The results are validated using Wilcoxon rank sum test which shows that all the propsed algorithms have a strong potential to solve these test suites efficiently in comparison to other state-of-the-art algorithms. The computational complexity of majority of the proposed algorithms has also been calculated which further affirms the power of these new algorithms.

Overall eight new algorithms have been proposed in this thesis and each of these proposed algorithms uses adaptive parametric settings, enhanced exploration and exploitation properties, decreasing distributions for enhanced convergence properties, adaptive population matrices, concepts of hybridization and a combination of division of population and generations. All the proposed algorithms aim at overcoming the inherent drawbacks of existing algorithms and experimental, statistical results and application on real-world scenarios

prove the efficiency of the proposed algorithms in producing high quality solutions in comparison to other algorithms from literature.

11.2 Recommendations

This thesis has presented various new improved algorithms to solve diverse set of optimization problems. For future work, more enhancements can be made to the proposed algorithms. One of them is to improve these algorithm by employing new enhanced mutation and adaptive strategies to solve prospective research problems. Apart from parametric modifications, hybridization of two or more nature inspired algorithms into one can be done for improving the performance without compromising the complexity. This could require widening of existing taxonomies to account for mixture of algorithms in hybrid search techniques. Till date no such technique has been proposed and once such hybrid algorithms are proposed, they will definitely provide significant improvement over traditional methods. So it is imperative that the research community should modify algorithm with respect to already proposed challenging algorithms in order to reflect the importance of their proposal in the field.

Apart from this more work can be done in specific challenging conditions. Such proposal include not only single-objective optimization problems but problems relating to other diverse fields including dynamic and stochastic optimization where problem dimension varies with respect to time. Multi-and many-objective optimization is also a challenging task and goal here is to optimize two or more conflicting objectives simultaneously. Multi-modal and large scale optimization where there are large number of global optima and variable dimension size of the order of thousand respectively. Memetic algorithms can also be designed where there is hybridization of certain techniques with others to improve their performance of proposed algorithms. Apart from that, parameter tuning and parametric adaptations are also followed to adapt algorithms with respect to the problem during search. It is required that above said points should be kept in mind for further investigation before proposing a new algorithm.

Also it would be interesting if the new proposed algorithms are compared with respect to state-of-the-art variants and not the classical optimization algorithms. Unfortunately most of the research done over the past few years have dealt with the proposal of new

algorithms and comparison with respect to basic classical techniques, making it relatively easy and not so competitive. For most of the proposed algorithms when compared with other new proposal, they find limited applications and hence loose their significance. So here researchers should be encouraged to increase the number of algorithms used in comparative analysis along with the state-of-the-art algorithms and not the classical algorithms. And unless and until they are proved to be competitive with respect to prospective algorithms, they will not be used in practice and hence will not attract the research community. The algorithm should also be tested on highly challenging datasets such as CEC 2015, CEC 2014, CEC 2017 and not just the classical benchmarks. The comparison should include high dimensional, multi-modal, hybrid and composite functions from the above said datasets. Apart from this, it would also be interesting if the source code of the proposed algorithms is made available for the research community. This will provide a clean implementation of the proposed algorithm and further improve the visibility of the proposed research.

Overall, these are exciting times for research on nature inspired algorithms and dynamic taxonomies should be followed to design new prospective algorithms and their application to respective fields. We hope that the above discussed critical analysis will help the prospective researchers to take a sensible step in designing new algorithms and contribute in achieving scientific and technical soundness in this field.

Bibliography

Abbass, H.A. (2002). The self-adaptive pareto differential evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 1, 831–836, IEEE.

Alexander, R.D., Jarvis, J.V. and Sherman, P.W. (1991). *The biology of the naked mole-rat*. 599.32 BIO.

Ali, M., Storey, C. and Törn, A. (1997). Application of stochastic global optimization algorithms to practical problems. *Journal of Optimization Theory and Applications*, **95**, 545–563.

Awad, N., Ali, M., Liang, J., Qu, B. and Suganthan, P. (2016a). Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. In *Technical Report*, Nanyang Technological University Singapore.

Awad, N., Ali, M., Liang, J., Qu, B., Suganthan, P. and Definitions, P. (2016b). Evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization, nanyang technological university, jordan university of science and technology and zhengzhou university. Tech. rep., Tech. Rep.

Awad, N.H., Ali, M.Z. and Suganthan, P.N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, 372–379, IEEE.

Bäck, T. and Schwefel, H.P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, **1**, 1–23.

Balanis, C.A. (2016). *Antenna theory: analysis and design*. John wiley & sons.

Balasubramani, K. and Marcus, K. (2014). A study on flower pollination algorithm and its applications. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, **3**, 230–235.

Beni, G. and Wang, J. (1993). Swarm intelligence in cellular robotic systems. In *Robots and biological systems: towards a new bionics?*, 703–712, Springer.

Bhandari, A.K., Singh, V.K., Kumar, A. and Singh, G.K. (2014). Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using kapur's entropy. *Expert Systems with Applications*, **41**, 3538–3560.

Bindu, A.H. and Reddy, M.D. (2013). Economic load dispatch using cuckoo search algorithm. *Int. Journal Of Engineering Research and Apllications*, **3**, 498–502.

Blouin, S.F. and Blouin, M. (1988). Inbreeding avoidance behaviors. *Trends in Ecology & Evolution*, **3**, 230–233.

Bonabeau, E., Dorigo, M., Marco, D.d.R.D.F., Theraulaz, G., Théraulaz, G. *et al.* (1999). *Swarm intelligence: from natural to artificial systems*. 1, Oxford university press.

Brett, R.A. (1991). The population structure of naked mole-rat colonies. *The biology of the naked mole-rat*, 97–136.

Buffenstein, R., Jarvis, J.U., Opperman, L.A., Cavaleros, M., Ross, F.P. and Pettifor, J.M. (1994). Subterranean mole-rats naturally have an impoverished calciol status, yet synthesize calciol metabolites and calbindins. *European journal of endocrinology*, **130**, 402–409.

Cengiz, Y. and Tokat, H. (2008). Linear antenna array design with use of genetic, memetic and tabu search optimization algorithms. *Progress In Electromagnetics Research*, **1**, 63–72.

Chakraborty, D., Saha, S. and Dutta, O. (2014). De-fpa: a hybrid differential evolution-flower pollination algorithm for function minimization. In *2014 international conference on high performance computing and applications (ICHPCA)*, 1–6, IEEE.

Chaowanawatee, K. and Heednacram, A. (2012). Implementation of cuckoo search in rbf neural network for flood forecasting. In *2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks*, 22–26, IEEE.

Chellapilla, K. (1998). Combining mutation operators in evolutionary programming. *IEEE transactions on Evolutionary Computation*, **2**, 91–96.

Chen, G., Huang, X., Jia, J. and Min, Z. (2006). Natural exponential inertia weight strategy in particle swarm optimization. In *2006 6th World Congress on Intelligent Control and Automation*, vol. 1, 3672–3675, IEEE.

Cheng, C.Y., Li, S.F. and Lin, Y.C. (2019). Self-adaptive parameters in differential evolution based on fitness performance with a perturbation strategy. *Soft Computing*, **23**, 3113–3128.

Chowdhury, A., Giri, R., Ghosh, A., Das, S., Abraham, A. and Snasel, V. (2010). Linear antenna array synthesis using fitness-adaptive differential evolution algorithm. In *IEEE Congress on Evolutionary Computation*, 1–8, IEEE.

Cobos, C., Muñoz-Collazos, H., Urbano-Muñoz, R., Mendoza, M., León, E. and Herrera-Viedma, E. (2014). Clustering of web search results based on the cuckoo search algorithm and balanced bayesian information criterion. *Information Sciences*, **281**, 248–264.

Črepinšek, M., Liu, S.H. and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, **45**, 1–33.

Das, A., Mandal, D., Ghoshal, S. and Kar, R. (2017). An efficient side lobe reduction technique considering mutual coupling effect in linear array antenna using bat algorithm.

Swarm and evolutionary computation, **35**, 26–40.

Das, S. and Suganthan, P.N. (2010). Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems. *Jadavpur University, Nanyang Technological University, Kolkata*, 341–359.

Derrac, J., García, S., Molina, D. and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, **1**, 3–18.

Dey, N., Samanta, S., Yang, X.S., Das, A. and Chaudhuri, S.S. (2013). Optimisation of scaling factors in electrocardiogram signal watermarking using cuckoo search. *International Journal of Bio-Inspired Computation*, **5**, 315–326.

Dib, N.I., Goudos, S.K. and Muhsen, H. (2010). Application of taguchi's optimization method and self-adaptive differential evolution to the synthesis of linear antenna arrays. *Progress In Electromagnetics Research*, **102**, 159–180.

Draa, A. (2015). On the performances of the flower pollination algorithm–qualitative and quantitative analyses. *Applied Soft Computing*, **34**, 349–371.

Dukic, M.L. and Dobrosavljevic, Z.S. (1990). A method of a spread-spectrum radar polyphase code design. *IEEE Journal on Selected Areas in Communications*, **8**, 743–749.

Eiben, A.E., Marchiori, E. and Valko, V. (2004). Evolutionary algorithms with on-the-fly population size adjustment. In *International Conference on Parallel Problem Solving from Nature*, 41–50, Springer.

El-henawy, I. and Ismail, M. (2014). An improved chaotic flower pollination algorithm for solving large integer programming problems. *International Journal of Digital Content Technology and Its Applications*, **8**, 72.

Elsayed, S.M., Sarker, R.A. and Essam, D.L. (2011). Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, 1041–1048, IEEE.

Erlich, I., Rueda, J.L., Wildenhues, S. and Shewarega, F. (2014). Evaluating the mean-variance mapping optimization on the ieee-cec 2014 test suite. In *2014 IEEE congress on evolutionary computation (CEC)*, 1625–1632, IEEE.

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M. and Heidari, A.A. (2020). Salp swarm algorithm: theory, literature review, and application in extreme learning machines. In *Nature-Inspired Optimizers*, 185–199, Springer.

Feng, Y., Wang, G.G., Feng, Q. and Zhao, X.J. (2014). An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0-1 knapsack problems. *Computational intelligence and neuroscience*, **2014**.

Fister, I., Yang, X.S. and Fister, D. (2014). Cuckoo search: a brief literature review. In *Cuckoo search and firefly algorithm*, 49–62, Springer.

Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposito, W.R., Gümüs, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A. and Schweiger, C.A. (2013). *Handbook of test problems in local and global optimization*, vol. 33. Springer Science & Business Media.

Gandomi, A.H., Yang, X.S. and Alavi, A.H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers*, **29**, 17–35.

Gill, P.E. and Murray, W. (1972). Quasi-newton methods for unconstrained optimization. *IMA Journal of Applied Mathematics*, **9**, 91–108.

Glover, F. (1986). Future paths for integer programming and links to ar tifi cial intelli g en ce. *Computers operations research*, **13**, 533–549.

Goudos, S.K., Moysiadou, V., Samaras, T., Siakavara, K. and Sahalos, J.N. (2010). Application of a comprehensive learning particle swarm optimizer to unequally spaced linear array synthesis with sidelobe level suppression and null control. *IEEE antennas and wireless propagation letters*, **9**, 125–129.

Guney, K. and Durmus, A. (2015). Pattern nulling of linear antenna arrays using backtracking search optimization algorithm. *International Journal of Antennas and Propa-*

gation, **2015**.

Guney, K. and Onay, M. (2011). Optimal synthesis of linear antenna arrays using a harmony search algorithm. *Expert Systems with Applications*, **38**, 15455–15462.

Gutjahr, W.J. (2009). Convergence analysis of metaheuristics. 159–187.

Hallam, J.W., Akman, O. and Akman, F. (2010). Genetic algorithms with shrinking population size. *Computational Statistics*, **25**, 691–705.

Heidari, A.A., Abbaspour, R.A. and Jordehi, A.R. (2017). Gaussian bare-bones water cycle algorithm for optimal reactive power dispatch in electrical power systems. *Applied Soft Computing*, **57**, 657–671.

Hoare, M. (1979). Structure and dynamics of simple microclusters. *Advances in Chemical Physics*, **40**, 49–135.

Holland, J.H. (1992). Genetic algorithms. *Scientific american*, **267**, 66–73.

Horner, A., Beauchamp, J. and Haken, L. (1993). Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis. *Computer Music Journal*, **17**, 17–29.

Huang, L., Ding, S., Yu, S., Wang, J. and Lu, K. (2016). Chaos-enhanced cuckoo search optimization algorithms for global optimization. *Applied Mathematical Modelling*, **40**, 3860–3875.

Jeanne, R.L. (1986). The evolution of the organization of work in social insects. *Monitore Zoologico Italiano-Italian Journal of Zoology*, **20**, 119–133.

Kanagaraj, G., Ponnambalam, S. and Jawahar, N. (2013). A hybrid cuckoo search and genetic algorithm for reliability–redundancy allocation problems. *Computers & Industrial Engineering*, **66**, 1115–1124.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer ....

Karaboga, D. and Basturk, B. (2008). On the performance of artificial bee colony (abc)

algorithm. *Applied soft computing*, **8**, 687–697.

Kennedy, J. (2003). Bare bones particle swarms. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, 80–87, IEEE.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, 1942–1948, IEEE.

Khodier, M. (2013). Optimisation of antenna arrays using the cuckoo search algorithm. *IET Microwaves, Antennas & Propagation*, **7**, 458–464.

Khodier, M.M. and Al-Aqeel, M. (2009). Linear and circular array optimization: A study using particle swarm intelligence. *Progress In Electromagnetics Research*, **15**, 347–373.

Khodier, M.M. and Christodoulou, C.G. (2005). Linear array geometry synthesis with minimum sidelobe level and null control using particle swarm optimization. *IEEE transactions on antennas and propagation*, **53**, 2674–2679.

Koenig, A.C. (2002). A study of mutation methods for evolutionary algorithms. *University of Missouri-Rolla*.

Koza, J.R. and Koza, J.R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press.

Kumar, A. and Chakarverty, S. (2011). Design optimization for reliable embedded system using cuckoo search. In *2011 3rd International Conference on Electronics Computer Technology*, vol. 1, 264–268, IEEE.

Kumar, M. and Rawat, T.K. (2015). Optimal design of fir fractional order differentiator using cuckoo search algorithm. *Expert Systems with Applications*, **42**, 3433–3449.

Lampinen, J., Zelinka, I. *et al.* (2000). On stagnation of the differential evolution algorithm. In *Proceedings of MENDEL*, 76–83.

Li, X., Wang, J. and Yin, M. (2014). Enhancing the performance of cuckoo search algorithm using orthogonal learning method. *Neural Computing and Applications*, **24**, 1233–1247.

Liang, J., Qu, B., Suganthan, P. and Hernández-Díaz, A.G. (2013). Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, **201212**, 281–295.

Lim, W.C.E., Kanagaraj, G. and Ponnambalam, S. (2014). Pcb drill path optimization by combinatorial cuckoo search algorithm. *The Scientific World Journal*, **2014**.

Lin, C., Qing, A. and Feng, Q. (2010). Synthesis of unequally spaced antenna arrays by using differential evolution. *IEEE Transactions on Antennas and Propagation*, **58**, 2553–2561.

Liu, J. and Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing*, **9**, 448–462.

Long, W., Liang, X., Huang, Y. and Chen, Y. (2014). An effective hybrid cuckoo search algorithm for constrained global optimization. *Neural Computing and Applications*, **25**, 911–926.

Łukasik, S. and Kowalski, P.A. (2015). Study of flower pollination algorithm for continuous optimization. In *Intelligent Systems' 2014*, 451–459, Springer.

Luus, R. (2000). *Iterative dynamic programming*. Chapman and Hall/CRC.

Man, K.F., Tang, K.S. and Kwong, S. (1996). Genetic algorithms: concepts and applications [in engineering design]. *IEEE transactions on Industrial Electronics*, **43**, 519–534.

Marichelvam, M., Prabaharan, T. and Yang, X.S. (2014). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Applied Soft Computing*, **19**, 93–101.

Merad, L., Bendimerad, F. and Meriah, S. (2008). Design of linear antenna arrays for side lobe reduction using the tabu search method. *International Arab Journal of Information Technology (IAJIT)*, **5**.

Michalewicz, Z. (2013). *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media.

Ming, B., Chang, J.x., Huang, Q., Wang, Y.m. and Huang, S.z. (2015). Optimal operation of multi-reservoir system based-on cuckoo search algorithm. *Water Resources Management*, **29**, 5671–5687.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in engineering software*, **83**, 80–98.

Mirjalili, S. and Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, **95**, 51–67.

Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, **69**, 46–61.

Moloi, N. and Ali, M. (2005). An iterative global optimization algorithm for potential energy minimization. *Computational Optimization and Applications*, **30**, 119–132.

Nabil, E. (2016). A modified flower pollination algorithm for global optimization. *Expert Systems with Applications*, **57**, 192–203.

Nawi, N.M., Khan, A. and Rehman, M. (2013). A new levenberg marquardt based back propagation algorithm trained with cuckoo search. *Procedia Technology*, **11**, 4th.

Nelder, J.A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, **7**, 308–313.

Nguyen, T.T., Truong, A.V. and Phung, T.A. (2016). A novel method based on adaptive cuckoo search for optimal network reconfiguration and distributed generation allocation in distribution network. *International Journal of Electrical Power & Energy Systems*, **78**, 801–815.

Niu, P., Niu, S., Chang, L. *et al.* (2019). The defect of the grey wolf optimization algorithm and its verification method. *Knowledge-Based Systems*, **171**, 37–43.

Ochoa, A., González, S., Margain, L., Padilla, T., Castillo, O. and Melín, P. (2014). Implementing flower multi-objective algorithm for selection of university academic credits. In *2014 Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC 2014)*, 7–11, IEEE.

Omran, M.G., Salman, A. and Engelbrecht, A.P. (2005). Self-adaptive differential evolution. In *International Conference on Computational and Information Science*, 192–199, Springer.

Omran, M.G., Engelbrecht, A.P. and Salman, A. (2009). Bare bones differential evolution. *European Journal of Operational Research*, **196**, 128–139.

Oster, G.F. and Wilson, E.O. (1978). *Caste and ecology in the social insects*. Princeton University Press.

Ouaarab, A., Ahiod, B. and Yang, X.S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications*, **24**, 1659–1669.

Pal, S., Qu, B., Das, S. and Suganthan, P. (2010). Optimal synthesis of linear antenna arrays with multi-objective differential evolution. *Progress In Electromagnetics Research B*, **21**, 87–111.

Pani, P.R., Nagpal, R.K., Malik, R. and Gupta, N. (2013). Design of planar ebg structures using cuckoo search algorithm for power/ground noise suppression. *Progress In Electromagnetics Research*, **28**, 145–155.

Pappula, L. and Ghosh, D. (2014). Linear antenna array synthesis using cat swarm optimization. *AEU-International Journal of Electronics and Communications*, **68**, 540–549.

Pappula, L. and Ghosh, D. (2017). Synthesis of linear aperiodic array using cauchy mutated cat swarm optimization. *AEU-International Journal of Electronics and Communications*, **72**, 52–64.

Passino, K.M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems magazine*, **22**, 52–67.

Pavlyukevich, I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, **226**, 1830–1844.

Prathiba, R., Moses, M.B. and Sakthivel, S. (2014). Flower pollination algorithm applied for different economic load dispatch problems. *International Journal of Engineering and Technology*, **6**, 1009–1016.

Price, K., Awad, N., Ali, M. and Suganthan, P. (2018). Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization. In *Technical Report*, Nanyang Technological University.

Qin, A.K., Huang, V.L. and Suganthan, P.N. (2008). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, **13**, 398–417.

Qin, A.K., Huang, V.L. and Suganthan, P.N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, **13**, 398–417.

Raja, S.B., Pramod, C.S., Krishna, K.V., Ragunathan, A. and Vinesh, S. (2015). Optimization of electrical discharge machining parameters on hardened die steel using firefly algorithm. *Engineering with Computers*, **31**, 1–9.

Ram, G., Kar, R., Mandal, D. and Ghoshal, S.P. (2019). Optimal design of linear antenna arrays of dipole elements using flower pollination algorithm. *IETE Journal of Research*, **65**, 694–701.

Ram, J.P. and Rajasekar, N. (2016). A novel flower pollination based global maximum power point method for solar maximum power point tracking. *IEEE Transactions on Power Electronics*, **32**, 8486–8499.

Rashedi, E., Nezamabadi-Pour, H. and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences*, **179**, 2232–2248.

Rattan, M., Patterh, M.S. and Sohi, B. (2007). Synthesis of aperiodic linear antenna arrays using genetic algorithm. In *2007 19th International Conference on Applied Electromagnetics and Communications*, 1–4, IEEE.

Rechenberg, I. (1978). Evolutionsstrategien. 83–114.

Ronkkonen, J., Kukkonen, S. and Price, K.V. (2005). Real-parameter optimization with differential evolution. In *2005 IEEE congress on evolutionary computation*, vol. 1, 506–513, IEEE.

Sakib, N., Kabir, M.W.U., Subbir, M. and Alam, S. (2014). A comparative study of flower pollination algorithm and bat algorithm on continuous optimization problems. *International Journal of Soft Computing and Engineering*, **4**, 13–19.

Salgotra, R. and Singh, U. (2017). Application of mutation operators to flower pollination algorithm. *Expert Systems with Applications*, **79**, 112–129.

Salgotra, R. and Singh, U. (2018). A novel bat flower pollination algorithm for synthesis of linear antenna arrays. *Neural Computing and Applications*, **30**, 2269–2282.

Salgotra, R. and Singh, U. (2019). The naked mole-rat algorithm. *Neural Computing and Applications*, **31**, 8837–8857.

Salgotra, R., Singh, U. and Saha, S. (2018a). Improved cuckoo search with better search capabilities for solving cec2017 benchmark problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, 1–7, IEEE.

Salgotra, R., Singh, U. and Saha, S. (2018b). New cuckoo search algorithms with enhanced exploration and exploitation properties. *Expert Systems with Applications*, **95**, 384–420.

Sarangi, S.K., Panda, R. and Dash, M. (2014). Design of 1-d and 2-d recursive filters using crossover bacterial foraging and cuckoo search techniques. *Engineering applications of artificial intelligence*, **34**, 109–121.

Saxena, P. and Kothari, A. (2016a). Linear antenna array optimization using flower pollination algorithm. *SpringerPlus*, **5**, 306.

Saxena, P. and Kothari, A. (2016b). Optimal pattern synthesis of linear antenna array using grey wolf optimization algorithm. *International Journal of Antennas and Propagation*, **2016**.

Scholz, F. (2008). Inference for the weibull distribution. *Stat 498B Industrial Statistics*, **632**, 59.

Schramm, H. and Zowe, J. (1992). A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM journal on optimization*, **2**, 121–152.

Sharaqa, A. and Dib, N. (2014). Design of linear and elliptical antenna arrays using biogeography based optimization. *Arabian Journal for Science and Engineering*, **39**, 2929–2939.

Sharawi, M., Emary, E., Saroit, I.A. and El-Mahdy, H. (2014). Flower pollination optimization algorithm for wireless sensor network lifetime global optimization. *International Journal of Soft Computing and Engineering*, **4**, 54–59.

Shehab, M., Khader, A.T. and Al-Betar, M.A. (2017). A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing*, **61**, 1041–1059.

Shi, Y. and Eberhart, R.C. (1998). Parameter selection in particle swarm optimization. In *International conference on evolutionary programming*, 591–600, Springer.

Shor, N.Z. (1985). The subgradient method. In *Minimization methods for non-differentiable functions*, 22–47, Springer.

Simon, D. (2008). Biogeography-based optimization. *IEEE transactions on evolutionary computation*, **12**, 702–713.

Singh, H.K. and Ray, T. (2011). Performance of a hybrid ea-de-memetic algorithm on cec 2011 real world optimization problems. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, 1322–1326, IEEE.

Singh, U. and Kamal, T.S. (2012). Optimal synthesis of thinned arrays using biogeography based optimization. *Progress In Electromagnetics Research*, **24**, 141–155.

Singh, U. and Rattan, M. (2014). Design of linear and circular antenna arrays using cuckoo optimization algorithm. *Progress in Electromagnetics Research*, **46**, 1–11.

Singh, U. and Salgotra, R. (2016). Optimal synthesis of linear antenna arrays using modified spider monkey optimization. *Arabian Journal for Science and Engineering*, **41**, 2957–2973.

Singh, U. and Salgotra, R. (2017). Pattern synthesis of linear antenna arrays using enhanced flower pollination algorithm. *International Journal of Antennas and Propagation*, **2017**.

Singh, U. and Salgotra, R. (2018). Synthesis of linear antenna array using flower pollination algorithm. *Neural Computing and Applications*, **29**, 435–445.

Singh, U., Kumar, H. and Kamal, T.S. (2010). Linear array synthesis using biogeography based optimization. *Progress In Electromagnetics Research*, **11**, 25–36.

Soto, R., Crawford, B., Galleguillos, C., Monfroy, E. and Paredes, F. (2014). A prefiltered cuckoo search algorithm with geometric operators for solving sudoku problems. *The Scientific World Journal*, **2014**.

Speed, E.R. (2010). Evolving a mario agent using cuckoo search and softmax heuristics. In *2010 2nd International IEEE Consumer Electronics Society's Games Innovations Conference*, 1–7, IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, **11**, 341–359.

Subhashini, K.R. and Satapathy, J.K. (2017). Development of an enhanced ant lion optimization algorithm and its application in antenna array synthesis. *Applied Soft Computing*, **59**, 153–173.

Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A. and Tiwari, S. (2005). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL report*, **2005005**, 2005.

Tanabe, R. and Fukunaga, A.S. (2014). Improving the search performance of shade using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*, 1658–1665, IEEE.

Thornhill, N.W. (1993). *The natural history of inbreeding and outbreeding: theoretical and empirical perspectives*. University of Chicago Press.

Valenzuela, L., Valdez, F. and Melin, P. (2017). Flower pollination algorithm with fuzzy approach for solving optimization problems. In *Nature-Inspired Design of Hybrid Intelligent Systems*, 357–369, Springer.

Walia, G.S. and Kapoor, R. (2014). Intelligent video target tracking using an evolutionary

particle filter based upon improved cuckoo search. *Expert Systems with Applications*, **41**, 6315–6326.

Wang, G.G., Gandomi, A.H., Zhao, X. and Chu, H.C.E. (2016). Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing*, **20**, 273–285.

Wang, H., Rahnamayan, S., Sun, H. and Omran, M.G. (2013). Gaussian bare-bones differential evolution. *IEEE Transactions on Cybernetics*, **43**, 634–647.

Wang, J.s., Han, S., Shen, N.n. and Li, S.x. (2014). Features extraction of flotation froth images and bp neural network soft-sensor model of concentrate grade optimized by shuffled cuckoo searching algorithm. *The Scientific World Journal*, **2014**.

Wang, L. and Zhong, Y. (2015). Cuckoo search algorithm with chaotic maps. *Mathematical Problems in Engineering*, **2015**.

Wang, R. and Zhou, Y. (2014). Flower pollination algorithm with dimension by dimension improvement. *Mathematical Problems in Engineering*, **2014**.

Wang, W.B., Feng, Q. and Liu, D. (2011). Application of chaotic particle swarm optimization algorithm to pattern synthesis of antenna arrays. *Progress In Electromagnetics Research*, **115**, 173–189.

Weihull, W. (1951). A statistical distribution function of wide applicability. *J Appl Mech*, **18**, 290–293.

Wolpert, D.H. and Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, **1**, 67–82.

Xin, J., Chen, G. and Hai, Y. (2009). A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In *2009 International Joint Conference on Computational Sciences and Optimization*, vol. 1, 505–508, IEEE.

Yang, X.S. (2010a). Firefly algorithm, stochastic test functions and design optimisation. *arXiv preprint arXiv:1003.1409*.

Yang, X.S. (2010b). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, 65–74, Springer.

Yang, X.S. (2012). Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation*, 240–249, Springer.

Yang, X.S. and Deb, S. (2009). Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 210–214, IEEE.

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

Yang, X.S., Karamanoglu, M. and He, X. (2013). Multi-objective flower algorithm for optimization. *Procedia Computer Science*, **18**, 861–868.

Yang, Z., Tang, K. and Yao, X. (2008). Self-adaptive differential evolution with neighborhood search. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 1110–1116, IEEE.

Yao, X., Lin, G. and Liu, Y. (1997). An analysis of evolutionary algorithms based on neighbourhood and step sizes. In *International Conference on Evolutionary Programming*, 297–307, Springer.

Yao, X., Liu, Y. and Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, **3**, 82–102.

Yue, C., Price, K., uganthan, P., Liang, J., Ali, M., Qu, B., NH, A. and Biswas, P. (2019). Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. In *Technical Report*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University Singapore.

Zaman, M.A. and Abdul Matin, M. (2012). Nonuniformly spaced linear antenna array design using firefly algorithm. *International Journal of Microwave Science and Technology*, **2012**.

Zhang, J. and Sanderson, A.C. (2009). Jade: adaptive differential evolution with optional

external archive. *IEEE Transactions on evolutionary computation*, **13**, 945–958.

Zhang, Y., Wang, L. and Wu, Q. (2012). Modified adaptive cuckoo search (macs) algorithm and formal description for global optimisation. *International Journal of Computer Applicationsin Technology*, **44**, 73.

Zhou, X., Wu, Z., Wang, H. and Rahnamayan, S. (2016a). Gaussian bare-bones artificial bee colony algorithm. *Soft Computing*, **20**, 907–924.

Zhou, Y., Wang, R. and Luo, Q. (2016b). Elite opposition-based flower pollination algorithm. *Neurocomputing*, **188**, 294–310.

Appendices

APPENDIX A

Test Suites

Table A.1 CEC 2015 Test Suite

	No.	**Functions**	**Global Minima**
Unimodal Function	F_1	Rotated High Conditional Elliptic	100
	F_2	Rotated Cigar	200
Simple Multimodal Functions	F_3	Shifted and Rotated Ackley	300
	F_4	Shifted and Rotated Rastrigin	400
	F_5	Shifted and Rotated Schwefel	500
Hybrid Functions	F_6	Hybrid 1 (N=3)	600
	F_7	Hybrid 2 (N=4)	700
	F_8	Hybrid 3 (N=5)	800
Composition Functions	F_9	Composition 1 (N=3)	900
	F_{10}	Composition 2 (N=3)	1000
	F_{11}	Composition 3 (N=5)	1100
	F_{12}	Composition 4 (N=5)	1200
	F_{13}	Composition 5 (N=5)	1300
	F_{14}	Composition 6 (N=7)	1400
	F_{15}	Composition 7 (N=10)	1500
Search Range : $[-100, 100]^D$			

Table A.2 CEC 2005 Test Suite

Test Problems	Objective	Search Range	Optimum Value	D
Unimodal Functions				
Ackley	$f_1(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20 + e$	$[-100.\ 100]$	0	30
Fixed Dimension Functions				
Colville	$f_2(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1)$	$[-10, 10]$	0	4
De Jong function N. 5	$f_3(x) = \left(0.002 + \sum_{i=1}^{25} 1/i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6\right)^{-1}$	$[-65.536, 65.536]$	0	2
Dixon & Price	$f_4(x) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$	$[-10,10]$	0	30
Michalewicz	$f_5(x) = -\sum_{i=1}^{D}\sin(x_i)sin^{2m}(\frac{ix_i^2}{\pi})$	$[0,\ \pi]$	-9.6601	10
Powell	$f_6(x) = \sum_{i=1}^{d}[(x_{4i-3} - 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^2]$	$[-4,\ 5]$	0	30
Quartic	$f_7(x) = \sum_{i=1}^{D} x_i^4 + random(0, 1)$	$[-1.28,\ 1.28]$	0	30
Salomon	$f_8(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + \left(0.1\sqrt{\sum_{i=1}^{D} x_i^2}\right)$	$[-100,100]$	0	30
Sphere	$f_9(x) = \sum_{i=1}^{D} x_i^2$	$[-100,\ 100]$	0	30
Step	$f_{10}(x) = \sum_{i=1}^{D}(\lvert x_i + 0.5\rvert)^2$	$[-100,\ 100]$	0	30
Sum of different powers	$f_{11}(x) = \sum_{i=1}^{D} x_i^{i+1}$	$[-1,\ 1]$	0	30
Tablet	$f_{12}(x) = 10^6 x_i^2 + \sum_{i=1}^{D} x_i^2$	$[-5,\ 5]$	0	30
Zakharov	$f_{13}(x) = \sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^4$	$[-5,\ 10]$	0	30
Beale	$f_{14}(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	$[-4.5,\ 4.5]$	0	2
Cigar	$f_{15}(x) = x_0^2 + 10000\sum_{i=1}^{D} x_i^2$	$[-10,\ 10]$	0	30
Easom	$f_{16}(x) = -cosx_1 cosx_2 e^{(-(x_1-\pi)^2-(x_2-\pi)^2)}$	$[-10,\ 10]$	-1	2
Elliptic	$f_{17}(x) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} x_i^2$	$[-100,\ 100]$	0	30
Goldstein & Price	$f_{18}(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3{x_1}^2 - 14 x_2 + 6x_1x_2 + 3{x_2}^2))(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12{x_1}^2 + 48x_2 - 36x_1x_2 + 27{x_2}^2))$	$[-2,\ 2]$	3	2
Griewank	$f_{19} = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600,\ 600]$	0	30
Hartmann 3	$f_{20}(x) = -\sum_{i=1}^{4}\alpha_i\exp[-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2]$	$[0,\ 1]$	-3.86278	3
Hartmann 6	$f_{21}(x) = -\sum_{i=1}^{4}\alpha_i\exp[-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2]$	$[0,\ 1]$	-3.32237	6
Penalized 1	$f_{22} = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)2\} + \sum_{i=1}^{n} u(x_i,\ 10,\ 100,\ 4)$ $y_i = 1 + \frac{x_{i+1}}{4}$ $\mathrm{u}(x_i, \mathrm{a}, \mathrm{k}, \mathrm{m}) = \begin{cases} k(x_i - a)^m x_i > a; 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	$[-50,\ 50]$	0	30
Penalized 2	$f_{23} = 0.1\{(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + sin^2(2\pi x_n)\right]\} + \sum_{i=1}^{n} u(x_i,\ 5,\ 100,\ 4)$ $\mathrm{u}(x_i, \mathrm{a}, \mathrm{k}, \mathrm{m}) = \begin{cases} k(x_i - a)^m x_i > a; 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	$[-50,\ 50]$	0	30
Rastrigin	$f_{24}(x) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12,\ 5.12]$	0	30
Scaffer	$f_{25}(x) = \left[\frac{1}{n-1}\sqrt{s_i}.(\sin\left(50.0s_i^{\frac{1}{5}}\right) + 1)\right]^2 s_i = \sqrt{x_i^2 + x_{i+1}^2}$	$[-100,\ 100]$	0	30
Six Hump Camel	$f_{26}(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$[-5,\ 5]$	-1.0316	2
Weierstrass	$f_{27}(x) = \sum_{i=1}^{D}\sum_{k=0}^{kmax}[a^k\cos\left(2\pi b^k(x_i + 0.5)\right)] - D\sum_{k=0}^{kmax}[a^k\cos(2\pi b^k.0.5)]$; *where a=0.5, b=3, kmax=20*	$[-0.5,\ 0.5]$	0	30

Table A.3 CEC 2017 Test Suite

	No.	Functions	$F_i^* = F_i(x^*)$
Unimodal Functions	F_1	Shifted and Rotated Bent Cigar Function	100
	F_2	Shifter and Rotated Sum of Different Power Function*	200
	F_3	Shifted and Rotated Zakharov Function	300
Multimodal Functions	F_4	Shifted and Rotated Rosenbrock's Function	400
	F_5	Shifted and Rotated Rastrigin's Function	500
	F_6	Shifted and Rotated Expanded Scaffer's $F6$ Function	600
	F_7	Shifted and Rotated Lunacek Bi-Rastrigin Function	700
	F_8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	F_9	Shifted and Rotated Levy Function	900
	F_{10}	Shifted and Rotated Schwefel's Function	1000
Hybrid Functions	F_{11}	Hybrid Function 1($N = 3$)	1100
	F_{12}	Hybrid Function 2($N = 3$)	1200
	F_{13}	Hybrid Function 3($N = 3$)	1300
	F_{14}	Hybrid Function 4($N = 4$)	1400
	F_{15}	Hybrid Function 5($N = 4$)	1500
	F_{16}	Hybrid Function 6($N = 4$)	1600
	F_{17}	Hybrid Function 6($N = 5$)	1700
	F_{18}	Hybrid Function 6($N = 5$)	1800
	F_{19}	Hybrid Function 6($N = 5$)	1900
	F_{20}	Hybrid Function 6($N = 6$)	2000
Composition Functions	F_{21}	Composition Function 1($N = 3$)	2100
	F_{22}	Composition Function 2($N = 3$)	2200
	F_{23}	Composition Function 3($N = 4$)	2300
	F_{24}	Composition Function 4($N = 4$)	2400
	F_{25}	Composition Function 5($N = 5$)	2500
	F_{26}	Composition Function 6($N = 5$)	2600
	F_{27}	Composition Function 7($N = 6$)	2700
	F_{28}	Composition Function 8($N = 6$)	2800
	F_{29}	Composition Function 9($N = 3$)	2900
	F_{30}	Composition Function 10($N = 3$)	3000
		Search Range: $[-100, 100]^D$	

Table A.4 CEC 2020 Test Suite

No.	**Functions**	**Global Minima**	**Search Range**
F_1	Shifted and Rotated Bent Cigar Function	100	[-100, 100]
F_2	Shifted and Rotated Schwefel's Function	1100	[-100, 100]
F_3	Shifted and Rotated Lunacek bi-Rastrigin Function	700	[-100, 100]
F_4	Expanded Rosenbrock's plus Griewangk's Function	1900	[-100, 100]
F_5	Hybrid Function 1 (N=3)	1700	[-100, 100]
F_6	Hybrid Function 2 (N=4)	1600	[-100, 100]
F_7	Hybrid Function 3 (N=5)	2100	[-100, 100]
F_8	Composite Function 1 (N=3)	2200	[-100, 100]
F_9	Composite Function 2 (N=4)	2400	[-100, 100]
F_{10}	Composite Function 3 (N=5)	2500	[-100, 100]

APPENDIX B

Parameter Settings Tables

Table B.1 Parameter Settings of Various Algorithms

Algorithm	Parameters	Values
FA	Number of fireflies	20
	Alpha (α)	0.5
	Beta (β)	0.2
	Gamma (γ)	1
	Maximum number of iterations	1000
	Stopping Criteria	Max Iteration.
BA	Population size	20
	Loudness	0.5
	Pulse rate	0.5
	Maximum number iterations	1000
	Stopping Criteria	Max Iteration.
FPA	Population Size	20
	Probability Switch	0.8
	Maximum Iterations	500
	Stopping Criteria	Max Iteration.
GWO	$\overrightarrow{a}$	Linearly decreased from 2 to 0.
	Population size	60
	Maximum number of generations	1000
	Stopping Criteria	Max Iteration.
CV 1.0, CV 2.0 & CV 3.0	Switch probability	0.5
	Maximum Cycles	500
	Stopping Criteria	Max Iteration.
BFP	Population size	20
	Loudness	0.25
	Pulse rate	0.25
	Probability switch	0.8
	Maximum number iterations	1000
	Stopping Criteria	Max Iteration.
ALFPA	Population Size	60
	Probability Switch	Dynamic
	Maximum Iterations	1000
	Stopping Criteria	Max Iteration.
EFPA	Population Size	30
	Probability Switch	Dynamic and linearly decreasing
	Maximum Iterations	1000
	Stopping Criteria	Max Iteration.
NMR	Population Size	50
	Breeding probability	0.5
	λ	U(0,1)
	Maximum Iterations	500
	Stopping Criteria	Max Iteration.

APPENDIX C

Experimental Result Tables

Table C.1 CV1.0 algorithm: Comparison for $p = 0.05$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	21.5032	3.16E+02	99.68	57.3408
	CV 1.0	6.39E-85	1.17E-69	2.36E-71	1.66E-70
	CV 2.0	2.15E-89	1.90E-70	4.60E-72	2.73E-71
	CV 3.0	8.08E-78	5.20E-50	1.77E-51	8.91E-51
f_6	CS	2.31E+04	3.69E+05	1.50E+05	9.53E+04
	CV 1.0	5.35E-83	3.21E-68	8.80E-70	4.59E-69
	CV 2.0	1.31E-85	1.57E-69	3.59E-71	2.22E-70
	CV 3.0	4.54E-69	1.70E-45	3.42E-47	2.41E-46
f_{17}	CS	0.00E-00	7.53E-13	2.56E-14	1.15E-13
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{20}	CS	3.0000	3.0000	3.0000	1.57E-15
	CV 1.0	3.0000	3.0000	3.0000	1.92E-15
	CV 2.0	3.0000	3.0000	3.0000	1.74E-15
	CV 3.0	3.0000	3.0000	3.0000	1.81E-15
f_{23}	CS	-3.3224	-3.2032	-3.3176	0.0235
	CV 1.0	-3.3224	-3.3224	-3.3224	9.27E-09
	CV 2.0	-3.3224	-3.2032	-3.3200	0.0168
	CV 3.0	-3.3224	-3.3224	-3.3224	2.55E-08
f_{24}	CS	-1.0316	-1.0316	-1.0316	6.72E-16
	CV 1.0	-1.0316	-1.0316	-1.0316	8.80E-16
	CV 2.0	-1.0316	-1.0316	-1.0316	7.93E-16
	CV 3.0	-1.0316	-1.0316	-1.0316	6.72E-16

Table C.2 CV1.0 algorithm: Comparison for $p = 0.25$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	2.8442	35.4595	10.1883	6.7095
	CV 1.0	4.48E-85	1.88E-70	4.50E-72	2.69E-71
	CV 2.0	8.42E-92	9.60E-75	2.18E-76	1.35E-75
	CV 3.0	1.68E-75	4.89E-47	9.83E-49	6.92E-48
f_2	CS	10	95	28.9400	15.7485
	CV 1.0	0	0	0	0
	CV 2.0	0	0	0	0
	CV 3.0	0	0	0	0
f_6	CS	3.61E+03	3.34E+04	1.63E+04	7.40E+03
	CV 1.0	9.61E-83	4.04E-68	1.13E-69	5.91E-69
	CV 2.0	1.25E-90	1.11E-71	2.60E-73	1.58E-72
	CV 3.0	2.50E-66	1.08E-44	2.57E-46	1.54E-45
f_8	CS	-8.67E+03	-7.25E+03	-7.90E+03	3.01E+02
	CV 1.0	-8.66E+03	-7.16E+03	-7.86E+03	3.78E+02
	CV 2.0	-8.44E+03	-6.84E+03	-7.65E+03	3.78E+02
	CV 3.0	-8.84E+03	-6.86E+03	-7.67E+03	3.99E+02
f_{14}	CS	0.00E-00	5.41E-14	2.63E-15	1.06E-14
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{15}	CS	2.5332	10.1373	4.8707	1.5239
	CV 1.0	0.0052	7.6306	1.3972	2.2321
	CV 2.0	0.0085	12.773	2.3308	3.2601
	CV 3.0	0.0013	7.2351	1.8664	2.1370
f_{16}	CS	3.9431	49.543	21.214	12.134
	CV 1.0	0.2229	1.0431	0.4698	0.1660
	CV 2.0	0.1218	0.5092	0.2770	0.1040
	CV 3.0	0.0407	0.4741	0.1917	0.1039
f_{17}	CS	0.00E-00	2.62E-13	1.01E-14	4.50E-14
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{18}	CS	6.6224	15.5536	11.3191	1.9997
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{20}	CS	3.0000	3.0000	3.0000	2.76E-15
	CV 1.0	3.0000	3.0000	3.0000	2.76E-15
	CV 2.0	3.0000	3.0000	3.0000	2.76E-15
	CV 3.0	3.0000	3.0000	3.0000	2.72E-15
f_{22}	CS	-3.8628	-3.8628	-3.8628	3.02E-15
	CV 1.0	-3.8628	-3.8628	-3.8628	2.89E-15
	CV 2.0	-3.8628	-3.8628	-3.8628	2.83E-15
	CV 3.0	-3.8628	-3.8628	-3.8628	2.90E-15
f_{23}	CS	-3.3224	-3.3224	-3.3224	3.67E-07
	CV 1.0	-3.3224	-3.3224	-3.3224	8.24E-09
	CV 2.0	-3.3224	-3.3224	-3.3224	4.13E-07
	CV 3.0	-3.3224	-3.3224	-3.3224	7.91E-08
f_{24}	CS	-1.0316	-1.0316	-1.0316	6.72E-16
	CV 1.0	-1.0316	-1.0316	-1.0316	9.12E-16
	CV 2.0	-1.0316	-1.0316	-1.0316	9.22E-16
	CV 3.0	-1.0316	-1.0316	-1.0316	6.72E-16

Table C.3 CV1.0 algorithm: Comparison for $p = 0.50$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	2.7124	12.7685	6.2805	2.6178
	CV 1.0	4.30E-84	6.70E-72	1.37E-73	9.47E-73
	CV 2.0	3.32E-88	2.57E-69	5.15E-71	3.64E-70
	CV 3.0	2.94E-72	1.63E-45	3.26E-47	2.30E-46
f_2	CS	9	36	17.1600	5.8392
	CV 1.0	0	0	0	0
	CV 2.0	0	0	0	0
	CV 3.0	0	0	0	0
f_6	CS	6.22E+03	3.28E+04	1.56E+04	5.68E+03
	CV 1.0	1.17E-80	4.67E-68	1.86E-69	9.20E-69
	CV 2.0	1.72E-89	5.44E-69	1.10E-70	7.70E-70
	CV 3.0	6.53E-68	2.69E-43	5.39E-45	3.81E-44
f_8	CS	-8.87E+03	-7.65E+03	-8.21E+03	2.86E+02
	CV 1.0	-9.12E+03	-7.29E+03	-8.04E+03	4.72E+02
	CV 2.0	-8.69E+03	-6.95E+03	-7.78E+03	3.55E+02
	CV 3.0	-8.81E+03	-7.34E+03	-8.03E+03	3.32E+02
f_{14}	CS	0.00E-00	5.15E-10	3.21E-11	8.30E-11
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{15}	CS	1.4836	6.4729	3.6821	1.0459
	CV 1.0	0.0052	2.0501	0.0643	0.2877
	CV 2.0	0.0086	5.0918	0.1752	0.8034
	CV 3.0	0.0027	3.5288	0.3224	0.8327
f_{16}	CS	1.8493	14.5429	6.6227	3.5481
	CV 1.0	0.1482	0.8791	0.4309	0.1766
	CV 2.0	0.1179	0.7175	0.3403	0.1419
	CV 3.0	0.0314	0.2375	0.1269	0.0509
f_{17}	CS	0.00E-00	5.67E-11	2.33E-12	1.02E-11
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{18}	CS	7.7972	15.7809	11.8559	1.7420
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{20}	CS	3.0000	3.0000	3.0000	4.87E-14
	CV 1.0	3.0000	3.0000	3.0000	2.95E-15
	CV 2.0	3.0000	3.0000	3.0000	5.02E-15
	CV 3.0	3.0000	3.0000	3.0000	5.25E-14
f_{22}	CS	-3.8628	-3.8628	-3.8628	1.29E-12
	CV 1.0	-3.8628	-3.8628	-3.8628	3.22E-15
	CV 2.0	-3.8628	-3.8628	-3.8628	6.51E-15
	CV 3.0	-3.8628	-3.8628	-3.8628	2.23E-14
f_{23}	CS	-3.3224	-3.3223	-3.3224	8.77E-06
	CV 1.0	-3.3224	-3.3224	-3.3224	1.78E-08
	CV 2.0	-3.3224	-3.3224	-3.3224	1.15E-07
	CV 3.0	-3.3224	-3.3224	-3.3224	3.13E-07
f_{24}	CS	-1.0316	-1.0316	-1.0316	1.05E-15
	CV 1.0	-1.0316	-1.0316	-1.0316	1.11E-14
	CV 2.0	-1.0316	-1.0316	-1.0316	6.08E-15
	CV 3.0	-1.0316	-1.0316	-1.0316	5.13E-15

Table C.4 CV1.0 algorithm: Comparison for $p = 0.75$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	6.9762	22.9576	12.84	3.6103
	CV 1.0	1.05E-83	4.74E-73	1.40E-74	6.77E-74
	CV 2.0	6.69E-93	1.84E-68	3.69E-70	2.60E-69
	CV 3.0	1.71E-70	3.89E-48	8.24E-50	5.51E-49
f_2	CS	14	53	27.6000	6.9282
	CV 1.0	0	0	0	0
	CV 2.0	0	0	0	0
	CV 3.0	0	0	0	0
f_6	CS	1.61E+04	9.91E+04	4.53E+04	1.57E+04
	CV 1.0	2.29E-81	1.31E-70	4.71E-72	1.89E-71
	CV 2.0	3.73E-94	2.95E-67	6.03E-69	4.17E-68
	CV 3.0	1.04E-69	3.59E-44	1.15E-45	5.48E-45
f_8	CS	-9.48E+03	-8.34E+03	-8.77E+03	2.32E+02
	CV 1.0	-9.22E+03	-7.63E+03	-8.31E+03	3.37E+02
	CV 2.0	-8.83E+03	-7.67E+03	-8.18E+03	2.51E+02
	CV 3.0	-9.55E+03	-7.98E+03	-8.66E+03	3.64E+02
f_{14}	CS	2.29E-10	1.89E-04	9.45E-06	3.29E-05
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{15}	CS	1.7396	4.3531	2.8953	0.5117
	CV 1.0	0.0070	0.2231	0.0271	0.0361
	CV 2.0	0.0063	0.2749	0.0276	0.0518
	CV 3.0	0.0010	1.1258	0.0329	0.1587
f_{16}	CS	2.8388	7.8916	4.7047	1.1451
	CV 1.0	0.1036	1.1096	0.3984	0.1724
	CV 2.0	0.1298	0.5774	0.2918	0.1130
	CV 3.0	0.0480	0.2979	0.1240	0.0586
f_{17}	CS	5.08E-13	3.24E-07	2.53E-08	6.60E-08
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{18}	CS	8.5907	11.2844	1.1912	13.6158
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{20}	CS	3.0000	3.0000	3.0000	2.18E-06
	CV 1.0	3.0000	3.0000	3.0000	3.24E-15
	CV 2.0	3.0000	3.0000	3.0000	1.62E-14
	CV 3.0	3.0000	3.0000	3.0000	1.06E-08
f_{22}	CS	-3.8628	-3.8628	-3.8628	3.57E-08
	CV 1.0	-3.8628	-3.8628	-3.8628	8.27E-15
	CV 2.0	-3.8628	-3.8628	-3.8628	7.40E-13
	CV 3.0	-3.8628	-3.8628	-3.8628	2.32E-10
f_{23}	CS	-3.3224	-3.3224	-3.3224	6.77E-07
	CV 1.0	-3.3224	-3.3224	-3.3224	9.37E-09
	CV 2.0	-3.3224	-3.3224	-3.3224	9.92E-08
	CV 3.0	-3.3224	-3.3224	-3.3224	3.89E-07
f_{24}	CS	-1.0316	-1.0316	-1.0316	1.64E-12
	CV 1.0	-1.0316	-1.0316	-1.0316	1.21E-13
	CV 2.0	-1.0316	-1.0316	-1.0316	3.99E-13
	CV 3.0	-1.0316	-1.0316	-1.0316	3.97E-12

Table C.5 CV1.0 algorithm: Comparison for $p = 0.95$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	61.6714	2.85E+02	1.48E+02	50.2003
	CV 1.0	1.09E-86	6.77E-70	1.35E-71	9.58E-71
	CV 2.0	1.93E-89	1.07E-71	3.48E-73	1.65E-72
	CV 3.0	1.26E-72	1.04E-48	2.21E-50	1.48E-49
f_2	CS	118	366	2.16E+02	64.6726
	CV 1.0	0	0	0	0
	CV 2.0	0	0	0	0
	CV 3.0	0	0	0	0
f_6	CS	3.69E+05	2.21E+06	9.75E+05	4.39E+05
	CV 1.0	1.59E-82	6.87E-68	2.33E-69	1.13E-68
	CV 2.0	4.17E-85	7.54E-69	2.32E-70	1.12E-69
	CV 3.0	1.79E-76	1.36E-39	2.73E-41	1.93E-40
f_8	CS	-1.03E+04	-9.26E+03	-9.82E+03	2.90E+02
	CV 1.0	-9.56E+03	-8.28E+03	-8.92E+03	2.68E+02
	CV 2.0	-9.68E+03	-8.38E+03	-8.82E+03	2.69E+02
	CV 3.0	-1.06E+04	-9.31E+03	-9.79E+03	3.18E+02
f_{14}	CS	3.99E-04	0.1482	0.0594	0.0519
	CV 1.0	0.00E-00	0.1456	0.0058	0.0288
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{15}	CS	1.3909	3.6053	2.7314	0.4917
	CV 1.0	0.0049	0.1497	0.0218	0.0264
	CV 2.0	0.0050	3.6824	0.0902	0.5184
	CV 3.0	0.0026	1.0463	0.0359	0.1496
f_{16}	CS	6.5011	16.839	11.857	2.5382
	CV 1.0	0.1884	0.8525	0.4546	0.1516
	CV 2.0	0.1254	0.5908	0.3052	0.1022
	CV 3.0	0.0503	0.3658	0.1672	0.0772
f_{17}	CS	1.15E-05	0.3626	0.0421	0.0773
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
f_{18}	CS	9.6206	14.0261	11.7413	1.0328
	CV 1.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 2.0	0.00E-00	0.00E-00	0.00E-00	0.00E-00
	CV 3.0	0.00E-00	7.10E-15	1.42E-16	1.00E-15
f_{20}	CS	3.0001	7.6876	3.1438	0.6627
	CV 1.0	3.0000	3.0000	3.0000	3.63E-15
	CV 2.0	3.0000	3.0000	3.0000	3.71E-14
	CV 3.0	3.0000	3.0001	3.0000	1.25E-05
f_{22}	CS	-3.8628	-3.8404	-3.8620	4.00E-03
	CV 1.0	-3.8628	-3.8628	-3.8628	4.94E-15
	CV 2.0	-3.8628	-3.8628	-3.8628	2.30E-11
	CV 3.0	-3.8628	-3.8627	-3.8628	1.44E-05
f_{23}	CS	-3.3223	-3.3179	-3.3218	7.66E-04
	CV 1.0	-3.3224	-3.3224	-3.3224	5.99E-08
	CV 2.0	-3.3224	-3.3223	-3.3224	5.33E-06
	CV 3.0	-3.3224	-3.3217	-3.3222	1.61E-04
f_{24}	CS	-1.0316	-1.0316	-1.0316	3.33E-08
	CV 1.0	-1.0316	-1.0316	-1.0316	1.41E-12
	CV 2.0	-1.0316	-1.0316	-1.0316	1.59E-10
	CV 3.0	-1.0316	-1.0316	-1.0316	4.32E-07

Table C.6 CV1.0 algorithm: Comparison for population size 40

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	9.5134	33.57	17.824	5.2876
	CV 1.0	1.21E-97	5.29E-85	1.41E-86	7.59E-86
	CV 2.0	**5.71E-104**	6.22E-86	**1.27E-87**	**8.79E-87**
	CV 3.0	6.08E-88	**9.55E-71**	2.58E-72	1.40E-71
f_3	CS	0.0458	0.2506	0.1352	0.0406
	CV 1.0	**6.89E-05**	**0.0020**	**8.07E-04**	**5.33E-04**
	CV 2.0	3.61E-04	0.0105	0.0025	0.0022
	CV 3.0	4.13E-04	0.0241	0.0060	0.0059
f_4	CS	0.0402	0.1981	0.0973	0.0275
	CV 1.0	2.10E-101	1.84E-83	3.70E-85	2.61E-84
	CV 2.0	**3.79E-106**	**3.17E-92**	**7.29E-94**	**4.49E-93**
	CV 3.0	5.11E-93	8.89E-69	1.77E-70	1.25E-69
f_6	CS	1.70E+04	9.31E+04	3.76E+04	1.35E+04
	CV 1.0	1.10E-93	1.90E-77	3.88E-79	2.68E-78
	CV 2.0	**1.90E-103**	**1.64E-84**	**3.99E-86**	**2.35E-85**
	CV 3.0	3.32E-84	9.85E-65	2.28E-66	1.40E-65
f_9	CS	6.8935	16.6332	10.4030	1.9881
	CV 1.0	8.88E-16	4.44E-15	**1.17E-15**	**9.73E-16**
	CV 2.0	8.88E-16	4.44E-15	1.66E-15	1.48E-15
	CV 3.0	8.88E-16	4.44E-15	2.30E-15	1.75E-15
f_{13}	CS	1.64E+02	5.60E+02	3.45E+02	95.1599
	CV 1.0	8.57E-95	2.35E-84	5.82E-86	3.36E-85
	CV 2.0	**1.73E-104**	**1.25E-85**	**2.62E-87**	**1.77E-86**
	CV 3.0	2.94E-88	1.97E-64	3.95E-66	2.79E-65
f_{15}	CS	2.4028	5.1099	3.8519	0.5694
	CV 1.0	**0.0010**	**0.0157**	**0.0057**	**0.0039**
	CV 2.0	0.0031	0.2624	0.0232	0.0522
	CV 3.0	0.0025	0.5953	0.0361	0.1046
f_{16}	CS	5.5846	12.160	8.3277	1.5837
	CV 1.0	**0.0432**	0.5007	0.1670	0.1027
	CV 2.0	0.0709	0.7461	0.2374	0.1310
	CV 3.0	0.0725	**0.4160**	**0.1649**	**0.0651**
f_{18}	CS	12.4613	18.1087	15.7333	1.4143
	CV 1.0	**0**	**0**	**0**	**0**
	CV 2.0	**0**	**0**	**0**	**0**
	CV 3.0	**0**	**0**	**0**	**0**
f_{20}	CS	3.0000	3.0000	3.0000	6.40E-14
	CV 1.0	3.0000	3.0000	3.0000	**3.36E-15**
	CV 2.0	3.0000	3.0000	3.0000	9.42E-15
	CV 3.0	3.0000	3.0000	3.0000	1.98E-14
f_{21}	CS	3.51E-16	6.61E-10	2.65E-11	1.00E-10
	CV 1.0	**2.95E-23**	**3.34E-19**	**1.90E-20**	**5.18E-20**
	CV 2.0	1.34E-20	2.20E-13	5.37E-15	3.12E-14
	CV 3.0	4.65E-17	2.66E-12	1.48E-13	3.89E-13
f_{22}	CS	-3.8628	-3.8628	-3.8628	2.63E-13
	CV 1.0	-3.8628	-3.8628	-3.8628	**5.15E-15**
	CV 2.0	-3.8628	-3.8628	-3.8628	6.21E-13
	CV 3.0	-3.8628	-3.8628	-3.8628	9.37E-14
f_{23}	CS	-3.3224	-3.3224	-3.3224	4.04E-07
	CV 1.0	-3.3224	-3.3224	-3.3224	**2.48E-08**
	CV 2.0	-3.3224	-3.3224	-3.3224	2.08E-07
	CV 3.0	-3.3224	-3.3224	-3.3224	9.37E-08
f_{24}	CS	-1.0316	-1.0316	-1.0316	7.78E-16
	CV 1.0	-1.0316	-1.0316	-1.0316	5.36E-15
	CV 2.0	-1.0316	-1.0316	-1.0316	5.43E-15
	CV 3.0	-1.0316	-1.0316	-1.0316	**5.38E-16**

Table C.7 CV1.0 algorithm: Comparison for population size 60

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	13.689	49.352	31.7465	6.550
	CV 1.0	1.74E-106	2.90E-96	1.39E-97	5.23E-97
	CV 2.0	**3.12E-119**	**3.37E-96**	**7.41E-98**	**4.78E-97**
	CV 3.0	6.80E-98	1.52E-78	3.17E-80	2.15E-79
f_3	CS	0.0511	0.3154	0.1641	0.0528
	CV 1.0	**6.22E-05**	**0.0020**	**6.05E-04**	**4.83E-04**
	CV 2.0	1.46E-04	0.0047	0.0013	0.0010
	CV 3.0	1.02E-04	0.0197	0.0031	0.0030
f_4	CS	0.0982	0.2878	0.1678	0.0399
	CV 1.0	2.03E-110	7.53E-96	2.39E-97	1.20E-96
	CV 2.0	**1.68E-119**	**7.92E-99**	**2.90E-100**	**1.36E-99**
	CV 3.0	2.70E-105	7.91E-83	2.56E-84	1.22E-83
f_6	CS	3.26E+04	1.12E+05	6.11E+04	1.59E+04
	CV 1.0	3.21E-107	1.27E-91	3.46E-93	1.85E-92
	CV 2.0	**6.13E-113**	**9.61E-95**	**2.23E-96**	**1.36E-95**
	CV 3.0	1.00E-92	7.76E-74	1.57E-75	1.09E-74
f_9	CS	9.3244	15.1384	12.0733	1.3251
	CV 1.0	8.88E-16	4.44E-15	**1.38E-15**	**1.24E-15**
	CV 2.0	8.88E-16	4.44E-15	1.74E-15	1.53E-15
	CV 3.0	8.88E-16	4.44E-15	2.23E-15	1.74E-15
f_{13}	CS	3.94E+02	9.90E+02	6.55E+02	1.51E+02
	CV 1.0	3.33E-109	8.23E-94	1.76E-95	1.16E-94
	CV 2.0	**2.12E-113**	**1.00E-96**	**2.77E-98**	**1.43E-97**
	CV 3.0	5.56E-96	1.42E-77	4.40E-79	2.25E-78
f_{15}	CS	3.5339	6.5283	4.8091	0.7260
	CV 1.0	**2.78E-04**	**0.0066**	**0.0015**	**0.0012**
	CV 2.0	0.0027	0.0140	0.0069	0.0026
	CV 3.0	0.0027	0.3207	0.0153	0.0441
f_{16}	CS	6.4102	20.170	12.317	2.7595
	CV 1.0	**0.0148**	**0.3169**	**0.0847**	**0.0665**
	CV 2.0	0.0657	0.8018	0.2480	0.1500
	CV 3.0	0.0570	0.3539	0.1850	0.0722
f_{18}	CS	15.706	20.274	17.908	0.9880
	CV 1.0	**0**	**0**	**0**	**0**
	CV 2.0	**0**	**0**	**0**	**0**
	CV 3.0	**0**	**0**	**0**	**0**
f_{20}	CS	3.0000	3.0000	3.0000	4.23E-14
	CV 1.0	3.0000	3.0000	3.0000	**3.51E-15**
	CV 2.0	3.0000	3.0000	3.0000	1.08E-14
	CV 3.0	3.0000	3.0000	3.0000	1.72E-14
f_{21}	CS	7.59E-15	1.24E-10	7.36E-12	2.01E-11
	CV 1.0	**3.36E-23**	**1.00E-18**	**5.61E-20**	**1.50E-19**
	CV 2.0	2.83E-19	2.80E-14	2.58E-15	6.01E-15
	CV 3.0	4.68E-17	1.06E-12	9.10E-14	1.95E-13
f_{22}	CS	-3.8628	-3.8628	-3.8628	4.45E-13
	CV 1.0	-3.8628	-3.8628	-3.8628	**1.42E-14**
	CV 2.0	-3.8628	-3.8628	-3.8628	1.41E-12
	CV 3.0	-3.8628	-3.8628	-3.8628	1.33E-13
f_{23}	CS	-3.3224	-3.3224	-3.3224	1.25E-07
	CV 1.0	-3.3224	-3.3224	-3.3224	**2.16E-08**
	CV 2.0	-3.3224	-3.3224	-3.3224	3.78E-07
	CV 3.0	-3.3224	-3.3224	-3.3224	4.00E-08
f_{24}	CS	-1.0316	-1.0316	-1.0316	**5.73E-16**
	CV 1.0	-1.0316	-1.0316	-1.0316	7.71E-15
	CV 2.0	-1.0316	-1.0316	-1.0316	1.22E-14
	CV 3.0	-1.0316	-1.0316	-1.0316	6.65E-16

Table C.8 CV1.0 algorithm: Comparison for population size 80

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	30.8498	69.9445	45.1331	9.1596
	CV 1.0	1.68E-115	3.86E-100	8.69E-102	5.48E-101
	CV 2.0	**7.20E-120**	**6.55E-102**	**1.31E-103**	**9.27E-103**
	CV 3.0	1.47E-104	7.50E-86	1.59E-87	1.06E-86
f_3	CS	2.32E-15	1.70E-11	2.00E-12	3.71E-12
	CV 1.0	**5.81E-22**	**1.38E-18**	**8.90E-20**	**2.36E-19**
	CV 2.0	3.63E-18	5.62E-14	4.13E-15	1.09E-14
	CV 3.0	3.01E-16	4.02E-13	6.14E-14	7.82E-14
f_4	CS	0.1453	0.3462	0.2206	0.0486
	CV 1.0	6.18E-117	1.35E-102	7.80E-104	3.10E-103
	CV 2.0	**2.02E-121**	**1.02E-105**	**2.22E-107**	**1.45E-106**
	CV 3.0	9.20E-106	8.94E-85	1.78E-86	1.26E-85
f_6	CS	4.25E+04	1.22E+05	7.63E+04	1.71E+04
	CV 1.0	3.08E-112	8.97E-96	2.05E-97	1.27E-96
	CV 2.0	**1.47E-117**	**1.63E-102**	**5.81E-104**	**2.67E-103**
	CV 3.0	1.98E-98	1.57E-82	3.36E-84	2.22E-83
f_9	CS	9.4238	15.1142	12.5551	1.1907
	CV 1.0	8.88E-16	4.44E-15	**1.17E-15**	**9.73E-16**
	CV 2.0	8.88E-16	4.44E-15	1.81E-15	1.57E-15
	CV 3.0	8.88E-16	4.44E-15	2.02E-15	1.67E-15
f_{13}	CS	4.90E+02	1.18E+03	8.36E+02	1.70E+02
	CV 1.0	2.39E-112	9.22E-100	2.12E-101	1.30E-100
	CV 2.0	**1.19E-116**	**1.17E-102**	**2.48E-104**	**1.66E-103**
	CV 3.0	8.23E-100	1.87E-85	5.13E-87	2.73E-86
f_{15}	CS	3.8544	6.4161	5.3254	0.6618
	CV 1.0	**2.51E-04**	**0.0040**	**7.98E-04**	**6.65E-04**
	CV 2.0	0.0023	0.0128	0.0059	0.0021
	CV 3.0	0.0029	0.0175	0.0067	0.0033
f_{16}	CS	10.3038	24.152	16.0158	3.5767
	CV 1.0	**0.0035**	**0.1414**	**0.0515**	**0.0411**
	CV 2.0	0.0275	0.4431	0.2217	0.0901
	CV 3.0	0.0653	0.3807	0.1733	0.0768
f_{18}	CS	15.9489	21.3931	18.6821	1.1379
	CV 1.0	**0**	**0**	**0**	**0**
	CV 2.0	**0**	**0**	**0**	**0**
	CV 3.0	**0**	**0**	**0**	**0**
f_{20}	CS	3.0000	3.0000	3.0000	4.62E-14
	CV 1.0	3.0000	3.0000	3.0000	**4.10E-15**
	CV 2.0	3.0000	3.0000	3.0000	1.16E-14
	CV 3.0	3.0000	3.0000	3.0000	1.77E-14
f_{21}	CS	2.32E-15	1.70E-11	2.00E-12	3.71E-12
	CV 1.0	**5.81E-22**	**1.38E-18**	**8.90E-20**	**2.36E-19**
	CV 2.0	3.63E-18	5.62E-14	4.13E-15	1.09E-14
	CV 3.0	3.01E-16	4.02E-13	6.14E-14	7.82E-14
f_{22}	CS	-3.8628	-3.8628	-3.8628	2.45E-13
	CV 1.0	-3.8628	-3.8628	-3.8628	**1.40E-14**
	CV 2.0	-3.8628	-3.8628	-3.8628	7.13E-13
	CV 3.0	-3.8628	-3.8628	-3.8628	7.88E-14
f_{23}	CS	-3.3224	-3.3224	-3.3224	1.46E-07
	CV 1.0	-3.3224	-3.3224	-3.3224	**2.94E-08**
	CV 2.0	-3.3224	-3.3224	-3.3224	4.50E-07
	CV 3.0	-3.3224	-3.3224	-3.3224	5.31E-08
f_{24}	CS	-1.0316	-1.0316	-1.0316	7.80E-16
	CV 1.0	-1.0316	-1.0316	-1.0316	4.34E-15
	CV 2.0	-1.0316	-1.0316	-1.0316	5.07E-15
	CV 3.0	-1.0316	-1.0316	-1.0316	**5.53E-16**

Table C.9 CV1.0 algorithm: Comparison for population size 100

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	CS	30.7174	80.6873	54.3018	9.7061
	CV 1.0	1.13E-118	1.34E-104	3.21E-106	1.93E-105
	CV 2.0	**1.27E-123**	**2.29E-106**	**4.59E-108**	**3.25E-107**
	CV 3.0	6.10E-109	6.55E-91	1.63E-92	9.49E-92
f_3	CS	0.0802	0.3007	0.2100	0.0455
	CV 1.0	9.40E-05	**0.0017**	**4.92E-04**	**3.63E-04**
	CV 2.0	4.79E-05	0.0035	9.50E-04	8.45E-04
	CV 3.0	**4.63E-05**	0.0035	0.0013	8.37E-04
f_4	CS	0.1381	0.3843	0.2466	0.0543
	CV 1.0	2.01E-122	7.18E-107	2.01E-108	1.03E-107
	CV 2.0	**2.64E-128**	**2.57E-111**	**5.24E-113**	**3.64E-112**
	CV 3.0	3.20E-108	1.54E-93	3.32E-95	2.19E-94
f_6	CS	4.59E+04	1.14E+05	8.00E+04	1.61E+04
	CV 1.0	2.84E-115	2.34E-99	4.70E-101	3.32E-100
	CV 2.0	**1.93E-125**	**1.19E-106**	**4.13E-108**	**2.02E-107**
	CV 3.0	2.95E-102	5.72E-88	1.29E-89	8.10E-89
f_9	CS	10.6530	15.8066	12.9135	1.1769
	CV 1.0	8.88E-16	4.44E-15	**1.24E-15**	**1.07E-15**
	CV 2.0	8.88E-16	4.44E-15	1.38E-15	1.24E-15
	CV 3.0	8.88E-16	4.44E-15	2.09E-15	1.70E-15
f_{13}	CS	6.86E+02	1.64E+03	1.05E+03	2.12E+02
	CV 1.0	1.49E-117	2.02E-103	4.50E-105	2.86E-104
	CV 2.0	**4.54E-122**	**1.33E-109**	**3.30E-111**	**1.89E-110**
	CV 3.0	6.39E-105	2.95E-89	6.02E-91	4.18E-90
f_{15}	CS	4.2395	7.4123	5.9287	0.6526
	CV 1.0	**7.67E-05**	**0.0019**	**4.41E-04**	**3.55E-04**
	CV 2.0	0.0023	0.1228	0.0124	0.0271
	CV 3.0	0.0022	2.6276	0.0659	0.3715
f_{16}	CS	12.1557	24.3941	18.4708	2.7416
	CV 1.0	**0.0011**	**0.0911**	**0.0306**	**0.0227**
	CV 2.0	0.0734	0.5081	0.2131	0.1073
	CV 3.0	0.0449	0.4640	0.1860	0.0819
f_{18}	CS	17.0423	21.4807	19.1492	0.9542
	CV 1.0	**0**	**0**	**0**	**0**
	CV 2.0	**0**	**0**	**0**	**0**
	CV 3.0	**0**	**0**	**0**	**0**
f_{20}	CS	3.0000	3.0000	3.0000	1.98E-14
	CV 1.0	3.0000	3.0000	3.0000	**3.82E-15**
	CV 2.0	3.0000	3.0000	3.0000	9.71E-15
	CV 3.0	3.0000	3.0000	3.0000	1.06E-14
f_{21}	CS	9.52E-15	5.66E-11	2.28E-12	8.23E-12
	CV 1.0	**4.83E-23**	**2.08E-18**	**1.25E-19**	**3.21E-19**
	CV 2.0	1.35E-18	5.30E-14	1.96E-15	7.55E-15
	CV 3.0	1.72E-16	3.50E-12	1.47E-13	5.06E-13
f_{22}	CS	-3.8628	-3.8628	-3.8628	1.34E-13
	CV 1.0	-3.8628	-3.8628	-3.8628	**2.26E-14**
	CV 2.0	-3.8628	-3.8628	-3.8628	9.42E-13
	CV 3.0	-3.8628	-3.8628	-3.8628	1.47E-13
f_{23}	CS	-3.3224	-3.3224	-3.3224	9.04E-08
	CV 1.0	-3.3224	-3.3224	-3.3224	**1.71E-08**
	CV 2.0	-3.3224	-3.3224	-3.3224	3.90E-07
	CV 3.0	-3.3224	-3.3224	-3.3224	3.23E-08
f_{24}	CS	-1.0316	-1.0316	-1.0316	**7.43E-16**
	CV 1.0	-1.0316	-1.0316	-1.0316	2.65E-15
	CV 2.0	-1.0316	-1.0316	-1.0316	7.67E-15
	CV 3.0	-1.0316	-1.0316	-1.0316	7.94E-16

Table C.10 CV1.0 algorithm: Comparison with respect to other algorithms

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	BA	9.60E+03	7.00E+04	2.63E+04	1.16E+04
	CV 1.0	**3.04E-86**	**5.01E-71**	**1.86E-72**	**8.03E-72**
	DE	2.14E+04	8.93E+04	6.84E+04	1.24E+04
	FA	0.00539	0.0370	0.01622	0.0072
	FPA	1.74E+03	6.35E+03	3.28E+03	9.91E+02
	GWO	8.23E-28	1.80E-25	3.56E-26	4.22E-26
f_2	BA	11392	8.57E+03	48990	2.61E+04
	CV 1.0	**0**	**0**	**0**	**0**
	DE	18459	1.47E+04	90398	6.48E+04
	FA	0	0.2740	1	0.0800
	FPA	968	1.20E+03	6656	3.36E+03
	GWO	**0**	**0**	**0**	**0**
f_3	BA	4.9646	37.256	15.078	7.4888
	CV 1.0	**7.03E-05**	0.0090	0.0017	0.0018
	DE	22.538	1.97E+02	1.32E+02	33.493
	FA	0.0175	0.1839	0.0716	0.0353
	FPA	0.1363	1.5017	0.5290	0.2563
	GWO	2.73E-04	**0.0048**	**0.0015**	**0.0010**
f_5	BA	1.74E-07	3.81E-06	8.03E-07	5.97E-07
	CV 1.0	**7.36E-183**	**2.47E-156**	**5.17E-158**	**3.50E-157**
	DE	0.0946	1.5088	0.8967	0.2807
	FA	3.17E-07	4.87E-06	2.08E-06	1.38E-06
	FPA	5.33E-08	6.62E-05	1.12E-05	1.46E-05
	GWO	1.81E-101	4.36E-86	2.04E-87	8.49E-87
f_7	BA	6.08E-13	0.4052	0.1231	0.1027
	CV 1.0	**0**	**0**	**0**	**0**
	DE	4.80E-05	0.4094	0.1064	0.0975
	FA	3.75E-12	9.09E-10	2.23E-10	2.37E-10
	FPA	2.31E-10	1.40E-06	1.04E-07	2.10E-07
	GWO	**0**	**0**	**0**	**0**
f_9	BA	14.7796	19.3891	16.9588	1.0246
	CV 1.0	**8.88E-16**	**4.44E-15**	**1.38E-15**	**1.24E-15**
	DE	17.9096	20.9201	20.1551	0.8220
	FA	0.0260	0.63806	0.08979	0.1138
	FPA	8.5186	13.3582	11.6734	1.2049
	GWO	1.00E-13	2.95E-13	1.66E-13	4.60E-14
f_{10}	BA	1.09E+02	5.45E+03	1.33E+03	9.71E+02
	CV 1.0	**1.15E-72**	**2.85E-07**	**1.33E-08**	**5.48E-08**
	DE	7.33E+02	2.63E+04	4.91E+03	4.08E+03
	FA	0.42377	41.0753	10.3029	9.0744
	FPA	42.0220	5.07E+02	1.67E+02	1.16E+02
	GWO	1.96E-07	1.45E-04	2.18E-05	2.81E-05
f_{11}	BA	2.15E+04	7.96E+05	2.21E+05	1.52E+05
	CV 1.0	0.6666	0.6667	0.6667	**1.02E-05**
	DE	1.16E+06	2.83E+06	2.01E+06	3.80E+05
	FA	0.7208	1.27E+02	13.7882	23.5559
	FPA	1.17E+03	2.33E+04	5.10E+03	3.93E+03
	GWO	0.6666	0.6667	0.6666	1.09E-05
f_{12}	BA	65.3099	2.14E+05	9.74E+03,	3.48E+04
	CV 1.0	8.9617	80.9374	29.5431	17.2651
	DE	2.40E+03	9.46E+04	2.02E+04	2.05E+04
	FA	53.8694	2.67E+02	1.38E+02	52.0701
	FPA	1.85E+02	1.86E+03	6.48E+02	3.28E+02
	GWO	**3.22E-06**	**0.0355**	**0.0030**	**0.0069**
f_{13}	BA	1.00E+05	7.98E+05	3.12E+05	1.63E+05
	CV 1.0	**3.42E-84**	**4.27E-70**	**1.32E-71**	**6.66E-71**
	DE	5.01E+05	1.96E+06	1.61E+06	3.26E+05
	FA	7.8610	79.5311	28.8670	17.0079
	FPA	1.98E+04	1.39E+05	7.67E+04	2.75E+04
	GWO	1.09E-26	2.56E-24	5.87E-25	6.04E-25

Table C.10 CV1.0 algorithm: (Contd.) Comparison with respect to other algorithms

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_{15}	BA	3.30E+06	2.84E+08	6.37E+07	6.33E+07
	CV 1.0	7.91E-03	8.1687	1.03270	2.0149
	DE	2.51E+08	1.00E+09	6.35E+08	1.66E+08
	FA	**1.17E-04**	**0.2940**	**0.0152**	0.0493
	FPA	11.493	3.19E+04	2.53E+03	6.13E+03
	GWO	0.0133	0.1675	0.0635	0.0343
f_{16}	BA	2.33E+07	4.43E+08	1.78E+08	1.00E+08
	CV 1.0	0.1722	0.8710	0.4055	0.1591
	DE	7.27E+08	1.56E+09	1.13E+09	2.31E+08
	FA	**0.0011**	**0.0237**	**0.0055**	**0.0043**
	FPA	1.38E+03	1.60E+07	6.52E+05	2.24E+06
	GWO	0.3258	1.5953	0.8001	0.2882
f_{19}	BA	0.0028	1.80E+04	1.12E+03	2.94E+03
	CV 1.0	**2.31E-12**	**1.51E-08**	**1.01E-09**	**2.36E-09**
	DE	1.73E-06	53.479	3.0651	9.0782
	FA	6.34E-06	7.7380	0.7251	1.5308
	FPA	0.0288	4.5457	1.1077	1.0511
	GWO	6.76E-04	7.8248	2.7033	2.8619
f_{20}	BA	3.0000	84.000	9.4800	20.086
	CV 1.0	3.0000	3.0000	3.0000	**3.14E-15**
	DE	3.0000	3.0544	3.0022	0.0094
	FA	3.0000	3.0000	3.0000	3.71E-08
	FPA	3.0000	3.0000	3.0000	1.53E-09
	GWO	3.0000	84.000	4.6200	11.4551
f_{21}	BA	1.94E-11	0.8670	0.1935	0.2952
	CV 1.0	2.10E-27	**2.34E-21**	**1.32E-22**	**4.02E-22**
	DE	**0**	0.5635	0.0113	0.0796
	FA	3.79E-12	2.64E-09	8.17E-10	7.05E-10
	FPA	1.55E-12	7.62E-05	2.05E-06	1.08E-05
	GWO	3.10E-09	0.4928	0.0195	0.0968
f_{22}	BA	-3.8628	-5.45E-14	-1.4366	1.5536
	CV 1.0	-3.8628	-3.8627	-3.8627	**2.77E-15**
	DE	-3.8628	-3.8015	-3.8614	0.0086
	FA	-3.8628	-3.8627	-3.8627	7.79E-08
	FPA	-3.8628	-1.1816	-3.7655	0.3939
	GWO	-3.8628	-3.8482	-3.8596	0.0048
f_{23}	BA	-3.3224	-3.1996	-3.2793	0.0579
	CV 1.0	-3.3224	**-3.3223**	**-3.3223**	**3.80E-08**
	DE	-3.3224	-2.0863	-3.0937	0.2528
	FA	-3.3224	-3.1671	-3.2984	0.0519
	FPA	-3.3224	-3.2024	-3.2789	0.0355
	GWO	-3.3224	-3.0106	-3.2520	0.0863
f_{24}	BA	-1.0316	-0.2154	-0.9336	0.2679
	CV 1.0	-1.0316	-1.0316	-1.0316	**7.33E-16**
	DE	-1.0316	-1.0288	-1.0315	4.39E-04
	FA	-1.0316	-1.0316	-1.0316	3.53E-09
	FPA	-1.0316	-1.0316	-1.0316	1.58E-07
	GWO	-1.0316	-1.0316	-1.0316	2.04E-08
f_{25}	BA	-7.7786	-3.2439	-5.1317	1.0332
	CV 1.0	-9.0488	-6.8746	-7.7861	**0.4564**
	DE	**-9.2632**	-6.4068	**-8.1956**	0.6071
	FA	-9.2250	-6.6216	-7.8223	0.6336
	FPA	-6.7349	-5.0073	-5.7008	0.4634
	GWO	-8.8881	-4.0628	-7.2119	1.1269
f_{26}	BA	-1.0000	-0.9779	-8.11E-05	0.1419
	CV 1.0	-1.0000	-1.0000	-1.0000	**0**
	DE	-1.0000	-0.9985	-0.7911	0.0295
	FA	-1.0000	-1.0000	-1.0000	3.44E-09
	FPA	-1.0000	-1.0000	-1.0000	2.49E-10
	GWO	-1.0000	-1.0000	-1.0000	5.86E-07

Table C.11 CV1.0 algorithm: Comparison for $D = 50$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	BA	1.98E+04	8.38E+04	4.59E+04	1.43E+04
	CV 1.0	**2.08E-80**	**6.18E-70**	**1.78E-71**	**9.18E-71**
	DE	9.62E+04	1.47E+05	1.27E+05	1.19E+04
	FA	0.0497	0.1602	0.1029	0.0269
	FPA	3.06E+03	1.14E+04	7.38E+03	1.93E+03
	GWO	7.48E-20	1.81E-17	1.63E-18	2.79E-18
f_3	BA	3.0792	59.841	16.2428	10.1490
	CV 1.0	**1.80E-04**	0.0059	0.00185	0.00134
	DE	59.252	1.85E+02	1.37E+02	29.6543
	FA	0.0125	0.2212	0.06857	0.04600
	FPA	0.2264	1.4726	0.54233	0.26164
	GWO	2.17E-04	**0.0040**	**0.00163**	**8.40E-04**
f_4	BA	45.2618	1.49E+03	2.82E+02	2.64E+02
	CV 1.0	**2.21E-82**	**4.58E-69**	9.27E-71	**6.48E-70**
	DE	1.47E+02	1.72E+03	4.54E+02	3.36E+02
	FA	49.5183	1.36E+02	87.0246	21.9144
	FPA	15.4087	71.542	38.3178	13.2628
	GWO	1.73E-22	3.15E-20	5.61E-21	6.56E-21
f_6	BA	1.21E+09	3.06E+09	2.31E+08	7.33E+08
	CV 1.0	**3.60E-69**	**1.79E-67**	**6.91E-82**	**2.53E-68**
	DE	8.15E+08	2.33E+09	8.94E+07	5.50E+08
	FA	9.12E+06	2.86E+07	1.54E+06	5.64E+06
	FPA	2.27E+07	7.24E+07	6.02E+06	1.40E+07
	GWO	1.52E-22	1.99E-21	1.84E-24	2.90E-22
f_9	BA	16.8778	20.0203	18.6442	0.6944
	CV 1.0	**8.88E-16**	**4.44E-15**	**1.95E-15**	**1.64E-15**
	DE	21.1209	21.2046	21.1595	0.0149
	FA	18.2671	19.2910	18.7513	0.2223
	FPA	13.4733	14.9931	14.1148	0.3224
	GWO	0.17097	2.08976	0.59249	0.4448
f_{10}	BA	4.69E+02	1.09E+04	3.60E+03	1.91E+03
	CV 1.0	**1.29E-79**	**2.68E-09**	**5.44E-11**	**3.79E-10**
	DE	4.99E+03	5.54E+04	2.88E+04	1.44E+04
	FA	9.1504	77.3188	31.5871	16.7779
	FPA	2.67E+02	1.16E+03	5.98E+02	2.25E+02
	GWO	2.51E-06	1.33E-04	3.47E-05	3.19E-05
f_{11}	BA	1.64E+05	3.05E+06	7.11E+05	4.84E+05
	CV 1.0	**0.6666**	**0.6668**	**0.6666**	2.75E-05
	DE	3.74E+06	8.13E+06	6.09E+06	9.16E+05
	FA	2.8738	2.69E+02	31.8000	45.0463
	FPA	7.81E+03	8.05E+04	2.98E+04	1.47E+04
	GWO	0.6666	0.6668	0.6667	**2.14E-05**
f_{13}	BA	3.11E+05	1.27E+06	7.22E+05	2.63E+05
	CV 1.0	**3.48E-78**	**6.32E-66**	**1.31E-67**	**8.94E-67**
	DE	2.56E+06	3.72E+06	3.12E+06	2.46E+05
	FA	2.03E+02	1.17E+03	5.23E+02	1.64E+02
	FPA	8.48E+04	2.82E+05	1.91E+05	3.88E+04
	GWO	8.51E-19	1.62E-16	2.36E-17	2.61E-17
f_{14}	BA	0.5364	1.19E+02	17.670	24.687
	CV 1.0	**0**	**0**	**0**	**0**
	DE	0	4.9221	0.5267	1.0669
	FA	5.96E-08	7.75E-06	2.19E-06	1.84E-06
	FPA	3.13E-09	1.64E-04	1.65E-05	3.46E-05
	GWO	0	0.1456	0.0116	0.0399
f_{17}	BA	9.51E-08	3.0445	2.6470	9.9495
	CV 1.0	**0**	**0**	**0**	**0**
	DE	5.07E-11	2.0863	1.7087	9.3593
	FA	5.38E-10	1.47E-07	2.09E-07	1.21E-06
	FPA	1.74E-06	0.0012	0.0024	0.0139
	GWO	**0**	**0**	**0**	**0**

Table C.12 CV1.0 algorithm: Comparison for $D = 100$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	BA	5.83E+04	2.33E+05	1.02E+05	3.54E+04
	CV 1.0	**1.06E-76**	**2.50E-65**	**7.78E-67**	**3.85E-66**
	DE	2.47E+05	3.03E+05	2.79E+05	1.28E+04
	FA	1.0225	95.4161	17.6251	22.1848
	FPA	1.41E+04	2.44E+04	1.78E+04	2.24E+03
	GWO	9.15E-13	3.05E-11	1.23E-11	7.72E-12
f_3	BA	4.5814	42.6601	14.1015	6.9912
	CV 1.0	**2.64E-04**	0.0095	0.0021	0.00172
	DE	47.9976	1.79E+02	1.18E+02	29.6255
	FA	0.0117	0.19693	0.05869	0.0419
	FPA	0.1569	1.4745	0.5697	0.2892
	GWO	3.65E-04	**0.0045**	**0.0015**	**8.38E-04**
f_5	BA	1.24E-07	6.97E-06	1.41E-06	1.26E-06
	CV 1.0	**3.49E-180**	**3.67E-145**	**7.34E-147**	**5.19E-146**
	DE	1.0298	2.5205	1.6995	0.38684
	FA	5.11E-08	3.34E-05	3.29E-06	5.02E-06
	FPA	5.25E-08	2.12E-04	2.83E-05	3.95E-05
	GWO	1.70E-67	6.54E-35	2.17E-36	1.10E-35
f_8	BA	-5.78E+227	-1.80E+54	-1.16E+226	9.63E+139
	CV 1.0	**-3.00E+04**	-1.60E+04	-1.89E+04	2.78E+03
	DE	-4.47E+125	-1.36E+114	-1.38E+124	7.02E+124
	FA	-2.75E+04	-1.70E+04	-2.08E+04	2.27E+03
	FPA	-5.48E+42	-1.62E+32	-1.10E+41	7.76E+41
	GWO	-1.36E+33	-8.41E+22	-2.87E+31	1.92E+32
f_{10}	BA	3.04E+03	3.81E+04	1.30E+04	8.01E+03
	CV 1.0	**8.66E-76**	**5.58E-13**	**1.12E-14**	**7.90E-14**
	DE	8.11E+04	1.49E+05	1.14E+05	1.84E+04
	FA	71.03779	2.82E+02	1.62E+02	53.60089
	FPA	9.57E+02	3.44E+03	2.05E+03	5.71E+02
	GWO	1.20E-05	6.62E-04	1.10E-04	1.08E-04
f_{11}	BA	1.04E+06	1.55E+07	3.63E+06	2.58E+06
	CV 1.0	**0.66666**	**0.66677**	**0.66670**	**2.74E-05**
	DE	2.32E+07	3.59E+07	2.93E+07	2.35E+06
	FA	2.40E+02	4.56E+03	1.12E+03	8.71E+02
	FPA	6.87E+04	4.55E+05	1.84E+05	8.26E+04
	GWO	0.66666	1.0000	0.6935	0.09129
f_{12}	BA	5.27E+03	3.21E+10	6.58E+08	4.53E+09
	CV 1.0	**2.52E+02**	**1.16E+03**	**6.52E+02**	**2.20E+02**
	DE	3.22E+04	8.70E+06	1.62E+06	2.01E+06
	FA	1.21E+03	2.64E+03	1.71E+03	3.34E+02
	FPA	4.10E+03	8.76E+04	1.84E+04	1.48E+04
	GWO	7.71E+02	6.84E+03	2.55E+03	1.31E+03
f_{15}	BA	3.37E+07	8.79E+08	2.73E+08	2.01E+08
	CV 1.0	**0.00942**	2.4829	**0.08385**	**0.347292**
	DE	2.36E+09	3.43E+09	2.93E+09	2.75E+08
	FA	4.02147	24.19976	10.5855	4.18550
	FPA	2.22E+04	4.57E+06	4.64E+05	7.31E+05
	GWO	0.05462	**0.86295**	0.23091	0.181021
f_{16}	BA	1.59E+08	2.23E+09	6.87E+08	4.22E+08
	CV 1.0	**0.24185**	**1.23187**	**0.70997**	**0.23383**
	DE	4.67E+09	6.42E+09	5.56E+09	4.58E+08
	FA	12.4119	72.6320	44.73665	11.25752
	FPA	1.33E+06	2.24E+07	8.23E+06	4.92E+06
	GWO	1.245858	2.54473	1.899744	0.30828
f_{17}	BA	1.76E-08	17.9092	3.40275	4.13951
	CV 1.0	**0**	**0**	**0**	**0**
	DE	2.13E-11	8.90626	2.16313	1.78390
	FA	6.82E-10	0.9949	0.0198	0.140708
	FPA	2.73E-08	0.0101	0.00130	0.00228
	GWO	**0**	**0**	**0**	**0**

Table C.13 CV1.0 algorithm: Comparison for $D = 500$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	BA	2.90E+05	1.41E+06	5.17E+05	1.81E+05
	CV 1.0	**1.08E-71**	**4.56E-58**	**9.22E-60**	**6.44E-59**
	DE	1.47E+06	1.61E+06	1.54E+06	3.51E+04
	FA	1.57E+05	2.81E+05	2.27E+05	2.80E+04
	FPA	8.87E+04	1.34E+05	1.10E+05	1.05E+04
	GWO	0.0052	0.0709	0.01967	0.0113
f_3	BA	5.0394	53.699	16.3432	9.961578
	CV 1.0	**9.88E-05**	0.0099	0.0018	0.001821
	DE	42.7086	1.56E+02	1.16E+02	25.24161
	FA	0.0194	0.1748	0.0790	0.037667
	FPA	0.0928	1.2723	0.5850	0.290977
	GWO	2.26E-04	**0.0036**	**0.0014**	**8.26E-04**
f_5	BA	7.41E-08	7.76E-06	1.18E-06	1.24E-06
	CV 1.0	**1.97E-177**	**2.64E-148**	**5.33E-150**	**3.74E-149**
	DE	1.4647127	4.0348332	2.8096894	0.562728
	FA	9.44E-06	1.2658304	0.4906256	0.378969
	FPA	1.18E-06	0.0041075	1.85E-04	6.04E-04
	GWO	1.05E-09	1.59E-04	1.85E-05	3.63E-05
f_6	BA	3.36E+08	2.73E+09	1.22E+09	6.10E+08
	CV 1.0	**2.37E-81**	**1.16E-68**	**2.81E-70**	**1.65E-69**
	DE	4.95E+07	4.09E+09	1.54E+09	1.02E+09
	FA	2.71E+06	2.05E+07	8.37E+06	3.81E+06
	FPA	4.48E+06	1.01E+08	2.61E+07	1.94E+07
	GWO	5.06E-24	1.23E-21	1.52E-22	2.56E-22
f_{10}	BA	3.68E+04	5.21E+05	9.78E+04	7.30E+04
	CV 1.0	**5.34E-74**	**1.25E-10**	**2.50E-12**	**1.77E-11**
	DE	6.19E+05	8.70E+05	7.76E+05	5.54E+04
	FA	1.90E+04	4.77E+04	2.96E+04	5.40E+03
	FPA	1.13E+04	1.94E+04	1.45E+04	1.97E+03
	GWO	0.011109	1.56E+02	3.900867	22.02712
f_{11}	BA	2.52E+07	3.23E+08	1.16E+08	5.85E+07
	CV 1.0	**0.666669**	**0.666905**	**0.666733**	**5.27E-05**
	DE	8.10E+08	9.50E+08	8.81E+08	2.88E+07
	FA	1.86E+07	4.42E+07	3.02E+07	5.88E+06
	FPA	3.82E+06	1.06E+07	6.93E+06	1.52E+06
	GWO	17.64874	1.44E+03	1.53E+02	2.54E+02
f_{12}	BA	6.25E+04	1.60E+17	4.66E+15	2.37E+16
	CV 1.0	**4.26E+03**	2.54E+07	1.85E+06	5.71E+06
	DE	5.79E+06	8.78E+09	4.62E+08	1.28E+09
	FA	1.09E+04	1.82E+04	1.49E+04	1.82E+03
	FPA	2.85E+05	3.94E+08	9.92E+06	5.55E+07
	GWO	4.88E+04	**4.24E+05**	**1.19E+05**	**6.68E+04**
f_{13}	BA	6.07E+06	2.15E+07	1.14E+07	3.81E+06
	CV 1.0	**6.11E-71**	**1.21E-58**	**3.08E-60**	**1.76E-59**
	DE	3.59E+07	4.01E+07	3.84E+07	9.42E+05
	FA	4.50E+06	6.98E+06	5.66E+06	5.76E+05
	FPA	2.14E+06	3.60E+06	2.77E+06	3.21E+05
	GWO	0.213989	1.093949	0.504789	0.205431
f_{16}	BA	9.46E+08	1.29E+10	3.76E+09	2.32E+09
	CV 1.0	**0.684512**	7.48E+06	1.50E+05	1.06E+06
	DE	3.04E+10	3.47E+10	3.25E+10	1.00E+09
	FA	3.47E+08	1.20E+09	7.52E+08	2.03E+08
	FPA	4.79E+07	1.70E+08	8.46E+07	2.52E+07
	GWO	3.039398	**24.24042**	**8.33681**	**5.003857**
f_{17}	BA	1.63E-08	16.9142	3.16396	3.313272
	CV 1.0	**0**	**0**	**0**	**0**
	DE	**0**	5.6605	1.892614	1.214637
	FA	3.07E-09	0.9949	0.059697	0.238688
	FPA	1.11E-06	0.0356	0.002207	0.006333
	GWO	**0**	0	0	0

Table C.14 CV1.0 algorithm: Comparison for $D = 1000$

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	BA	5.18E+05	1.85E+06	9.44E+05	2.62E+05
	CV 1.0	**1.04E-68**	**1.62E-59**	**8.25E-61**	**2.85E-60**
	DE	3.02E+06	3.26E+06	3.16E+06	5.12E+04
	FA	7.46E+05	9.77E+05	8.61E+05	5.71E+04
	FPA	1.83E+05	2.81E+05	2.29E+05	2.22E+04
	GWO	3.6573	58.71844	15.544	9.02437
f_2	BA	598231	1694554	9.86E+05	2.49E+05
	CV 1.0	**0**	**0**	**0**	**0**
	DE	3013795	3256350	3.17E+06	5.07E+04
	FA	724475	1016982	8.46E+05	7.04E+04
	FPA	182382	299982	2.25E+05	2.22E+04
	GWO	89	263	1.53E+02	38.7688
f_4	BA	1.90E+03	1.04E+04	4.31E+03	1.63E+03
	CV 1.0	**1.36E-71**	**2.55E-60**	**6.07E-62**	**3.62E-61**
	DE	4.13E+03	1.25E+04	6.65E+03	1.83E+03
	FA	3.51E+03	3.99E+03	3.79E+03	1.23E+02
	FPA	6.23E+02	1.10E+03	8.26E+02	1.16E+02
	GWO	0.0184	1.8529	0.2616	0.409
f_6	BA	4.35E+08	2.67E+09	1.29E+09	5.23E+08
	CV 1.0	**2.89E-83**	**1.61E-68**	**6.05E-70**	**2.86E-69**
	DE	1.19E+08	4.51E+09	1.91E+09	1.18E+09
	FA	2.22E+06	2.52E+07	8.96E+06	5.19E+06
	FPA	5.43E+06	6.96E+07	2.04E+07	1.53E+07
	GWO	2.56E-24	1.52E-21	1.38E-22	2.30E-22
f_7	BA	0.0093	0.4844	0.1747	0.1387
	CV 1.0	**0**	**0**	**0**	**0**
	DE	9.70E-04	0.4545	0.1494	0.1208
	FA	2.21E-12	0.00312	6.25E-05	4.42E-04
	FPA	2.56E-11	1.76E-06	1.51E-07	3.05E-07
	GWO	**0**	**0**	**0**	**0**
f_8	BA	-5.79E+225	-1.37E+86	-1.16E+224	3.45E+159
	CV 1.0	-3.79E+05	-2.40E+05	-2.80E+05	1.99E+04
	DE	-2.09E+119	-2.65E+104	-6.94E+117	3.07E+118
	FA	**-1.38E+05**	**-9.21E+04**	-1.10E+05	1.26E+04
	FPA	-2.42E+43	-1.81E+32	-4.85E+41	3.43E+42
	GWO	-1.44E+31	-4.05E+04	-3.09E+29	2.03E+30
f_9	BA	17.2858	19.5875	18.4739	0.6199
	CV 1.0	**8.88E-16**	**4.44E-15**	**1.88E-15**	**1.61E-15**
	DE	21.0937	21.1929	21.1554	0.02027
	FA	18.2716	19.1248	18.7271	0.2175
	FPA	13.493	14.8069	14.087	0.3251
	GWO	0.1821	3.881	0.7645	0.7879
f_{10}	BA	6.60E+04	1.31E+06	2.42E+05	2.05E+05
	CV 1.0	**3.34E-72**	**1.38E-33**	**2.76E-35**	**1.95E-34**
	DE	1.55E+06	1.76E+06	1.65E+06	5.51E+04
	FA	1.05E+05	1.67E+05	1.35E+05	1.38E+04
	FPA	2.26E+04	3.82E+04	3.01E+04	3.12E+03
	GWO	24.3337	3.05E+03	5.68E+02	5.34E+02
f_{12}	BA	3.60E+05	3.20E+19	7.16E+18	2.04E+20
	CV 1.0	**1.26E+04**	4.34E+09	2.94E+09	1.00E+10
	DE	1.25E+07	1.79E+11	6.05E+10	1.07E+12
	FA	2.68E+04	**3.03E+03**	**3.31E+04**	**3.89E+04**
	FPA	7.34E+05	1.00E+08	2.80E+07	7.14E+08
	GWO	1.30E+05	1.26E+05	3.42E+05	9.44E+05
f_{16}	BA	1.57E+09	6.48E+10	8.16E+09	9.28E+09
	CV 1.0	**0.6345**	1.00E+10	6.03E+09	4.89E+09
	DE	6.31E+10	7.03E+10	6.76E+10	1.50E+09
	FA	3.75E+09	7.67E+09	5.38E+09	8.82E+08
	FPA	1.04E+08	3.40E+08	1.87E+08	5.96E+07
	GWO	1.48E+04	**6.72E+07**	**3.81E+06**	**9.76E+06**

Table C.15 CVnew algorithm: CEC 2017 Results for $10D$

Function	Best	Worst	Mean	Std dev.
F1	0.00E+00	4.12E-07	1.26E-06	5.92E-06
F2	0.00E+00	1.00E+10	5.88E+02	2.37E+09
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F5	3.58E+00	2.44E+01	1.38E+01	4.61E+00
F6	7.90E-03	1.24E-01	3.27E-02	2.31E-02
F7	1.35E+01	3.40E+01	2.47E+01	4.64E+00
F8	5.05E+00	2.18E+01	1.34E+01	4.33E+00
F9	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F11	2.52E-02	4.40E+00	1.68E+00	9.84E-01
F12	1.09E+01	8.46E+01	3.62E+01	1.90E+01
F13	1.54E+00	1.13E+01	7.26E+00	2.26E+00
F14	3.29E+00	1.22E+01	7.38E+00	2.26E+00
F15	1.60E-01	2.11E+00	8.30E-01	3.40E-01
F16	1.26E+00	1.44E+01	2.82E+00	1.84E+00
F17	1.25E+01	3.45E+01	2.35E+01	4.81E+00
F18	6.54E-01	4.76E+00	2.30E+00	1.08E+00
F19	1.01E+00	2.67E+00	1.66E+00	2.91E-01
F20	4.88E+00	3.53E+01	1.95E+01	6.89E+00
F21	1.07E-04	1.00E+02	9.60E+01	1.96E+01
F22	9.69E-08	1.01E+02	6.44E+01	3.72E+01
F23	1.61E-02	3.23E+02	3.07E+02	4.40E+01
F24	4.72E-02	2.03E+02	1.00E+02	3.51E+01
F25	1.00E+02	3.97E+02	1.86E+02	1.21E+02
F26	2.72E-07	3.00E+02	1.47E+02	9.80E+01
F27	1.73E+00	2.89E+02	3.81E+02	5.42E+01
F28	1.21E-06	3.00E+02	2.35E+02	1.09E+02
F29	1.95E+02	3.08E+02	2.58E+02	1.71E+01
F30	3.95E+02	3.96E+02	3.95E+02	3.52E-01

Table C.16 CVnew algorithm: CEC 2017 Results for 30D

Function	Best	Worst	Mean	Std dev.
F1	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F2	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F3	2.32E+01	5.43E+02	1.39E+02	9.30E+01
F4	3.20E-03	6.79E+01	1.43E+01	2.18E+01
F5	7.22E+01	1.75E+02	1.10E+02	2.51E+01
F6	2.52E+00	3.66E+01	1.96E+01	7.90E+00
F7	8.21E+01	1.59E+02	1.15E+02	2.06E+01
F8	7.13E+01	1.77E+02	1.16E+02	2.46E+01
F9	2.35E+02	5.16E+03	1.95E+03	1.04E+03
F10	2.43E+03	3.98E+03	3.31E+03	2.68E+02
F11	2.06E+01	9.24E+01	4.20E+01	1.68E+01
F13	3.89E+01	1.85E+02	8.20E+01	2.66E+01
F14	3.85E+01	6.50E+01	4.95E+01	6.68E+00
F15	1.59E+01	5.42E+01	3.41E+01	8.39E+00
F16	3.17E+02	1.10E+03	7.43E+02	1.81E+02
F17	7.65E+01	3.65E+02	1.93E+02	7.33E+01
F18	2.60E+01	5.89E+01	3.85E+01	7.36E+00
F19	1.40E+01	2.87E+01	2.11E+01	3.18E+00
F20	1.04E+02	4.96E+02	2.83E+02	8.51E+01
F21	1.00E+02	3.82E+02	2.50E+02	9.93E+01
F22	1.00E+02	4.18E+03	1.65E+03	1.81E+03
F23	2.24E+02	5.20E+02	4.34E+02	5.30E+01
F24	1.08E+02	5.67E+02	4.21E+02	1.61E+02
F25	3.83E+02	3.86E+02	3.83E+02	8.20E-01
F26	2.00E+02	3.04E+02	2.57E+02	4.88E+01
F27	4.88E+02	5.33E+02	5.12E+02	9.32E+00
F28	3.00E+02	4.03E+02	3.02E+02	1.44E+01
F29	5.97E+02	1.07E+03	8.04E+02	1.02E+02
F30	1.95E+03	2.16E+03	2.03E+03	5.34E+01

Table C.17 CVnew algorithm: CEC 2017 Results for $50D$

Function	Best	Worst	Mean	Std dev.
F1	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F2	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F3	1.79E+03	1.87E+04	8.71E+03	4.08E+03
F4	1.00E+00	3.25E+01	2.67E+01	5.92E+00
F5	1.55E+02	3.39E+02	2.39E+02	3.80E+01
F6	2.64E+01	5.42E+01	4.07E+01	8.14E+00
F7	1.67E+02	3.13E+02	2.22E+02	3.49E+01
F8	1.66E+02	3.73E+02	2.50E+02	4.51E+01
F9	5.13E+03	1.67E+04	1.06E+04	3.10E+03
F10	4.95E+03	6.83E+03	6.09E+03	3.55E+02
F11	8.28E+01	1.80E+02	1.18E+02	1.91E+01
F13	2.77E+04	1.00E+10	9.80E+09	1.40E+09
F14	1.67E+01	1.33E+02	3.98E+01	1.62E+01
F15	9.43E+01	1.94E+03	2.85E+02	3.54E+02
F16	1.04E+03	1.83E+03	1.44E+03	2.10E+02
F17	5.31E+01	1.49E+03	1.13E+02	1.92E+02
F18	8.79E+01	3.00E+02	1.51E+02	4.43E+01
F19	3.46E+01	8.25E+01	5.57E+01	1.10E+01
F20	1.73E+02	1.25E+03	2.81E+02	1.65E+02
F21	1.16E+02	5.68E+02	1.18E+02	8.77E+01
F22	6.15E+03	7.59E+03	5.77E+03	3.64E+02
F23	5.91E+02	8.00E+02	1.87E+02	5.11E+01
F24	1.92E+02	8.58E+02	3.25E+02	8.95E+01
F25	4.28E+02	5.19E+02	4.70E+02	2.26E+01
F26	1.00E+02	4.24E+03	1.16E+03	1.56E+03
F27	4.08E+02	7.85E+02	4.53E+02	7.17E+01
F28	4.58E+02	4.60E+02	4.58E+02	2.33E-01
F29	1.02E+03	1.79E+03	1.45E+03	1.68E+02
F30	5.79E+05	6.67E+05	6.02E+05	2.99E+04

Table C.18 CVnew algorithm: CEC2017 Results in comparison to other algorithms

	SaDE	JADE	SHADE	MVMO	CV1.0	CV_{new}
$F1$	1.21E+03 (1.97E+03) +	5.23E-14 (2.51E-14) +	0.00E+00 (0.00E-00) +	1.33E-05 (5.60E-06) +	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E-00)
$F2$	9.27E+01 (4.12E+01) +	1.31E+13 (8.53E+13) -	1.08E+12 (4.39E+12) -	1.80E+17 (1.27E+18) -	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
$F3$	2.71E+02 (8.28E+02) -	1.77E+04 (3.70E+04) +	0.00E+00 (0.00E+00) -	5.30E-07 (1.09E-07) +	1.95E+04 (6.27E+03) -	8.71E+03 (4.08E+03)
$F4$	8.92E+01 (4.21E+01) -	4.96E+01 (4.71E+01) -	5.68E+01 (8.80E+00) -	3.58E+01 (3.66E+01) -	1.16E+02 (6.27E+03) -	2.67E+01 (5.92E+00)
$F5$	9.23E+01 (1.86E+01) -	5.42E+01 (8.80E+00) +	3.28E+01 (5.03E+00) +	8.07E+01 (1.64E+01) +	3.41E+02 (8.02E+01) -	2.39E+02 (3.80E+01)
$F6$	7.43E-03 (2.35E-02) +	1.44E-13 (9.11E-14) +	8.38E-04 (1.01E-03) +	5.43E-03 (3.30E-03) -	4.85E+01 4.85E+01 -	4.07E+01 (8.14E+00)
$F7$	1.40E+02 (1.97E+01) +	1.01E+02 (6.48E+00) +	8.09E+01 (3.78E+00) +	1.23E+02 (1.27E+01) +	2.74E+02 (7.29E+01) +	2.22E+02 (3.49E+01)
$F8$	9.42E+01 (1.77E+01) +	5.52E+01 (7.76E+00) +	3.23E+01 (3.82E+00) +	7.59E+01 (1.61E+01) +	3.29E+02 (7.29E+01) +	2.50E+02 (4.51E+01)
$F9$	4.83E+01 (6.29E+01) +	1.17E+00 (1.31E+00) +	1.11E+00 (9.37E-01) +	7.38E+00 (5.77E+00) +	1.00E+04 (2.90E+03) +	1.06E+04 (3.10E+03)
$F10$	6.60E+03 (1.63E+03) -	3.75E+03 (2.54E+02) +	3.34E+03 (2.94E+02) +	3.49E+03 (4.31E+02) +	7.10E+03 (5.34E+02) -	6.09E+03 (3.55E+02)
$F11$	1.09E+02 (3.54E+01) -	1.36E+02 (3.39E+01) -	1.20E+02 (2.93E+01) -	4.74E+01 (8.72E+00) +	1.66E+02 (3.38E+01) -	1.18E+02 (1.91E+01)
$F12$	1.11E+05 (6.20E+04) +	5.14E+03 (3.32E+03) +	5.13E+03 (2.87E+03) +	1.29E+03 (2.79E+02) +	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
$F13$	1.21E+03 (1.45E+03) +	3.03E+02 (2.69E+02) +	2.65E+02 (1.49E+02) +	4.37E+01 (1.76E+01) +	1.00E+10 (0.00E+00) -	9.80E+09 (1.40E+09)
$F14$	2.18E+03 (2.20E+03) -	1.05E+04 (3.11E+04) -	2.15E+02 (7.29E+01) -	4.85E+01 (1.21E+01) -	2.05E+02 (2.13E+01) -	3.98E+01 (1.62E+01)
$F15$	3.35E+03 (2.79E+03) -	3.49E+02 (4.42E+02) -	3.22E+02 (1.42E+02) -	4.46E+01 (1.12E+01) +	1.37E+09 (3.47E+09) -	2.85E+02 (3.54E+02)

Table C.18 CVnew algorithm: (Contd.) CEC2017 Results in comparison to other algorithms

	SaDE	JADE	SHADE	MVMO	CV1.0	CV_{new}
$F16$	8.17E+02 (2.34E+02) +	8.56E+02 (1.75E+02) +	7.33E+02 (1.88E+02) -	8.40E+02 (1.93E+02) +	1.53E+03 (2.74E+02) -	1.44E+03 (2.10E+02)
$F17$	5.08E+02 (1.53E+02) -	6.00E+02 (1.21E+02) -	5.16E+02 (1.11E+02) -	5.19E+02 (1.33E+02) -	1.25E+03 (1.85E+02) -	1.13E+02 (1.92E+02)
$F18$	3.24E+04 (1.68E+04) -	1.89E+02 (1.25E+02) -	1.89E+02 (1.03E+02) -	4.17E+01 (1.94E+01) +	5.21E+02 (1.19E+02) -	1.51E+02 (4.43E+01)
$F19$	1.13E+04 (1.68E+04) -	3.24E+02 (1.25E+03) -	1.59E+02 (568E+01) -	1.73E+01 (5.13E+00) +	1.73E+02 (4.17E+02) -	5.57E+01 (1.10E+01)
$F20$	3.52E+02 (1.50E+02) -	4.38E+02 (1.33E+02) -	3.33E+02 (1.20E+02) -	3.29E+02 (1.47E+02) -	1.05E+03 (2.14E+02) -	2.81E+02 (1.65E+02)
$F21$	2.87E+02 (1.36E+01) -	2.51E+02 (9.63E+00) -	2.33E+02 (5.11E+00) -	2.77E+02 (1.60E+01) -	5.41E+02 (6.27E+01) -	1.18E+02 (8.77E+01)
$F22$	2.92E+03 (3.24E+03) +	3.33E+03 (1.80E+03) +	3.17E+03 (1.55E+03) +	3.26E+03 (1.71E+03) +	7.33E+03 (1.99E+03) -	5.77E+03 (3.64E+02)
$F23$	5.22E+02 (2.05E+01) -	4.79E+02 (1.17E+01) -	4.59E+02 (8.75E+00) -	5.04E+02 (1.71E+03) -	7.74E+02 (8.06E+01) -	1.87E+02 (5.11E+01)
$F24$	5.89E+02 (1.86E+01) -	5.31E+02 (7.62E+00) -	5.31E+02 (7.45E+00) -	5.83E+02 (1.69E+01) -	8.32E+02 (1.21E+01) -	3.25E+02 (8.95E+01)
$F25$	5.71E+02 (3.05E+01) -	5.19E+02 (3.48E+01) -	5.06E+02 (3.64E+01) -	5.09E+02 (3.12E+01) -	5.43E+02 (1.51E+01) -	4.70E+02 (2.26E+01)
$F26$	2.52E+03 (3.37E+02) -	1.61E+03 (1.21E+02) -	1.41E+03 (9.78E+01) -	1.93E+03 (2.86E+02) -	2.48E+03 (1.88E+03) -	1.16E+03 (1.56E+03)
$F27$	7.10E+02 (6.65E+01) -	5.50E+02 (2.34E+01) -	5.49E+02 (2.78E+01) -	5.43E+02 (1.75E+01) -	7.38E+02 (8.21E+01) -	4.53E+02 (7.17E+01)
$F28$	4.99E+02 (1.53E+01) -	4.91E+02 (2.08E+01) -	4.79E+02 (2.41E+01) -	4.64E+02 (1.50E+01) -	4.94E+02 (1.93E+01) -	4.58E+02 (2.33E-01)
$F29$	5.11E+02 (1.37E+02) +	4.77E+02 (8.06E+01) +	4.87E+02 (1.05E+02) +	4.89E+02 (1.40E+01) +	1.69E+03 (2.29E+02) -	1.45E+03 (1.68E+02)
$F30$	8.07E+05 (8.33E+04) -	6.68E+05 (9.25E+04) -	6.82E+05 (8.51E+04) -	5.81E+05 (1.02E+04) -	4.64E+06 (8.59E+06) -	6.02E+05 (2.99E+04)
$w/t/l$	12/18/0	13/17/0	13/17/0	15/15/0	1/26/3	

Table C.19 CSsin algorithm: CEC 2017 Results for $10D$

Function	Best	Worst	Mean	Std dev.
F_1	8.45E-02	8.83E+00	1.91E+00	1.87E+00
F_3	0.00E+00	4.80E-08	4.53E-09	1.08E-08
F_4	1.86E-05	2.34E-01	1.22E-02	3.41E-02
F_5	3.97E+00	2.48E+01	1.38E+01	5.63E+00
F_6	2.63E-05	8.94E-04	2.73E-04	1.85E-04
F_7	1.49E+01	4.66E+01	2.90E+01	7.32E+00
F_8	2.98E+00	2.28E+01	1.37E+01	3.75E+00
F_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{10}	1.24E-01	4.24E+02	2.33E+02	1.09E+02
F_{11}	1.48E-04	5.13E+00	1.42E+00	1.02E+00
F_{12}	1.06E+03	5.71E+03	2.30E+03	8.49E+02
F_{13}	7.96E+00	2.34E+01	1.45E+01	3.12E+00
F_{14}	2.48E+00	1.00E+01	5.59E+00	1.81E+00
F_{15}	6.53E-01	2.51E+00	1.40E+00	4.96E-01
F_{16}	4.01E-01	2.38E+02	5.88E+01	7.18E+01
F_{17}	4.43E-01	1.96E+01	8.12E+00	7.20E+00
F_{18}	3.56E+00	2.29E+01	1.47E+01	5.13E+00
F_{19}	7.86E-01	2.42E+00	1.56E+00	3.71E-01
F_{20}	5.52E-06	7.41E-01	1.30E-01	2.09E-01
F_{21}	0.00E+00	1.02E+02	9.42E+01	2.38E+01
F_{22}	1.51E+01	1.01E+01	8.94E+01	2.46E+01
F_{23}	0.00E+00	3.19E+02	2.18E+02	1.33E+02
F_{24}	5.27E-06	1.00E+02	9.41E+01	2.37E+01
F_{25}	1.00E+02	3.99E+02	3.91E+02	4.18E+01
F_{26}	0.00E+00	2.00E+02	3.92E+01	8.01E+01
F_{27}	3.87E+02	3.92E+02	3.89E+02	7.67E-01
F_{28}	1.00E+02	3.00E+02	2.96E+02	2.80E+01
F_{29}	1.58E+02	2.77E+02	2.48E+02	1.65E+01
F_{30}	4.56E+02	7.26E+03	1.97E+03	1.42E+03

Table C.20 CSsin algorithm: CEC 2017 Results for $30D$

Function	Best	Worst	Mean	Std dev.
F_1	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_3	2.34E+03	9.02E+03	4.58E+03	1.51E+03
F_4	1.07E-02	8.81E+01	3.30E+01	3.40E+01
F_5	1.32E+02	2.16E+02	1.68E+02	1.85E+01
F_6	5.92E-02	1.61E+00	5.17E-01	4.21E-01
F_7	1.03E+02	3.12E+02	2.00E+02	4.83E+01
F_8	8.45E+01	1.43E+02	1.16E+02	1.25E+01
F_9	1.15E+03	3.96E+02	3.17E+03	5.68E+02
F_{10}	1.57E+03	3.17E+03	2.48E+02	3.33E+02
F_{11}	6.53E+00	9.77E+01	3.61E+01	1.86E+01
F_{12}	4.14E+04	1.00E+10	2.56E+09	4.39E+09
F_{13}	9.80E+02	6.14E+04	1.23E+04	1.45E+04
F_{14}	1.61E+02	1.02E+03	3.36E+02	1.42E+02
F_{15}	2.88E+02	7.44E+02	4.72E+02	1.03E+02
F_{16}	1.37E+02	5.54E+02	3.97E+02	1.03E+02
F_{17}	3.17E+01	1.65E+02	7.27E+01	2.53E+01
F_{18}	1.93E+04	7.59E+04	3.91E+04	1.16E+04
F_{19}	1.05E+02	5.87E+02	2.54E+02	8.41E+01
F_{20}	4.21E+01	2.00E+02	1.58E+02	3.84E+01
F_{21}	1.00E+02	1.04E+02	1.00E+02	7.70E-01
F_{22}	1.00E+02	1.00E+02	1.00E+02	3.09E-02
F_{23}	1.00E+02	4.17E+02	2.82E+02	1.30E+02
F_{24}	1.00E+02	2.00E+02	1.66E+02	4.76E+01
F_{25}	3.83E+02	3.87E+02	3.84E+02	1.61E+00
F_{26}	2.00E+02	2.16E+02	2.02E+02	2.58E+00
F_{27}	4.77E+02	5.09E+02	4.97E+02	8.51E+00
F_{28}	3.00E+02	4.09E+02	3.48E+02	4.43E+01
F_{29}	4.10E+02	5.91E+02	5.11E+02	4.23E+01
F_{30}	9.78E+03	3.16E+04	1.84E+04	5.34E+03

Table C.21 CSsin algorithm: CEC 2017 Results for $50D$

Function	Best	Worst	Mean	Std dev.
F_1	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_2	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_3	1.59E+03	4.75E+04	1.07E+04	6.68E+03
F_4	2.74E-01	1.50E+02	1.88E+01	2.45E+01
F_5	2.65E+02	3.53E+02	3.09E+02	2.10E+01
F_6	1.46E+00	2.16E+01	1.00E+01	5.28E+00
F_7	5.15E+01	9.31E+02	1.39E+02	9.71E+01
F_8	2.69E+02	3.63E+02	3.17E+02	2.43E+01
F_9	7.81E+03	1.33E+04	1.11E+04	1.00E+03
F_{10}	3.66E+03	6.72E+03	4.97E+03	5.83E+03
F_{11}	6.52E+01	1.80E+02	1.17E+02	2.91E+01
F_{13}	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_{14}	1.70E+03	7.53E+04	2.23E+04	1.72E+04
F_{15}	1.64E+03	2.06E+04	1.13E+04	6.02E+03
F_{16}	3.75E+02	1.07E+03	7.23E+02	1.79E+02
F_{17}	2.86E+02	9.28E+02	6.50E+02	1.16E+02
F_{18}	4.75E+04	4.31E+04	1.73E+05	7.91E+04
F_{19}	9.06E+02	1.32E=04	5.84E+03	3.15E+03
F_{20}	2.80E+01	6.94E+02	2.31E+01	9.73E+01
F_{21}	1.08E+02	4.17E+02	1.57E+02	9.74E+01
F_{22}	1.0E+02	1.02E+02	1.00E+02	3.91E-01
F_{23}	1.30E+02	6.82E+02	3.51E+02	7.88E+01
F_{24}	5.84E+02	7.40E+02	6.87E+02	3.57E+01
F_{25}	3.61E+02	5.70E+02	4.26E+02	2.08E+01
F_{26}	3.00E+02	3.00E+02	3.00E+02	4.57E-02
F_{27}	5.39E+02	6.51E+02	5.97E+02	3.25E+01
F_{28}	2.59E+02	5.15E+02	4.13E+02	1.83E+01
F_{29}	4.98E+02	1.03E+03	8.03E+02	1.24E+02
F_{30}	8.39E+04	3.45E+06	1.64E+05	6.29E+05

Table C.22 CSsin algorithm: CEC2017 Results in comparison to other algorithms

	SaDE	**JADE**	**SHADE**	**MVMO**	**CV1.0**	CV_{new}	**CSsin**
F_1	1.21E+03 (1.97E+03) +	5.23E-14 (2.51E-14) +	0.00E+00 (0.00E+00) +	1.33E-05 (5.60E-06) +	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
F_2	9.27E+01 (4.12E+01) +	1.31E+13 (8.53E+13) -	1.08E+12 (4.39E+12) -	1.80E+17 (1.27E+18) -	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
F_3	2.71E+02 (8.28E+02) +	1.77E+04 (3.70E+04) -	0.00E+00 (0.00E+00) +	5.30E-07 (1.09E-07) +	1.95E+04 (6.27E+03) -	8.71E+04 (4.08E+03) -	1.07E+04 (6.68E+03)
F_4	8.92E+01 (4.21E+01) -	4.96E+01 (4.71E+01) -	5.68E+01 (8.80E+00) -	3.58E+01 (3.66E+01) -	1.16E+02 (6.27E+03) +	2.67E+01 (5.92E+00) -	1.88E+01 (3.45E+01)
F_5	9.23E+01 (1.86E+01) +	5.42E+01 (8.80E+00) +	3.28E+01 (5.03E+00) +	8.07E+01 (1.64E+01) +	3.41E+02 (8.02E+01) -	2.39E+02 (3.80E+01) +	3.09E+02 (2.10E+01)
F_6	7.43E-03 (2.35E-02) +	1.44E-13 (9.11E-14) +	8.38E-04 (1.01E-03) +	5.43E-03 (3.30E-03) +	4.85E+01 4.85E+01 -	4.07E+01 (8.14E+00) -	1.00E+01 (5.20E+00)
F_7	1.40E+02 (1.97E+01) -	1.01E+02 (6.48E+00) +	8.09E+01 (3.78E+00) +	1.23E+02 (1.27E+01) +	2.74E+02 (7.29E+01) -	2.22E+02 (3.49E+01) -	1.39E+02 (9.71E+01)
F_8	9.42E+01 (1.77E+01) +	5.52E+01 (7.76E+00) +	3.23E+01 (3.82E+00) +	7.59E+01 (1.61E+01) +	3.29E+02 (7.29E+01) +	2.50E+02 (4.51E+01) +	3.17E+02 (2.43E+01)
F_9	4.83E+01 (6.29E+01) +	1.17E+00 (1.31E+00) +	1.11E+00 (9.37E-01) +	7.38E+00 (5.77E+00) +	1.00E+04 (2.90E+03) =	1.06E+04 (3.10E+03)	1.11E+04 (1.00E+03)
F_{10}	6.60E+03 (1.63E+03) -	3.75E+03 (2.54E+02) +	3.34E+03 (2.94E+02) +	3.49E+03 (4.31E+02) +	7.10E+03 (5.34E+02) -	6.09E+03 (3.55E+02) -	4.97E+03 (5.83E+02)
F_{11}	1.09E+02 (3.54E+01) -	1.36E+02 (3.39E+01) -	1.20E+02 (2.93E+01) -	4.74E+01 (8.72E+00) -	1.66E+02 (3.38E+01) -	1.18E+02 (1.91E+01) -	1.17E+01 (2.91E+01)
F_{12}	1.11E+05 (6.20E+04) +	5.14E+03 (3.32E+03) +	5.13E+03 (2.87E+03) +	1.29E+03 (2.79E+02) +	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
F_{13}	1.21E+03 (1.45E+03) +	3.03E+02 (2.69E+02) +	2.65E+02 (1.49E+02) +	4.37E+01 (1.76E+01) +	1.00E+10 (0.00E+00) =	9.80E+09 (1.40E+09) +	1.10E+10 (0.00E+00)
F_{14}	2.18E+03 (2.20E+03) +	1.05E+04 (3.11E+04) +	2.15E+02 (7.29E+01) +	4.85E+01 (1.21E+01) +	2.05E+02 (2.13E+01) +	3.98E+01 (1.62E+01) +	2.23E+04 (1.72E+04)
F_{15}	3.35E+03 (2.79E+03) +	3.49E+02 (4.42E+02) +	3.22E+02 (1.42E+02) +	4.46E+01 (1.12E+01) +	1.37E+09 (3.47E+09) +	2.85E+02 (3.54E+02) +	1.13E+04 (6.02E+03)
$w/t/l$	13/0/17	14/0/16	14/0/16	14/0/16	8/3/19	10/3/17	

Table C.22 CSsin algorithm: (Contd.) CEC2017 Results in comparison to other algorithms

	SaDE	**JADE**	**SHADE**	**MVMO**	**CV1.0**	CV_{new}	**CSsin**
F_{16}	8.17E+02 (2.34E+02) -	8.56E+02 (1.75E+02) -	7.33E+02 (1.88E+02) -	8.40E+02 (1.93E+02) -	1.53E+03 (2.74E+02) -	1.44E+03 (2.10E+02) -	7.23E+02 (1.79E+02)
F_{17}	5.08E+02 (1.53E+02) -	6.00E+02 (1.21E+02) -	5.16E+02 (1.11E+02) -	5.19E+02 (1.33E+02) -	1.25E+03 (1.85E+02) -	1.13E+02 (1.92E+02) +	1.50E+02 (1.16E+02)
F_{18}	3.24E+04 (1.68E+04) +	1.89E+02 (1.25E+02) +	1.89E+02 (1.03E+02) +	4.17E+01 (1.94E+01) +	5.21E+02 (1.19E+02) +	1.51E+02 (4.43E+01) +	1.73E+05 (7.91E+04)
F_{19}	1.13E+04 (1.68E+04) -	3.24E+02 (1.25E+03) +	1.59E+02 (568E+01) +	1.73E+01 (5.13E+00) +	1.73E+02 (4.17E+02) +	5.57E+01 (1.10E+01) +	5.84E+03 (3.15E+03)
F_{20}	3.52E+02 (1.50E+02) -	4.38E+02 (1.33E+02) -	3.33E+02 (1.20E+02) -	3.29E+02 (1.47E+02) -	1.05E+03 (2.14E+02) -	2.81E+02 (1.65E+02) -	2.31E+02 (9.73E+01)
F_{21}	2.87E+02 (1.36E+01) -	2.51E+02 (9.63E+00) -	2.33E+02 (5.11E+00) -	2.77E+02 (1.60E+01) -	5.41E+02 (6.27E+01) -	1.18E+02 (8.77E+01) +	1.57E+02 (9.74E+01)
F_{22}	2.92E+03 (3.24E+03) -	3.33E+03 (1.80E+03) -	3.17E+03 (1.55E+03) -	3.26E+03 (1.71E+03) -	7.33E+03 (1.99E+03) -	5.77E+03 (3.64E+02) -	1.00E+02 (3.91E-01)
F_{23}	5.22E+02 (2.05E+01) -	4.79E+02 (1.17E+01) -	4.59E+02 (8.75E+00) -	5.04E+02 (1.71E+03) +	7.74E+02 (8.06E+01) +	1.87E+02 (5.11E+01) -	4.51E+02 (7.88E+01)
F_{24}	5.89E+02 (1.86E+01) +	5.31E+02 (7.62E+00) +	5.31E+02 (7.45E+00) +	5.83E+02 (1.69E+01) +	8.32E+02 (1.21E+01) -	3.25E+02 (8,95E+01) +	6.87E+02 (3.57E+01)
F_{25}	5.71E+02 (3.05E+01) -	5.19E+02 (3.48E+01) -	5.06E+02 (3.64E+01) -	5.09E+02 (3.12E+01) -	5.43E+02 (1.51E+01) -	4.70E+02 (2.26E+01) -	4.26E+02 (2.08E+01)
F_{26}	2.52E+03 (3.37E+02) -	1.61E+03 (1.21E+02) -	1.41E+03 (9.78E+01) -	1.93E+03 (2.86E+02) -	2.48E+03 (1.88E+03) -	1.16E+03 (1.56E+03) -	3.00E+02 (4.57E-02)
F_{27}	7.10E+02 (6.65E+01) -	5.50E+02 (2.34E+01) +	5.49E+02 (2.78E+01) +	5.43E+02 (1.75E+01) +	7.38E+02 (8.21E+01) -	4.53E+02 (7.17E+01) -	5.97E+02 (3.22E+01)
F_{28}	4.99E+02 (1.53E+01) -	4.91E+02 (2.08E+01) -	4.79E+02 (2.41E+01) -	4.64E+02 (1.50E+01) -	4.94E+02 (1.93E+01) -	4.58E+02 (2.33E-01) -	4.13E+02 (1.83E+01)
F_{29}	5.11E+02 (1.37E+02) +	4.77E+02 (8.06E+01) +	4.87E+02 (1.05E+02) +	4.89E+02 (1.40E+01) +	1.69E+03 (2.29E+02) -	1.45E+03 (1.68E+02) -	8.03E+02 (1.24E+02)
F_{30}	8.07E+05 (8.33E+04) -	6.68E+05 (9.25E+04) -	6.82E+05 (8.51E+04) -	5.81E+05 (1.02E+04) -	4.64E+06 (8.59E+06) -	6.02E+05 (2.99E+04) -	1.64E+05 (6.25E+05)
$w/t/l$	13/0/17	14/0/16	14/0/16	14/0/16	8/3/19	10/3/17	

Table C.23 CSsin algorithm: CEC2020 Results in comparison to other algorithms

Dimension	Function	Best	Worst	Mean	Median	Std
D=5	F1	0.00E+00	1.36E-06	2.81E-07	2.06E-07	3.01E-07
	F2	1.08E-04	1.19E+02	8.75E+00	2.49E-01	2.81E+01
	F3	5.14E+00	7.58E+00	5.65E+00	5.68E+00	4.74E-01
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	9.45E-06	3.80E-03	3.94E-04	1.67E-04	6.52E-04
	F6	6.67E-05	1.90E-03	5.77E-04	4.68E-04	4.14E-04
	F7	3.90E-07	1.49E-05	4.18E-06	2.29E-06	3.94E-06
	F8	0.00E+00	1.00E+02	3.67E+01	4.87E-06	4.77E+01
	F9	2.74E-05	1.00E+02	1.17E+01	1.98E-03	3.25E+01
	F10	1.00E+02	3.47E+02	2.88E+02	3.00E+02	7.00E+01
D=10	F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F2	4.37E-01	4.02E+01	1.58E+01	1.12E+01	1.21E+01
	F3	1.11E+01	1.71E+01	1.39E+01	1.40E+01	1.71E+00
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	1.99E+00	9.12E+00	4.43E+00	4.33E+00	1.82E+00
	F6	3.07E-02	3.85E-01	1.70E-01	1.50E-01	1.20E-01
	F7	3.65E-02	4.61E-01	1.55E-01	1.22E-01	9.95E-02
	F8	1.84E+01	1.00E+02	8.01E+01	1.00E+02	3.18E+01
	F9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.38E-13
	F10	1.00E+02	3.97E+02	3.77E+02	3.97E+02	7.55E+01
D=15	F1	0.00E+00	1.00E+10	3.33E+08	0.00E+00	1.82E+09
	F2	4.72E+00	2.41E+02	7.25E+01	4.82E+01	5.99E+01
	F3	1.64E+01	2.11E+01	1.80E+01	1.76E+01	1.25E+00
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	1.24E+00	2.57E+01	1.30E+01	1.34E+01	6.19E+00
	F6	5.23E-02	8.70E+00	1.44E+00	3.83E-01	2.85E+00
	F7	4.93E-01	1.24E+00	8.85E-01	8.98E-01	1.96E-01
	F8	0.00E+00	1.00E+02	8.75E+01	1.00E+02	2.76E+01
	F9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	2.30E-13
	F10	4.00E+02	4.00E+02	4.00E+02	4.00E+02	0.00E+00
D=20	F1	0.00E+00	1.00E+10	9.33E+09	1.00E+10	2.53E+09
	F2	3.54E+00	2.75E+02	9.83E+01	1.27E+02	8.33E+01
	F3	2.12E+01	3.17E+01	2.55E+01	2.53E+01	2.27E+00
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	1.96E+01	2.22E+02	1.16E+02	1.38E+02	6.34E+01
	F6	2.87E-01	1.42E+00	6.72E-01	6.58E-01	82.23E-01
	F7	7.90E-01	1.03E+01	2.62E+00	1.96E+00	2.26E+00
	F8	6.93E+01	1.00E+02	9.89E+01	1.00E+02	5.59E+00
	F9	1.00E+02	2.00E+02	1.03E+02	1.00E+02	1.82E+01
	F10	3.99E+02	4.00E+02	3.99E+02	3.99E+02	2.44E-01

Table C.24 SACS algorithm: Analysis of switch probability

Function	Algorithm	$p_a = 0.25$		$p_a = 0.50$		$p_a = 0.80$		$p_a = adaptive$	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
F_1	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.10E+10	0.00E+00
F_2	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.10E+10	0.00E+00
F_3	CS	1.83E+05	2.81E+04	2.38E+05	3.45E+04	2.52E+05	3.02E+04		
	SACS	1.39E+04	**4.31E+03**	1.43E+04	5.18E+03	1.40E+04	4.76E+03	**1.21E+04**	5.54E+03
F_4	CS	1.28E+02	4.66E+01	1.15E+02	4.42E+01	1.28E+02	2.44E+01		
	SACS	9.90E+01	2.88E+01	1.13E+02	3.56E+01	**8.31E+01**	**3.26E+01**	9.26E+02	3.66E+01
F_5	CS	3.42E+02	4.24E+01	4.17E+02	4.67E+01	4.86E+02	4.66E+01		
	SACS	2.37E+02	3.17E+01	**2.82E+02**	4.51E+01	3.17E+02	5.14E+01	3.04E+02	**2.49E+01**
F_6	CS	6.30E+01	9.57E+00	6.75E+01	8.41E+00	4.13E+01	6.32E+00		
	SACS	2.92E+01	1.00E+01	2.13E+01	8.05E+00	5.85E+00	**2.46E+00**	**1.68E+00**	3.81E+00
F_7	CS	6.46E+02	1.03E+02	5.39E+02	6.33E+01	5.51E+02	4.08E+01		
	SACS	1.68E+02	2.35E+01	1.93E+02	2.58E+01	2.48E+02	2.87E+01	**1.28E+02**	9.03E+01
F_8	CS	3.52E+02	4.25E+01	4.05E+02	4.60E+01	4.82E+02	4.67E+01		
	SACS	**2.30E+02**	3.23E+01	2.72E+02	5.01E+01	3.11E+02	3.95E+01	3.15E+02	2.74E+02
F_9	CS	1.95E+04	5.68E+03	3.06E+04	7.15E+03	3.53E+04	4.82E+09		
	SACS	3.29E+03	1.09E+03	5.41E+03	1.22E+03	5.84E+03	1.50E+03	**1.09E+03**	9.58E+02
F_{10}	CS	7.77E+03	3.08E+02	7.74E+03	3.76E+02	7.39E+03	3.26E+02		
	SACS	5.23E+03	5.47E+02	5.15E+03	4.47E+02	5.02E+03	3.87E+02	**4.97E+03**	6.63E+02
F_{11}	CS	2.99E+02	6.75E+01	3.23E+02	4.54E+01	3.45E+02	4.16E+01		
	SACS	9.92E+01	1.90E+01	1.38E+02	3.32E+01	1.61E+02	3.94E+01	**1.47E+02**	3.97E+01
F_{12}	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_{13}	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	9.80E+03	1.40E+09	1.10E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_{14}	CS	3.81E+02	7.49E+01	1.03E+05	4.03E+04	3.26E+05	1.60E+05		
	SACS	**9.31E+02**	1.88E+02	1.16E+04	7.98E+03	3.00E+04	2.29E+04	3.51E+03	2.86E+04
F_{15}	CS	3.47E+09	4.71E+09	3.58E+09	4.78E+09	7.85E+09	4.12E+09		
	SACS	8.62E+03	3.67E+03	9.23E+03	5.73E+03	5.79E+03	7.18E+03	**1.10E+03**	6.03E+02
F_{16}	CS	2.07E+03	2.43E+02	2.18E+03	1.60E+02	1.76E+03	2.37E+02		
	SACS	1.04E+03	2.50E+02	1.03E+03	1.90E+02	1.03E+03	1.61E+02	**7.44E+02**	1.57E+02
F_{17}	CS	1.47E+03	1.93E+02	1.58E+03	1.83E+02	1.18E+03	1.78E+02		
	SACS	9.31E+02	1.82E+02	7.81E+02	1.86E+02	6.59E+02	1.50E+02	**6.23E+02**	1.03E+02
F_{18}	CS	2.80E+05	1.66E+05	1.72E+06	6.67E+)5	1.43E+06	1.39E+09		
	SACS	9.02E+04	3.42E+04	2.60E+05	1.15E+05	2.29E+05	1.14E+05	1.79E+05	1.05E+05
F_{19}	CS	8.02E+02	1.41E+03	5.46E+04	2.37E+05	1.99E+08	1.67E+02		
	SACS	6.94E+03	2.81E+03	3.86E+03	3.39E+03	**9.22E+02**	1.14E+03	6.62E+03	4.01E+03
F_{20}	CS	1.41E+03	1.41E+02	1.43E+03	1.78E+02	1.04E+03	7.93E+01		
	SACS	7.76E+02	1.48E+02	6.33E+02	1.76E+02	4.74E+02	1.26E+02	**4.44E+02**	1.36E+02
F_{21}	CS	5.21E+02	3.96E+01	5.68E+02	4.70E+01	6.55E+02	4.08E+02		
	SACS	4.18E+02	5.51E+01	4.63E+02	4.34E+01	4.56E+02	1.22E+02	**1.59E+02**	9.57E+01
F_{22}	CS	8.30E+03	3.89E+02	8.36E+03	3.70E+02	8.19E+03	4.59E+01		
	SACS	5.57E+03	1.21E+03	5.26E+03	1.75E+03	4.55E+03	2.11E+03	**1.02E+02**	**6.50E-01**
F_{23}	CS	8.25E+02	5.67E+01	8.29E+02	5.86E+01	9.14E+02	4.59E+01		
	SACS	6.54E+02	4.82E+01	6.71E+02	6.15E+01	6.74E+02	1.05E+02	**5.52E+02**	7.47E+01
F_{24}	CS	8.64E+02	6.69E+01	8.76E+02	6.05E+01	1.01E+03	6.38E+01		
	SACS	7.38E+02	4.89E+01	7.51E+02	9.49E+01	7.88E+02	1.32E+02	**6.88E+02**	9.00E+01
F_{25}	CS	5.57E+02	2.92E+01	5.47E+02	2.70E+01	5.53E+02	1.66E+01		
	SACS	**4.93E+02**	2.11E+01	5.02E+02	2.36E+01	5.11E+02	2.87E+01	5.28E+02	2.50E+01
F_{26}	CS	5.39E+03	6.70E+02	4.88E+03	9.07E+02	4.57E+03	1.82E+03		
	SACS	9.32E+02	1.29E+03	5.12E+02	7.59E+02	3.12E+02	4.43E+01	**3.00E+02**	1.87E-01
F_{27}	CS	8.69E+02	9.35E+01	9.03E+02	7.89E+01	8.17E+02	5.68E+01		
	SACS	6.62E+02	4.96E+01	6.73E+02	5.31E+01	6.38E+02	4.34E+01	**6.11E+02**	3.64E+01
F_{28}	CS	5.22E+02	2.95E+01	5.05E+02	2.59E+01	5.12E+02	1.88E+01		
	SACS	4.77E+02	2.20E+01	**4.72E+02**	1.83E+01	4.82E+02	1.89E+01	4.87E+02	2.43E+01
F_{29}	CS	2.00E+03	2.12E+02	2.00E+03	1.65E+02	1.67E+03	1.79E+02		
	SACS	1.16E+03	1.98E+02	1.09E+03	1.88E+02	8.39E+02	1.70E+02	**6.45E+02**	1.34E+02
F_{30}	CS	6.22E+08	2.37E+09	1.42E+09	3/45E+09	2.95E+09	4.59E+09		
	SACS	2.17E+06	6.36E+05	1.47E+06	4.94E+05	1.05E+06	2.60E+05	**1.22E+05**	3.53E+05

Table C.25 SACS algorithm: Analysis of effect of population

Function	Algorithm	$pop_{size} = 20$		$pop_{size} = 40$		$pop_{size} = 80$		$pop_{size} = 100$		$pop_{adaptive}$	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F_1	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_2	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_3	CS	2.52E+05	3.02E+04	2.24E+05	2.41E+04	2.15E+05	2.43E+04	2.08E+05	2.28E+04		
	SACS	5.98E+04	1.92E+04	5.96E+04	1.47E+04	3.62E+04	1.30E+04	2.68E+04	1.03E+04	**2.27E+04**	6.15E+03
F_4	CS	1.28E+02	2.44E+01	1.09E+02	2.97E+01	1.05E+02	2.23E+01	1.01E+02	2.01E+01		
	SACS	**2.93E+01**	4.25E+01	3.71E+01	2.70E+01	2.95E+01	3.55E+01	3.87E+01	2.56E+01	8.75E+01	3.81E+01
F_5	CS	4.86E+02	4.66E+01	4.01E+02	3.89E+01	4.00E+02	3.40E+01	4.03E+02	3.48E+01		
	SACS	**1.85E+02**	4.55E+01	2.02E+02	4.51E+01	2.12E+02	4.50E+01	2.29E+02	3.42E+01	2.35E+02	2.57E+01
F_6	CS	4.13E+02	6.32E+00	6.70E+01	6.02E+00	6.64E+01	5.00E+00	6.62E+01	4.47E+00		
	SACS	2.58E+01	1.32E+01	1.54E+01	1.53E+01	1.01E+01	1.27E+01	6.09E+00	1.44E+01	**1.89E+00**	**1.37E+00**
F_7	CS	5.51E+02	4.08E+01	4.49E+02	4.07E+01	4.29E+02	2.73E+01	4.34E+02	3.29E+01		
	SACS	3.03E+02	7.68E+01	2.09E+02	5.45E+01	1.98E+02	3.78E+01	1.88E+02	4.71E+01	**1.44E+02**	1.06E+02
F_8	CS	4.82E+02	4.67E+01	3.96E+02	4.51E+01	4.04E+02	4.01E+01	3.92E+01	3.90E+01		
	SACS	5.06E+03	4.78E+01	2.18E+02	3.70E+01	2.26E+02	4.10E+01	2.05E+02	3.76E+01	**2.13E+02**	2.93E+01
F_9	CS	3.53E+04	4.82E+03	3.20E+04	6.97E+03	3.19E+04	4.29E+03	3.10E+04	4.25E+03		
	SACS	**1.03E+02**	5.33E+03	4.67E+03	2.89E+03	2.69E+03	2.60E+03	3.58E+03	1.56E+03	7.89E+03	1.52E+03
F_{10}	CS	7.39E+03	3.26E+02	7.39E+03	2.62E+02	7.08E+03	2.62E+02	6.83E+03	3.18E+02		
	SACS	5.06E+03	8.81E+02	4.32E+03	6.56E+02	4.94E+03	5.10E+02	4.76E+03	5.75E+02	**3.53E+03**	6.33E+02
F_{11}	CS	3.45E+02	4.16E+01	2.98E+02	3.20E+01	2.96E+02	2.22E+01	2.92E+02	2.04E+01		
	SACS	1.03E+02	5.09E+01	**6.58E+01**	4.30E+01	8.14E+01	3.83E+01	6.91E+01	2.91E+01	8.93E+01	4.98E+01
F_{12}	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_{13}	CS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	9.82E+09	1.24E+09	1.00E+10	0.00E+00		
	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00	**4.86E+04**	2.37E+09
F_{14}	CS	3.26E+05	1.60E+05	8.61E+04	3.57E+04	5.64E+05	2.10E+04	5.27E+04	2.00E+04		
	SACS	9.26E+03	5.28E+04	5.94E+03	3.11E+04	2.31E+03	2.04E+04	4.03E+03	2.18E+04	**5.46E+03**	3.10E+04
F_{15}	CS	7.85E+09	4.12E+09	2.15E+09	4.14E+09	5.99E+08	2.37E+09	1.13E+07	2.55E+07		
	SACS	**3.81E+03**	5.09E+09	3.96E+03	4.28E+09	3.19E+03	3.00E+09	1.80E+03	6.95E+03	3.88E+03	6.50E+03
F_{16}	CS	1.79E+03	2.37E+02	2.05E+03	1.53E+02	1.87E+03	1.65E+02	1.86E+03	1.41E+02		
	SACS	8.48E+02	3.41E+02	6.57E+02	2.79E+02	5.23E+02	2.35E+02	9.23E+02	2.13E+02	**4.04E+02**	2.16E+02
F_{17}	CS	1.18E+03	1.78E+02	1.45E+03	1.50E+02	1.35E+03	1.38E+02	1.33E+03	1.11E+02		
	SACS	7.96E+02	2.19E+02	3.73E+02	2.61E+02	6.26E+02	2.25E+02	4.95E+02	1.93E+02	**2.85E+02**	1.37E+02
F_{18}	CS	1.43E+06	5.89E+05	1.08E+)3	3.62E+05	8.72E+05	2.91E+05	7.30E+05	2.26E+05		
	SACS	2.09E+05	5.57E+05	1.13E+05	2.96E+05	9.37E+04	3.01E+05	**1.34E+02**	1.93E+02	8.24E+04	1.55E+05
F_{19}	CS	1.99E+08	1.39E+09	2.19E+04	1.57E+04	1.52E+04	1.18E+04	1.56E+04	1.12E+04		
	SACS	**4.21E+02**	1.33E+04	7.03E+02	8.47E+03	8.45E+02	7.19E+03	4.25E+02	5.79E+03	3.86E+03	1.06E+04
F_{20}	CS	1.04E+03	1.67E+02	1.30E+03	1.72E+02	1.19E+03	1.22E+02	1.14E+03	1.24E+02		
	SACS	6.59E+02	2.25E+02	5.16E+02	2.10E+02	4.40E+02	1.93E+02	4.92E+02	2.14E+02	**2.35E+02**	1.23E+02
F_{21}	CS	6.55E+02	7.93E+01	5.70E+02	3.75E+01	5.86E+02	3.50E+01	5.86E+02	3.44E+01		
	SACS	3.26E+02	4.54E+01	3.84E+02	4.05E+01	3.58E+02	4.39E+01	3.96E+02	4.36E+01	**1.24E+02**	1.04E+02
F_{22}	CS	8.19E+03	4.08E+02	8.04E+03	2.76E+02	7.60E+03	8.32E+02	7.41E+03	1.28E+02		
	SACS	1.17E+02	1.66E+03	1.16E+02	1.18E+03	1.12E+02	1.36E+02	1.05E+02	1.30E+03	**1.10E+02**	3.85E+01
F_{23}	CS	9.14E+02	4.59E+01	8.37E+02	3.64E+01	8.41E+02	3.00E+01	8.45E+02	3.16E+01		
	SACS	6.69E+02	5.02E+01	6.54E+02	5.12E+01	6.37E+02	4.59E+01	6.19E+02	4.43E+01	**4.59E+02**	5.73E+01
F_{24}	CS	1.01E+03	6.38E+01	8.84E+02	3.86E+01	8.84E+02	3.92E+01	8.97E+02	3.41E+01		
	SACS	6.45E+02	5.62E+01	6.81E+02	5.40E+01	6.46E+02	5.23E+01	6.76E+02	5.22E+01	**6.01E+02**	3.52E+01
F_{25}	CS	5.53E+02	1.66E+01	5.31E+02	1.63E+01	5.23E+02	9.19E+00	5.17E+02	1.23E+01		
	SACS	4.66E+02	2.94E+01	4.64E+02	1.82E+01	4.60E+02	2.29E+01	4.67E+02	1.69E+01	**4.30E+02**	2.22E+01
F_{26}	CS	4.57E+03	1.82E+03	4.12E+03	1.21E+03	3.56E+03	1.44E+03	3.26E+03	1.38E+03		
	SACS	7.52E+02	1.09E+03	3.39E+02	1.38E+03	3.05E+02	1.77E+03	3.04E+02	1.81E+03	**3.01E+02**	1.03E+01
F_{27}	CS	8.17E+02	5.68E+01	8.63E+02	5.19E+01	8.30E+02	4.78E+01	8.04E+02	4.15E+01		
	SACS	5.87E+02	1.01E+02	**5.66E+02**	7.08E+01	5.95E+02	6.62E+01	5.78E+02	6.19E+01	5.89E+02	4.33E+01
F_{28}	CS	5.12E+02	1.88E+01	4.76E+02	1.51E+01	4.71E+02	7.32E+00	4.68E+02	4.69E+00		
	SACS	4.61E+02	3.16E+01	4.95E+02	1.89E+01	**4.59E+02**	1.20E+01	4.59E+02	8.76E+00	4.62E+02	2.25E+01
F_{29}	CS	1.57E+03	1.79E+02	1.89E+03	1.62E+02	1.79E+03	1.16E+02	1.70E+03	1.44E+02		
	SACS	9.00E+02	3.28E+02	6.45E+02	3.05E+02	1.17E+03	2.21E+02	9.13E+02	2.27E+02	**5.28E+02**	1.79E+02
F_{30}	CS	2.95E+09	4.59E+09	3.44E+07	7.56E+07	1.31E+07	1.13E+07	1.05E+07	4.44E+06		
	SACS	1.02E+06	3.47E+09	1.14E+06	4.61E+07	**9.41E+05**	6.38E+05	9.83E+05	6.60E+05	1.31E+06	9.13E+05

Table C.26 SACS algorithm: Analysis of effect of dimension size

Function	Algorithm	$D = 10$		$D = 30$		$D = 50$	
		Mean	Std	Mean	Std	Mean	Std
F_1	CS	2.94E+09	4.59E+09	1.00E+10	**0.00E+00**	1.00E+10	**0.00E+00**
	CV_{new}	1.26E-06	5.92E-06	1.00E+10	**0.00E+00**	1.00E+10	**0.00E+00**
	SACS	**6.97E-08**	**7.17E-08**	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_2	CS	5.69E+03	4.99E+09	1.10E+10	0.00E+00	1.00E+10	0.00E+00
	CV_{new}	1.00E+10	2.37E+09	1.00E+10	**0.00E+00**	1.00E+10	**0.00E+00**
	SACS	**0.00E+00**	**0.00E+00**	1.00E+10	**0.00E+00**	1.00E+10	**0.00E+00**
F_3	CS	2.56E+00	4.69E+00	7.70E+04	1.50E+04	2.52E+05	3.02E+04
	CV_{new}	**0.00E+00**	**0.00E+00**	1.39E+02	9.30E+01	8.71E+03	4.08E+03
	SACS	**0.00E+00**	**0.00E+00**	1.61E+03	4.29E+02	4.09E+03	1.50E+03
F_4	CS	3.58E-01	6.70E-01	**7.66E+01**	1.25E+01	1.28E+02	2.44E+01
	CV_{new}	**0.00E+00**	**0.00E+00**	1.43E+01	2.18E+01	**2.67E+01**	5.92E+00
	SACS	3.69E-04	1.70E-03	8.13E+01	1.66E+01	4.23E+01	2.71E+01
F_5	CS	1.21E+01	3.26E+00	1.91E+02	2.43E+01	4.86E+02	4.66E+01
	CV_{new}	1.38E+01	4.61E+00	1.10E+02	2.51E+01	**2.39E+02**	3.80E+01
	SACS	**4.20E+00**	1.87E+00	**1.05E+02**	2.47E+01	2.79E+02	2.34E+01
F_6	CS	2.40E-03	1.70E-03	1.05E+01	3.10E+00	4.13E+01	6.32E+00
	CV_{new}	3.27E-02	2.31E-02	1.96E+01	7.90E+00	4.07E+01	8.14E+00
	SACS	**1.16E-08**	1.77E-08	**1.99E-01**	1.61E-01	**4.87E-01**	3.41E-01
F_7	CS	2.27E+01	4.47E+00	2.30E+02	2.50E+01	5.51E+02	4.08E+01
	CV_{new}	2.47E+01	4.64E+00	1.15E+02	2.06E+01	2.22E+02	3.49E+01
	SACS	**1.41E+01**	1.51E+00	**8.83E+01**	1.32E+01	**1.70E+02**	2.02E+01
F_8	CS	1.36E+01	4.30E+00	1.94E+02	2.27E+01	4.82E+02	4.67E+01
	CV_{new}	1.34E+01	4.33E+00	1.16E+02	2.46E+01	2.50E+02	4.51E+01
	SACS	**8.07E+00**	3.00E+00	**1.07E+02**	1.72E+01	**2.53E+02**	3.05E+01
F_9	CS	2.67E-01	3.95E-01	6.76E+03	1.62E+03	3.53E+04	4.02E+03
	CV_{new}	0.00E+00	0.00E+00	1.95E+03	1.04E+03	1.06E+04	3.10E+03
	SACS	**0.00E+00**	0.00E+00	2.02E+03	**8.69E+02**	**8.93E+03**	1.73E+03
F_{10}	CS	3.75E+02	1.31E+02	3.70E+03	2.76E+02	7.39E+03	3.26E+02
	CV_{new}	5.11E+02	1.29E+02	3.31E+03	2.68E+02	6.09E+03	3.55E+02
	SACS	**1.07E+02**	8.43E+01	**2.79E+03**	3.91E+02	**5.30E+03**	6.12E+02
F_{11}	CS	408E+00	1.66E+00	9.88E+01	2.01E+01	3.45E+02	4.16E+01
	CV_{new}	1.68E+00	9.84E-01	4.20E+01	1.68E+01	1.18E+02	1.91E+01
	SACS	**3.90E-01**	5.29E-01	**2.85E+01**	1.04E+01	**6.58E+01**	1.06E+01
F_{12}	CS	5.17E+03	3.16E+03	9.04E+09	2.92E+09	1.00E+10	0.00E+00
	CV_{new}	3.62E+01	1.90E+01	4.50E+09	5.02E+09	1.00E+10	0.00E+00
	SACS	**1.68E+02**	6.98E+01	**1.97E+08**	1.40E+09	1.00E+10	0.00E+00
F_{13}	CS	1.02E+01	3.49E+00	4.34E+09	4.97E+09	1.00E+10	0.00E+00
	CV_{new}	7.26E+00	2.26E+00	8.20E+01	2.66E+01	9.80E+09	1.40E+09
	SACS	**2.77E+00**	1.26E+01	1.37E+04	1.52E+04	**9.21E+09**	2.71E+09
F_{14}	CS	3.97E+01	1.70E+00	4.36E+04	3.21E+04	3.26E+05	1.60E+05
	CV_{new}	7.38E+00	2.26E+00	**4.95E+01**	6.68E+00	**3.98E+01**	1.62E+01
	SACS	**4.87E-01**	4.74E-01	3.35E+02	1.18E+02	2.66E+03	1.58E+03
F_{15}	CS	1.43E+00	6.23E-01	1.38E+03	1.76E+03	7.88E+09	4.12E+09
	CV_{new}	8.30E-01	3.40E-01	3.41E+01	8.39E+00	2.85E+02	3.54E+02
	SACS	**9.99E-02**	5.29E-02	**1.08E+01**	1.45E+01	**2.40E+02**	2.99E+03

Table C.26 SACS algorithm (Contd.) Analysis of effect of dimension size

Function	Algorithm	$D = 10$		$D = 30$		$D = 50$	
		Mean	Std	Mean	Std	Mean	Std
F_{16}	CS	2.63E+01	4.22E+01	8.02E+02	1.56E+02	1.79E+03	2.37E+02
	CV_{new}	2.82E+00	1.84E+00	7.43E+02	1.81E+02	1.44E+03	2.10E+02
	SACS	**2.50E+00**	1.66E+01	**3.02E+02**	1.41E+02	**7.21E+02**	1.74E+02
F_{17}	CS	5.28E+00	4.27E+00	3.28E+02	1.28E+02	1.18E+03	1.78E+02
	CV_{new}	2.35E+01	4.81E+00	1.93E+02	7.33E+01	**1.13E+02**	1.92E+02
	SACS	**3.40E-01**	2.53E-01	**6.55E+01**	1.45E+01	5.03E+02	1.18E+02
F_{18}	CS	5.01E+00	2.53E+00	2.48E+05	1.25E+05	1.43E+06	5.89E+05
	CV_{new}	2.30E+00	1.08E+00	3.85E+01	7.36E+00	**1.51E+02**	4.43E+01
	SACS	**2.92E-01**	1.46E-01	3.66E+04	1.46E+04	1.06E+05	4.17E+04
F_{19}	CS	3.33E-01	2.43E+00	1.37E+05	1.64E+03	1.99E+08	1.39E+09
	CV_{new}	1.66E+00	2.91E-01	2.11E+01	3.18E+00	**5.57E+01**	1.10E+01
	SACS	**8.52E-02**	3.67E-02	**5.83E+02**	2.01E+02	1.18E+02	1.13E+03
F_{20}	CS	1.43E+00	1.41E+00	3.75E+02	1.20E+02	1.04E+03	1.67E+02
	CV_{new}	1.95E+01	6.89E+00	2.83E+02	8.51E+01	2.81E+02	1.65E+02
	SACS	**8.89E-08**	3.12E-07	**1.25E+02**	6.40E+01	**3.23E+02**	8.66E+01
F_{21}	CS	1.13E+02	3.30E+01	3.47E+02	9.97E+01	6.55E+02	7.93E+01
	CV_{new}	9.60E+01	1.96E+01	2.50E+02	9.93E+01	**1.18E+02**	8.77E+01
	SACS	**9.80E+01**	1.40E+01	**1.00E+02**	6.61E-01	2.54E+02	1.14E+02
F_{22}	CS	8.65E+01	2.76E+01	2.10E+03	2.02E+03	8.19E+03	4.08E+02
	CV_{new}	6.44E+01	3.72E+01	1.65E+03	1.81E+03	5.77E+03	3.64E+02
	SACS	**8.34E+01**	2.75E+01	**1.00E+02**	2.94E-02	**1.00E+02**	6.10E-03
F_{23}	CS	3.17E+02	3.79E+00	5.15E+02	1.44E+01	9.14E+02	4.59E+01
	CV_{new}	3.07E+02	4.40E+01	4.34E+02	5.30E+01	**1.87E+02**	5.11E+01
	SACS	**1.95E+02**	1.27E+02	**2.97E+02**	1.29E+02	5.14E+02	4.76E+01
F_{24}	CS	1.07E+02	4.35E+01	5.98E+02	1.17E+02	1.01E+03	6.38E+01
	CV_{new}	1.00E+02	3.51E+01	4.21E+02	1.61E+02	3.25E+02	8.95E+01
	SACS	**9.60E+01**	1.96E+01	**2.32E+02**	1.32E+02	**5.33E+02**	2.63E+01
F_{25}	CS	**2.50E+02**	1.34E+02	3.85E+02	1.48E+00	5.53E+02	1.66E+01
	CV_{new}	1.86E+02	1.21E+02	3.83E+02	8.20E-01	**4.70E+02**	2.26E+01
	SACS	3.74E+02	8.08E+01	3.86E+02	**1.45E+00**	4.83E+02	8.65E+00
F_{26}	CS	1.84E+02	8.98E+01	1.32E+03	1.00E+03	4.57E+03	1.82E+03
	CV_{new}	1.47E+02	9.80E+01	2.57E+02	4.88E+01	1.16E+03	1.56E+03
	SACS	**5.09E+00**	8.80E+01	**2.74E+02**	4.25E+01	**3.00E+02**	2.41E-04
F_{27}	CS	3.91E+02	1.56E+00	5.22E+02	8.59E+00	8.17E+02	5.68E+01
	CV_{new}	3.81E+02	5.42E+01	5.12E+02	9.32E+00	4.53E+02	7.71E+01
	SACS	**3.88E+02**	6.33E-01	4.98E+02	**5.30E+00**	**5.27E+02**	2.80E+01
F_{28}	CS	3.25E+02	6.25E+01	4.15E+02	6.00E+00	5.12E+02	1.88E+01
	CV_{new}	2.35E+02	1.09E+02	3.02E+02	1.44E+01	4.58E+02	2.33E-01
	SACS	**2.94E+02**	4.20E+01	**3.44E+02**	4.67E+01	**4.58E+02**	7.56E-04
F_{29}	CS	2.65E+02	1.93E+01	8.06E+02	1.16E+02	1.57E+03	1.79E+02
	CV_{new}	2.58E+02	1.71E+01	8.04E+02	1.02E+02	1.45E+03	1.68E+02
	SACS	**2.20E+02**	3.03E+01	**5.29E+02**	5.48E+01	**7.28E+02**	1.50E+02
F_{30}	CS	1.36E+04	1.42E+04	**1.20E+04**	3.57E+03	2.95E+09	4.59E+09
	CV_{new}	3.95E+02	3.52E+01	2.03E+03	5.34E+01	6.02E+05	2.09E+04
	SACS	**8.36E+02**	3.36E+02	1.84E+04	5.93E+03	**1.16E+05**	4.13E+04

Table C.27 SACS algorithm: Analysis of effect of each parameter

Function	Algorithm	$pop_{adaptive}$&$BareBones$		$p_a = adaptive$&$BareBones$		$All_{adaptations}$	
		Mean	Std	Mean	Std	Mean	Std
F_1	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_2	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_3	SACS	2.21E+04	5.47E+03	3.11E+04	6.91E+03	**4.09E+03**	1.50E+03
F_4	SACS	9.26E+01	3.66E+01	1.44E+02	3.70E+01	**4.23E+01**	2.71E+01
F_5	SACS	3.04E+02	2.49E+01	2.81E+02	2.89E+01	**2.79E+02**	2.34E+01
F_6	SACS	8.68E+00	3.81E+00	4.55E+00	1.14E+00	**4.87E-01**	3.41E-01
F_7	SACS	7,28E+02	9.03E+01	2.37E+02	8.71E+01	**1.70E+02**	2.02E+01
F_8	SACS	3.15E+02	2.74E+01	2.69E+02	2.61E+01	**2.53E+02**	3.05E+01
F_9	SACS	1.09E+04	9.58E+02	1.08E+04	1.53E+03	**8.93E+03**	1.73E+03
F_{10}	SACS	4.97E+03	6.63E+02	5.23E+03	6.87E+02	**5.30E+03**	6.12E+02
F_{11}	SACS	1.47E+02	3.97E+01	1.65E+02	4.18E+01	**6.58E+01**	1.06E+01
F_{12}	SACS	1.00E+10	0.00E+00	1.00E+10	0.00E+00	1.00E+10	0.00E+00
F_{13}	SACS	1.00E+10	0.00E+00	9.80E+09	1.40E+09	9.21E+09	2.71E+09
F_{14}	SACS	3.51E+04	2.86E+04	3.85E+04	2.43E+04	**2.66E+03**	1.58E+03
F_{15}	SACS	1.10E+04	6.40E+03	1.38E+04	6.97E+03	**9.40E+03**	5.99E+03
F_{16}	SACS	7.44E+02	1.57E+02	8.81E+02	2.35E+02	**7.21E+02**	1.74E+02
F_{17}	SACS	6.23E+02	1.03E+02	5.91E+02	1.29E+02	**5.03E+02**	1.18E+02
F_{18}	SACS	1.79E+05	1.05E+05	2.88E+05	1.44E+05	**1.06E+05**	4.17E+04
F_{19}	SACS	6.62E+03	4.01E+03	1.72E+04	9.81E+03	**1.18E+02**	1.13E+03
F_{20}	SACS	4.44E+02	1.36E+02	4.30E+02	1.27E+02	**3.23E+02**	8.66E+01
F_{21}	SACS	1.59E+02	9.57E+01	2.90E+02	1.09E+02	**2.54E+02**	1.14E+02
F_{22}	SACS	1.02E+02	6.50E-01	1.10E+02	1.99E+00	**1.00E+02**	6.10E-03
F_{23}	SACS	5.52E+02	7.47E+01	5.86E+02	6.46E+01	**5.14E+02**	4.76E+01
F_{24}	SACS	6.88E+02	9.00E+01	6.79E+02	3.67E+01	**5.33E+02**	2.63E+01
F_{25}	SACS	5.28E+02	2.50E+01	5.48E+02	1.51E+01	**4.83E+02**	8.65E+00
F_{26}	SACS	3.00E+02	1.87E-01	3.09E+02	4.15E+00	**3.00E+02**	2.41E-04
F_{27}	SACS	6.11E+02	3.64E+01	6.48E+02	4.48E+01	**5.27E+02**	2.80E+01
F_{28}	SACS	4.87E+02	2.43E+01	4.93E+02	1.87E+01	**4.58E+02**	7.56E-04
F_{29}	SACS	**6.45E+02**	1.34E+02	7.64E+02	1.73E+02	7.28E+02	1.50E+02
F_{30}	SACS	1.22E+06	3.53E+05	2.39E+06	8.29E+05	**1.16E+05**	4.13E+04

Table C.28 SACS algorithm: Statistical results

		SaDE	JADE	SHADE	**LSHADE**	MVMO	CV1.0	CV_{new}	CS	SACS
	mean	1.21E+03	5.23E-14	**0.00E+00**	**0.00E+00**	1.33E-05	1.00E+10	1.00E+10	1.00E+10	1.00E+10
F_1	std	(1.97E+03)	(2.51E-14)	(0.00E+00)	(0.00E+00)	(5.60E-06)	(0.0000E+00)	(0.0000E+00)	(0.00E+00)	(0.00E+00)
	rank	+	+	+	+	+	=	=	=	
	mean	9.27E+01	1.31E+13	1.08E+12	**4.11E-01**	1.80E+17	1.00E+10	1.00E+10	1.00E+10	1.00E+10
F_2	std	(4.12E+01)	(8.53E+13)	(4.39E+12)	(6.68E-01)	(1.27E+18)	(0.00E+00)	(0.00E+00)	(0.00E+00)	(0.00E+00)
	rank	+	+	+	+	+	=	=	=	
	mean	2.71E+02	1.77E+04	**0.00E+00**	**0.00E+00**	5.30E-07	1.95E+04	8.71E+03	2.52E+05	4.09E+03
F_3	std	(8.28E+02)	(3.70E+04)	(0.00E+00)	(0.00E+00)	(1.09E-07)	(6.27E+03)	(4.08E+03)	(3.02E+04)	(1.50E+03)
	rank	+	-	+	+	+	-	-	-	
	mean	8.92E+01	4.96E+01	5.68E+01	8.18E+01	3.58E+01	1.16E+02	**2.67E+01**	1.28E+02	4.23E+01
F_4	std	(4.21E+01)	(4.71E+01)	(8.80E+00)	(4.08E+01))	(3.66E+01)	(6.27E+03)	(5.92E+00)	(2.44E+01)	(2.71E+01)
	rank	-	-	-	-	+	-	+	-	
	mean	9.23E+01	5.42E+01	3.28E+01	**1.22E+01**	8.07E+01	3.41E+02	2.39E+02	4.86E+02	2.29E+02
F_5	std	(1.86E+01)	(8.80E+00)	(5.03E+00)	(2.04E+00)	(1.64E+01)	(8.02E+01)	(3.80E+01)	(4.66E+01)	(2.34E+01)
	rank	+	+	+	+	+	-	-	-	
	mean	7.43E-03	**1.44E-13**	8.38E-04	5.69E-05	5.43E-03	4.85E+01	4.07E+01	4.13E+01	4.87E-01
F_6	std	(2.35E-02)	(9.11E-14)	(1.01E-03)	(3.71E-04)	(3.30E-03)	4.85E+01	(8.14E+00)	(6.32E+00)	(3.41E-01)
	rank	+	+	+	+	+	-	-	-	
	mean	1.80E+02	1.01E+02	8.09E+01	**6.32E+01**	1.23E+02	2.74E+02	2.22E+02	5.51E+02	1.70E+02
F_7	std	(1.97E+01)	(6.48E+00)	(3.78E+00)	(1.70E+00)	(1.27E+01)	(7.29E+01)	(3.49E+01)	(4.08E+01)	(2.02E+01)
	rank	-	+	+	+	+	-	-	-	
	mean	9.42E+01	5.52E+01	3.23E+01	**1.19E+01**	7.59E+01	3.29E+02	2.59E+02	4.82E+02	2.53E+02
F_8	std	(1.77E+01)	(7.76E+00)	(3.82E+00)	(2.27E+00)	(1.61E+01)	(7.29E+01)	(4.51E+01)	(4.67E+01)	(3.05E+01)
	rank	+	+	+	+	+	-	-	-	
	mean	4.83E+01	1.17E+00	1.11E+00	**0.00E+00**	7.38E+00	1.00E+04	1.06E+04	3.53E+04	8.93E+03
F_9	std	(6.29E+01)	(1.31E+00)	(9.37E-01)	(0.00E+00)	(5.77E+00)	(2.90E+03)	(3.10E+03)	(4.82E+03)	(1.73E+03)
	rank	+	+	+	+	+	-	-	-	
	mean	6.60E+03	3.75E+03	3.34E+03	**3.17E+03**	3.49E+03	7.10E+03	6.09E+03	7.39E+03	5.30E+03
F_{10}	std	(1.63E+03)	(2.54E+02)	(2.94E+02)	(2.54E+02)	(4.31E+02)	(5.34E+02)	(3.55E+02)	(3.26E+02)	(6.12E+02)
	rank	-	+	+	+	+	-	-	-	
	mean	1.09E+02	1.36E+02	1.20E+02	6.86E+01	6.74E+01	1.66E+02	1.18E+02	3.45E+02	**6.58E+01**
F_{11}	std	(3.54E+01)	(3.39E+01)	(2.93E+01)	(7.91E+00)	(8.72E+00)	(3.38E+01)	(1.91E+01)	(4.16E+01)	(1.06E+01)
	rank	-	-	-	-	-	-	-	-	
	mean	1.11E+05	5.14E+03	5.13E+03	2.16E+03	**1.29E+03**	1.00E+10	1.00E+10	1.00E+10	1.00E+10
F_{12}	std	(6.20E+04)	(3.32E+03)	(2.87E+03)	(4.51E+02)	(2.79E+02)	(0.00E+00)	(0.00E+00)	(0.00E+00)	(0.00E+00)
	rank	+	+	+	+	+	=	=	=	
	mean	1.21E+03	3.03E+02	2.65E+02	6.62E+01	**4.37E+01**	1.00E+10	9.80E+09	1.00E+10	9.21E+09
F_{13}	std	(1.45E+03)	(2.69E+02)	(1.49E+02)	(2.83E+01)	(1.76E+01)	(0.00E+00)	(1.40E+09)	(0.00E+00)	(2.79E+09)
	rank	+	+	+	+	+	-	-	-	
	mean	2.88E+03	1.05E+04	2.15E+02	2.76E+03	4.85E+01	2.05E+02	**3.98E+01**	3.26E+05	2.66E+03
F_{14}	std	(2.20E+03)	(3.11E+04)	(7.29E+01)	(4.51E+02)	(1.21E+01)	(2.13E+01)	(1.62E+01)	(1.60E+05)	(1.58E+02)
	rank	-	-	-	-	+	-	-	-	
	mean	3.35E+03	3.49E+02	3.22E+02	4.07E+01	4.46E+01	1.37E+09	2.85E+02	7.85E+09	**2.40E+02**
F_{15}	std	(2.79E+03)	(4.42E+02)	(1.42E+02)	(9.91E+00)	(1.12E+01)	(3.47E+09)	(3.54E+02)	(4.12E+09)	(5.99E+03)
	rank	-	-	-	-	-	-	-	-	

Table C.28 SACS algorithm: (Contd.) Statistical results

		SaDE	JADE	SHADE	**LSHADE**	MVMO	CV1.0	CV_{new}	CS	SACS
	mean	8.17E+02	8.56E+02	7.33E+02	**6.26E+01**	8.40E+02	1.53E+03	1.44E+03	1.76E+03	7.21E+02
F_{16}	std	(2.34E+02)	(1.75E+02)	(1.88E+02)	(2.83E+01)	(1.93E+02)	(2.74E+02)	(2.10E+02)	(2.37E+02)	(1.74E+02)
	rank	-	-	-	-	-	-	-	-	
	mean	5.48E+02	6.00E+02	5.16E+02	5.54E+02	5.19E+02	1.25E+03	**1.13E+02**	1.18E+03	5.03E+02
F_{17}	std	(1.53E+02)	(1.21E+02)	(1.11E+02)	(7.45E+01)	(1.33E+02)	(1.85E+02)	(1.92E+02)	(1.78E+02)	(1.18E+02)
	rank	-	-	-	-	-	-	-	-	
	mean	3.24E+04	1.89E+02	1.89E+02	3.92E+01	4.17E+01	5.21E+02	**1.51E+02**	1.43E+06	1.06E+05
F_{18}	std	(1.68E+04)	(1.25E+02)	(1.03E+02)	(1.10E+01)	(1.94E+01)	(1.19E+02)	(4.43E+01)	(5.89E+05)	(4.17E+04)
	rank	+	+	+	+	+	+	+	-	
	mean	1.13E+04	3.24E+02	1.59E+02	2.45E+01	**1.73E+01**	1.73E+02	5.57E+01	1.99E+08	1.18E+02
F_{19}	std	(1.68E+04)	(1.25E+03)	(568E+01)	(8.81E+00)	(5.13E+00)	(4.17E+02)	(1.10E+01)	(1.39E+09)	(1.13E+03)
	rank	-	-	-	+	+	-	+	-	
	mean	3.52E+02	4.38E+02	3.33E+02	**1.73E+02**	3.49E+02	1.05E+03	2.81E+02	1.04E+03	3.23E+02
F_{20}	std	(1.50E+02)	(1.33E+02)	(1.20E+02)	(7.92E+01)	(1.47E+02)	(2.14E+02)	(1.65E+02)	(1.67E+02)	(8.66E+01)
	rank	-	-	-	+	-	-	-	-	
	mean	2.87E+02	2.59E+02	2.33E+02	2.62E+02	2.77E+02	5.41E+02	**1.18E+02**	6.55E+02	2.54E+02
F_{21}	std	(1.36E+01)	(9.63E+00)	(5.11E+00)	(1.94E+01)	(1.60E+01)	(6.27E+01)	(8.77E+01)	(7.93E+01)	(1.14E+02)
	rank	-	-	+	-	-	-	-	-	
	mean	2.92E+03	3.33E+03	3.17E+03	2.49E+03	3.26E+03	7.33E+03	5.77E+03	8.19E+03	**1.00E+02**
F_{22}	std	(3.24E+03)	(1.80E+03)	(1.55E+03)	1.60E+03	(1.71E+03)	(1.99E+03)	(3.64E+02)	(4.08E+02)	(6.10E-03)
	rank	-	-	-	-	-	-	-	-	
	mean	5.22E+02	4.79E+02	4.59E+02	4.30E+02	5.04E+02	7.74E+02	**1.87E+02**	9.14E+02	5.14E+02
F_{23}	std	(2.05E+01)	(1.17E+01)	(8.75E+00)	(5.07E+02)	(1.71E+03)	(8.06E+01)	(5.11E+01)	(4.59E+01)	(4.76E+01)
	rank	-	+	+	+	+	-	+	-	
	mean	5.89E+02	5.31E+02	5.31E+02	5.62E+02	5.83E+02	8.32E+02	**3.25E+02**	1.01E+03	5.33E+02
F_{24}	std	(1.86E+01)	(7.62E+00)	(7.45E+00)	(2.33E+02)	(1.69E+01)	(1.21E+01)	(8.95E+01)	(6.38E+01)	(2.63E+01)
	rank	-	=	=	-	-	-	+	-	
	mean	5.71E+02	5.19E+02	5.06E+02	4.85E+02	5.09E+02	5.43E+02	**4.70E+02**	5.33E+02	4.83E+02
F_{25}	std	(3.05E+01)	(3.48E+01)	(3.64E+01)	(1.63E+01)	(3.12E+01)	(1.51E+01)	(2.26E+01)	(1.66E+01)	**(8.65E+00)**
	rank	-	-	-	-	-	-	+	-	
	mean	2.52E+03	1.61E+03	1.41E+03	1.14E+03	1.93E+03	2.48E+03	1.16E+03	4.57E+03	**3.00E+02**
F_{26}	std	(3.37E+02)	(1.21E+02)	(9.78E+01)	(4.49E+01)	(2.86E+02)	(1.88E+03)	(1.56E+03)	(1.82E+03)	**(2.41E-04)**
	rank	-	-	-	-	-	-	-	-	
	mean	7.10E+02	5.50E+02	5.49E+02	5.33E+02	5.43E+02	7.38E+02	**4.53E+02**	8.17E+02	5.27E+02
F_{27}	std	(6.65E+01)	(2.34E+01)	(2.78E+01)	(1.91E+01)	(1.75E+01)	(8.21E+01)	(7.17E+01)	(5.68E+01)	**(2.80E-01)**
	rank	-	-	-	-	-	-	+	-	
	mean	4.99E+02	4.91E+02	4.79E+02	4.73E+02	4.64E+02	4.94E+02	4.58E+02	5.12E+02	**4.58E+02**
F_{28}	std	(1.53E+01)	(2.08E+01)	(2.41E+01)	(2.24E+01)	(1.50E+01)	(1.93E+01)	(2.33E-01)	(1.88E+01)	**(7.56E-04)**
	rank	-	-	-	-	-	-	-	-	
	mean	5.11E+02	4.77E+02	4.87E+02	**3.51E+02**	4.89E+02	1.69E+03	1.45E+03	1.57E+03	7.28E+02
F_{29}	std	(1.37E+02)	(8.06E+01)	(1.05E+02)	(1.04E+01)	(1.40E+01)	(2.29E+02)	(1.68E+02)	(1.79E+02)	(1.50E+02)
	rank	+	+	+	+	+	-	-	-	
	mean	8.07E+05	6.68E+05	6.82E+05	6.53E+05	5.81E+05	4.64E+06	6.02E+05	2.95E+09	**1.16E+05**
F_{30}	std	(8.33E+04)	(9.25E+04)	(8.51E+04)	(7.32E+04)	(1.02E+04)	(8.59E+06)	(2.99E+04)	(4.59E+09)	**(4.13E+04)**
	rank	-	-	-	-	-	-	-	-	
w/l/t		10/20/0	12/17/1	14/16/0	16/14/0	15/15/0	1/26/3	7/20/3	0/27/3	

Table C.29 ALFPA algorithm: Comparison for population Size of 40

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	ALFPA	**0.00373**	**19.9900**	**3.1356**	3.3740
	AMMFPA	20.9300	21.1399	21.036	0.0549
	CFPA	20.8745	21.1110	21.022	0.0560
	GFPA	20.9000	21.1521	21.053	0.0569
	MMFPA	20.5278	21.0611	20.968	0.0902
	FPA	20.8640	21.1032	21.016	0.0560
f_2	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	3.63E−26	1.05E−16	6.37E−18	2.10E−17
	CFPA	1.12E−22	9.15E−15	1.86E−16	1.29E−15
	GFPA	4.22E−21	9.13E−13	4.76E−14	1.58E−13
	MMFPA	1.25E−23	5.05E−15	1.48E−16	7.21E−16
	FPA	6.47E−21	6.47E−15	5.27E−16	1.26E−15
f_3	ALFPA	0.3978	0.3970	0.3978	**3.36E−16**
	AMMFPA	0.3978	0.3970	0.3978	2.15E−08
	CFPA	0.3978	0.3978	0.3978	1.80E−07
	GFPA	0.3978	0.3978	0.3978	1.88E−07
	MMFPA	0.3978	0.3978	0.3978	6.55E−10
	FPA	0.3978	0.3978	0.3978	6.46E−08
f_4	ALFPA	6.05E+03	5.04E+06	6.58E+05	8.97E+05
	AMMFPA	8.0300	14.3300	11.703	1.4130
	CFPA	**7.5653**	15.4924	**11.600**	1.8970
	GFPA	15.360	21.6418	18.634	1.3218
	MMFPA	7.6993	13.5500	10.605	1.2262
	FPA	9.2850	18.1850	12.561	1.6038
f_5	ALFPA	2.6080	36.0340	12.619	8.1170
	AMMFPA	**1.0139**	1.13970	1.0540	0.0260
	CFPA	1.0214	1.11559	1.0503	0.0183
	GFPA	3.7351	14.9425	7.5357	2.4477
	MMFPA	1.1041	1.64950	1.2470	0.0989
	FPA	1.4316	4.98320	2.6520	0.7200
f_6	ALFPA	−3.3223	−3.2031	−3.2961	0.0498
	AMMFPA	−3.8620	−3.8620	−3.8620	2.90E−15
	CFPA	−3.8627	−3.8627	−3.8620	2.98E−15
	GFPA	−3.8627	−3.8627	−3.8627	2.97E−14
	MMFPA	−3.8620	−3.8627	−3.8627	2.68E−15
	FPA	−3.8620	−3.8627	−3.8627	**2.62E−15**
f_7	ALFPA	−3.3220	−3.2000	−3.2746	0.0589
	AMMFPA	−3.3220	−3.3220	−3.3223	2.16E−05
	CFPA	−3.3220	−3.3223	−3.3220	**2.76E−07**
	GFPA	−3.3223	−3.2031	−3.3156	0.0220
	MMFPA	−3.3223	−3.3223	−3.3223	8.05E−06
	FPA	−3.3223	−3.3223	−3.3223	5.93E−05
f_{10}	ALFPA	**32.795**	**84.7300**	**48.510**	**10.3373**
	AMMFPA	68.500	1.25E+02	93.788	13.7404
	CFPA	68.054	1.21E+02	97.5995	12.5135
	GFPA	1.68E+02	2.18E+02	1.93E+02	11.9709
	MMFPA	63.002	1.11E+02	84.0498	13.2060
	FPA	74.116	1.32E+02	97.7010	13.6325
f_{15}	ALFPA	−10.530	−2.87114	−9.96175	1.9733
	AMMFPA	−10.530	−10.5300	−10.5300	5.81E−07
	CFPA	−10.536	−10.5360	−10.5360	**9.02E−08**
	GFPA	−10.536	−10.5360	−10.5300	8.34E−05
	MMFPA	−10.536	−10.5360	−10.5360	2.55E−06
	FPA	−10.536	−10.5360	−10.5364	8.01E−06
f_{16}	ALFPA	−1.0316	−1.0316	−1.03162	**6.72E−16**
	AMMFPA	−1.0310	−1.0316	−1.03160	8.10E−15
	CFPA	−1.0316	−1.0316	−1.03162	6.50E−15
	GFPA	−1.0316	−1.0316	−1.03162	3.62E−12
	MMFPA	−1.0316	−1.0316	−1.03162	9.21E−14
	FPA	−1.0316	−1.0316	−1.03162	2.15E−12

Table C.30 ALFPA algorithm: Comparison for population Size of 60

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	ALFPA	**0.0066**	**2.9669**	**1.3161**	0.9676
	AMMFPA	20.8448	21.0875	20.9943	0.0449
	CFPA	20.8232	21.1178	21.0025	0.0557
	GFPA	20.7637	21.1612	21.0286	0.0738
	MMFPA	20.8230	21.0813	20.9578	0.0555
	FPA	20.8609	21.1054	20.9924	0.0591
f_2	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	5.83E−22	1.61E−17	1.11E−18	2.70E−18
	CFPA	1.77E−22	5.26E−17	2.20E−18	8.99E−18
	GFPA	1.43E−17	2.98E−12	1.651E−13	4.83E−13
	MMFPA	1.98E−20	9.94E−16	8.36E−17	1.88E−16
	FPA	8.43E−20	1.96E−14	1.11E−15	3.49E−15
f_3	ALFPA	0.3979	0.3979	0.3979	**3.36E−16**
	AMMFPA	0.3979	0.3979	0.3979	2.03E−10
	CFPA	0.3979	0.3979	0.3979	2.80E−10
	GFPA	0.3979	0.3979	0.3979	8.08E−09
	MMFPA	0.3979	0.3979	0.3979	3.03E−08
	FPA	0.3979	0.3979	0.3979	**6.33E−08**
f_4	ALFPA	4.1370	14.1461	9.3538	2.3602
	AMMFPA	8.5498	13.7908	11.6130	1.1440
	CFPA	7.9767	16.3025	12.5144	1.5910
	GFPA	17.4437	23.3525	20.0652	1.3255
	MMFPA	8.092	13.590	10.847	1.2510
	FPA	9.2407	15.7750	12.8020	1.2343
f_5	ALFPA	0.9780	3.0006	1.4037	**0.4652**
	AMMFPA	1.0301	1.1244	1.0682	0.0207
	CFPA	1.0387	1.1277	1.0656	0.0179
	GFPA	3.0325	10.8677	6.4368	1.5913
	MMFPA	1.157	1.58	1.309	0.0846
	FPA	1.7427	4.2795	2.6473	0.5529
f_6	ALFPA	−3.8628	−3.8628	−3.8628	**3.14E−15**
	AMMFPA	−3.8628	−3.8628	−3.8628	2.98E−15
	CFPA	−3.8628	−3.8628	−3.8628	2.90E−15
	GFPA	−3.8628	−3.8628	−3.8628	3.21E−14
	MMFPA	−3.8628	−3.8628	−3.8628	2.53E−15
	FPA	−3.8628	−3.8628	−3.8628	3.44E−15
f_{11}	ALFPA	1.65E+04	9.24E+06	1.48E+06	2.11E+06
	AMMFPA	5.50E+03	4.10E+04	1.74E+04	**9.09E+03**
	CFPA	**4.94E+03**	7.80E+04	1.77E+04	1.26E+04
	GFPA	9.58E+05	1.19E+07	1.19E+07	2.86E+06
	MMFPA	1.39E+04	1.77E+05	7.65E+04	3.82E+04
	FPA	2.21E+05	1.69E+06	6.26E+05	3.25E+05
f_{12}	ALFPA	**0.00E−00**	**0.00E−00**	**0.00E−00**	**0.00E−00**
	AMMFPA	0.00E−00	4.34E−13	3.84E−14	7.60E−14
	CFPA	0.00E−00	1.12E−12	4.83E−14	1.72E−13
	GFPA	4.44E−15	6.87E−10	2.17E−11	9.83E−11
	MMFPA	6.66E−16	2.11E−11	1.51E−12	3.67E−12
	FPA	0.000000	1.24E−12	1.97E−13	3.10E−13
f_{14}	ALFPA	−10.4029	−10.4029	−10.4029	**8.97E−15**
	AMMFPA	−10.4029	−10.4029	−10.4029	6.52E−08
	CFPA	−10.4029	−10.4029	−10.4029	5.97E−08
	GFPA	−10.4029	−10.4034	−10.4012	3.18E−04
	MMFPA	−10.4029	−10.4029	−10.4029	3.68E−06
	FPA	−10.4029	−10.4029	−10.4029	3.42E−05
f_{16}	ALFPA	−1.0316	−1.0316	−1.0316	**6.72E−16**
	AMMFPA	−1.0316	−1.0316	−1.0316	3.61E−15
	CFPA	−1.0316	−1.0316	−1.0316	8.71E−16
	GFPA	−1.0316	−1.0316	−1.0316	3.53E−12
	MMFPA	−1.0316	−1.0316	−1.0316	7.81E−14
	FPA	−1.0316	−1.0316	−1.0316	4.54E−12

Table C.31 ALFPA algorithm: Comparison for population Size of 80

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	ALFPA	**0.00626**	**2.14065**	**0.41643**	0.6685
	AMMFPA	20.8654	21.0827	20.9928	0.0423
	CFPA	20.8136	21.0619	20.9733	0.0567
	GFPA	20.8312	21.0996	21.0214	0.0564
	MMFPA	20.8157	21.0452	20.9444	0.0541
	FPA	20.8486	21.0715	20.9938	0.0465
f_4	ALFPA	**5.52654**	**12.7332**	**8.44947**	1.6115
	AMMFPA	9.84276	15.2794	12.6325	1.3574
	CFPA	10.0031	16.4138	12.8217	1.4941
	GFPA	18.1922	24.1149	21.5050	1.4275
	MMFPA	7.1676 3	13.9665	11.2519	1.3002
	FPA	9.9236	16.8457	13.7463	1.3274
f_5	ALFPA	**0.0434**	**1.0897**	**0.5328**	0.2555
	AMMFPA	1.0566	1.2028	1.0904	0.0245
	CFPA	1.0457	1.1218	1.0846	0.0164
	GFPA	3.6287	4.0729	9.4301	2.4844
	MMFPA	1.2440	1.6534	1.3946	0.0863
	FPA	2.3040	4.3090	3.0930	0.4896
f_6	ALFPA	–3.8628	–3.8628	–3.8628	3.14E–15
	AMMFPA	–3.8628	–3.8628	–3.8628	2.90E–15
	CFPA	–3.8628	–3.8628	–3.8628	2.86E–15
	GFPA	–3.8628	–3.8628	–3.8628	7.83E–14
	MMFPA	–3.8628	–3.8628	–3.8628	2.45E–15
	FPA	–3.8628	–3.8628	–3.8628	**2.52E–15**
f_7	ALFPA	–3.3224	–3.3224	–3.3224	0.0230
	AMMFPA	–3.3224	–3.2031	–3.3170	1.60E–07
	CFPA	**–3.3224**	**–3.3224**	**–3.3224**	**1.63E–07**
	GFPA	–3.3224	–.3224	–3.3224	0.0017
	MMFPA	–3.3224	–.3224	–3.3224	5.77E–06
	FPA	–3.3224	–3.3224	–3.3224	6.98E–05
f_8	ALFPA	**4.24E–04**	8.54	2.023	1.908
	AMMFPA	0.855	3.10	1.867	0.504
	CFPA	1.1781	3.0387	1.9842	0.4685
	GFPA	7.3902	44.8770	15.9797	6.0123
	MMFPA	1.5601	4.2341	2.8578	0.5523
	FPA	4.3020	10.259	6.7020	1.4720
f_9	ALFPA	**0.335**	47.131	14.658	11.019
	AMMFPA	3.886	10.229	6.418	1.549
	CFPA	3.4130	9.9119	6.1760	1.3344
	GFPA	41.2730	8.07E+04	7.40E+03	1.34E+04
	MMFPA	6.4905	7.6718	1.7207 26.605	2.6676
	FPA	12.514	45.990		7.5090
f_{10}	ALFPA	**25.658**	72.952	**49.319**	11.261
	AMMFPA	77.488	1.31E+02	1.01E+02	12.747
	CFPA	76.6478	125.5905	100.6522	11.6035
	GFPA	174.5894	253.5865	212.980	217.3889
	MMFPA	65.8232 73.813	21.5544	89.2597	10.7760
	FPA		1.34E+02	1.03E+02	12.237
f_{11}	ALFPA	**2.86E+02**	5.35E+04	**7.33E+03**	**9.49E+03**
	AMMFPA	1.00E+04	6.80E+04	2.43E+04	1.16E+04
	CFPA	8.28E+03	4.88E+04	2.55E+04	1.26E+04
	GFPA	2.61E+06	3.54E+07	1.24E+07	7.03E+06
	MMFPA	4.37E+04	2.35E+05	1.16E+05	4.56E+04
	FPA	3.43E+05	3.34E+06	1.18E+06	6.73E+05
f_{17}	ALFPA	–1	–1	–1	**0.00E–00**
	AMMFPA	–1	–1	–1	**0.00E–00**
	CFPA	–1	–1	–1	**0.00E–00**
	GFPA	–1	–1	–1	1.89E–16
	MMFPA	–1	–1	–1	**0.00E–00**
	FPA	–1	–1	–1	**0.00E–00**

Table C.32 ALFPA algorithm: Comparison with respect to other algorithms

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	ABC	20.0000	20.0120	20.0032	1.26E−02
	DE	20.0000	20.0000	20.0000	3.02E−09
	BA	19.9990	20.0000	19.9999	2.22E−04
	GWO	20.5887	20.9715	21.8017	9.21E−02
	FA	19.9976	20.0023	20.0007	6.97E−04
	ALFPA	**6.60E−03**	**2.96E−00**	**1.31E−00**	9.67E−01
f_4	ABC	**2.97E−00**	**8.32E−00**	5.48E−00	2.38E−00
	DE	2.60E+01	4.27E+02	1.14E+02	9.03E+01
	BA	1.27E+01	2.32E+02	6.46E+01	4.59E+01
	GWO	1.10E−00	1.99E+01	5.79E−00	4.38E−00
	FA	2.60E−00	1.00E+01	4.96E−00	1.37E−00
	ALFPA	4.13E−00	1.41E+01	9.35E−00	**2.36E−00**
f_5	ABC	7.38E−01	1.06E−00	8.84E−01	8.01E−02
	DE	4.67E+01	3.03E+02	1.36E+02	5.69E+01
	BA	6.26E+01	2.00E+02	1.20E+02	3.40E+01
	GWO	**0.00E−00**	1.34E−02	1.20E−03	3.40E−03
	FA	2.63E−03	6.40E−03	4.20E−03	1.00E−03
	ALFPA	9.78E−01	3.00E−00	1.40E−00	4.65E−01
f_7	ABC	−3.3223	−3.2030	−3.3148	2.85E−02
	DE	−3.3224	−2.8657	−3.1628	7.07E−02
	BA	−3.3224	−3.2032	−3.2842	5.62E−02
	GWO	−3.3224	−3.1327	−3.2752	6.48E−02
	FA	−3.3223	−3.1666	−3.2612	6.19E−02
	ALFPA	**−3.3224**	**−3.3224**	**−3.3224**	**2.45E−08**
f_8	ABC	1.56E−02	1.02E−00	1.87E−01	5.01E+07
	DE	3.24E+06	6.54E+08	1.85E+08	1.94E+08
	BA	8.51E+00	7.72E+06	8.21E+05	1.41E+07
	GWO	8.43E−07	7.19E−02	2.35E−02	1.68E−02
	FA	1.23E−05	**6.82E−05**	**2.55E−05**	**1.00E−05**
	ALFPA	5.59E−01	1.32E+03	4.23E+01	1.90E+02
f_9	ABC	1.35E−01	3.07E−00	8.32E−01	6.10E−01
	DE	3.54E+07	1.39E+09	3.19E+08	3.35E+08
	BA	6.37E+04	5.64E+07	9.39E+06	1.20E+07
	GWO	9.63E−06	6.86E−01	2.67E−01	1.93E−01
	FA	1.10E−04	**1.14E−02**	**5.82E−04**	**1.50E−03**
	ALFPA	4.62E−00	2.76E+04	1.75E+03	4.77E+03
f_{10}	ABC	6.06E−00	3.07E+01	1.44E+01	6.04E−00
	DE	9.87E+01	2.14E+02	1.53E+02	2.55E+01
	BA	3.18E+01	2.08E+02	7.45E+01	2.69E+01
	GWO	**0.00E−00**	**4.50E−00**	**1.10E−01**	**6.50E−01**
	FA	1.79E+01	6.16E+01	3.44E+01	1.01E+01
	ALFPA	2.20E+01	6.32E+01	4.26E+01	1.12E+01
f_{13}	ABC	−10.1525	−5.0589	−9.8957	9.97E−01
	DE	−10.1532	−1.8513	−5.3191	2.96E−00
	BA	−10.1532	−2.6305	−6.2930	3.26E−00
	GWO	−10.1532	−5.0552	−9.6468	1.53E−00
	FA	−10.1531	−2.6828	−9.8543	1.47E−00
	ALFPA	**−10.1532**	**−10.1532**	**−10.1532**	**8.97E−15**
f_{14}	ABC	−10.5056	−5.1516	−10.2474	1.05E−00
	DE	−10.5064	−2.8027	−9.3903	2.32E−00
	BA	−10.5048	−1.6766	−5.6230	3.63E−00
	GWO	−10.5049	−2.4217	−10.1585	7.57E−04
	FA	−10.4028	−10.4026	−10.4022	8.30E−07
	ALFPA	**−10.4028**	**−10.4028**	**−10.4028**	**8.95E−15**
f_{17}	ABC	−1.0000	−0.9996	−0.9996	1.80E−01
	DE	−1.0000	−0.9702	−0.9988	5.70E−03
	BA	−1.0000	−1.0000	−1.0000	5.92E−11
	GWO	−1.0000	−1.0000	−1.0000	8.71E−08
	FA	−1.0000	−1.0000	−1.0000	1.36E−09
	ALFPA	−1.0000	−1.0000	−1.0000	**0.00E−00**

Table C.33 EFPA algorithm: CEC 2005 results

Function	Algorithm	Best	Worst	Mean	Standard Deviation
$f_1(x)$	DE	4.43E+03	2.97E+04	1.54E+04	6.22E+03
	ABC	2.74E+03	3.67E+04	2.74E+04	5.93E+05
	BA	8.08E+03	3.54E+04	1.96E+04	7.01E+03
	FPA	1.00E+02	1.00E+03	3.61E+02	1.66E+02
	EFPA	3.35E−15	4.72E−07	1.82E−08	6.85E−08
$f_2(x)$	DE	1.64E+01	3.82E+01	2.54E+01	4.51E+00
	ABC	2.53E+00	5.90E+00	3.90E+00	8.40E−01
	BA	2.51E+01	4.30E+01	3.36E+01	4.24E+00
	FPA	3.83E+01	5.72E+01	5.12E+01	4.64E+00
	EFPA	1.34E+01	2.66E+01	2.00E+01	3.10E+00
$f_3(x)$	DE	7.95E−05	2.29E−01	4.64E−02	5.37E−02
	ABC	0.00E−00	0.00E−00	0.00E−00	0.00E−00
	BA	3.06E−14	3.47E−01	8.95E−02	9.22E−02
	FPA	0.00E−00	1.98E−12	1.03E−13	3.39E−13
	EFPA	0.00E−00	0.00E−00	0.00E−00	0.00E−00
$f_4(x)$	DE	4.89E+01	3.68E+02	1.54E+02	7.19E+01
	ABC	1.06E+00	1.76E+00	1.25E+00	1.73E−01
	BA	2.94E+00	1.18E+01	5.77E+00	1.63E+00
	FPA	5.71E−01	1.04E+00	9.25E−01	9.76E−02
	EFPA	0.00E−00	1.45E−01	2.90E−03	2.06E−02
$f_5(x)$	DE	1.78E+06	6.92E+08	2.24E+08	2.27E+08
	ABC	3.49E−02	2.77E+07	1.07E+08	8.91E+07
	BA	3.40E+05	5.55E+07	1.58E+07	1.53E+07
	FPA	2.81E+00	1.11E+01	6.81E+00	1.87E+00
	EFPA	1.25E+00	3.1E+01	8.93E+00	6.40E+00
$f_6(x)$	DE	2.00E+01	2.00E+01	2.00E+01	0.00E+00
	ABC	2.00E+01	2.02E+01	2.00E+01	5.30E−03
	BA	1.97E+01	2.00E+01	1.99E+01	2.92E−02
	FPA	2.05E+01	2.11E+01	2.10E+01	8.37E−02
	EFPA	7.29E+00	1.33E+01	9.95E+00	1.29E+00
$f_7(x)$	DE	4.21E+08	3.78E+10	8.63E+09	1.00E+10
	ABC	2.18E+04	3.61E+05	2.96E+04	5.39E+04
	BA	8.62E+08	1.29E+10	3.07E+09	2.10E+09
	FPA	1.48E+05	6.49E+06	2.30E+06	1.72E+06
	EFPA	0.00E−00	0.00E−00	0.00E−00	0.00E−00

Table C.34 BFP algorithm: CEC 2005 results

Objective Function	Algorithm	Best	Worst	Mean	Standard Deviation
$f_1(x)$	BFP	**1.15E−05**	**1.70E+01**	**1.20E−00**	**3.92E−00**
	BA	1.31E+06	8.04E+07	3.01E+07	1.77E+07
	FPA	1.03E+01	6.81E+01	3.77E+01	1.56E+01
	FA	1.78E−00	8.97E+01	2.99E+01	2.77E+01
$f_2(x)$	BFP	1.23E+01	**1.75E+01**	**1.54E+01**	**9.80E−01**
	BA	1.99E+01	2.00E+01	1.99E+01	1.40E**−02**
	FPA	2.09E+01	2.10E+01	2.10E+01	1.56E−02
	FA	1.99E+01	2.00E+01	1.99E+00	9.00E+00
$f_3(x)$	BFP	1.85E+06	1.76E+08	3.73E+07	**4.85E+06**
	BA	1.69E+08	1.51E+09	4.87E+08	2.80E+08
	FPA	1.86E+05	1.92E+06	**7.61E+05**	**3.78E+05**
	FA	**1.83E+06**	1.23E+07	5.30E+06	5.84E+06
$f_4(x)$	BFP	**1.03E+01**	4.01E+01	**3.07E+01**	**1.63E+00**
	BA	2.34E+01	3.81E+01	3.09E+01	3.56E+00
	FPA	4.24E+01	5.66E+01	5.30E+01	3.29E+00
	FA	2.19E+01	3.45E+01	2.87E+01	3.10E+00
$f_5(x)$	BFP	**0.00E−00**	1.05E−01	2.86E−02	2.82E−02
	BA	3.10E−03	3.12E−01	7.36E−02	7.21E−02
	FPA	4.44E−16	4.48E−12	3.24E−13	9.66E−13
	FA	**1.37E−14**	**5.83E−10**	**1.18E−10**	**1.33E−10**
$f_6(x)$	BFP	**6.62E−00**	2.77E+01	1.58E+01	**5.52E−00**
	BA	6.14E+04	4.83E+07	6.09E+06	9.55E+06
	FPAFA	2.93E−00	**8.18E−00**	4.72E−00	1.24E−00
		1.71E−00	2.67E−00	1.93E−00	**2.50E−01**
$f_7(x)$	BFP	2.82E+01	7.78E+01	5.39E+01	**1.43E−01**
	BA	2.14E+06	6.48E+07	2.06E+07	1.70E+07
	FPAFA	1.33E+01	1.44E+02	3.32E+01	2.94E+01
		6.16E−05	**1.11E−01**	**5.15E−04**	**2.00E−02**
$f_8(x)$	BFP	2.06E+01	**2.92E+01**	**2.29E+01**	**2.17E−00**
	BA	4.60E+02	1.95E+04	4.18E+03	4.37E+03
	FPAFA	1.37E+02	1.25E+03	4.48E+02	3.07E+02
		2.69E+01	1.00E+04	1.67E+03	3.02E+03
$f_9(x)$	BFP	1.54E−00	4.07E+01	1.80E+01	1.27E+01
	BA	1.03E+02	4.03E+02	1.72E+02	5.83E+01
	FPA	**0.00E−00**	**2.54E−13**	**4.36E−14**	**6.24E−14**
	FA	5.24E−04	1.75E−02	3.50E−03	4.80E−03
$f_{10}(x)$	BFP	0.3979	0.3979	0.3979	**0.00E−00**
	BA	0.3979	0.3979	0.3979	1.86E−10
	FPA	0.3979	0.3979	0.3979	7.30E−09
	FA	0.3979	0.3979	0.3979	5.77E−10
$f_{11}(x)$	BFP	1.68E−09	**1.38E−06**	**1.01E−07**	**3.73E−07**
	BA	5.85E+03	2.97E+04	1.53E+04	5.70E+03
	FPA	5.81E+01	3.60E+01	2.16E+02	7.52E+01
	FA	**0.00E−00**	1.58E−05	2.51E−06	3.70E−06
$f_{12}(x)$	BFP	**0.00E−00**	6.37E−01	3.19E−02	1.42E−01
	BA	4.88E−12	6.67E−01	7.48E−02	1.96E−01
	FPAFA	1.24E−20	**4.40E−16**	**8.52E−17**	**1.24E−16**
		2.37E−13	2.74E−10	1.17E−10	7.99E−11
$f_{13}(x)$	BFP	−1	−1	−1	**0.00E−00**
	BA	−1	−1	−1	1.36E−10
	FPA	−1	−1	−1	**1.04E−10**
	FA	−1	−1	−1	2.93E−09

Table C.35 NMR algorithm: Comparison with respect to other algorithms

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_1	BA	1.29E+01	1.72E+01	1.53E+01	1.04E−00
	DE	1.93E+01	2.09E+01	2.05E+01	3.00E−01
	FA	2.82E−02	1.21E−01	6.67E−02	2.09E−02
	FPA	1.10E+01	1.59E+01	1.37E+01	1.05E−00
	NMR	3.76E−12	2.70E−05	1.25E−06	5.00E−06
f_2	BA	1.20E−04	7.27E+02	2.86E+01	1.18E+02
	DE	1.29E−09	4.17E−00	1.77E−01	7.25E−01
	FA	7.90E−06	3.17E−00	1.99E−01	6.73E−01
	FPA	1.65E−01	2.51E−00	1.02E−00	6.28E−01
	NMR	8.60E−02	4.04E+01	1.02E+01	9.46E−00
f_3	BA	9.98E−01	1.71E+01	6.89E−00	4.63E−00
	DE	9.98E−01	5.92E−00	1.15E−00	7.28E−01
	FA	9.98E−01	3.96E−00	1.85E−00	7.78E−01
	FPA	9.98E−01	9.98E−01	9.98E−01	4.68E−05
	NMR	9.98E−01	8.86E−00	2.10E−00	1.57E−00
f_4	BA	8.10E+01	1.00E+05	1.51E+04	2.04E+04
	DE	1.07E+06	2.20E+06	1.64E+06	2.85E+05
	FA	7.21E−01	2.03E+01	3.75E−00	4.08E−00
	FPA	2.28E+03	2.16E+04	1.05E+04	4.61E+03
	NMR	9.86E−01	1.00E−00	9.98E−01	5.16E−05
f_5	BA	−7.4811	−3.5039	−5.7971	9.47E−01
	DE	−9.4553	−5.0448	−7.8624	1.13E−00
	FA	−9.5663	−7.1786	−8.3731	5.76E−01
	FPA	−6.6852	−5.0034	−6.0206	3.35E−01
	NMR	−8.0542	−5.5740	−6.4179	5.42E−01
f_6	BA	6.15E−01	3.21E+01	5.16E+02	7.47E+01
	DE	2.23E+03	6.87E+03	1.44E+04	3.24E+03
	FA	5.52E−01	3.36E−00	1.52E+01	2.86E−00
	FPA	1.37E+02	2.78E+02	4.83E+02	8.98E+01
	NMR	3.59E−22	2.21E−11	1.04E−09	1.48E−10
f_7	BA	1.11E−00	1.26E+01	4.51E−00	2.30E−00
	DE	1.62E+01	1.56E+02	1.07E+02	2.33E+01
	FA	7.40E−03	8.90E−02	2.77E−02	1.65E−02
	FPA	3.90E−01	2.27E−00	9.53E−01	4.29E−01
	NMR	1.14E−04	8.40E−03	1.80E−03	3.03E−04
f_8	BA	8.99E−00	1.60E+01	1.27E+01	1.73E−00
	DE	1.70E+01	2.84E+01	2.57E+01	2.28E−00
	FA	2.99E−01	7.99E−01	4.71E−01	1.29E−01
	FPA	6.04E−00	1.05E+01	7.99E−00	8.66E−01
	NMR	1.27E−09	9.99E−02	3.37E−02	4.43E−02
f_9	BA	8.02E+03	2.65E+04	1.42E+04	3.78E+03
	DE	2.18E+04	8.02E+04	6.28E+04	1.07E+04
	FA	4.20E−03	3.25E−02	1.55E−02	6.50E−03
	FPA	2.12E+03	7.90E+03	3.82E+03	1.14E+03
	NMR	7.57E−25	1.34E−08	2.69E−10	1.89E−09
f_{11}	BA	5.91E−09	1.41E−07	5.71E−08	3.02E−08
	DE	1.77E−01	1.43E−00	7.14E−01	3.03E−01
	FA	1.86E−08	2.61E−06	6.09E−07	5.04E−07
	FPA	8.31E−06	1.30E−03	2.51E−04	2.68E−04
	NMR	1.22E−32	2.40E−13	6.06E−15	3.49E−14
f_{12}	BA	4.16E−00	2.15E+02	5.55E+01	4.46E+01
	DE	5.91E+01	5.74E+02	1.82E+02	1.06E+02
	FA	4.92E−00	2.31E+01	1.25E+01	4.05E−00
	FPA	2.17E+01	8.23E+01	4.64E+01	1.38E+01
	NMR	2.29E−25	1.45E−08	4.50E−10	2.22E−09
f_{13}	BA	4.95E−00	7.94E+03	1.01E+03	1.30E+03
	DE	6.15E+02	3.28E+04	7.71E+03	7.54E+03
	FA	1.97E−00	1.54E+02	6.55E+01	2.64E+01
	FPA	3.13E+02	3.08E+03	1.05E+03	5.17E+02
	NMR	6.93E−24	1.05E+03	6.42E+01	1.93E+02

Table C.35 NMR algorithm: (Contd.) Comparison with respect to other algorithms

Function	Algorithm	Best	Worst	Mean	Standard Deviation
f_{14}	BA	1.01E−12	7.65E−01	1.03E−01	2.21E−01
	DE	0.00E−00	1.96E−12	3.92E−14	2.77E−13
	FA	9.44E−12	3.10E−09	8.63E−10	8.45E−10
	FPA	6.56E−10	1.21E−05	7.84E−07	2.30E−06
	NMR	2.73E−07	5.46E−02	4.00E−03	1.04E−02
f_{15}	BA	3.63E−01	7.76E+04	2.03E+04	2.19E+04
	DE	8.01E+05	1.82E+06	1.49E+06	2.11E+05
	FA	3.16E−00	7.34E+01	1.94E+01	1.56E+01
	FPA	4.10E+04	1.35E+05	8.84E+04	2.57E+04
	NMR	2.36E−20	5.51E−09	1.22E−10	7.80E−10
f_{16}	BA	−1.0000	−8.11E−05	−0.9800	1.41E−01
	DE	−1.0000	−1.0000	−1.0000	0.00E−00
	FA	−1.0000	−1.0000	−1.0000	3.48E−09
	FPA	−1.0000	−1.0000	−1.0000	6.26E−10
	NMR	−1.0000	−1.0000	−1.0000	4.22E−02
f_{17}	BA	1.35E+08	1.75E+09	5.12E+08	2.96E+08
	DE	1.31E+08	1.12E+09	5.86E+08	2.75E+08
	FA	9.00E+05	1.85E+07	5.17E+06	3.72E+06
	FPA	1.91E+07	8.78E+07	5.00E+07	1.67E+07
	NMR	2.93E−19	3.02E−05	6.46E−07	4.27E−06
f_{18}	BA	3.0000	3.0000	3.0000	1.14E+01
	DE	3.0000	3.0000	3.0000	2.71E−15
	FA	3.0000	3.0000	3.0000	2.96E−08
	FPA	3.0000	3.0000	3.0000	7.49E−09
	NMR	3.0000	3.0000	3.0000	7.74E−01
f_{19}	BA	1.45E−01	2.17E+01	3.71E−00	4.25E−00
	DE	0.00E−00	1.92E−01	2.09E−02	5.38E−02
	FA	4.43E−08	5.54E−06	1.41E−06	1.44E−06
	FPA	2.76E−09	5.66E−05	1.03E−05	1.37E−05
	NMR	0.00E−00	1.20E−03	3.26E−04	1.80E−03
f_{20}	BA	−3.8628	−6.32E−08	−3.1546	1.19E−00
	DE	−3.8628	−3.8628	−3.8628	2.96E−15
	FA	−3.8628	−3.5663	−3.8535	4.77E−02
	FPA	−3.8628	−3.7700	−3.8493	2.04E−02
	NMR	−3.8628	−3.8511	−3.8611	2.05E−03
f_{21}	BA	−3.3224	−3.2031	−3.2747	5.90E−02
	DE	−3.3224	−3.2029	−3.2241	4.54E−02
	FA	−3.3224	−3.0783	−3.2638	6.91E−02
	FPA	−3.3180	−3.2172	−3.2784	2.34E−02
	NMR	−3.3224	−3.0722	−3.1125	2.01E−02
f_{22}	BA	3.11E+04	3.96E+07	6.48E+06	9.39E+06
	DE	2.53E+08	7.38E+08	5.34E+08	1.03E+08
	FA	1.42E−04	3.90E−03	6.37E−04	5.88E−04
	FPA	3.24E+01	4.79E+05	5.02E+04	8.90E+04
	NMR	6.46E−01	1.66E−00	1.32E−00	2.70E−01
f_{25}	BA	4.46E−14	2.62E−01	5.44E−02	6.03E−02
	DE	0.00E−00	4.29E−02	6.80E−03	1.03E−02
	FA	2.64E−12	1.71E−02	1.60E−03	3.80E−03
	FPA	1.48E−09	1.47E−06	1.97E−07	3.20E−07
	NMR	0.00E−00	1.70E−04	3.46E−06	2.41E−05
f_{26}	BA	−1.0316	−1.0316	−1.0316	9.89E−08
	DE	−1.0316	−1.0316	−1.0316	6.72E−06
	FA	−1.0316	−1.0316	−1.0316	2.55E−09
	FPA	−1.0316	−1.0316	−1.0316	6.67E−08
	NMR	−1.0316	−1.0316	−1.0316	4.47E−09
f_{27}	BA	2.28E+01	3.83E+01	3.14E+01	3.23E−00
	DE	1.92E+01	3.93E+01	2.86E+01	4.77E−00
	FA	8.10E−00	2.15E+01	1.60E+01	2.80E−00
	FPA	3.16E+01	3.63E+01	3.36E+01	1.06E−00
	NMR	6.39E−13	5.50E−03	3.02E−04	8.45E−08

Table C.36 NMR algorithm: Comparison with respect to other algorithms from literature

Function		WOA	GWO	PSO	GSA	FEP	NMR
f_1	Mean	7.40E−00	1.06E−13	2.76E−01	6.20E−02	1.80E−02	2.69E−05
	Std	9.89E−00	7.78E−02	5.09E−01	2.36E−01	2.10E−03	6.94E−05
f_3	Mean	2.11E−00	4.04E−00	3.62E−00	5.85E−00	1.22E−00	2.42E−00
	Std	2.94E−00	4.25E−00	2.56E−00	3.83E−00	5.60E−01	1.79E−00
f_9	Mean	1.41E−30	6.28E−28	1.36E−03	2.53E−16	5.70E−04	9.11E−09
	Std	4.91E−30	6.34E−05	2.02E−04	9.67E−17	1.30E−04	4.86E−08
f_{10}	Mean	3.11E−00	8.16E−01	1.02E−03	2.50E−16	0.00E−00	0.00E−00
	Std	5.32E−01	1.26E−03	8.28E−05	1.74E−16	0.00E−00	0.00E−00
f_{19}	Mean	2.89E−04	4.48E−03	9.21E−03	2.77E+01	1.60E−02	2.70E−03
	Std	1.58E−03	6.65E−03	7.72E−03	5.04E−00	2.20E−01	1.28E−02
f_{20}	Mean	−3.85616	−3.86263	−3.86278	−3.86278	−3.86000	−3.85060
	Std	2.70E−03	−3.86278*	2.58E−15	2.29E−15	1.40E−05	3.99E−01
f_{21}	Mean	−2.98105	−3.28865	−3.26634	−3.31778	−3.27000	−3.31760
	Std	3.76E−01	−3.25065*	6.05E−02	2.30E−02	5.90E−02	3.84E−01
f_{22}	Mean	3.39E−01	5.34E−02	6.91E−03	1.79E−00	9.20E−06	1.28E−00
	Std	2.14E−01	2.07E−02	2.63E−02	9.51E−01	3.60E−06	3.20E−01
f_{23}	Mean	1.88E−00	6.54E−01	6.67E−03	8.89E−00	1.60E−04	2.90E−00
	Std	2.66E−01	4.47E−03	8.90E−03	7.12E−00	7.30E−05	1.00E−03
f_{24}	Mean	0.00E−00	3.10E−01	4.67E+01	2.59E+01	4.60E−02	9.00E−03
	Std	0.00E−00	4.73E+01	1.16E+01	7.47E−00	1.20E−02	4.90E−02
f_{26}	Mean	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
	Std	4.20E−07	−1.03163*	6.25E−16	4.88E−16	4.90E−07	4.30E−03

APPENDIX D

Statistical Test Result Tables

Table D.1 CV1.0 algorithm: p-test values with respect to algorithms

Function	DE	BA	GWO	FA	FPA	CV 1.0
f_1	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_2	3.31E-20	3.31E-20	∼	4.33E-02	3.31E-20	NA
f_3	7.06E-18	7.06E-18	NA	7.06E-18	7.06E-18	5.01E-01
f_4	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_5	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_6	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_7	3.31E-20	3.31E-20	∼	3.31E-20	3.31E-20	NA
f_8	7.06E-18	7.06E-18	7.06E-18	3.42E-13	7.06E-18	NA
f_9	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_{10}	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_{11}	7.06E-18	7.06E-18	1.54E-02	7.06E-18	7.06E-18	NA
f_{12}	7.06E-18	8.46E-18	NA	1.44E-17	7.06E-18	7.06E-18
f_{13}	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_{14}	4.67E-19	3.31E-20	8.22E-02	3.31E-20	3.31E-20	NA
f_{15}	7.06E-18	7.06E-18	7.58E-04	NA	7.06E-18	4.83E-13
f_{16}	7.06E-18	7.06E-18	1.40E-12	NA	7.06E-18	7.06E-18
f_{17}	4.69E-19	3.31E-20	∼	3.31E-20	3.31E-20	NA
f_{18}	3.31E-20	3.31E-20	3.31E-20	3.31E-20	3.31E-20	NA
f_{19}	7.06E-18	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_{20}	8.53E-01	5.96E-18	5.96E-18	5.96E-18	5.96E-18	NA
f_{21}	2.40E-05	7.06E-18	7.06E-18	7.06E-18	7.06E-18	NA
f_{22}	1.40E-03	7.55E-19	7.55E-19	7.55E-19	7.55E-19	NA
f_{23}	7.06E-18	7.06E-18	7.06E-18	3.89E-14	7.06E-18	NA
f_{24}	3.54E-01	9.47E-19	9.47E-19	9.47E-19	9.47E-19	NA
f_{25}	4.28E-05	4.20E-17	2.19E-02	7.43E-01	7.06E-18	NA
f_{26}	3.23E-05	3.31E-20	3.31E-20	3.31E-20	3.31E-20	NA

Table D.2 ALFPA algorithm: p-test values with respect to other algorithms

Function	ABC	DE	BA	GWO	FA	ALFPA
f_1	7.06E−18	5.18E−14	7.06E−18	∼	7.06E−18	NA
f_2	∼	8.34E−18	3.31E−20	3.31E−20	3.31E−20	NA
f_5	7.06E−18	3.31E−20	7.06E−18	7.72E−12	NA	7.06E−18
f_7	6.90E−16	6.90E−16	6.47E−17	1.20E−17	1.20E−17	NA
f_8	7.06E−18	3.31E−20	7.06E−18	2.29E−15	NA	7.06E−18
f_{10}	2.13E−19	4.18E−20	2.13E−19	NA	2.13E−19	2.13E−19
f_{11}	7.06E−18	3.31E−20	7.06E−18	NA	7.06E−18	7.06E−18
f_{13}	3.31E−20	3.04E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{14}	3.31E−20	1.87E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{15}	3.31E−20	2.33E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{16}	3.31E−20	2.74E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{17}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA

Table D.3 EFPA algorithm: p-test values for CEC 2005 results

Objective Function	BA	FPA	ABC	DE	EFPA
$f_1(x)$	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
$f_2(x)$	7.06E−18	7.06E−18	NA	3.25E−09	7.06E−18
$f_3(x)$	3.31E−20	6.89E−17	NA	3.31E−20	NA
$f_4(x)$	4.73E−20	4.73E−20	4.73E−20	4.73E−20	NA
$f_5(x)$	7.06E−18	7.70E−02	4.21E−04	7.06E−18	NA
$f_6(x)$	7.06E−18	7.06E−18	7.06E−18	3.31E−20	NA
$f_7(x)$	3.31E−20	1.06E−17	3.31E−20	3.31E−20	NA

Table D.4 BFP algorithm: p-test values for CEC 2005 results

Objective Function	FA	FPA	BA	BFP
$f_1(x)$	5.22E−07	6.79E−08	1.43E−07	NA
$f_2(x)$	6.79E−08	2.15E−01	5.16E−06	NA
$f_3(x)$	3.41E−07	NA	6.79E−08	6.79E−08
$f_4(x)$	6.79E−08	6.59E−06	4.53E−07	NA
$f_5(x)$	NA	9.29E−04	8.00E−09	2.95E−08
$f_6(x)$	NA	6.79E−08	6.79E−08	6.79E−08
$f_7(x)$	NA	6.79E−08	6.79E−08	6.79E−08
$f_8(x)$	6.61E−05	6.79E−08	6.79E−08	NA
$f_9(x)$	NA	6.79E−08	6.79E−08	6.79E−08
$f_{10}(x)$	8.00E−09	8.00E−09	8.00E−09	NA
$f_{11}(x)$	6.60E−03	6.79E−08	6.79E−08	NA
$f_{12}(x)$	1.08E−06	2.79E−06	7.17E−07	NA
$f_{13}(x)$	8.00E−09	NA	8.00E−09	NA

Table D.5 NMR algorithm: p-test values with respect to other algorithms

Function	BA	DE	FA	FPA	NMR
f_1	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_2	3.13E−20	NA	3.13E−20	3.13E−20	3.31E−20
f_3	3.13E−20	3.13E−20	3.13E−20	NA	3.31E−20
f_4	3.13E−20	3.13E−20	3.13E−20	3.13E−20	NA
f_5	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_6	3.31E−20	3.31E−20	3.31E−20	NA	3.31E−20
f_7	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_8	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_9	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_{10}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{11}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{12}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{13}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{14}	0.0123	NA	0.0123	0.0123	0.0123
f_{15}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{16}	3.31E−20	NA	3.31E−20	3.31E−20	3.31E−20
f_{17}	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_{18}	9.45E−19	NA	9.45E−19	9.45E−19	9.45E−19
f_{19}	5.87E−04	5.87E−04	5.87E−04	5.87E−04	NA
f_{20}	1.91E−18	NA	1.91E−18	1.91E−18	1.91E−18
f_{21}	5.27E−05	5.27E−05	5.27E−05	5.27E−05	NA
f_{22}	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_{23}	7.06E−18	7.06E−18	7.06E−18	7.06E−18	NA
f_{24}	3.85E−07	3.85E−07	3.85E−07	3.85E−07	NA
f_{25}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA
f_{26}	7.06E−18	7.06E−18	NA	7.06E−18	7.06E−18
f_{27}	3.31E−20	3.31E−20	3.31E−20	3.31E−20	NA

APPENDIX E

Real World Applications Tables

Table E.1 SACS algorithm: Results of real-world problem

Function		BA	DE	FA	FPA	GWO	SAMODE	EADEMA	SACS
FM	Best	11.5046	13.0145	1.17E-05	13.4103	5.24E-02	0.00E+00	1.16E-11	**2.94E-14**
	Worst	28.6796	38.1591	23.3135	25.6242	25.1636	10.9422	1.54E+01	1.51E+01
	Mean	24.6598	25.4501	15.1450	19.6346	14.2894	12.2025	7.60E+00	8.50E+00
	Std	3.1561	3.0654	5.9781	3.4362	5.4369	3.37E+00	6.02E+00	**2.23E+00**
LJ	Best	-28.4220	-15.6824	-27.4797	-18.3146	-28.3907	-28.4225	-28.4225	**-28.4225**
	Worst	-20.0129	-5.7267	-17.9139	-13.2319	-8.8083	-26.1004	-16.5337	-26.1291
	Mean	-26.2378	-8.6792	-23.5119	-16.1327	-23.4835	-27.0697	-23.0199	-27.2343
	Std	1.6103	1.8565	2.1488	1.1080	3.4505	6.62E-01	3.56E+00	**6.62E-01**
STR	Best	13.7709	14.3291	13.9436	13.7708	13.7709	13.7707	13.8542	**13.7708**
	Worst	21.8332	21.003	20.9045	13.7942	13.9543	14.3291	20.878	13.7708
	Mean	16.5098	17.8709	15.5661	13.7746	16.5098	13.9406	15.4119	13.7708
	Std	3.21E+00	3.18E+00	1.81E+00	4.70E-03	5.56E-02	2.50E-01	1.96E+00	**2.24E-08**
$SPRP$	Best	0.8686	1.3616	0.5561	1.1351	0.7200	0.5000	0.5000	0.5000
	Worst	2.2973	0.0960	0.1378	0.0972	0.3689	0.9943	0.7996	0.1595
	Mean	1.4044	1.7510	0.8743	1.3824	1.2503	0.8166	0.5847	0.7533
	Std	0.2791	0.0960	0.1378	0.0972	0.3689	1.19E-01	**7.79E-02**	1.59E-01

Table E.2 EFPA algorithm: Comparison of element amplitude excitationvalues for 12-element LAA

Element	1	2	3	4	5	6
Position	0.3765	0.9734	1.7376	2.5051	3.4827	4.6744

Table E.3 EFPA algorithm: Comparison of SLL values for 12-element LAA

Algorithm	**FiADE**	**MA**	**GA**	**PSO**	**FPA**	**TS**	**EFPA**
SLL (dB)	-18.96	-19.10	-18.77	-17.90	-20.764	-18.40	-21.07

Table E.4 EFPA algorithm: Comparison of element amplitude excitation values for 22-element LAA

Element	1	2	3	4	5	6	7	8	9	10	11
Position	0.4937	0.7965	1.2153	2.1313	2.5003	3.1327	3.9654	4.5877	5.5435	6.6391	8.0603

Table E.5 EFPA algorithm: Comparison of SLL values for 22-element LAA

Algorithm	**BSA**	**TS**	**MOEA/D-DE**	**HSA**	**PSO**	**GA**	**MA**	**FPA**	**EFPA**
SLL (dB)	-23.54	-17.17	-20.93	-23.28	-20.68	-15.73	-18.11	-23.81	-26.31
Null (dB) (99^0)	-104.61	-67.94	-69.64	-103.3	-49.94	-54.29	-73.92	-101.71	-75.58

Table E.6 EFPA algorithm: Comparison of element amplitude excitation values for 28-element LAA

Element	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Position	0.5880	1.2142	2.5291	2.8291	4.1205	4.9374	6.1006	6.9646	8.1374	9.5186	10.7399	12.4399	14.1399	15.8399

Table E.7 EFPA algorithm: Comparison of SLL values for 28-element LAA

Algorithm	**BSA**	**QPM**	**PSO**	**COA**	**ACO**	**GA**	**CLPSO**	**FPA**	**EFPA**
SLL (dB)	-21.90	-13.24	-13.6	-21.86	-14.64	-13.60	-21.63	-22.60	-22.90
Null (dB) (120^0)	-82.49	-48.49	-52.74	-60.08	-52.74	-59.25	-60.04	-78.45	-60
Null (dB) (122.5^0)	-93.59	-48.35	-51.66	-60.05	-59.20	-75.53	-60.01	-96.51	-60
Null (dB) (125^0)	-80.49	-89.3	-61.46	-60.10	-43.58	-62.52	-60.00	-82.58	-67.66

Table E.8 EFPA algorithm: Comparison of element amplitude excitation values for 32-element LAA

Element	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Position	0.5268	1.1101	2.2759	2.9840	3.8883	4.5802	5.7817	6.6474
Element	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Position	7.7093	8.8370	9.7490	11.0885	12.6565	14.3043	16.0043	17.7043

Table E.9 EFPA algorithm: Comparison of SLL values for 32-element LAA

Algorithm	**BSA**	**PSO**	**HSA**	**QPM**	**CS**	**GA**	**CLPSO**	**FPA**	**EFPA**
SLL (dB)	-20.50	-18.80	-19.51	-17.73	-22.83	-16.24	-22.75	-23.10	-23.73
Null (dB) (99^0)	-107.50	-62.20	-88.08	-34.74	-62.63	-62.94	-60	-130.60	-60.00

Table E.10 BFP algorithm: Comparison of SLL values for 12-element unequally spaced LAA

Element	1	2	3	4	5	6
Position (λ)	0.3961	1.2014	2.0626	2.9957	4.1667	5.5601

Table E.11 BFP algorithm: Comparison of SLL values for 12-element unequally spaced LAA

Algorithm	**FADE [16]**	**TS [7]**	**MA [7]**	**GA [7]**	**PSO [16]**	**BFP**
SLL (dB)	-18.96	−18.40	-19.10	-18.7770	-17.90	-20.77

Table E.12 BFP algorithm: Comparison of SLL values for 28-element unequally spaced LAA with nulls at 120^0, 122.5^0, and 125^0

Element	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Position (λ)	0.6076	1.3407	2.2932	3.1064	3.9422	4.9963	6.1585	6.9878	8.1819	9.4864	10.7808	12.4942	14.1769	15.8685

Table E.13 BFP algorithm: Comparison of SLL values for 28-element unequally spaced LAA with nulls at 120^0, 122.5^0, and 125^0

Algorithm	**BSA [20]**	**PSO [10]**	**COA [17]**	**QPM [10]**	**ACO [19]**	**GA [2]**	**CLPSO [12]**	**BFP**
SLL (dB)	-21.90	-13.60	-21.86	-13.24	-14.64	-13.60	-21.63	-22.88
Null (dB) (120^0)	-82.49	-52.74	-60.08	-48.49	-52.74	-59.25	-60.04	-60.72
Null (dB) (122.5^0)	-93.59	-51.66	-60.05	-48.35	-59.2	-75.53	−60.01	-72.85
Null (dB) (125^0)	-80.49	-61.46	-60.10	-89.30	-43.58	-62.52	−60.00	-64.55

Table E.14 BFP algorithm: Comparison of SLL values for 32-element unequally spaced LAA with null at 99^0

Element	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Position (λ)	0.5638	1.1664	2.2930	2.8370	3.8556	4.6802	5.8146	6.6742
Element	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Position (λ)	7.5390	8.7713	9.8758	11.1151	12.5145	14.2816	16.0941	17.7024

Table E.15 BFP algorithm: Comparison of SLL values for 28-element unequally spaced LAA with null at 99^0

Algorithm	**HSA [18]**	**PSO [10]**	**CPSO [13]**	**ILPSO [11]**	**CS [5]**	**GA [2]**	**MSMO [6]**	**COA [17]**	**CLPSO [12]**	**BFP**
SLL (dB)	-19.51	-18.80	-23.17	-23.75	-22.83	-16.24	-23.85	-23.81	-22.75	-23.99
Null (dB) (99^0)	-62.20	-62.20	-63.16	-73	-62.63	-62.94	-60	-79.85	-60	-61.51

List of Publications

Journal Papers

1. **Rohit Salgotra**, and Urvinder Singh. "Application of mutation operators to flower pollination algorithm." Expert Systems with Applications 79 (2017): 112-129.

2. **Rohit Salgotra** and Urvinder Singh "The naked mole-rat algorithm" Neural Computing and Applications 31, no. 12 (2019):8837-8857.

3. **Rohit Salgotra**, and Urvinder Singh. "A novel bat flower pollination algorithm for synthesis of linear antenna arrays." Neural Computing and Applications 30, no. 7 (2018): 2269-2282.

4. Urvinder Singh, and **Rohit Salgotra**. "Pattern synthesis of linear antenna arrays using enhanced flower pollination algorithm." International Journal of Antennas and Propagation 2017 (2017).

5. **Rohit Salgotra**, Urvinder Singh, and Sriparna Saha. "New cuckoo search algorithms with enhanced exploration and exploitation properties." Expert Systems with Applications 95 (2018): 384-420.

6. **Rohit Salgotra**, Urvinder Singh, Sriparna Saha and Amir H. Gandomi. "Self-adaptive cuckoo search algorithm: Analysis and experimentation." Swarm and Evolutionary Computation (2020).

Conferences Papers

1. **Rohit Salgotra**, Urvinder Singh, and Sriparna Saha. "Improved Cuckoo Search with Better Search Capabilities for Solving CEC2017 Benchmark Problems." In 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-7. IEEE, 2018 (Presented).

2. **Rohit Salgotra**, Urvinder Singh, Sriparna Saha, and Amir H Gandomi. "Improving Cuckoo Search: Incorporating Changes for CEC 2017 and CEC 2020 Benchmark Problems." In 2020 IEEE Congress on Evolutionary Computation (CEC), (Presented).

Awards & Honors

- **2016-2020: INSPIRE** Fellow, *Department of Science and Technology*, Govt. of India.
- **July 2020:** IEEE *Computational Intelligence Society* Registration grant for attending virtual conference **WCCI 2020**, Glasgow, Scotland.
- **June 2019:** IEEE *Computational Intelligence Society* Travel grant for attending **CEC 2019**, Wellington, New Zealand.
- **June 2019:** Travel grant for attending **CEC 2019**, Wellington, New Zealand by *Council of Scientific & Industrial Research*, Govt. of India.
- **July 2018:** IEEE *Computational Intelligence Society* Travel grant for attending **WCCI 2018**, Rio de Janerio, Brazil.
- **July 2018:** International Travel Support (ITS) for attending **WCCI 2018**, Rio de Janerio, Brazil by *Science and Engineering Research Board*, Govt. of India.
- **July 2018:** RF-25 Travel grant for attending **WCCI 2018**, Rio de Janerio, Brazil by *Thapar Institute of Engineering & Technology*, India.
- **2012-2014:** Academic Excellence Award for Securing **First Position** in Masters of Engineering, Chandigarh University, India.

Rohit Salgotra

Professional Recognition Gained

- **Member:**

  2018-2020: IEEE Student Member

- **Intern:**

  AI-NLP-ML Lab, *Indian Institute of Technology*, Patna, India (Dec. 2017-June 2018).

- **Editor:**

  Academic Editor: Mathematical Problems in Engineering

- **Reviewer:**

  IEEE Access

  Applied Intelligence

  Swarm & Evolutionary Computation

  Mathematical Problems in Engineering

  International Journal of Antennas and Propagation

  International Transactions on Electrical Energy Systems

  Concurrency and Computation: Practice and Experience

  Journal of Ambient Intelligence & Humanized Computing

  International Journal of RF and Microwave Computer-Aided Engineering

  International Journal of Numerical Modelling:Electronic Networks,Devices & Fields

Note: All the journals listed above are Science Citation Indexed (SCI) by *Web of Science*.

Rohit Salgotra

Acknowledgments

In this grand endeavor of my life, I have always been positive in attitude and working for my greater good. In the initial days it felt like things are a bit challenging but with the first publication, things changed and I started to enjoy every bit of it. I travelled a lot and worked remotely from various locations across the country on leave days, just to enjoy the wave and learn diverse cultures. It is during my this tenure, I analysed that travelling and photography have become my passion and this craving has grown many folds now. I made an Instagram account named **'xpedicr'** during the initial days of this adventure and has been followed by more than 10,000 people around the globe.

A lot of people joined and guided me in this executive journey and with them, I share a perfect healthy relationship. **Dr. Urvinder Singh** my PhD supervisor has been the key to my success. He is more of a guardian to me and provided every possible opportunity for excelling in everything I desired. **Dr. Sriparna Saha** from Indian Institute of Technology, Patna has served as my co-mentor and has been my constant motivation. It would be suffice to say that without their proper guidance, I might not have got such good publication track. Thirdly, I would like to thank **Dr. Amir H Gandomi** from University of Technology, Sydney. He is one among the constant motivators who has helped me define new objectives and add more flavor to this journey. At last I would like to thank **Directorate of Science & Technology, Govt. of India** for providing the INSPIRE fellowship to support my PhD course.

"*...may the LORD reward everyone of you for your kindness... to me.*"

Throughout this adventure, I felt as if *I was only chasing the moments of fire that were already predefined meticulously by Someone.* I am thankful to that **Marvelous Creator** for encouraging me in accomplishing this thesis.

A big $\mathcal{THANK\ YOU}$ to my parents and my brother for their love and care.

Rohit Salgotra