BIO-INSPIRED ALGORITHMS HARNESSING CRICKET CHIRPING BEHAVIOUR FOR SINGLE AND MULTI-OBJECTIVE OPTIMIZATION

Thesis submitted in partial fulfillment of the requirements for the award of the Degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

JONTI DEURI

Under the Supervision of

Dr. S. SIVA SATHYA

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF ENGINEERING AND TECHNOLOGY
PONDICHERRY UNIVERSITY, PONDICHERRY– 605014

FEBRUARY 2018

DEDICATION

To the almighty God

&

To my Father ***Mr. Bidyasing Deuri*** *& my Mother*

Mrs. Chenniprova Deuri

Thank you for your blessings and filling me with so much Hope, Purpose and Ambition and giving me this wonderful opportunity.

CERTIFICATE

This is to certify that this thesis titled "**BIO-INSPIRED ALGORITHMS HARNESSING CRICKET CHIRPING BEHAVIOUR FOR SINGLE AND MULTI-OBJECTIVE OPTIMIZATION**" submitted by **Ms. Jonti Deuri**, to the Department of Computer Science, School of Engineering and Technology, Pondicherry University, Puducherry, India for the award of the degree of **Doctor of Philosophy in Computer Science and Engineering** is a record of bonafide research work carried out by her under my guidance and supervision.

This work is original and has not been submitted, in part or full of this or any other University/Institution for the award of any other degree.

.

Place:

Date:

Dr. S. Siva Sathya
(Guide & Supervisor)
Associate Professor
Department of Computer Science
School of Engineering and Technology
Pondicherry University
Puducherry – 605 014
India**.**

DECLARATION

I hereby declare that this thesis titled **"BIO-INSPIRED ALGORITHMS HARNESSING CRICKET CHIRPING BEHAVIOUR FOR SINGLE AND MULTI-OBJECTIVE OPTIMIZATION"** submitted to the Department of Computer Science, School of Engineering and Technology, Pondicherry University, Puducherry, India for the award of the degree of **Doctor of Philosophy in Computer Science and Engineering** is a record of bonafide research work carried out by me under the guidance and supervision of **Dr. S. Siva Sathya**.

This work is original and has not been submitted, in part or full of this or any other University/Institution for the award of any other degree.

Place: **JONTI DEURI**

Date:

ACKNOWLEDGEMENT

To begin with, I thank God for blessing me with some astonishing people who have spun a web of support around me. There can never be enough words to express my gratefulness to those incredible people, by whom this research can be made possible. This is just an attempt to "THANK" them for making my period of research as a treasure to be cherished forever.

It is an immense pleasure to express my deep sense of thanks and gratitude to my guide and supervisor **Dr. S. Siva Sathya**, Associate Professor, Department of Computer Science, School of Engineering and Technology, Pondicherry University, Pondicherry for being beyond my guide, encouraging and soothing me with her invaluable suggestions, support and direction throughout my research period. I am indebted to her for her discerning guidance, for the copious trust she had on me and for the privilege she granted me to explore my own path in research.

I gratefully acknowledge the perceptive remarks and guidance of my Doctoral Committee members **Dr. M. Ezhilarasan**, Professor, Department of Information Technology, Pondicherry Engineering College, Pondicherry and **Dr. S. Ravi,** Assistant Professor, Department of Computer Science, School of Engineering and Technology, Pondicherry University, Puducherry. Their comments, suggestions, and ideas have helped me a lot to bring out this research successfully.

My sincere thanks to **Dr. T. Chithralekha,** the Head of the Department of Computer Science, for providing institutional support to carry out the research work. I owe my gratitude to our Dean **Dr. P. Dhanavanthan**, Dean, School of Engineering and Technology, Pondicherry University, Puducherry for his support.

I would like to thank **Dr. R. Sunitha, Dr. V. Uma, and Dr. P. Shanthi Bala** for their suggestions, advice, and encouragement throughout this research period and other faculty members of the Department of Computer Science, Pondicherry University, Puducherry.

.

I also would like to express my deepest gratitude to the Vice Chancellor of Pondicherry University, the Registrar and the Assistant Registrar of all the Departments, at Pondicherry University, Pondicherry for their official supports to make this thesis in time.

I am also thankful to the system administrators who maintained all the machines in my Lab so efficiently and I express my deepest gratitude to the office staff of my Department for their support. I extend my thanks towards the **staffs of the Ananda Rangapillai Library** of Pondicherry University for their service and efforts to conduct many workshops and seminar related to research, which helped to furnish the research in the form of a thesis.

I must acknowledge and express my gratitude and deep appreciation to **Ms. R. Vasundhara Devi** for her sisterly advice, suggestions and care and my research mates **Ms. M. Chitra, Mr. Naveen Kumar and Mr. Rajesh Kumar** for their support and invaluable help during my research period. I also thank all my colleagues and friends, including but not limited to **Richa Mishra, Ajit Kumar, Abhishek Kumar, Bithin Thakur, T. Subair, V. Balaji, Chanti, A. Sivagnana Ganesan, Suresh Matla, R. Gayatri, S. Sharmila Devi, Gunikhan Sonowal, Abid Islam and Anushreya**.

My thanks must also go to my hostel friends **Ms. Tripti Gogoi, Ms. Suchismita Jena, Ms. Swatilekha Mahato, Ms. Samapti Dasmahapatra, Ms. Priyanka Sharma and Ms. Arundhati Lahan.** They have consistently helped me keep perspective on what is important in life and shown me how to deal with reality.

Above all, I am much obliged to **my parents, brothers, sisters and all my family members** for supporting me spiritually throughout my life in general and my best friend **Mr. Kalyan Jyoti Deori** for his constant support and help.

JONTI DEURI

ABSTRACT

Optimization techniques find its application in almost every field of concern. The task of optimization is obtaining the maxima or minima, subject to the various constraints specified. The problems can be single objective or multi-objective and correspondingly the techniques can be categorized as a single objective optimization technique and multi-objective optimization techniques. Some techniques aim at finding only the optimal solution and are termed as exact methods. They take exponential time in order to achieve their motive. An alternative to this is the approximate methods that attempt to determine a near optimal solution in a reasonable amount of time. The majority of the approximate approaches derives their concepts from Biology and Mother Nature. Evolutionary and swarm-based algorithms can be quoted as typical examples for this case.

Though there are numerous optimization techniques, there also exist certain barriers that hurdle in attaining the maximum efficiency. They include the premature convergence, rigorous parameter tuning, non-generalization, high computational cost and difficult implementation. Further, the “No Free Lunch Theorem”, which states that all algorithms perform similarly when averaged on all functions and designing an algorithm to suit all applications will result in vain, encourages the formulation of new optimization algorithms.

In this research, new optimization techniques to overcome the stated barriers have been formulated. The chirping behavior and movement of the insect named cricket have placed the primary emphasis while devising the algorithm. Initially, a cricket chirping algorithm (CCA) for single objective optimization is designed. In this algorithm, the unique chirping nature of male crickets while mating and aggression are exploited. The male crickets chirp with an exclusive sound in order to attract the females for mating and simultaneously repel the males. Another kind of chirping sound is emitted during aggression when another male cricket nears it with the intent to fight. In the case of aggression, the winner survives and takes the position of the loser and the loser is discarded. This behavior of cricket has been harnessed for the design and development of the bio-inspired algorithms for single and multi-objective optimization. The single

objective CCA is tested on many test functions. The parameters are fine-tuned based on an exhaustive analysis of the algorithm.

Subsequently, the cricket chirping algorithm is extended to deal with the MOO problems. In order to achieve this, two approaches, namely the weighted sum and Pareto based are adopted and thus two variants of Multi-Objective Cricket Chirping Algorithm (MOCCA) namely MOCCA-W and MOCCA-P are formulated.

The CCA and MOCCA are then applied on various real-time problems and compared against the existing popular and effective optimization algorithms in order to exhibit its potential. The CCA is tested on engineering optimization problems such as Tension and Compression Spring Design Optimization and Welded Beam Design Optimization and Multilevel Threshold Optimization for image segmentation. The MOCCA is evaluated on multi-objective Disc Brake Design Optimization and Welded Beam Design Optimization. The outcomes along with statistical analysis reveal the outstanding performance of the formulated optimization techniques.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AE	Average Error
ANOVA	Analysis of variance
A_r	Aggression rate.
C_r	Crossover Rate.
CCA	Cricket Chirping Algorithm.
F_s	Female Selection.
GD	Generational Distance
MOCCA	Multi-Objective Cricket Chirping Algorithm
MOCCA- W	Multi-Objective Cricket Chirping Algorithm Using Weighted Sum
MOCCA- P	Multi-Objective Cricket Chirping Algorithm Using Pareto Approach.
MOO	Multi-Objective Optimization.
MOOT	Multi Objective Optimization Techniques
MOP	Multi Objective Optimization Problem.
MS	Maximum Spread
MT	Multilevel Threshold
SD	Standard Deviation.
SOO	Single Objective Optimization
SP	Spacing
SDOP	Spring Design Optimization Problem.
SPSS	Statistical Package for the Social Sciences
T_c	Temperature.
WBDOP	Welded Beam Design Optimization Problem.

Chapter 1

INTRODUCTION

Nature inspired meta-heuristic algorithms have been recognized to be very proficient in solving complex optimization problems in the recent times. Literature reports several inspirations from nature and biology that have been exploited to solve complex computational problems. This research is yet another effort in the journey towards the utilization of bio-inspired techniques for seeking solutions to complex optimization problems. In this chapter, initially, the significance of optimizing is highlighted. Then, the solutions towards optimization are presented. Followed by that, a glimpse of various nature and bio-inspired algorithms are provided. Subsequently, the biological aspects of cricket, the insect that has inspired in accomplishing this research is presented. After that, the research overview of the thesis and the organization of the chapters are provided. The following section presents the importance of optimization in real-world problems.

1.1 OPTIMIZATION

Generally speaking, optimization can be termed as the process of identifying the most cost-effective method for accomplishing the maximum performance under the provided constraints, by maximizing the desired factors and minimizing the undesired aspects. It can also be visualized as a minimization or maximization problem based on the problem at hand. In day to day life, every individual is posed with many options and is forced to choose one of them to get through the situation. Naturally, individuals choose one of the many available choices such that it is beneficial for them in some way or the other. The benefits can be related to finance, quality, personal development, satisfaction and many more. It is to be noted that sometimes, the benefits can be related to one or more aspects.

To typify the maximization problem, a general business scenario is considered. In a business, the objective will be to optimize the effectiveness of the production process or the quality and desirability of their existing goods and commodities with minimal runtime or resources, etc. To illustrate an example of the minimization problem, the scenario of purchase of mobile can be quoted. While buying a mobile, optimization can mean cost to some customers. In this context, minimizing cost will be the objective and hence it is easy to provide a solution. In other cases, purchasing mobile may be related to many factors such as cost, power backup, screen size, front camera resolution, back camera resolution, music quality, RAM capacity and many more. In such cases, the goal of a selfie lover will be to find mobiles that satisfy conditions such as the affordable price, average power backup, large screen size, high front camera resolution, average back camera resolution, average music quality and high RAM capacity. In the case of a music lover, the optimization constraints can differ such as the affordable price, high power back up, medium screen size, average front and back camera resolution, high music quality and high RAM capacity.

The first scenario of purchasing mobile can be described as a Single Objective Optimization (SOO) problem while the second and third scenarios can be defined as Multi-Objective Optimization (MOO). Therefore, optimization can be generally categorized into single objective and multi-objective optimization. SOO is relatively easy whereas MOO problems are complex, expensive and time-consuming.

In computational terms, optimization involves minimization or maximization of one or more objective functions involving some integer or real variables. On being provided with a specific domain along with its constraints, the main motive of optimization is to investigate the means of attaining the best value of the objective function. A simple mathematical representation of the optimization problem can be formulated as follows:

Given a function $f: B \to S$ from some set of the real numbers, the goal is to find an element x_0 in B such that $f(x_0) \leq f(x)$ for all x in B in the case of minimization or $f(x_0) \geq f(x)$ for all x in B in the case of maximization problems. Here B refers to a subset of the Euclidean space S. S is a collection of entities, namely constraints, equalities, and inequalities. The elements of B owe to satisfy these entities and are called as a candidate or feasible solutions. B, which defines the domain of *f, is* referred to as the search space. A feasible solution which optimizes the objective is called the optimal solution. Thus, optimization can also be realized by exploring and exploiting the search space of solutions to a problem in order to identify the optimal solution. The following sub-sections provide a detailed overview of SOO and MOO techniques.

1.1.1 SINGLE OBJECTIVE OPTIMIZATION (SOO)

As the name suggests, the primary motive of SOO is to identify the best solution that is associated with the minimum or maximum of a single objective function. There can be only one global solution in this case. Hence it is relatively easy to identify the solution. MOO problems are more complex than SOO problems and are discussed in the following sub-section.

1.1.2 MULTI-OBJECTIVE OPTIMIZATION (MOO)

MOO deals with two or more objectives [1] and these objectives may be conflicting and contrary. In such cases, it is very difficult to get a single optimal solution. The interactions among the different objectives may give rise to a collection of compromised solutions. It is often termed as trade-off or pseudo-optimal or quasi-optimal solutions. Many real-world problems involve many objectives and MOO can best fit the scenario to identify the optimal solution. Some basic and primary definitions of the support of MOO are stated briefly as follows.

MOO problems [2] can either be convex or non-convex. All the objective functions of an MOO problem are convex when the function is convex. A function f: $Rn \rightarrow R$ is convex if for any two pairs of solutions x_1, $x_2 \in R_n$, it satisfies the condition $f(\delta x_1 + (1-\delta)x_2 \leq \delta f(x_1) + (1-\delta)f(x_2)$ for all $0 \leq \delta \leq 1$

A characteristic means to express a solution is by means of Pareto Optimality [3]. The strategy has been initially introduced in [2] and later worked out by Vilfredo Pareto. A solution of $x \in R$ is considered Pareto Optimal with regard to R only when there exists no $x \in R$ for which $v = F(x) = (f_1\,(x), \ldots, f_k(x))$ dominates $u = F(x) = (f_1(x), \ldots, f_k(x))$. Such a solution is also known as non-dominated solution. In other words, a solution is called Pareto optimal if there are no other solutions that can dominate it. This solution cannot be enhanced through any one of the objectives without adversely affecting at least one other objective.

Another commonly used terminology is Pareto Dominance [4]. A vector $u = (u_1, , \ldots, u_k)$ is considered to dominate another vector $v = (v_1, \ldots, v_k)$ (denoted by u $\leq$ v) only when u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. This concept is incorporated in multi-objective optimization for the purpose of comparing and ranking the decision vectors: if u dominates v in the Pareto sense, then it signifies that *F(u)* is better than *F(v)* objectives, and there is at least one objective function for which *F(u)* is strictly better than *F(v).*

Let $F(x)$ denote a MOO problem. Then, the Pareto Optimal Set P^* for $F(x)$ can be interpreted as $P^* = \{x \in \Omega \,|\neg\exists\, x' \in \Omega\, F(x') \leq F(x)\}$. Based on this, the Pareto Front PF^* can be interpreted as $PF^* = \{u = F(x) \mid x \in P^*\}$. Typically, the Pareto Optimal set P^* constitutes the entire collection of Pareto-optimal solutions while Pareto-Optimal Front PF^* comprises the mapping of Pareto-Optimal solutions in the objective space. In the Pareto spirit, minimum in the Pareto sense becomes the border of the design space. Alternatively, it can also be defined as the locus of the tangent points of the objective functions [5]. Figure 1.1 shows this strategy evidently. In the figure, a bold line is used to denote the border line of a problem that has two objectives for optimization. The region of points defined by this bold line is called the Pareto Front.

.

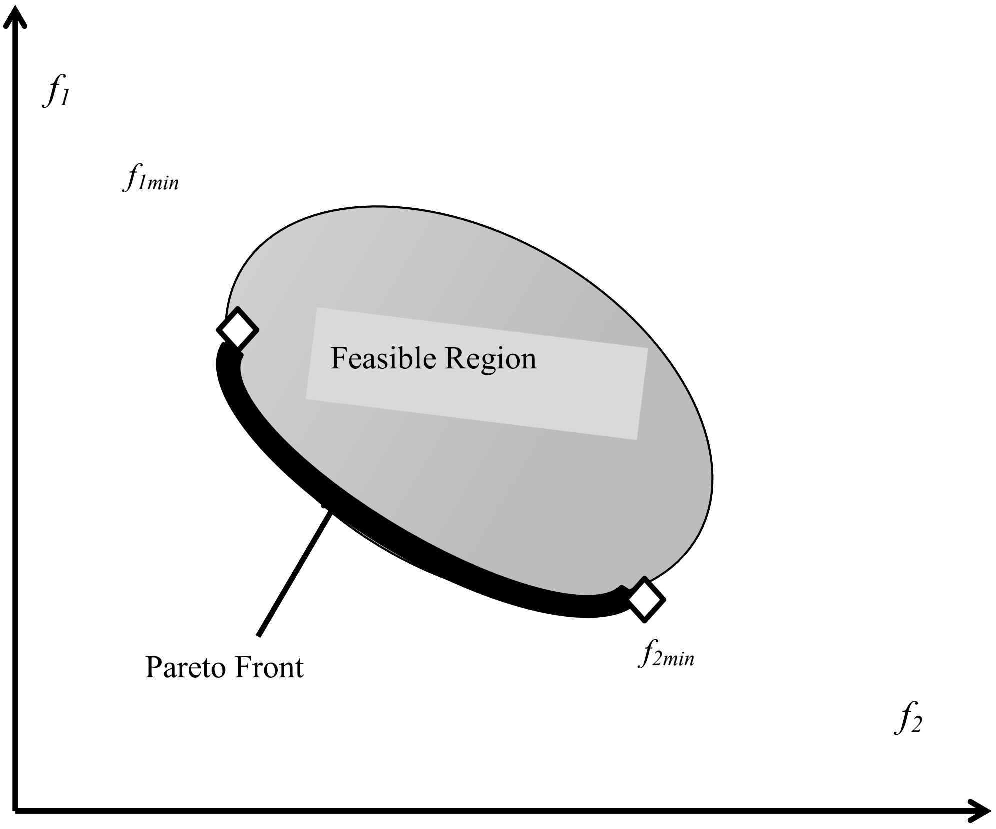


Figure 1.1: Pareto Front

The concept of trade-off in the MOO can be defined in terms of Pareto optimality where the goal is to achieve the set of all Pareto optimal solutions or, at least, a good approximation of this set. An alternative method would be to convert the MOO problem into a single objective by using an approximation approach of the multiple objectives and can be found Pareto optimal solutions by solving these SOO problems using various weights. However, if certain conditions are not met, not all Pareto optimal solutions will be found by means of such method.

Some strategies are followed for computing the fitness based on the various conflicting objectives. The fitness assignment for MOO techniques can be grouped into different categories such as Aggregative, Lexicographic, Sub-population, Pareto-based, and Hybrid methods. Both SOO and MOO problems find manifold applications. The next sub-section spotlights on these applications.

1.1.3 APPLICATIONS OF SINGLE OBJECTIVE AND MULTI-OBJECTIVE OPTIMIZATION

Optimization is indeed a part of human life. It has its application in almost every field of concern. However, a few domains demand computationally effective solutions. They include engineering optimization problems such as tension/compression spring design optimization, welded beam design optimization, pressure vessel design optimization, speed reducer design optimization, disc brake optimization etc., computer vision and image processing optimization problems such as curve fitting optimization, threshold optimization, segmentation optimization, registration optimization, filtering optimization etc. Out of the diverse applications, tension/compression spring design optimization, welded beam design optimization, disc brake optimization and multi-level threshold optimization have been taken as case studies in this research.

In the perspective of SOO optimization, tension and compression spring design optimization problem deal with weight minimization of the spring, subject to constraints of minimum deflection, surge frequency, shear stress, and limits on outside diameter and on design variables. Similarly, the motive of welded beam design optimization is cost minimization, subject to a set of constraints on shear stress, bending stress in the beam, buckling load on the bar PC, end reflection of the beam and side constraints. In the case of multi-level Thresholding for image segmentation, the task is to determine the optimal value of the threshold, a parameter used for Thresholding.

In the perspective of MOO, the goal of the welded beam problem is to minimize both the end deflection and the fabrication cost, subject to length of the welded area, the thickness of the main beam and its width and depth. Similarly, the aim of disc brake design optimization is to minimize the braking time and overall mass, subject to design variables such as outer radius of the discs, the inner radius, the number of the friction surface and the engaging force and design constraints such as temperature, pressure, length of the brake and torque.

Having presented an overview of the problem of optimization and its kinds, the following section presents the various optimization techniques adopted.

1.2 OPTIMIZATION TECHNIQUES

As mentioned earlier, optimization is the finding of the optimal (maxima or minima) solution of a given problem under some circumstances. It may be single objective or multi-objective and constrained or unconstrained in nature.

Optimization techniques can be categorized into two types namely exact methods and stochastic (approximate) methods [6]. The exact methods identify the best possible solution. Brute force search, branch and bound, dynamic programming, cutting plane method etc., come under this category. These are highly efficient for small sized problems. On the other hand, stochastic (approximate) approaches are efficient for large and complex NP-hard problems [7]. These do not guarantee for optimal solutions but attempt to obtain quasi-optimal solutions in a reasonable amount of time. Evolutionary algorithms, stochastic hill climbing, swarm algorithms, simulated annealing etc., belong to this category. Most of these stochastic approaches have been derived from the concepts of nature and biology. A brief overview of these techniques is provided in the following sub-section.

1.2.1 EXACT METHODS

Exact methods are designed to find only the optimal solution without any compromise. They consume long time to arrive at the solutions and hence become inappropriate when the search space is very large. On the other hand, for problems with a smaller search space, they provide the best solutions. A few kinds of exact methods are presented here.

➢ BRUTE FORCE SEARCH

Brute Force search involves building all the admissible solutions and thereby attaining the optimal solution. This is the best method to find the optimal solution but is not efficient as it will take exponentially longer times even if there is a marginal growth in the size of the search space. Hence, other techniques which do not explore the entire search space but find the global optimal solution is sought. One such method is a branch and bound technique and is discussed subsequently.

➢ BRANCH AND BOUND

The branch and bound technique assume the candidate solutions as a tree with the root holding all the solutions. The branches depict the subset of the solutions. Hence, the branches are explored from top to bottom till a particular optimal solution is identified. Prior to the exploration of a particular branch, it is checked against the upper and lower bounds on the optimal solution. The branch is discarded without further exploration if it cannot yield a better solution than the one identified by the algorithm till then. Thus, a few admissible solutions, which cannot be optimal solutions, are discarded without building it thus saving time.

➢ DYNAMIC PROGRAMMING

Dynamic programming views a complicated problem as a collection of simpler sub-problems. It finds the solutions of the simpler sub-problems only once and saves it for future use. In case, the sub-problem is encountered again, then it is not computed but the already obtained solution is utilized thus, saving the computational time. Dynamic programming is advantageous if the complex problem can be divided into overlapping simpler sub-problems.

➢ CUTTING PLANES

The cutting plane technique involves cutting planes which are hyperplanes delineating the current point from the optimal point. The process refines the candidate set by means of linear equalities called cuts. They are highly appropriate for mixed integer linear programming problems. The technique operates through solving a non-integer linear program with an assumption that an extreme or corner point that is optimal can be found. The hence obtained optimum is checked for being an integer solution. If it is not an integer solution, then a linear inequality that separates the optimum from the convex hull of the candidate solution set will exist. Determining this inequality is the problem of identifying the cut. This process is repeated until an optimal integer solution is found.

Having provided an account on a few of the exact optimization techniques, the following sub-section presents the approximate techniques for optimization.

1.2.2 APPROXIMATE METHODS

The approximate methods aim to identify the quasi-optimal solution in a considerable time. These methods primarily adopt many heuristics to explore the search space and can also be called as meta-heuristic algorithms [8]. Some heuristics have their roots in the concepts of nature and biology and hence these techniques are formulated based on the concepts of nature and biology and referred to as bio-inspired algorithms. The following section provides a brief note on the meta-heuristic and bio-inspired optimization methods [9].

➢ EVOLUTIONARY ALGORITHMS

Evolutionary algorithms, one of the typical illustrations of bio-inspired algorithms, are inspired by the biological evolution process [10], It incorporates concepts such as selection, reproduction, crossover, mutation, and survival of the fittest. Feasible candidate solution set forms the population and each solution represents an individual. Initially, a few individuals are selected for the process. The fitness function evaluates its quality. Then, the reproduction process is performed to produce offsprings. The process continues until a termination criterion is met. It may be the number of generations or the ideal fitness value. Genetic algorithms, Genetic programming, Evolutionary programming etc., are the widely adopted evolutionary algorithms in the context of optimization.

Genetic Algorithms are one of the extensively used evolutionary algorithms It realizes the solution in the form of bit vectors or string of numbers, characters etc.

Genetic Programming represents solutions in terms of computer programs or tree structures. The fitness function evaluates the potential of the program to solve a computational problem.

Evolutionary Programming is similar to Genetic Programming in the context that it also represents solutions in the form of computer programs. But in this case, the structure of the computer program is fixed while the numeric parameters of the program evolve.

➢ SWARM ALGORITHMS

Swarm algorithms are inspired by the biological ecosystems and mimic the interactions among various organisms and its interactions with the environment [11]. The intelligent agents interact with each other and with the environment through simple rules in a decentralized environment where no central control structure would instruct on how to behave and interact. Particle Swarm Optimization, Artificial Bee Colony, and Ant Colony Optimization etc. are the sterling examples of swarm algorithms.

Particle Swarm Optimization (PSO) algorithm considers the solution as a point or surface in the n-dimensional space [12]. Initially, the particles are randomly chosen with an initial velocity and a communication channel is established among the particles. The algorithm proceeds by moving the particles and computing the fitness at regular time intervals. Over time, the particles accelerate towards an optimal solution, thereby producing higher fitness values.

Ant Colony Optimization (ACO) algorithm depicts the behavior of ant colonies [13]. The ants lay down pheromones leading each other to resources while at the same time exploring the environment. In an ant colony algorithm, the intelligent ant agents marked their best positions and the potential of their solutions so that in a lesser amount of time more ants find better solutions.

Artificial Bee Colony Algorithm (ABC) is an optimization algorithm that is inspired by the honey bee [14] The algorithm imitates the food foraging behavior of bees. In a bee hive, scout bees go in search of food source. After some time they return to their hive and harvest the food. The bee that has identified the huge amount of profitable food source performs a waggle dance to notify other members about the rich source of food. The length of the dance will be proportional to the quality of profitability. Then, more foragers are recruited to proceed with further exploration of the identified area of the rich source of food. Similarly, in bee optimization, the candidate solutions are analogous to the food source and a population of bee agents is used to explore the search space. The algorithm involves recruitment, local search, neighborhood shrinking, site abandonment and global search. The cycle is iterated for a specified number of times or until an optimal solution is attained.

Cuckoo Search (CS) emulates the brooding behavior of cuckoos [15]. Cuckoo lays its egg in the nest of other hosts . When the host identifies it as an alien egg, it either gets rid of the egg or abandons the nest. In cuckoo search algorithm, the eggs form the potential solutions and the number of nests remains fixed. The host can identify the alien egg based on a probability. On identifying the egg, it abandons the nest and builds a completely new nest.

Bat Algorithm (BA) is inspired by the hunting behavior of bats [16]. It is rooted in the concept of echolocation behavior of microbats. During the search for its prey, pulse emission rate and loudness revealed by bats is mimicked in the bat algorithm. It incorporates tuning of frequency to elevate the diversity of the solution in the population, but at the same time, it adopts the automatic zooming concept and attempts to maintain a balance between the exploration and exploitation during the search process. The auto zooming ability in microbats is manifested as the automatic adjustment from exploration to exploitation to approach the global optimality. Bat Algorithms is considered as one of the first kind of algorithms that balance these two key components in the search process.

Firefly algorithm (FA) is another optimization algorithm that imitates the behavior of insects [17]. This algorithm is inspired by the flashing light behavior of fireflies. The flashing light is used for courtship signals and as a protective mechanism. The firefly search algorithm is based on the light intensity and attractiveness of fireflies. The brightest firefly represents the optimal solution.

➢ SIMULATED ANNEALING

Simulated Annealing (SA) is based on the strategies adopted in the field of metallurgy [18]. The concept of heating and controlled cooling to increase the size of the crystal and decrease its defect is grabbed in simulated annealing. The technique operates on a search space where the choice of moving to another solution is based on one of the two probabilities, which are based on whether the new solution is better or worse than the current solution. The temperature is decreased from a positive value towards zero and this affects both the probabilities.

➢ GRAVITATIONAL SEARCH OPTIMIZATION

Gravitational Search Optimization (GSO) is another class of optimization technique with a different strategy for searching [19] . It is primarily rooted on the basis of the law of gravity and the idea of mass interactions. This technique considers the distance between the neighboring agents to update the position of the currently considered agent. In this algorithm, the agent is characterized by four parameters namely (i) position (ii) inertial mass (iii) active gravitational mass and (iv) passive gravitational mass. The solution is indicated by the position of the mass. Fitness measures are incorporated for the purpose of calculating the gravitational and inertial masses. The inertia mass parameter, which is used for updating the agent movement, is inversely proportional to the motion of the agent. A bigger inertia mass facilitates slower motion of the agents in the search space. This leads to a more precise local search with increased diversity in search space. On the other hand, higher the gravitational mass, higher will be the attraction of agents, thus leading to a faster convergence. The algorithm proceeds by adjusting these two masses namely the gravitational and inertia masses, wherein each mass signifies a solution. The masses are attracted by the heaviest mass. Hence, the heaviest mass offers an optimal solution in the search space.

➢ ELECTRO-MAGNETISM OPTIMIZATION

Electro-Magnetism Optimization (EMO) is a metaheuristic algorithm based on the attraction-repulsion mechanism to move the sample points towards the optimality [20]. Here, each sample point is anticipated as a charged particle that is released to space. The objective function value is the charge of each point that has to be optimized. This charge defines the magnitude of attraction or repulsion of the point over the sample population; the higher the magnitude of attraction, the better the objective function value. After calculating these charges, they are used to find a direction for each point to move in subsequent iterations. This direction is selected by evaluating a combination force applied to the point via other points.

➢ STOCHASTIC HILL CLIMBING

Stochastic hill climbing is a variant of the deterministic hill climbing. The algorithm moves to a nearby solution only if it can yield improvement over the current solution. The stochastic version is implemented to overcome the local optima problem encountered by the deterministic version.

Mother Nature has always been an unending source of inspiration for the scientific community. The behavior of genes, bees, bacteria, glow worms, slime molds, cockroaches, mosquitoes, crickets, Firefly, cuckoo and other organisms have inspired researchers to develop new optimization algorithms for solving numerous SOO and MOO problems due to the inherent simplicity, effectiveness, and efficiency observed in their behavior. A few of these algorithms are spotlighted here.

Similarly, the behavior of many insects, birds, and animals are imitated to devise optimization algorithms. The following section presents an account of the swarm characteristics of another insect cricket, whose characteristics are exploited to formulate an optimization technique in this research.

1.3 CHARACTERISTICS OF CRICKET

Cricket belongs to the family of Gryllidae. They are insects resembling very close to bush crickets and grasshoppers. They are nocturnal and hide themselves during the day. For their defense, they adopt camouflaging, fleeing, colorings and aggression [21] [22]–[26]. Another unique characteristic of the crickets is their chirping. Mostly, only male crickets possess this feature while female crickets do not chirp. The male crickets produce a loud chirping sound by scraping two specially textured limbs together. The crickets chirp differently on different occasions. The chirping song can be categorised as (i) calling song, which attracts female crickets for mating and repels the male crickets (ii) counting song, which signifies that the female cricket is ready to mate (iii) the triumph song after mating to encourage the female to lay eggs and (iv) the aggressive calling, which is triggered when another male cricket nears it with the intention of fighting. Figure 1.2 shows the natural behaviors of cricket.

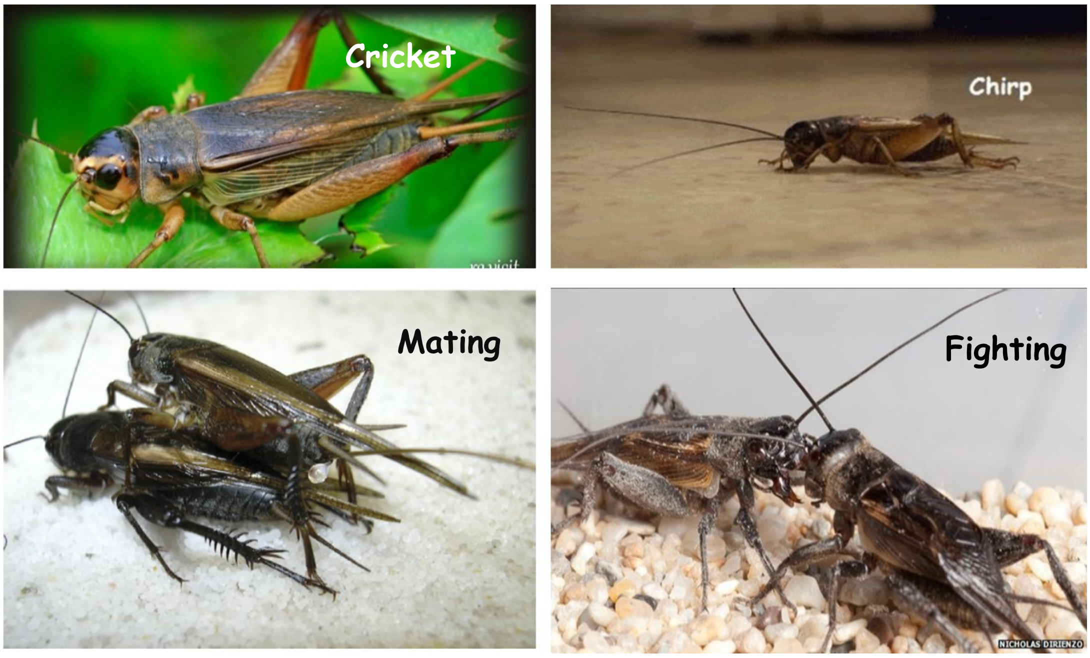


Figure 1.2 Cricket's Natural Behavior

The chirping rate varies among the various species of the crickets and is dependent on the temperature of the surroundings. The chirping rate increases with the increase in temperature. The forecasting of temperature through chirping rate and frequency tuning of chirping has been modeled to solve computational problems. The chirping behavior is a unique feature of these insects and they chirp with unique frequency and loudness for every action. The chirping of the crickets thus inspires this research to formulate an optimization algorithm that could solve both single and MOO problems.

The following section provides a note on the research overview.

1.4 RESEARCH OVERVIEW

The outline of the work done in this research is presented in this section. The research focuses on formulating a nature-inspired optimization technique that suits SOO and MOO problems. Primarily, the chirping behavior of crickets is mimicked to devise the optimization technique. The chirping behavior is a specialty for these insects. This specialty is incorporated to develop a new optimization technique that can solve real-world complex optimization problems.

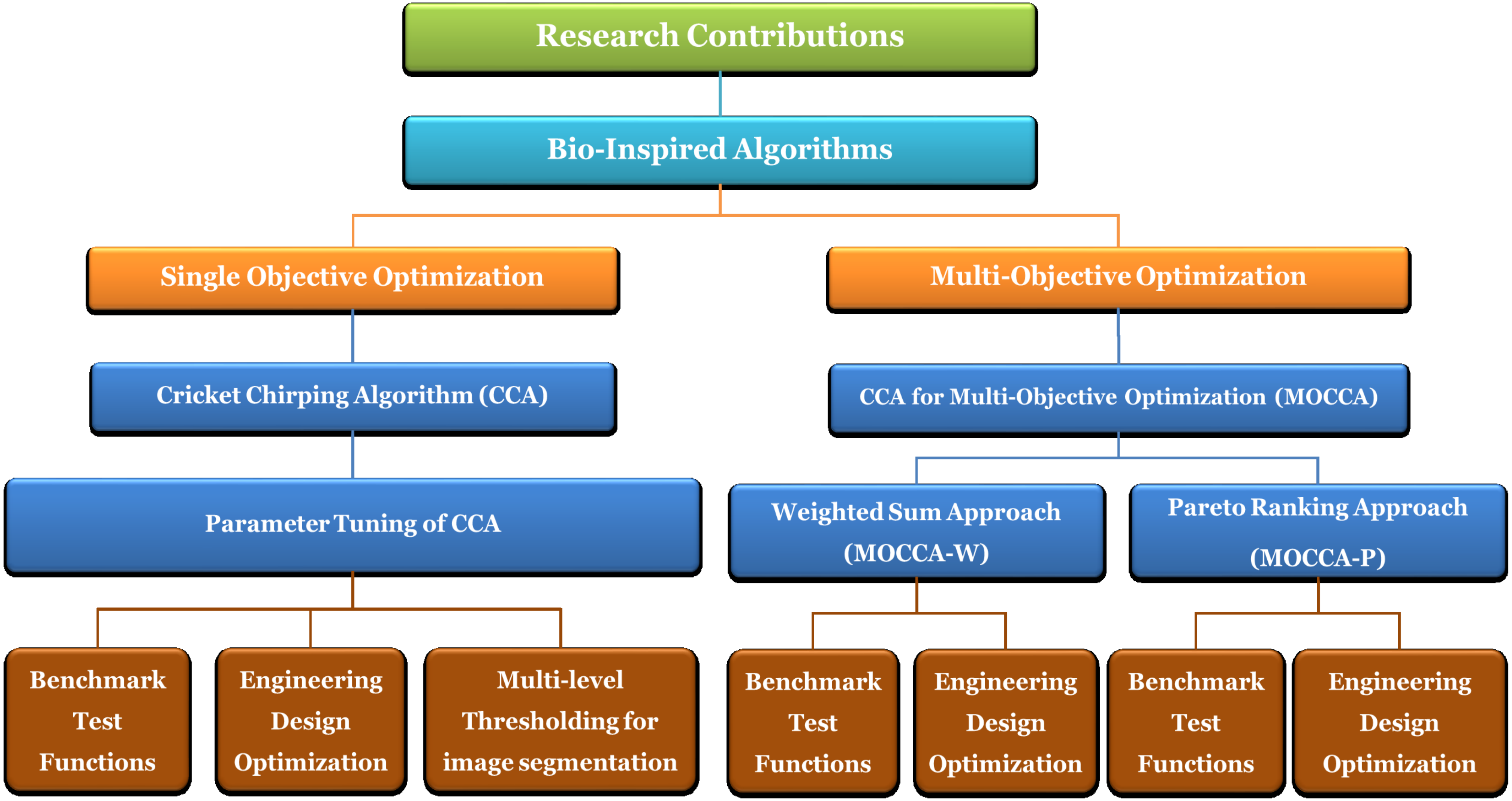

Research Contributions
Bio-Inspired Algorithms
Single Objective Optimization
Multi-Objective Optimization
Cricket Chirping Algorithm (CCA)
CCA for Multi-Objective Optimization (MOCCA)
Parameter Tuning of CCA
Weighted Sum Approach (MOCCA-W)
Pareto Ranking Approach (MOCCA-P)
Benchmark Test Functions
Engineering Design Optimization
Multi-level Thresholding for image segmentation
Benchmark Test Functions
Engineering Design Optimization
Benchmark Test Functions
Engineering Design Optimization

Figure 1.3: Research Overview

Initially, an SOO technique is developed. To devise this technique, the chirping characteristics of crickets and their movement in the environment for mating and aggression is emulated. Each cricket is considered as a feasible solution and is characterized by its position in the search space. Few of the crickets are assigned as females by the user. As mentioned already, only the male crickets can chirp and its chirping rate is based on the outside temperature.

As any meta-heuristic algorithm is greatly influenced by the parameter settings, the fixing of parameter values is very important. Thus, subsequently, the parameters are tuned to suit the problems in hand.

Subsequently, an MOO technique is designed to handle multi-objective problems. In this case, the algorithm has to handle conflicting objectives to provide an optimal solution.

The multi-objective variant is extended and two variants are proposed. The variation is based on the fitness computation strategy. In the first type, the weighted sum strategy is adopted while in another, the Pareto notion is incorporated. Then, the potential of the proposed algorithms is justified through appropriate case studies.

In figure 1.3, the complete research undertaken is shown. The Cricket Chirping Algorithm for single objective optimization (CCA) is inspired by the chirping of crickets during mating and aggression. It is formulated in the view of providing efficient solutions to SOO problems.

Further, as the parameters of optimization algorithms greatly influence their performance, the performance tuning of the algorithm is emphasized.

Then, Multi-Objective Cricket Chirping Algorithm (MOCCA) is developed as an extension to CCA in order to support the MOO problems. The strategy of weighted sum and Pareto has been used for this purpose.

Then, appropriate case studies are taken to justify the potential of the proposed algorithms. Optimization of tension and compression spring design, welded beam optimization and multi-level Thresholding for image segmentation are chosen as case studies for SOO and the design of welded beam and disc brakes are selected for multi-objective optimization.

The research thus focuses on providing efficient SOO and MOO techniques. The following section provides the organization of the remaining chapters of the thesis.

1.5 ORGANIZATION OF THE THESIS

This section describes in detail about the overall structure of the thesis.

CHAPTER 2

This chapter presents the literature review of the existing meta-heuristic algorithms proposed for single and multi-objective problems. It also presents the earlier work on optimization of the various case studies undertaken.

CHAPTER 3

In this chapter, the problem is defined clearly with the motivation and objectives of this research. The objective of this research is to develop a bio-inspired meta-heuristic optimization algorithm for SOO and MOO problems. Further, the scope of the work and research methodology is described.

CHAPTER 4

In this chapter, the details of the design and development of Cricket Chirping Algorithm (CCA) for SOO is projected. Further, the experimental results and analyses on benchmark problems are given along with statistical analysis using ANOVA.

CHAPTER 5

The performance of the meta-heuristic algorithms is primarily based on the apt selection of parameter values. In this chapter, the fine-tuning of various parameters of the proposed algorithm is discussed. The fine-tuned algorithm is compared with the existing state of the art algorithm and statistical analysis using ANOVA is carried out.

CHAPTER 6

The CCA is extended to suit the MOO problems. The design and implementation of this algorithm, Multi-Objective Cricket Chirping Algorithm (MOCCA) are detailed in this chapter. The two strategies adopted for handling the multi-objectives is presented in detail and the comparison with benchmark problems are highlighted. A statistical analysis using ANOVA is also carried out.

CHAPTER 7

Various case studies are undertaken to prove the effectiveness of the proposed algorithms. Appropriate case studies in terms of SOO and MOO are taken into consideration. The proposed algorithms are used to solve the problems and the experimental results are analyzed.

CHAPTER 8

This chapter provides the conclusions derived from this work and discusses the possible future enhancements.

Chapter 2

LITERATURE SURVEY

This chapter describes the existing work related to meta-heuristic optimization algorithms. The chapter initially presents a detailed overview of the various extensively used meta-heuristic algorithms in the view of the SOO. Then, the commonly used fitness strategies for extending the SOO to MOO techniques are reviewed. Then, the popular variants of meta-heuristic algorithms for MOO are presented. Subsequently, the earlier works on the optimization of the case studies undertaken are presented. The following section presents the various meta-heuristic algorithms for SOO.

2.1 META-HEURISTIC OPTIMIZATION TECHNIQUES FOR SINGLE OBJECTIVE OPTIMIZATION

This section presents some classes of popularly used meta-heuristic algorithms that have been used for SOO problems. They include Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony Optimization (ABCO), Cuckoo Search Optimization (CS), Bat Algorithm (BA), Firefly Algorithm (FA), Gravitational Search Algorithm (GSA) and Electro-Magnetism Optimization (EMO). The following sub-sections present these techniques and its variants in detail with the main emphasis on SOO.

2.1.1 GENETIC ALGORITHMS

Genetic Algorithms (GA) is one of the extensively adopted Evolutionary algorithms, developed by John Holland in the year 1970, being inspired by the theory of evolution [27]. The algorithm starts with an initial population. The initial population comprises of a set of individuals, representing feasible solutions and fitness of each individual is evaluated. Then a selection process is performed in order to decide which individuals should go to the mating pool for crossover and mutation. The selected individuals are operated through genetic operators namely crossover and mutation to produce new offspring. The crossover and mutation are carried out based on the pre-defined probabilities. Crossover is done in the view of identifying stronger individuals while mutation is done to bring out diversity in the solution. Then, the fitness of the new offspring is identified. The process is repeated for several iterations as set by the user or till convergence. This imitates the survival of the fittest.

Variations in Genetic Algorithms pertain to variations and proposals of new genetic operators, selection methods, representations etc. Variations can also pertain to hybrid methodologies that combine GA with other techniques to improve its potential. The concept of elitism has been introduced into the standard GA. This strategy retains the best individual of the current generation and carries it to the next generation. The individual is not altered by means of the genetic operators. Similar to crossover probability, and mutation probability, the number of elitists can also be set as a parameter to the algorithm.

Applications that use GA and its variants include various problems [28]. It has also been providing efficient results for engineering optimization problems. GA is an appropriate choice when fitness evaluation is very complex

2.1.2 PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is one of the oldest algorithms in the context of population-based swarm intelligence meta-heuristic optimization approach. It has been put forth by Eberhart and Kennedy in the year 1995 [29]. The algorithm is primarily motivated by the flocking behavior of birds and schooling of fishes. In PSO, the swarm is the population of the algorithm and particles (individual) are the member in swarms that represent the potential solution. Some basic terminologies in PSO are as follows:

With respect to position, there are three parameters, namely; (i) *pbest*, that represent the personal best position of a given particle till then, (ii) *lbest*, the local best, that depicts the position of the best particle member of the neighborhood of a given particle and (iii) *gbest*, the global best that signifies the location of the best particle in the entire swarm. The particles are initialized at random positions and they keep moving with a certain velocity till the global best improves no longer. The parameter *Velocity* (vector) is utilized to determine the direction and speed in which a particle should travel in order to enhance its present position. The *inertia weight* (w) is incorporated to govern the influence of the earlier velocities on the present velocity of a provided particle. There exist two Learning factors *C1* and *C2*. *C1* is the cognitive learning factor signifying the attraction of a particle towards its own success while *C2* is the social learning factor representing the attraction of a particle towards the success of its neighbors. These learning factors are usually constant that is defined during the inception of the procedure. Neighbourhood topology signifies the set of particles that contribute to the computation of the local best value (*lbest*) of a given particle.

The particles denote the individual feasible solutions. Each particle changes its position based on its own experience and also the experience of its neighbors. This is incorporated by storing the best position visited by it and its neighbors and based on this the local and global positions are determined. [13] PSO involves two main operations namely updation of velocity and updation of position. Every particle in the swarm is geared to march towards its best-known position and the global best position. After that, the velocity of each particle is recomputed based on its present velocity, the distance from its previous best position and the distance from the global best position. This recomputed velocity is then employed to estimate the next position of the particle in the solution search space. This process is iteratively performed for a predefined number of times or until a minimum error is accomplished. PSO or hybridization of PSO has been widely utilized in solving the single optimization problems because of its high convergence speed and relative simplicity [30]. However, the effectiveness of the algorithm greatly depends on the proper selection of parameter values as inappropriate parameter values easily pave the way to divergent results.

2.1.3 ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) has been introduced by Dorigo in the year 1992 [31]. It has been developed by emulating the activities of real ant colonies and has been put to use in solving optimization problems. The meta-heuristics involved in ACO is primarily based on the strategies adopted by ants while in search of food. During its search for food, the main motive of ants is to identify the shortest path between its nest and the food source. Every path established by the ants portrays a potential solution to the problem under consideration. During its forage, ants lay down a chemical substance known as pheromone. Through these deposits of pheromone, the ants tend to communicate with each other locally. This indirect communication mechanism is known as stigmergy. On identifying a path between its nest and food source, the ants deposit a certain quantity of pheromone in the path in order to influence other ants to take the same path. This is called positive feedback. As a result of successive deposits of pheromone on the same path, more ants tend to take this path which in turn results in still more pheromone getting deposited and subsequently still more ants will get attracted to it. Various variants of ACO are proposed for solving different types of SOO problems.

Ant systems [32], a variant of ant colony optimization, has been advocated with exclusive characteristics such as positive feedback, distributed computation and the utilization of a constructive greedy heuristic. It has been evaluated on the traveling salesman problem and the results report superior performance when compared with the performance of tabu search and simulated annealing. ACO has been employed to optimize the job shop scheduling problem [33]. The ACO approach adopts a local search technique namely food stepping in order to explore the solution space. In contrast to many other local search techniques, the food stepping technique is not problem specific and is flexible owing to the fact that it does not modify the ant system algorithm. On the other hand, it alters the information collected by the ants to enhance the current solution. The ACO methodology is used in the field of data mining for the purpose of classification. An ant-based classification technique called AntMiner [34], [35] is put forth with specific characteristics namely better performing max-min ant system, a clearly defined and augmented environment for the ant to move, the inclusion of class variables to tackle the multi-class problems and the capability to include interval rules in the rule list. The parameters have been tunes through an automated process. The performance reveals that it achieves better classification performance than traditional classification procedures. Subsequently, an ACO system named continuous orthogonal ant colony has been put forward for solving the continuous optimization problems more effectively [36], [37]. In this technique, the pheromone deposit mechanisms facilitate the artificial ant agents to search for solutions collaboratively and selectively. Through the incorporation of the orthogonal design strategy, ant agents in the feasible solution domain have the capacity to explore their chosen regions rapidly and efficiently. In addition to this, the inclusion of the adaptive regional radius reduces the risk of being caught into the local optima and therefore enhances the global search capability and accuracy [38]. An elitist strategy is also incorporated in the motive of retaining the most valuable points.

2.1.4 ARTIFICIAL BEE COLONY OPTIMIZATION

The Artificial bee colony (ABC) algorithm is one of the most popular swarm based evolutionary methods developed by D. Karaboga and B. Basturk [39] based on the foraging behavior of honey bees. The most essential and motivating characteristics of the bee are their foraging behavior, how they search the food source and collect the nectar and bring success to their hive. In real bee colony, the bees can be categorized into three types like scout bee, employed bee, and onlooker bee. The scout bees (unemployed bee) explore the food source and share the food source information with other bees by a special dance called waggle dance. The onlooker bees make the decision to choose food source by observing the dance regarding food source; the amount of nectar and direction of the source. In a powerful search process, both the process of exploration and exploitation should be performed simultaneously. In order to execute both the exploration and exploitation processes together, the exploration process is managed by the scout bees while the exploitation process is taken care of the employed bees and onlookers. The total cardinality of employed bees and the onlookers constitute the cardinality of the total population. The employed bee is transformed into a scout bee when its food source is exhausted. The position of a food source constituting a considerable amount of nectar depicts a possible solution to the optimization problem.

The ABC algorithm has been adopted for many applications. ABC algorithm has been utilized to deal with discrete optimization problem namely the leaf constrained minimum spanning tree problem [40]. In this problem, a spanning tree of minimum weight but having at least l leaves is sought for in an undirected, connected and weighted graph. ABC algorithm has been employed in order to reconstruct the gene regulatory network from the gene expression data [15]. In this technique, the notion of crossover and mutation has been incorporated to enhance the performance of ABC. The technique has also been put to use in noise-free and noisy time series datasets. The ABC root inference method has shown its potential in discovering considerable gene regulations and it has also shown comparable results when compared against its counterparts.

A memetic ABC algorithm has been put forth in [41]. In this, the technique only the best particle of the current swarm updates itself in its proximity. The memetic ABC algorithm involves four phases namely employed bee phase, onlooker bee phase, scout bee phase and the memetic search phase. Subsequently, a randomized memetic ABC has been suggested to improve the local search potential of the algorithm. It also involves four phases, out of which the first three are similar to memetic ABC while the fourth step is incorporated with two new parameters, whose values are arbitrarily chosen every time. During the fourth phase, global section search is utilized for producing solutions in proximity to the best solution. Experimental results show the improved performance in the context of reliability, efficiency, and accuracy.

2.1.5 CUCKOO SEARCH OPTIMIZATION

The Cuckoo Search (CS) Algorithm is introduced by Yang and Dev. It is rooted in the parasitic behavior and flight behavior of birds. It mimics the obligate brood parasitic behavior of few cuckoo species and the flight characteristics of some birds. The cuckoos lay its eggs in other bird's nest. In case the host bird identifies the alien egg, then either it gets rid of the egg or leaves the nest and build a completely new nest. This behavior is emulated in the CS algorithm [42]. Then, a Binary Cuckoo Search algorithm (BCS) has also been put forth to solve binary optimization problem based on a sigmoid function by [43]. They have potential applications in the domain of routing, job shop scheduling and flow shop scheduling. The standard cuckoo search algorithm represents the solutions in the form of a set of real numbers. In order to extend it to binary cuckoo search, these must be converted into binary. The binary cuckoo search is characterized by levy flights that are used to obtain a new cuckoo and binary representation to estimate the flipping chance of each cuckoo through sigmoid function. Other than that, the selection and objective function estimation is similar to standard algorithms.

Another variant of cuckoo search is the discrete cuckoo search algorithm (DCSA) [44]. It has been primarily adopted to solve Travelling salesman problem (TSP) in various ways. It has been operated on specific domain specific parameters to gear up the convergence. Then, Ouaarab et. al. have proposed a new category of cuckoos, thereby rebuilding the population to solve TSP, combinatorial and continuous problems effectively. Another variant of DCSA with two phases has been put forward

[45]. During the initial phase, discrete step size denoting the distance between the cuckoo and best cuckoo in the generation is computed. During the second phase, the cuckoos are updated using the step size and a random step length derived from levy distribution called levy flight.

Yet another variation of the cuckoo search algorithm is the modified cuckoo search algorithm (MCSA) [46]. With regard to unconstrained optimization problems, an MCSA has been advocated. The variation is proposed in the context of determining the step size. A random walk in a biased way with some random step sizes through the application of a different function set has been introduced to determine the step size. Subsequently, MCSA with rough sets has been introduced in [16]. In this method, the fitness function is formulated through two factors namely the number of features and the classification quality. The number of features is reduced which signify that the number of learning parameters is decreased, thereby yielding a faster convergence. Another variant called One-Rank CSA has been suggested. In this technique, the exploration and exploitation phases are integrated to produce new solutions. In basic CSA, exploiting new solutions is achieved based on Levy flights to achieve large moves. In some cases, the solutions can also skip the solution space as the step size is based on the scale of the problem. In this method, optimal utilization of Levy flight and elimination of invalid randomly selected solutions is efficiently handled. Then, Dinh et. al [47] has recommended an MCSA for Short-term hydrothermal scheduling by considering the existence of reservoir volume, fuel cost function thermal unit and power losses in the transmission line. The method tries to provide a new CSA solution based on alien egg discovery. Based on the value of their fitness function, all eggs are partitioned into high or low quality. The best egg selected will be used to obtain the increased value.

2.1.6 BAT ALGORITHM

Bat algorithm, inspired by echolocation behavior of microbats, is put forth by Yang [48]. Bat algorithm (BA) incorporates frequency tuning to increase the diversity of solutions while at the same time adopts automatic zooming in the view of trying to maintain a balance between exploration and exploitation during the process of searching thus mimicking the variations of pulse emission rates and loudness of bats when hunting for its prey. The characteristics of BA are as follows: All bats in the search space employ echolocation in order to sense distance. The bats are also capable of distinguishing the food/prey as against the background obstacles barriers. Bats make their flight randomly characterized with a velocity v_i at position x_i with a frequency f_{min}, varying wavelength λ and loudness A_0 in the view of recognizing its target prey. They have the potential to regulate the wavelength (or frequency) of their emitted pulses and also modify their rate of pulse emission $r \in [0,1]$ automatically in accordance with the proximity of their target. Though there are many possible variations in their loudness, for simplicity it is defined tat the loudness varies between a large (positive) A_0 to a minimum constant value A_{min}.

Though BA has many advantages, the primary advantage is attaining a quick convergence during its inception stage itself by switching from exploration to exploitation. Hence it becomes very appropriate for applications such as classification when faster results are expected. Though it converges very quickly, in some cases, it might get stuck in the local optimum. Hence many strategies have been adopted to increase the diversity of the solutions.

A variant which integrated the K-Means clustering procedure and the BA has also been advocated in the view of superior performance in clustering. The chaotic search has been incorporated into Bat Algorithms to result in Chaotic Bat Algorithms [49]. It adopts Levy flights and chaotic maps to perform parameter estimation in dynamic systems. Then, a binary version of the BA has also been put forward [17]. It is a discrete variant of the original BA and has been highly appropriate in dealing with classification and feature selection problems. Subsequently, Differential operators and Levy flight operators have been incorporated into the BA. The Differential operators and Levy flight Bat Algorithm has demonstrated its effectiveness in function optimization problems. After that, Jamil et al [17] have proposed a variation of BA by including a combination of Levy flights and minor variations in loudness and pulse emission rates.

Some other variants have also been proposed by grabbing a few concepts of other optimization algorithms. Mutation operator has been included in Bat algorithm with the view of improving the diversity of the solutions and tested in image matching applications. A hybrid version of Bat Algorithm and Harmony search algorithm has also been proposed and tested on numerical optimization of functions.

2.1.7 FIREFLY ALGORITHM

Firefly Algorithms (FA) is another class of optimization algorithms put forth by Yang during the year 2008, being inspired by the flashing light behavior of fireflies [50]. The flashing of lights acts as a courtship signal for mating. The males emit their light and to its response, the female emits back the flashlight. They tune among themselves emitting a particular pattern of light and then initiate mating. This behavior is emulated in the swarm intelligence based firefly algorithms.

The FA algorithm is primarily rooted in the physical characteristics of light intensity that decreases proportionally to the increase in the square of the distance. As the distance increase, the light may be absorbed and hence weakened. This concept is mainly utilized to design the objective function or fitness function. Some characteristics of fireflies that have been emulated in the optimization algorithm include the following: All fireflies are unisex and their attractiveness is proportional to their light intensity. The light intensity of the Firefly is based on the landscape of the fitness function. The original firefly algorithm has been very efficient in solving multi-modal optimization applications [51] and non-linear pressure vessel optimization [52].

Several variants have been proposed to the original firefly algorithms. The random motion of the brightest firefly has been modified. The modification attempts to improve the current position of the brightest firefly through the generation of m uniform random vectors and taking it towards the best performance. A large variety of binary firefly algorithms have been put forth for solving different optimization problems. In order to convert the traditional FA into Binary FA, almost all components need to be modified to suit the representation. A binary FA [53] has been proposed for cryptanalysis of Merkle-Hellman Knapsack cipher thereby deciphering the plaintext from the ciphered text. Another binary FA [54] that adopts binary encoding of the solution, an adaptive light absorption coefficient for gearing the search and domain knowledge to handle infeasible solutions.

A modified FA has been advocated to control the motion of fireflies. A Gaussian distribution has been utilized to control the speed and lead to convergence [55]. Though the randomization has been fixed, the parameters can be updated adaptively.

Another variant that incorporates the Levy flight into the motion of fireflies has been put forth [56]. Yet another variant that involves integrating the chaotic maps and the traditional FA has been introduced in the view of improving the convergence [57]. Then, a parallel version of FA has been put into operation [58] to improve the speed and quality of convergence.

To add more, many hybrid versions of FA have been formulated. An eagle strategy has been combined with FA to produce better results [59]. Eagle strategy emulates the foraging behaviors of eagles. The Eagles move in a random manner in search of their prey and once they find the prey, they try to capture it as efficiently as possible. It involves a random search by Levy flight and an intensive local search, which is replaced by FA in the proposed methodology. A hybrid version of GA and FA has also been put forth in which the FA algorithm utilizes the crossover and mutation operators to produce strong and diverse solutions [60]. After that, Evolutionary Firefly algorithm has been introduced which combines the classical firefly algorithm and the evolutionary Differential evolution algorithm [69]. This hybrid algorithm aims to improve the search accuracy and the information sharing among the fireflies. Then, another hybrid variant that combines the FA with the local search heuristics has been suggested and applied to graph coloring problem and has proved its efficiency [61]. The FA algorithms have also been used along with the back propagation method in order to train a feed-forward neural network [79]. In this methodology, the FA algorithm has been incorporated into the back propagation model in order to accomplish faster and improved convergence. Yet another hybrid FA that integrates the cellular learning automata into FA for increasing the diversity of the solutions has been advocated [62] . Also, a flexible neural tree for dealing with microarray data has been put forth [63].

FA and its variants can be used to solve optimization problems in any field. They have been extensively utilized in the fields of image processing, sensor networks and several other areas where optimization is very essential.

Having provided a detailed account of many bio-inspired meta-heuristic algorithms, the following section presents an optimization technique named Gravitational Search Algorithm that has been inspired by the law of gravity.

2.1.8 GRAVITATIONAL SEARCH ALGORITHM

Gravitational search algorithm (GSA) is developed by Rashedi et. al. in the year 2009 [64]. The optimization algorithm is formulated from the concepts of the universal law of gravity. In GSA, a collection of objects interact with each other based on the law of gravity and low of motion. Each object characterizes a mass that represents the performance of the object and is computed through an appropriate fitness function. The position of the mass of the object depicts the solution to the problem. These positions are updated during every iteration and the best fitness of the object is kept track of. The algorithm proceeds by tuning the gravitational and inertia masses. Intuitively, the objects with heavier mass attract other objects. After executing a pre-defined number of iterations, the best fitness of the corresponding object turns out to be the global solution to the problem. In general, there exist around nine parameters that have to be initialized and tuned for the operation of GSA. Some of them include the number of objects *N*, the number of objects with top fitness to be selected, number of iterations and a few parameters that control convergence, exploration, and exploitation.

GSA has been effective in providing optimal solutions in various optimization algorithms. The two highlighting issues in the search process include parameter convergence at local optimum due to rapid reduction of diversity and rapid convergence at the initial stage and slow convergence near the optimum of the local search resulting in ineffective iterations thereby failing inaccurate estimation of the optimum. Thus, a number of variants of GSA have been proposed in the literature to improve its performance.

Rashedi et. al., the founder of GSA, has proposed a variant of it namely the Binary Gravitational Search Algorithm (BGSA). It is based on the notion that if an object is very close to the global optimum, then its velocity should be near to zero. To incorporate this idea, a probability function is formulated for the absolute value of velocity such that the probability of changing the position is low for small values of velocity and probability of changing the position is high for large values of velocity. The main difference between continuous GSA and Binary GSA is that the position updates switches between 0 and 1 in BGSA whereas the updating of the force acceleration and velocity are all continuous as in the case of conventional GSA.The

BGSA has been evaluated on a range of uni-modal and multi-modal benchmark test functions and has demonstrated improved results.

The traditional GSA is memoryless. Attempts have been made to incorporate the concepts of memory and social behavior from PSO into GSA in the view of improvement [65]. As PSO uses a memory to store the best previous position of a particle and incorporates a velocity update mechanism, a similar technique is included in GSA also. Then, Li and Zhou [19] have put forth an improved GSA through the incorporation of moving strategy in the search space obeying the law of gravity, memory and social information of PSO. Two constants namely c_1 and c_2 are defined. The parameters are tuned such that a balance is maintained between the effectiveness of law of gravity and memory and social information. When these constants are set to 0, then it becomes the traditional GSA. The algorithm has been tested on parameter identification of hydraulic turbine growing systems. Khajezadeh et al [66] have proposed a controlled trajectory into the traditional GSA. This is done by defining a minimum and maximum velocity that an object can move. Also, a time-varying profile for velocity is defined.

2.1.9 ELECTRO-MAGNETISM OPTIMIZATION

The Electro-Magnetism Optimization (EMO) has been developed by getting inspired by the principles of electromagnetism [67] It searches a solution based on the attraction and repulsion among prototype candidates. It emulates the behavior of charged particles in an electromagnetic field in the view of evolving the members of the population thereby attaining an optimal solution. The primary benefit of this procedure is that even though it characterizes interesting search capabilities, it incurs only a very low computational complexity. On comparing the methodology with that GA, It does not involve the genetic operator namely crossover and mutation to explore feasible regions but incorporates collective attraction and repulsion to carry out the exploration process. It incurs a low computational cost in the context of memory allocation and execution time. It does not necessitate gradient information.

The methodology of EM-like algorithm initially involves generating a group of random solutions from the domain of feasible solutions. Each of the generated solutions is considered as a charged particle. The fitness function is utilized to estimate the charge of every particle. Owing to the charge designated to a particle, it moves with attraction or repulsion force among the population. The attraction-repulsion mechanism of this algorithm can be regarded analogous to the genetic operators of GA namely the reproduction, crossover, and mutation. The algorithm then computes the resultant force in the population for determining the direction of the considered particle's movement. This is done based on the Coulomb's law and superposition principle. The resultant force is estimated in accordance with the charges and distance associated with each particle. According to this technique, higher, the charge of the particle more will be the force of attraction or repulsion. The resultant force is negatively related to the distance between the particles. The EMO algorithm can enhance the current optimal solution through local search and move ahead of the feasibility of enhancing through global search. The EMO has been used for circle detection presented in an image by C.Erik Oliva et. al. [68]. Again this algorithm has been used in image segmentation for multilevel Thresholding. In this context, the search capabilities of EMO are integrated with the multi-threshold methods suggested by Kapur and Otsu. The methodology initiates by selecting a few samples randomly within the histogram of the image. These samples form the particles of the EMO algorithm. Its fitness is assessed based on the objective function that has been devised based on the methods advocated by Otsu or Kapur. On the basis of these objective values, a collection of solutions represented by the charged particles are evolved until an optimal solution is identified. The methodology evolves a multi-level algorithm for segmentation of images in the view of determining the threshold values within a fewer iteration and lesser computational complexity when compared to that of the originally proposed methods.

2.2 FITNESS COMPUTATION STRATEGIES

The fitness assignment for MOO techniques can be categorized broadly as (i) Aggregative (ii) Lexicographic (iii) Sub-population (iv) Indicator based (v) Pareto-based, and (vi) Hybrid methods. Out of these methods, most of the research works have placed its focus on Pareto-based approaches. All the extensions of SOO algorithms to MOO algorithms fall under any of the above-said categories. In this section, a brief explanation of each method is dealt and the classification of various fitness assignment methods is shown in figure 2.1.

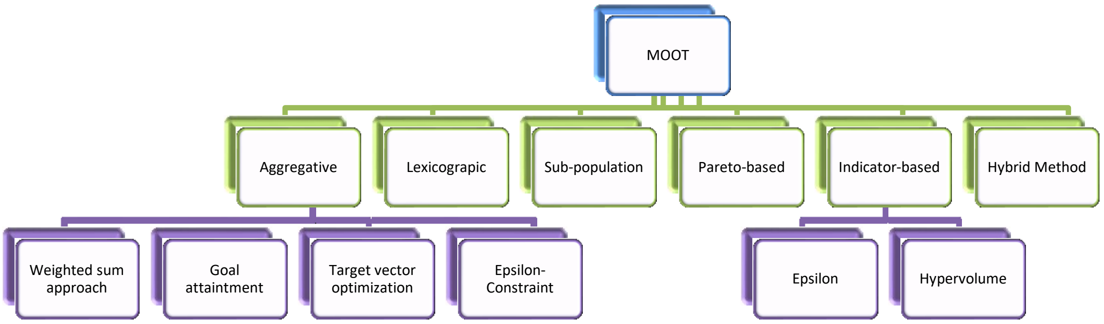


Figure 2.1 Fitness assignment method of MOOT

2.2.1 AGGREGATIVE APPROACHES

In aggregating approach, the multi-objectives are integrated to a single objective. The primary benefit owing to the aggregating method is that it results in one single solution. However, the challenge in defining such a goal function necessitates intense domain knowledge, which is unavailable many times. The widest aggregation approaches include the weighted-sum, goal attainment, target vector optimization, and Epsilon constraint method [69]. These popular aggregating approaches are briefed subsequently.

- ***WEIGHTED SUM APPROACH***

This approach has been the first attempt to obtain non-inferior solutions in the context of MOO. In this approach, the multi-objective context is transformed into a single objective function through the summation of the functions via different weight coefficients allocated for each one of them. This approach is also known as scalarization method [70]. It can be formulated mathematically as shown in equation 2.1.

$$\text{Min/Max} \qquad \sum_{i=1}^{k} w_i f_i\,(\bar{x}) \qquad (2.1)$$

In Equation 2.1, $w_i \geq 0$ refers to the weight coefficients that signify the relative significance of the objectives. Generally, the weight coeffients are assigned values such that $\sum_{i=1}^{k} w_i = 1$. Computational efficiency is one of the primary benefits of this approach. This technique is appropriate for generating a powerful non-dominated solution at the initial stage, which can be further, evolved using other methods. Nevertheless, determining the suitable weights during lack of enough information about the problem poses difficulty in using this approach. Besides this, it is also hard to identify all the non-dominated solutions through the weighted sum approach as long as every objective function and its feasible solution space possess the characteristic of convexity. As mentioned, the key challenge lies in associating suitable weight coefficients to each of the objectives. The weight coefficients are not actually proportional to the respective significance of the objectives or do not allow trade-offs between the objectives to be expressed. Moreover, the boundary of the non-inferior solution tend to be non-concurrent making a few solutions in accessible.

➢ *GOAL ATTAINMENT*

In goal attainment method, a collection of design goals is related to a collection of objectives [71]. The problem formulation permits the goals to be under-achieved or over-achieved. This makes the initial design goals relatively imprecise. The relative degree of under or over achievement of the objectives is governed by a vector of weight coefficients. This incorporates a component of flexibility into the problem. Otherwise, the condition would have been such that the objectives should be rigidly met. The weight vector, *w,* facilitates in exhibiting a measure of respective tradeoffs between the goals. For illustration, assigning the weight vector *w* to the inceptive goals signifies the attainment of the same degree of under or over achievement. The hard constraints are taken into account in the design by assigning a specific weight factor to zero. The goal attainment approach furnishes a handy understandable explanation for the design problem in hand. A set of coefficients of weights $w = [w_1, w_2, \ldots, w_k]$ interpreting the respective under or over-achievement of the desired motives has to be provided. For identification of the best optimal solution x^*, the equation 2.2 must be followed.

$$\begin{aligned} &\text{Minimize} && \alpha \\ &\text{Subject to,} && z_i^{ref} + \alpha.w_i \geq f_i(x); \quad i = 1,\ldots,k, \\ &&& x \in X \end{aligned} \tag{2.2}$$

In equation 2.2, α represents a scalar variable that can take any sign. The values assigned to the weight coefficients $w_1, w_2, ., . ., w_k$ are normalized so that $\sum_{i=1}^{k}|w_i| = 1$ holds true. In case, any weight coefficient, $w_i = 0$ *(i = 1,2.,.,., k),* it signifies that the maximum value achived by the objectives *$f_i(x)$* will be z_i^{ref}. Through this method, by incorporating variations in weights, all Pareto optimal solutions can be obtained with $w_i \geq 0$ *(i = 1,2.,.,., k)* even for problems that do not satisfy the convexity constraint. The vector z^{ref} is depicted by the decision motive of the decision maker. He is the one who is responsible for deciding on the direction of w also. On being provided with w and z^{ref}, the direction of the vector $z^{ref} + \alpha.w$ can be estimated. Hence, the problem defined in equation 2.2 can be regarded as equivalent to identifying a feasible point that is the closest to the origin on this vector. From equation 2.2, it is evident that the optimal solution is the first point at which z^{ref} + w cuts the feasible region in the objective space. If such a intersecting point exists, then it can be confirmed to have a Pareto optimal solution. The optimal value indicates the attainability of the goals. A negative value of z^{ref} indicates the attainability of the goal and hence an enhanced solution is sought after that.

➢ *TARGET VECTOR OPTIMIZATION*

In these approaches, targets or goals that are intended to be achieved in each objective have to be assigned. Goal Programming, Goal Attainment,and the min-max approach are some of the most popular techniques. The approach results in a dominated solution in case the objectives are selected in the feasible domain. This constraint can be considered as a bottleneck in applying this technique to many problems.

➢ *EPSILON-CONSTRAINT METHOD:*

An approach that solves a few of the convexity issues faced by the weight sum method is the ∊-constraint method. In this method, the most preferred or primary objective is to minimize F_p and expressing or considering the other objectives in the form of inequality constraints bound by some allowable levels ϵ_i as stated in equation 2.3.

$$\text{Minimize,} \quad F_p(x), \quad x \epsilon \boldsymbol{\Omega} \tag{2.3}$$

$$\text{Subject to,} \quad F_i(x) \leq \epsilon_i, \quad i = 1,2, \dots . n$$

In equation 2.3, ϵ_i are the assumed values of the objective functions. The search is stopped when a satisfactory solution is identified. Though the ∊-constraint method does not demand convexity, it yields only one non-dominated solution when certain particular conditions are satisfied.

2.2.2 LEXICOGRAPHIC METHOD

In this method, the goals are ranked on the basis of their order of significance. The objective functions are minimized one by one, initiating from the most significant objective and then continuing based on the rank of the objectives, in order to attain the optimal solution. The method is suitable only for a very less number of objectives, say two to three. Also, the performance of the approach is highly influenced by the ranking of the goals.

The subscripts of the objectives are intended to denote the objective function as well as the priority of the objectives. According to this assumption, $f_1(x)$ and $f_k(x)$ are the respective highest and the lowest significant objective functions. Initially, the first problem is framed according to equation 2.4 and its solution x_i and $f_1 = (x_1^*)$ is obtained.

$$\text{Minimize,} \quad f_1(x) \tag{2.4}$$

$$\text{Subject to} \quad g_j(x) \leq 0, \quad j = 1,2, \dots .m$$

After that, the second problem is formulated as in equation 2.5 and the solution of this problem is got as x_2 and $f_2 = f_2(x_2^*)$

$$\begin{aligned} &\text{Minimize} && f_2(x) \\ &\text{Subject to} && g_j(x) \leq 0, \quad j = 1,2, \dots .m \\ &&& f_1(x) = f_1^* \end{aligned} \tag{2.5}$$

This process is iterated until all k objectives have been taken into account. In general terms, the i^{th} problem is defined as in equation 2.6.

$$\begin{aligned} &\text{Minimize} && f_i(x) \\ &\text{Subject to} && g_j(x) \leq 0, \quad j = 1,2, \dots .m \\ &&& f_l(x) = f_l^* \end{aligned} \tag{2.6}$$

The solution obtained at the end, i.e. x_k, is considered to be the desired solution x^* of the problem.

2.2.3 SUB-POPULATION

In this technique, the whole population is partitioned into *m* smaller subpopulations and every subpopulation has the same size subject to the same constraints but different optimization objectives. The separation of the individuals into smaller groups allows a greater convergence speed in each sub-population. Also, if there exists certain independence between the sub-populations, each of them can result in converging at a different region in the solution search space thereby aiding to maintain some degree of diversity. These population-based methods make an effort to identify many Pareto-optimal solutions in one run of simulation.

2.2.4 PARETO-BASED METHOD

Pareto-based fitness assignment has been initially put forward in [72]. All methods based on this technique mandatorily and evidently utilize the concept of Pareto dominance the view of estimating the reproduction probability of each individual. The basis of a majority of MOO is the consideration that there are two contradicting motives namely (i) distance minimization towards the Pareto-optimal set and (ii) diversity maximization within the Pareto-optimal set. In general, there are mainly three goals in handling the multi-objective problems [71], [73], [74]. These are stated as (i) maximization of the cardinality of elements in the Pareto optimal set identified (ii) minimization of the Pareto front's distance generated by the optimization procedure with regard to the original (global) Pareto front and (iii) maximization of the spread of solutions identified, in order to gain a vector distribution that is as smooth and uniform as possible.

2.2.5 INDICATOR-BASED METHOD

The primary notion behind this technique is a formalization of preferences in terms of continuous generalizations of the dominance relation leading to a simple algorithmic concept. There are two types of indicator-based approaches namely Epsilon based and Hyper-volume Based. The Indicator based evolutionary algorithm permits adaptation towards arbitrary preference information and optimization cases. Also, it does not require any diversity maintenance methods [74]. It is more general owing to the flexibility that an arbitrary size of the population can be used. It is also faster as it considers only pairs of individuals for comparison and does involve the entire approximation sets.

2.2.6 HYBRID METHOD

In hybrid method, the above-mentioned approaches in Section 2.2.1 through 2.2.5 are used collaboratively on the bases of two factors namely the domain and the considered problem for optimization.

Having briefed on the various common strategies adopted during fitness computation in the context of multi-objective optimization, the subsequent section deals with the multi-objective variants of the popular meta-heuristic algorithms.

2.3 META-HEURISTIC OPTIMIZATION TECHNIQUES FOR MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

This section presents the variants of the popular meta-heuristic algorithms that have been proposed and applied for solving MOO problems.

2.3.1 EVOLUTIONARYAPPROACHESFOR MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

Evolutionary Algorithms are not very sensitive to the Pareto front's shape and continuity characteristics and deal with a set of Pareto optimal solutions. Genetic Algorithms is one of the widely used meta-heuristic algorithms. It has been utilized for MOO problems as well. By evolving a population of solutions, multi-objective evolutionary algorithms (MOEAs) are capable of approximating the Pareto optimal set in a single run [75]. A few popular methods are presented subsequently.

➢ VECTOR EVALUATED GENETIC ALGORITHM (VEGA)

By extending the Grefenstette's GENESIS program in order to solve the multiple objective functions, a variant namely Vector Evaluated Genetic Algorithm (**VEGA**) [76] has been put forth. In VEGA, the population is partitioned into N equal sub-populations. Each sub-population is designated a fitness based on the various objective functions. In the view of finding a trade-off solution, the crossover is permitted between two solutions in the entire population. The process of selection is carried out for each objective separately. Fitness proportionate selection technique is adopted during selection.

➢ NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA)

Non-dominated Sorting Genetic Algorithm **(NSGA)** [77] is another variation of GA that has been developed on the basis of various layers of classifications of the individuals. The ranking of the population is performed with respect to the non-dominated nature of the individuals. The entire sets of non-dominated individuals are categorized into a group wherein they share a dummy fitness value in order to maintain diversity within the population. The primary benefit while adopting this algorithm is that as many objectives can be reduced to a dummy fitness through the non-dominated sorting, there lies no restriction in the number of objectives that can be solved. Moreover, both maximization and minimization problems can be handled.

➢ NON-DOMINATED SORTING GENETIC ALGORITHM 2 (NSGA2)

This is the improved version of NSGA proposed by Deb et al. called NSGA 2 [78] . It is more efficient and uses elitism and a crowded comparison operator. It does not use an external memory and no additional parameter for diversity.Pareto rankings are used but keep tournament selection. It does not use an external memory and no additional parameters for diversity.

➢ NICHED PARETO GENETIC ALGORITHM (NPGA):

The Niched Pareto Genetic Algorithm (NPGA) [79] is a variant of GA that involves tournament selection grounded on Pareto dominance. A tie situation occurs in the scenario when both the individuals involved in the tournament are either dominated or non-dominated. In this case, the outcome that is the winner, of the tournament is decided on the basis of fitness sharing. In order to handle the noise during the selection method, a large population size is utilized in this approach. Yet another variation of NPGA has been proposed and named as NPGA2. In this technique, Pareto rankings are adopted along with tournament selection. No external memory is used and elitism mechanism is same as NSGA2.

➢ MULTI-OBJECTIVE GENETIC ALGORITHM (MOGA)

Multi-objective Genetic Algorithm (MOGA) is yet another variant of Genetic Algorithm in the view of solving multi-objective problems [80]. In this approach, an individual is ranked on the basis of the cardinality of chromosomes in the present population by which it is dominated. In this technique, fitness computation is done via three steps. Initially, the population is sorted based on the rank. Secondly, the fitness of the individuals is formulated through interpolation from the best to the worst rank. Finally, the fitnesses of individuals with the same rank are averaged so that all of them will be grouped at the same rate.

➢ MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM BASED ON DECOMPOSITION(MOEA/D)

Another variation is proposed based on decomposition and is called as Multi-Objective Evolutionary Algorithm based on Decomposition **(**MOEA/D) [81]. It is based on conventional aggregation approaches in which an MOO problem is decomposed into a number of scalar objective optimization problems, which can also be called as sub-problems. One main advantage of this approach is that a scalar objective local search can be used in each sub-problem in a natural way since its task is to optimize a scalar objective sub-problem.

➢ PARETO ARCHIVES EVOLUTION STRATEGY (PAES)

One more variant termed as Pareto Archived Evolution Strategy (PAES) [82] comprising of (1+1) evolution strategy has been introduced. In this approach, one parent can produce only one offspring. Then, it is included in the external archive in case it is a non-dominated solution. In this technique, the potential of the parent and the child is evaluated against each other. During evaluation of the potential, in case, if the child dominates the parent, then the child becomes the next parent and the iteration of the procedure continues. On a contrary situation, if the parent dominates the child, the child is discarded and mutation is executed subsequently in order to bring about diversity among the population. Suppose if both the child and the parent do not dominate each other, the individual to be retained is chosen between the child and the parent in the view of maintaining diversity among the solutions. In order to execute this strategy, an archive of non-dominated solutions is kept track of. In case of the mentioned scenario, the child is evaluated against all the candidates in the archive in order to examine if it dominates any one of the solutions. If it does, then the child becomes the new parent. Then, the dominated solution in the archive will be discarded. On the other hand, if the child does not dominate any candidates of the archive, then the nearness of both the parent and child to the candidates of the archive is investigated. During this computation, if the child lies in the least crowded region among the candidates of the archive, then it becomes the parent and is also included as a candidate of the archive. Subsequently, another variation of PAES called the multi-parent PAES has been put forward with similar principles and concepts as mentioned above.

- **STRENGTH PARETO EVOLUTIONARY ALGORITHM (SPEA)**

Another extension of GA to support MOO problems is the Strength Pareto Evolutionary Algorithm (SPEA) [83]. It employs an external archive to store all the non-dominated solutions obtained till then. During every generation, all the obtained non-dominated candidates are pushed to the archive. In case if any duplicates or dominated solutions exist among the candidates of the archive, they are discarded during the update operation. On encountering the maximum size of the archive, a few candidates are excluded from the archive through a clustering procedure that preserves the non-dominated nature of the archive. Moreover, the candidates in the archive are also permitted to take part in the genetic operations of the procedure. During every generation, a combined population is evolved through the integration of candidates from the archive and the current population. The fitness value that is designated to each individual of the combined population is estimated on the basis of the number of solutions dominated by the considered candidate individual. Also, the fitness values are allocated such that the dominated solutions get a value lower than the least fitness of any non-dominated solution. This strategy of fitness assignment assures that the search is geared towards the non-dominated solutions.

Subsequently, another version of SPEA namely **SPEA2** has been put forth [84]. It differs in terms of three aspects namely (i) fitness assignment that takes into account the number of solutions dominated by it and the number of solutions it is dominated by with respect to each individual (ii) nearest neighbor density estimation that facilitates a more precise guidance for the search process and (iii) a different archive truncation method that assures the preservation of boundary solutions. The main difference from SPEA to SPEA2 is with regard to archive updating operation.

2.3.2 PSO FOR MULTI-OBJECTIVE OPTIMIZATION

The basic PSO is not applicable directly to Multi-objective problems. The solution of a multi-objective problem comprises of a set of equally adequate solutions. Initially, PSO has been extended to multi-objective optimization by Moore and Chapman [85]. A *p-list* has been utilized in order to maintain a track of all the non-dominated solutions explored by the particle in the search space. During every updation of the x-vector of a particle, it is compared to the solutions in the p-list to examine if it is a non-dominated solution. If true, it is included in the p-list. Moreover, the p-list is also constantly updated to ensure that it comprises of only non-dominated solutions. Majority of the proposed multi-objective Particle Swarm Optimization (MOPSO) approaches to define the notion of leaders. Each particle may possess several leaders, out of which only one is chosen to update its position [86]. These leaders are stored in an external archive, which is a separate location from that of the swarm. It holds the responsibility of holding track of all the non-dominated solutions identified till then.The solutions stored in the external archive are considered as leaders when the positions of the particles in the swarm need to be updated. Further, the solutions stored in the external archive are also revealed as the final outcome of the procedure.

The most straightforward and elementary means to determine a leader selection is to deem all the non-dominated solutions as leaders and then select one of them. However, in this approach, a chief consideration is associated with the estimation of quality indicating how potential a leader is. Functions of density measure have also been employed for electing the leader. In the context of multi-objective optimization, the most widely used density estimators include those that are based on Nearest neighbor and Kernel [87]. Nearest neighbor density estimator provides an insight into the crowd density of the nearest neighbors of the considered particle. On the account of sharing its resources with others, the fitness of such particles is degenerated by a parameter in proportion to the cardinality and the nearness to the particles that bound it. The neighborhood of a particle, called a niche, which signifies the radius of the neighborhood, is defined in terms of a parameter known as σ^{share}.

The retaining of solutions throughout the entire search process is another important challenge in multi-objective optimization. Usage of an external archive is the most common means to retain solutions that are non-dominated with regard to all the earlier swarms. Such an archive permits the inclusion of a solution only if: (a) it is non-dominated with regard to the already existing solutions in the archive or (b) it dominates any of the solutions within the archive. The main disadvantage of the external archive is that the archive size increases very quickly as in every generation the archive has to be updated. This update may become very expensive if the size of the archive grows too high. In MOPSO, a set of leaders is also initialized with the non-dominated particles from the swarm and stored in an external archive. When it is mapped for MOO generally three archives are used. The first one is utilized to store the global best solutions, the second one is used for saving the personal best values and a third one is employed for keeping track of the local best. However, in reality, utilization of more than three repositories for implementation of MOPOSOs have been reported. Most of the existing MOPSOs apply some sort of mutation operator or turbulence operators after performing the flight [88].

X. Hu and R. Eberhart [88] have modified the PSO to deal with the MOO problem by considering a Dynamic Neighborhood strategy, a new particle updating strategy. In dynamic neighbor, each particle has a different neighbor in each generation based on the fitness value. The updating strategy updates only if they encounter those solutions that dominate the current pbest. Later, this Dynamic Neighborhood PSO has been modified by using extended memory to store the global optimal solution [89]. An extension of PSO has been advocated through the adoption of Pareto Dominance for estimation of the flight direction of a particle [90]. A global repository is incorporated to keep track of the non-dominated solutions earlier obtained. This can be exploited by other particles to regulate their own flight in future. A variant of this approach, incorporating secondary repository with the intention of regulating the flight and mutation operation, thereby enriching the exploratory capabilities of the algorithm, has been put forth [91]. The disadvantage of the original method is the multi-frontal problem, which has been overcome in the extended version. The main aim of mutation operator is to explore remote regions of the search space and to ensure that the full range of each decision variable is explored. The main objective of the external repository (or archive) is to maintain historical information of the non-dominated solutions found along the search process. The two primary constituents of the external repository are the archive controller that holds the responsibility of deciding whether a solution should be included in the archive or not and the adaptive grid that holds the responsibility of generating well-distributed Pareto fronts. The primary benefit of using the grid as against niching is that the computational cost incurred during grid implementation is lower.

Then, yet another variation in PSO has been put forward by integrating PSO with clustering for the purpose of partitioning all particles into various sub-swarms and then utilizing it for automated docking. Then, a Time-Variant MOPOSO (TV-MOPSO) that is adaptive with regard to its inertia weight and acceleration coefficients has been suggested [92]. Owing to its capability of adaptability, exploration of search space is still more effective thereby achieving a good balance between the exploration and the exploitation of the search space. A diversity parameter is also employed for assuring considerable diversity amongst the solutions of the non-dominated fronts. The parameter also takes care that the convergence to the Pareto-optimal front is maintained. An investigation on Pareto-ranking based quantum-behaved PSO (QPSO) has been presented [93], In this method, an external repository is employed for maintaining and keeping tracking of the non-dominated solutions. The global best position is selected from this archive. For selecting the elitists, three different schemes namely preference order, sigma value, and random selection methods have been adopted. Then, an optimality criterion grounded on preference order strategy has been propounded for obtaining the best compromise solution [94]. This preference order has been put forth to rank all the particles and thus to identify the global best particle.

V.L. Huang et.al. [95] have presented a multi-objective comprehensive learning particle swarm optimizer (MOCLPSO). Here a learning strategy is used which utilizes the history regarding the best position of all other particles to update the velocity of a particle. This strategy enhances the diversity and prevents premature convergence. A two-local-best (lbest)-based multi-objective PSO (2LB-MOPSO) technique that is different from canonical MOPSO has been suggested by S.Z. Zhao and P.N. Suganthan [96]. It employs two local bests rather than a single personal best and global best in order to direct each particle. In the view of improving the local search ability of the process, the selection of two local bests is made in such a way that they are in proximity to each other. This method demonstrates high benefits with regard to convergence speed and fine-searching ability. X. Yu and X. Zhang [97] have developed multi-swarm CLPSO (MSCLPSO) for multi-objective optimization. It incorporates multiple swarms wherein every swarm is attached to an exclusive original objective. It employs conventional archives in order to save the elitists. MSCLPSO varies from existing MOPSO in three aspects. Firstly, every swarm attempts to optimize the attached objective without gaining knowledge about the elitists or the other swarms. Secondly, elitists are subjected to mutation as the concept of mutation exploits the personal best positions and elitists suitably. Finally, a modified differential evolution (DE) concept is put to operate on a few extreme and least crowded elitists. The difference among the elitists is used as a basis by the DE for updating of elitists. The personal best positions characterize important information regarding the Pareto set while the mutation and DE strategies aid the MSCLPSO in discovering the true Pareto front.

There are endless applications for various MOPSO in different fields and domains. Molecular docking problem has been handled by a variant of MOPSO [98]. In this context, the particles are partitioned into groups. Then, the global best of a particle is identified from its own group. Then, a weighted-sum of the objectives is utilized to keep track of its local best. D.S. Liu et. al. [98] has devised a Multi-objective Evolutionary PSO (MOEPSO) algorithm that incorporates concepts of Evolutionary algorithms such as the use of mutation operator as a source of diversity. It has been used for solving multi-objective bin packing problem. It is characterized by the fact that particle movement is directed by means of either personal best or global best only. This is in contrast to the concept followed in earlier works, wherein the movement of a particle is influenced by both personal and global best at the same time.

2.3.3 ACO FOR MULTI-OBJECTIVE OPTIMIZATION

Variants of ACO are proposed for solving different types of SOO problem as well as MOO problems. The multi-objective algorithms manifest different design choices for dealing with the traits of multi-objective contexts. One of the significant characteristics of the Multi-objective ACO (MOACO) algorithms is the incorporation of heuristics in the context of enhancing the potential of the identified solutions. On the account that heuristics provides further insights into the problem at hand, it can be expected to yield much better solutions than that of those algorithms that do not incorporate it [99]. In MOACO, the management of the pheromone information is an intricate task. It involves defining the pheromone information such as (i) the approach utilized to aggregate the weights of various pheromones (ii) the strategy for selection of solutions that can update the pheromone information and (iii) the methodology adopted by these solutions to change the pheromone information. The incorporation of multiple colonies has also been put forward so that each of the colonies works independently weighing the relative importance of the multiple objectives variedly. When multiple colonies are taken into consideration, handling of pheromone information becomes even more complex [100]. Therefore, in some methods, the usage of local search methods is also considered. All these features can be viewed as various components of a specific configuration of a generic MOACO algorithm. Broadly, there are two different search strategies used to handle the MOACO problems. They are the dominance relations and several scalarizations of the objective vector. Some of the ACO algorithms that perform highly effective in the case of SOO are the Ant Colony System (ACS) and Min-Max Ant Systems (MMAS). These algorithms can be extended to MOACO with equivalent strategies to handle multi-objective problems. [101].

Since last twenty years, more interest has been shown on exploiting the potential of MOACO in various fields with various modification and improvement. For instance, MOACO has been widely used to solve problems such as traveling salesman, vehicle routing, flow-shop scheduling and portfolio selection [102],[103],[99],[100],[104] etc. An optimization strategy for MOACO has been put forth through optimization of the initialization of the pheromone matrix using the prior information obtained through Physarum-inspired Mathematical Model (PMM) [105]. This has been applied to solve binary-TSP.

An extension to the Population-based ACO algorithm is proposed by incorporating a crowding population replacement scheme to enhance the effectiveness of the search process and has been applied to solve multi-objective traveling salesman problem. The Crowding Population ACO (CPACO) algorithm has the capability of identifying and maintaining a diverse set of solutions across the Pareto front. This, in turn, facilitates in identifying better solutions from all regions of this front. As the CPACO builds solutions on the basis of pheromone matrix, which indicates the performance of the entire population irrespective of the position of the solution on the approximate

Pareto front, there can be some futile efforts during the implementation of this procedure.

T.B. Kurniawan et. al. [106] has proposed a Population-based Ant Colony Optimization (P-ACO) to solve the DNA sequence optimization. Sabino Jodelson A. et. al. [107] has developed two variants of the Ants algorithm to tackle the specific problem of switch engine scheduling in a railroad yard (SESR). This SESR problem is solved by using multiple ant colonies. For solving a multi-objective supply chain design problem, a Pareto ACO has been advocated by Moncayo-Martínez et. al. [108]. In this regard, a number of colonies ants are used in a sequence to explore the solution space and search for a successively better non-dominated set of supply chain designs. A multi-objective Ant Colony Optimization has been proposed by López-Ibánez et.al [109] for solving an automatic design problem. This MOACO algorithm provides various design choices for handling the characteristics of the multi-objective problems.

2.3.4 ABC FOR MULTI-OBJECTIVE OPTIMIZATION

As the ABC algorithm has proved its effectiveness in solving the SOO, it has been extended for solving multi-objective as well as many objective optimization problems. The conventional ABC algorithm has been extended to support multi-objective problems through the incorporation of a grid-based technique in order to maintain and adaptively evaluate the Pareto front. The Pareto set is utilized for the purpose of controlling the behavior of flight of individuals and structure of the bee colony. A fixed-sized archive is employed for keeping track of the good solutions. This archive is managed through the ∈-dominance method. While using ∈-dominance, the size of the external archive is based on the user-defined ∈ value. The employed bees incorporate the social information got from the external archive to adjust their flying trajectories. The grid manages in maintaining the diversity within the external archive. The solutions generated by the employed bees are assessed by the onlooker bees so they can update their next position based on the solution attained. Finally, the solutions that have attained trial limit are replaced by the scout bees with a new random solution in the search space.

Vector Evaluated ABC (VEABC) is a parallel vector evaluated variant of the ABC for solving MO problems. An extended version of this algorithm [110] primarily for discrete variables has been put forth in the context of optimization of composites. An Adaptive Multi-Objective Artificial Bee Colony (A-MOABC) Optimizer has been proposed [111] through the incorporation of Pareto dominance concept. It also involves the concepts of crowding distance and windowing mechanism. An adaptive windowing mechanism is employed by the employer bees in order to choose their own leaders and in order to change their current positions. In addition to this, the positions of the onlooker bees are modified based on the food sources advocated by the employer bees. Crowding distance technique employed in this procedure aids in managing the diversity in the archive.

Three variations of MOABC algorithms have been suggested with its roots in synchronous and asynchronous models employing Pareto dominance and non-dominated sorting [112]. The algorithms include (i) Asynchronous MOABC optimization with Pareto Dominance (A-MOABC/PD) (ii) Asynchronous MOABC Optimization with Non-dominated Sorting (A-MOABC/NS) and (iii) Synchronous MOABC Colony Optimization with Non-Dominated Sorting (S-MOABC/NS). S-MOABC/NS is demonstrated to be highly scalable and efficient when compared to the other two variations. The conventional Non-dominated Sorting ABC algorithm has been extended in order to obtain Pareto-optimal solutions effectively and efficiently even in the presence of noise on the fitness landscapes [113] For this purpose, three strategies have been devised. The first strategy involves the adaptive selection of sample-size in order to maintain the trade-off between accurate estimation of fitness and the computational complexity. The second strategy deals with estimating the statistical expectation as a metric of fitness for trial solutions rather than the usual averaging. The third strategy is associated with extending the Goldberg's approach to checking if a slightly inferior solution can be placed in the optimal Pareto front. Y. Xiang et. al. [114] has recommended an elitist MOABC (eMOABC) using an elitism strategy and a crowding-distance archive to keep a good spread of the obtained solutions. During every iteration, the algorithm chooses two elites which have a maximum crowding distance and are defined as the archived intermediate solution. These elites are them utilized in adjusting the trajectories of flight of both the employed and onlooker bees. The algorithm incorporates the elites as well as the neighbors to direct the bees' trajectories of flight. During the employed bees phase, an intermediate solution that has the maximum crowding-distance is chosen as the elite, which is then utilized to generate new food sources. After updating the entire bee colony, the crowding-distances are estimated once again. Then, another elite is chosen with the maximum distance. Now, this elite will be subjected to exploitation in the subsequent onlooker bees' phase. The elitism strategy is targeted at enhancing the exploitation potential of the eMOABC algorithm. The merit of this algorithm can be stated as the ability to exploit more potential non-dominated solutions and preserve the diversity of solutions.

A dynamic multi-colony MOABC algorithm (DMCMOABC) has been advocated by employing the multi-deme model and a dynamic information exchange concept. [115]. This algorithm is designed such that k different colonies search independently for the majority of the time and shares the essential information intermittently. Each colony comprises a fixed number of bees such that the number of the onlooker and employed bees are equal. For every source of food, either of the bees will explore temporary position generated through neighboring information. The richness of the food source is estimated through a greedy selection approach and the better one is retained for the subsequent iterations. An external archive is employed to save the non-dominated solution while the diversity within the archive is maintained through the crowding distance method. If a randomly generated number is smaller than the migration rate R, then an elite is selected and the food source with the worst fitness is replaced by this elite. During each migration, an elite is chosen from the external archive. The migration direction is dynamic as the elite may be selected by any colony. It is also to be noticed that the colony that receives the elite is also estimated stochastically.

By applying the fast non-dominated sorting and population selection strategy to measure the quality of the solution and select the better ones, Y. Huo et. al [116] has proposed an elite-guided MOABC. The elite-guided generation of the solution is devised in order to exploit the neighborhood of the existing solutions on the basis of the guidance attained from the position of the elite. In addition to this, a fitness calculation method has been discussed to compute the probability of choosing the onlookers._Selection model and searching scheme of artificial bee colony algorithm and diversity maintaining scheme have been improved by W. Y. Wu Chunming and Li Tingting in [117]. W. Zou et. al. [118] presented an MOO method based on the artificial bee colony using the concept of Pareto dominance to determine the flight direction and it maintains the non-dominated solution vectors in an external archive. They sort bees based on non-domination in the initialization phase and store them in the external archive. In their method, all the bees are regarded as onlookers and there does not exist employed and scout bees.

The different variants of MOABC are applied in various field like static routing and wavelength assignment problem [119], motif discovery problem and discovering novel transcription factor binding sites in DNA sequences [120], power and heating system [121], frequency assignment problems [122]., wireless sensor network [123], image segmentation [124], robot path planning [125], FIR filter design [126] etc.

2.3.5 CUCKOO SEARCH ALGORITHM FOR MULTI-OBJECTIVE OPTIMIZATION

The cuckoo search has been extended for MOO by Yang and Deb [127] For MOO problems with K different objectives, the CS algorithm is modified as follows: each cuckoo lays K eggs at a time and dumps them in a nest that is randomly chosen. The best nests depicting high quality will sustain and be carried over to the subsequent generations. The cardinality of available host nests is constant. The egg laid by a cuckoo is recognized by the host bird with a probability p_a. Once the host identifies the cuckoo egg, it can either get rid of the egg or abandon its nest and build a new nest. Also, the probability can be used by n host nests to replace the new nests, if better. Some random mixing can be used to generate diversity.

A. Layeb has proposed a cuckoo search for binary multi-objective optimization. Pareto dominance is used to find optimal Pareto solution. It has been evaluated on the knapsack problem. H. V. H et al. [128] has applied the multi-objective cuckoo search algorithm to Radial Basis Function Neural Networks Training for System Identification. I. Kahvazadeh and M. S. Abadeh have proposed a Pareto based multi-objective cuckoo search algorithm that evolves efficient association rules from numeric datasets. The parameters related to the generation of association rules namely the support, confidence, interestingness, and comprehensibility are regarded as the objectives to be optimized. The algorithm evolves rules incrementally such that during every run, a few efficient rules are generated. In order to perform task scheduling effectively on heterogeneous systems, M. Akbari and H. Rashidi [129] have put forward an algorithm based on multi-objective scheduling cuckoo optimization algorithm (MOSCOA) in the view of reducing execution time while allowing for maximum parallelization.

2.3.6 MULTI-OBJECTIVE BAT ALGORITHM

The Bat algorithm has been extended for the multi-objective problems by using the weighted sum approach, where all the objectives are combined into a single objective [130] In this case, the weights are assigned randomly based on uniform distribution. This provides the possibility to vary the weights with considerable diversity so that the Pareto front can be approximated suitably. In order to incorporate Bat algorithm effectively in binary space, a multi-objective binary bat algorithm (MBBA) [131] that employs a modified bat position updating strategy to suit binary problems has been put forward. It also characterizes a mutation operator for enhancing the local search potential and maintaining diversity. Then, an approach based on Pareto dominance along with the external elitist archive has been put forth to identify optimal Pareto solutions. It also involves a procedure to choose the leader of flight in order to aid in the flight of the bats. Then, Yang Nien-Che and Minh-DuyLe have developed a method to optimize the design of passive power filters (PPFs). The most useful inertia weight with the best effect has been selected to optimize performance [132]. The external archive has been utilized in order to retain the multi-objective solutions. Subsequently, Tharakeshwar T. K et. al. [133] has adopted Multi-objective BA for solving shell and tube heat exchange problem. Again, Yang Nien-Che and Minh-Duy Le [134] have proposed a MOOby using modified BA and Pareto front for solving passive power filters (PPFs) design problem in order to suppress critical harmonics and improve power factor.

Hybrid versions of BA have been introduced in the view of improved performance. Bat algorithm has been hybridized with Artificial Bee Colony Algorithm and has been used to solve the Multi-objective Radio Frequency Identification network planning problem [128]. In this technique, the search procedure of the original BA is enhanced by incorporating onlooker mechanism from ABC algorithm.

The concept of fuzzy logic has been introduced in the bat algorithm in [135] The fuzzy logic bat algorithm offers fuzzy good judgments in the set of rules, thereby providing an efficient solution. Again in the same year, Yang [136] extended BA to solve MOO problems. The technique has proved its effectiveness in many engineering design optimization problems.

2.3.7 FIREFLY ALGORITHM for MULTI-OBJECTIVE OPTIMIZATION

The original firefly algorithm is extended for tackling MOO problem. FA can be directly used for MOO by using weighted sum approach [137]. Another way to solve MOO problem is producing Pareto optimal front by modifying or improving the original methods. The FA has been extended to solve MOO problem and applied in engineering design optimization [138]. Fran SérgioLobato and Jr. Valder Steffen have extended FA for multi-objective by associating the classical FA with the fast non-dominated sorting and the crowding distance [139]. All the dominated solutions are removed from the population by using the fast non-dominated sorting and sorted into the non-dominated front. The FA has been used to generate new firefly population and when the number of individuals has increased, it is truncated by using a crowding distance operator. An anti-stagnation operator has been adopted in order to avoid the stagnation process. This MOFA has been utilized to solve classical (bio) chemical engineering system design. Subsequently, a multi-objective non-dominated sorting firefly algorithm (MONSFA) has been advocated [140]. In the view of updating the population with high-quality solutions, characteristics such as the global search ability, the non-dominated sorting, and population crowding distance selection are incorporated during every iteration. This updated strategy is analogous to that of the strategy adopted in NSGA-II only with an exception of a different formula to compute the crowding distance as the formula used in NSGA- II is not appropriate when the number of objectives is more. The method also facilitates in choosing better solutions in the non-dominated set that are more evenly distributed.

Two modified versions of the firefly algorithm, one using the weighted sum method and the other employing the Pareto-dominance method have been suggested to solve the multi-objective task scheduling problem. A decomposition-based firefly algorithm has been designed to solve FRID network planning problem [141]. Radio frequency identification (RFID) is widely used for item identification and tracking. Thus multi-objective firefly algorithm finds its application in many multi-objective problems.

2.4 OPTIMIZATION TECHNIQUES IN ENGINEERING OPTIMIZATION PROBLEMS

There are various optimization problems in real world. Out of these, five problems have been taken as a case study in this research. This section provides the earlier works performed in the context of optimization in these case studies [142]. The studies include (i) Tension and Compression Spring Design Optimization and Welded Beam Design Optimization in the context of SOO (ii) Welded Beam Design and Disc Brake Design Optimization with regard to MOO and (iii) Multi-level threshold optimization for image segmentation. The following sub-section deals with the existing works in these problems.

2.4.1 EARLIER WORK IN SINGLE OBJECTIVE OPTIMIZATION OF TENSION AND COMPRESSION SPRING DESIGN AND WELDED BEAM DESIGN

The existing algorithms to handle spring design optimization and beam design optimization are concisely presented here. Based on the socio-behavioral concept of society and civilization Akhtar et al. [143] has developed a method for solving single objective constrained optimization problems. The primary idea is to interact with leaders of all societies for the improvement of the society. They have tested their algorithm using Welded Beam Design problem. It requires 19,154 evaluations to get the objective value 2.4426. After that, Mahdavi et. al. (2007) have proposed an improved harmony search algorithm that generates new solutions to enhance the accuracy and the convergence rate of the harmony search. They have solved the spring design problem and welded beam design problem using 50,000 and 300,000 evaluations respectively. The best values identified have been good but it has taken a higher number of iterations compared to other algorithms. Then, Hernandez et al. [144] have introduced constraint optimization using PSO including two new perturbation operators to prevent premature convergence and applied to solve engineering design problem by [145] .

An extended version of ABC algorithm has been advocated through the inclusion of a constraint handling technique during the selection process so that the feasible regions are chosen rather than the entire search space. This extended algorithm has been employed for solving engineering design problems. Their method requires 30,000 evaluations to obtain the best value for both problems of spring design optimization and welded beam design optimization. Cagnina et al [146] have proposed a simple method using PSO to handle constraints and a different mechanism to update the velocity and position of each particle. Yang and Deb [147] has introduced cuckoo search method which is based on the breeding behavior of cuckoos to solve Engineering optimization problem.

The following sub-section provides an account of the earlier works with respect to Multi-Level Threshold Optimization and multi-objective welded beam design and disc brake design.

2.4.2 EARLIER WORK IN MULTI-LEVEL THRESHOLD OPTIMIZATION FOR IMAGE SEGMENTATION

The existing works pertaining to multi-level threshold optimization is briefly presented in this sub-section. The approaches usually select thresholds by optimizing (maximization or minimization) some criterion functions defined for images There exist several classical thresholding methods like Otsu's class variance method that maximizes the variance between classes, Kapur's Entropy Criterion Method that uses the maximization of the entropy to measure the homogeneity among classes, Non-extensive or Tsallis entropy method etc. Since the classical methods search for the best values exhaustively to optimize the objective function for multi-level thresholding, it is computationally expensive and the use of evolutionary approaches for optimization has proved to be efficient. Various meta-heuristic algorithms such as Genetic Algorithm, Particle Swarm Optimization (PSO), Bacterial Foraging Optimization (BFO), Differential Evaluation (DE), Artificial Bee Colony (ABC), Cuckoo Search (CS), Galaxy-based Search Algorithm, Harmony Search Optimization, Bat Algorithm, Electro-magnetism Optimization. Firefly Algorithm, hybrid method etc. are widely used for solving the optimal multi-level image segmentation problem. The classical and optimization algorithm based thresholding methods are employed to find the best possible threshold in the segmented histogram by satisfying some guiding parameters.

A general scheme to segment images through GA using an evaluation criterion is developed by S. Chabrier which quantifies the quality of the image segmentation result. This method utilizes the knowledge of the ground truth when available in the view of setting the desired level of precision of the final outcome. GA is then utilized to identify the best combination of information that has been elicited from by the selected criterion. A framework formulated on the basis of Multi-Agent System theory and hybrid GA has been proposed for the purpose of image segmentation. In this method, initially, every segmentation agent considers a sub-optimal image and implements the Iterated Conditional Modes algorithm and yields the segmented image to the coordinator agent. The coordinator then diversifies these initial sub-optimal images by subjecting it to hybrid genetic operators in order to produce new promising starting solutions which are refined once again by the segmentation agents. Then, Kamal H., et. al. has proposed a method for image segmentation by combining GA and wavelet transform [148]. Firstly, the length of the original histogram is diminished through the application of wavelet transform. With the histogram of this lower resolution image, the number of thresholds and values of the threshold are estimated through GA. Similarly using PSO, Akhilesh Chander, et. al. has presented a self-iterative method (Otsu's method) to find the appropriate number of thresholds in order to delineate an image. The thresholds that are attained as an outcome of this iterative procedure are considered as initial thresholds. Then, for the current PSO variant, the particles are generated randomly around these thresholds. This algorithm adapts social and momentum characteristics of the velocity equation in order to update the movement of the particles. A hybrid cooperative-comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding is developed by M. Maitra and A. Chatterjee where an improved variant of PSO employs cloning of fitter particles, at the expense of worst particles, in an attempt to further enhance the capability of the optimization strategy. PSO is also modified and hybridized with another algorithm for multilevel thresholding image segmentation [149]. Then, P. D. Sathya and R. Kayalvizhi [150], [151], [152],[153] have adopted bacterial foraging algorithm to find the optimal threshold values for maximizing the Tsallis, Kapur's and Otsu's objective functions. Using ABC algorithm, Ming-Huwi

Horng [154] has put forth the maximum entropy based ABC thresholding (MEABCT) method for image segmentation. After that, Miao Ma et. al.[155] and Kazim Hanbaya and M. Fatih Talu [156] have also incorporated ABC and improved ABC algorithm for SAR Image segmentation. Then, Diego Oliva et. al. has used the Harmony Search algorithm for multilevel thresholding in image segmentation that encoded random samples from a feasible search space inside the image histogram as candidate solutions[157], whereas their quality has been evaluated considering the objective functions that are employed by the Otsu's or Kapur's methods [158]. Some other meta-heuristics algorithm like cuckoo search [159], Galaxy-based Search Algorithm [20], [160]–[162], Bat Algorithm [163], Electro-magnetism Optimization [164], Firefly Algorithm [165] etc. are also used widely in image segmentation. Diego Oliva et. al. have put forth a method that integrates the characteristic search potential of the EMO algorithm with the objective functions of the widely used MT methods proposed by Otsu and Kapur.

2.4.3 EARLIER WORK IN MULTI-OBJECTIVE OPTIMIZATION OF WELDED BEAM DESIGN AND DISC BRAKE DESIGN

This sub-section elaborates on the existing works that have been carried out in the optimization of two multi-objective Engineering Design problems namely the welded beam design and the disc brake design.

The disc brake optimization problem has been formulated by Osyczka and Kundu [166]. The authors have utilized the modified distance method in GA to solve the disc brake problem and have compared their results with that of a plain stochastic method. Then, Ray and Liew [125] have adopted a swarm metaphor approach in which a new optimization algorithm based on behavioral concepts similar to real swarm have been proposed to solve the disc brake problem. After that, Yıldız et al. [135] have employed a hybrid GA combining Taguchi's method and GA. The incorporation of robust design of parameters with GA via a small population of individuals has resulted in optimal parameter settings for design optimization problems. In order to evaluate this method, L16 orthogonal arrays have been tested. On the basis of the impact of design parameters on constraints, objectives, and inequalities, ANOVA statistical tests have been used to identify the optimal levels of these parameters. Subsequently, a hybrid approach integrating immune algorithm and the hill climbing local search procedure has been advocated for attaining efficient solutions towards complex real-world optimization problems. The outcomes of this approach are reported with respect to the design of disc brake and have been compared against the solutions provided in the literature. In 2012, a multi-objective Bat Algorithm has been adopted to optimize the welded beam design [167]. Then, a multi-objective cuckoo search has been incorporated to solve the problems of welded beam design and the disc brake design [168]. In this method, the single objective Cuckoo search algorithm has been extended to suit the multi-objective design problem. The proposed algorithm yields comparable performance with regard to optimization of disc brake design and welded beam design at around 1000 iterations. Then, the same author [169] has used multi-objective firefly algorithm (MOFA) for solving this disc brake problem. By extending the basic ideas of FA, Yang et al. have developed multi-objective firefly algorithm. Then, a multi-objective flower algorithm using weight sum approach with random weights has been utilized to solve the two-objective disc brake problem [170]. The algorithm emulates the flower pollination aspects and shows good performance.

Reynoso-Meza et al. [171] have used the evaluation of design concepts and the analysis of multiple Pareto fronts in multi-criteria decision-making using level diagrams. They have addressed the multi-objective design optimization problem of disc brake by considering the friction surfaces as 4 and 6 to obtain Pareto fronts. Then parameter adaptive harmony search algorithm has been customized to suit the multi-objective disc brake problem [172], the weight sum approach has been adopted for fitness computation. In 2017, the multi-objective welded beam design is solved through t-norms, t-co-norms and fuzzy optimization [172]. An Intuitionistic fuzzy optimization technique has been formulated by incorporating the concepts of t-norm and t-co-norm and has proved to be efficient in handling real-world complex optimization problems. Subsequently, a multi-objective ant lion optimization algorithm has been recommended for solving both the welded beam design and disc brake problems [173]. In this technique, the solutions are initially stored in a repository. Then, solutions are chosen based on roulette wheel mechanism according to the coverage of solutions.

Thus a detailed review of the existing attempts to optimize single objective, multi-objective and multi-level thresholding has been investigated.

2.5 SUMMARY

The chapter discussed the related work pertaining to meta-heuristic algorithms and engineering design optimization. Eight extensively used meta-heuristic algorithms namely Particle Swarm Optimization, Ant Colony Optimization, Artificial Bee Colony Optimization, Cuckoo Search, Bat Algorithm, Firefly Algorithm, Genetic Algorithm and Gravitational Search Algorithm have been extensively investigated projecting its variants and applications. Initially, the algorithms were presented in the context of SOO. Then, the various strategies involved in the computation of fitness in MOO were presented. Subsequently, the multi-objective variants of the considered meta-heuristic algorithms were spotlighted. Then, the earlier works pertaining to the optimization of the case studies undertaken in this research were presented. Initially, with respect to SOO, a literature survey on optimization techniques adopted for tension and compression spring design and welded beam design was presented. Then review on optimization of multi-level segmentation techniques was discussed. After that, related works associated with the MOO of the welded beam, design, and disc brake design were detailed.

Chapter 3

PROBLEM STATEMENT AND RESEARCH METHODOLOGY

A detailed literature review related to meta-heuristic techniques for optimization problems has been presented in Chapter 2. After performing the review, the foundations for this research have been arrived at and presented in this chapter.

3.1 RESEARCH MOTIVATION

Even though the literature reports a variety of optimization techniques, it is observed that each type of techniques characterizes some certain merits and demerits. Usually, the choice of parameter values greatly impacts the performance of the algorithms. Only when optimal values are chosen for its parameters, which are not known in prior and are application specific, the optimization techniques perform effectively. The complexity is enhanced when dealing with MOO algorithms.

Many optimization algorithms have their inspiration from biology and nature. Bio-inspired meta-heuristic algorithms are of specific research interest as a result of their vital strength and efficiency. Over the billions of years, biological processes have their own develop capability such that they are endlessly becoming more effective. For the processes that are working in optimal ways, nature is a copious source. Sometimes the algorithms based on these processes are often very effective in the optimization of the objective functions. One of the merits of these algorithms is that they can easily overcome the local optima meritoriously due to the decision exercised in nature.

In order to devise a new optimization technique, the motivation is derived from the 'No Free Lunch Theorem' [20]. It offers both an initiative and an essential design guideline for developing algorithms. According to this theorem, the potential of all search algorithms will be identical if averaged over the entire set of possible objective functions [162]. This implies that if a search algorithm exhibits high efficiency with specific objective functions, then it will be incompetent with the other objective functions. Hence it can conjecture that there can be no algorithm that can demonstrate high efficiency in optimizing all possible objective functions. Therefore, owing to the surge in cardinality of the engineering applications that employ optimization, the necessity for the formulation of new algorithms also increase. This acts as a driving force towards perpetual innovation of novel optimization algorithms. The No Free Lunch Theorem also emphasizes the necessity for developing an optimization algorithm with respect to a particular application domain. According to this theorem, attempting to formulate a procedure that can exhibit superiority in performance in all domains will result in futility. Even though meta-heuristics can be applied to all problems related to optimization, high performance cannot be assured. Therefore, understanding the characteristics of search space for which the algorithm is designed will be greatly useful and supportive while formulating it. On the scenario of having to make a decision on which optimization algorithm to adopt, the algorithm that exhibits the highest potential for the specific application in hand is preferred rather than the ones that demonstrate acceptable performance with a wide range of applications. Thus, the theorem conjectures that designing algorithm based on the application will make it more reachable and successful.

After thoroughly investigating the earlier works, a set of goals are identified as desirable to be achieved by the new algorithm to be proposed. The first goal is that the algorithm should not make any assumption about the search domain. Therefore, algorithms should be designed to make it as general as possible. This means that the fitness values should not be incorporated directly into the algorithm. The second goal is related to efficiency in performance. This means producing results comparable to existing algorithms with a fewer number of iterations. In the case of a few optimization problems, the function evaluation may take a considerable amount of time. Therefore, the algorithm has to be efficient as well as converge faster without getting stuck in local optima. The third aim is to preserve simplicity. From the literature review, it has been found that some algorithms involve complex mathematical computations making them hard to comprehend. Hence, a new algorithm that can be easily adapted by anyone with good, general scientific or mathematical knowledge, rather than just by experts in the meta-heuristic field is sought for. This motivates to formulate a simple algorithm that performs efficiently for varied applications. This also signifies that the number of user-controlled parameters should be kept as minimum as possible. Thirdly for MOO also simpler algorithms that exploit the complete potential of biological aspects need to be developed.

3.2 PROBLEM STATEMENT

Issues such as premature convergence by unsuitable bias, need for careful fine-tuning of parameters, non-generalization, high computational cost and difficult implementations are among the main limitations of early optimization techniques. Moreover, several natural and bio-inspired phenomenon are still unexplored and if they are tapped properly, they may help in solving complex optimization problems.

The stated issues, the unexplored biological phenomena and the ideologies of no free lunch theorem problems have motivated this research to develop new methods towards optimization. The problem definition that is to be explored in this research is as follows:

"Develop bio-inspired algorithms for single and multi-objective optimization problems that overcome all the above-said problems in the existing techniques and test them for diverse applications".

3.3 RESEARCH OBJECTIVES

The performance of the bio-inspired algorithm is determined by its capability to converge to the optimal solution in a limited extent of time. Bearing in mind the above-mentioned issues, the objectives of this research work is to design and develop a simple and efficient optimization technique for SOO problems as well as MOO problems. With the intention to overcome the problem stated in the aforementioned section, the following research objectives are framed and achieved in this thesis.

- **OBJECTIVE 1**

  Design a simple, robust and efficient truly bio-inspired method to handle single objective optimization problems.

- **OBJECTIVE 2**

  Fine-tune the various parameters practice in the proposed algorithm by carrying out a complete analysis of the algorithm.

OBJECTIVE 3

Extend the algorithm for solving multi-objective optimization problems by considering both weighted sum approach and Pareto-based approach.

OBJECTIVE 4

Once the algorithms are designed, apply the algorithms to diverse problems for measuring the performance, efficiency, and simplicity. Ensure that the algorithm performs competitively well against existing bio-inspired algorithms on problems related to optimization.

3.4 RESEARCH CONTRIBUTIONS

The main contributions of the research work to fulfill the stated objectives are illustrated below:

CONTRIBUTIONS 1

The first contribution to fulfill the first objective of this research work is developing a novel bio-inspired algorithm harnessing the chirping behavior of cricket for the SOO problem. This proposed Cricket Chirping Algorithm (CCA) is tested on various test problems and analyzed by using different performance metrics to measure the performance, efficiency, and simplicity.

CONTRIBUTIONS 2

The second contribution of this work is fine-tuning the various parameters of the proposed single objective optimization algorithm by carrying out a complete analysis for enhancing the performance. A statistical analysis using ANOVA is also done.

CONTRIBUTIONS 3

The proposed algorithm is extended to solve MOO problem by designing and developing MOO Algorithm using two approaches.

A. Multi-objective Cricket Chirping Algorithm with Weighted Sum Approach (MOCCA-W)

B. Multi-objective Cricket Chirping Algorithm with Pareto Ranking Approach (MOCCA-P).

CONTRIBUTIONS 4

The proposed algorithm is applied to some real-life problems for validating its performance and efficiency. To test the Cricket Chirping Algorithm for SOO, two problems i.e. Mechanical engineering design optimization problems and the Multi-level Thresholding for Image Segmentation are considered. The developed Multi-objective Cricket Chirping Algorithm (MOCCA) is applied to solve benchmark test problems and the Engineering Design optimization problems.

3.5 SCOPE OF THE RESEARCH

The proposed algorithms are suitable for solving discrete and continuous optimization functions for both SOO problems as well as MOO problems. The CCA for the SOO problem is tested for 2 to 40 dimensions. In the case of MOO problems, the algorithm is limited to two objective functions. It is applied to various applications like mechanical engineering optimization problem for single and multiple objectives and in multi-level Thresholds for image segmentation. The algorithms are designed and implemented in MATLAB 2013b. Several experiments to list the efficiency of the algorithm in terms of correctness of results and the convergence speed are carried out. Also, the error rate is checked. To further substantiate the experimental results, statistical analysis is also done through ANOVA with SPSS package.

3.6 SUMMARY

The research motivations have been thoroughly analyzed and the research objectives have been framed in this chapter. A brief list of the research contributions is also provided along with the scope of the research.

Chapter 4

CRICKET CHIRPING ALGORITHM FOR SINGLE-OBJECTIVE OPTIMIZATION (CCA)

Optimization problems in Science and Engineering are viewed as problems that are difficult to solve in polynomial time. In the literature, several heuristics and meta-heuristic bio-inspired algorithms have evolved as powerful methods for solving these types of problems. Though there are a number of optimization techniques, it is observed that each type of technique retains certain advantages and disadvantages. Though these methods are easy to implement, they usually require some kind of parameter tuning. This makes them difficult to apply directly because there is no prior knowledge of the optimal values of these parameters and sometimes they are often problem-dependent.

This chapter introduces the proposed meta-heuristic algorithm motivated by the chirping behavior of the cricket insect. The chirping characteristics of crickets and their movement for mating and aggression serve the motivation for mapping it to solve optimization problems. In this chapter, a detailed study of cricket's behaviors as noticed in nature and the intuition behind their chirping behavior for function optimization is presented. The proposed algorithm is tested and validated by using standard benchmark mathematical functions and compared with recent meta-heuristics techniques to show the performance of the proposed algorithm.

4.1 CRICKET'S NATURAL CHIRPING BEHAVIOUR

Crickets are insects that somewhat resembles grasshoppers having trampled bodies and long antennae. There are more than 900 species of crickets. Crickets emit a peculiar sound, which is known as chirping. Scientifically, it is referred to as 'stridulation' since the stridulatory organ emits the sound. This is a large vein look like a comb running along the bottom of each wing covered with 'teeth'. Usually, only the male crickets chirp, however some female crickets chirp as well. As the male cricket chirps, he also holds the wings up and opens, so that the wing membranes also act as acoustical sails. The cricket chirp or song is divided into four types based on their chirping behavior [174], [174], [171].

Calling Chirp / Song: The calling chirp is produced for attracting female crickets to mate. This song is fairly loud and this is the song that is most commonly heard during summer nights.

Courting Chirp/Song: The courting chirp sounds more like a scraping noise of low intensity. This chirp is produced when a female cricket is near and a male attempts to mate with a female.

Copulatory Chirp/Song: A copulatory chirp is produced for a brief period after a successful mating.

Aggressive Chirp/Song: An aggressive chirp is triggered by chemoreceptor on the antennae that perceive the near presence of another male cricket. It is a very loud trill and is produced during or after combat with another cricket.

Though the cricket chirping is of different types, generally crickets chirp for two reasons: (1) for mating (2) for aggression. They produce the calling chirp for mating with female crickets and aggressive chirp to fight with other male crickets. It is rumored that crickets can tell the outside temperature. In addition, it was scientifically proved by Dolbear in 1987 and is known as Dolbear's law [16]. According to this law, there is a relation between the cricket's chirping rate and the temperature of the atmosphere. Depending on the species and the temperature of the environment Cricket's chirping rate may be different. Most species chirp at higher rates at a higher temperature.

4.2 MAPPING CRICKET BEHAVIOUR TO PROBLEM-SOLVING

The chirping characteristics of crickets and their mating and aggressive behavior to survive serve the motivation for mapping it to solve optimization problems. This forms the basis of the CCA that is presented in this section. Each cricket is assumed to be a solution in the search space and is characterized by its position in the search space. Out of the total cricket population, few of them as determined by the user is randomly designated as female populations. The male cricket can only chirp and its chirping rate is based on the outside temperature. The male cricket may chirp for mating or aggression. The male crickets move to new positions by emitting a mating song and mate with females producing offspring. Based on their chirping rate at a certain temperature, the velocity of the sound is calculated. The offspring represents a new position of the cricket. By emitting an aggressive song, crickets fight with other male crickets, the best-fit cricket is the winner of cricket and it reaches a new position in the search space. The cricket that has the highest fitness will be selected as the winner of cricket. For simplicity, crickets are assumed to be in two phases: a mating phase when they produce calling chirp and aggression phase when they produce aggressive chirp.

1. **Mating phase:** In this phase, the male cricket chirps for mating. It emits a peculiar sound that attracts the female crickets and other male crickets move away. The male cricket that has the highest chirping rate will attract more female crickets. It is assumed that after mating they produce offspring and move to a new place that means they are taken to new positions in the search space. The attraction is based on the loudness of the chirping sound. Based on the chirping rate the cricket moves to the new position.
2. **Aggression phase:** When the crickets chirp for aggression, they emit an aggressive chirp that other male crickets are warned or called and female crickets will move away. All crickets may not chirp for aggression. For simplicity, a simple representation is used i.e., the probability of chirping for aggression is chosen to be between [0, 1]. When crickets chirp for aggression, it is assumed that they randomly walk to other male crickets and fight. The winner of cricket takes the place of the new solution and removes the loser cricket. The fitness of the male cricket is calculated based on their attractiveness and replace the position of low fit cricket with high fit cricket. The main intention is to use the new and potentially better solutions (cricket) to replace a not so good solution.

The cricket's calling chirp and aggressive chirp in nature are shown in figure 4.1. The relationship between the environmental temperature and the chirping rate of crickets was first calculated by an American physicist and naturalist named Amos Dolbear. He expressed the relationship using a mathematical equation given in 4.1. It provides a way to estimate the temperature T_c in degree Celsius from the number of chirps per minute.

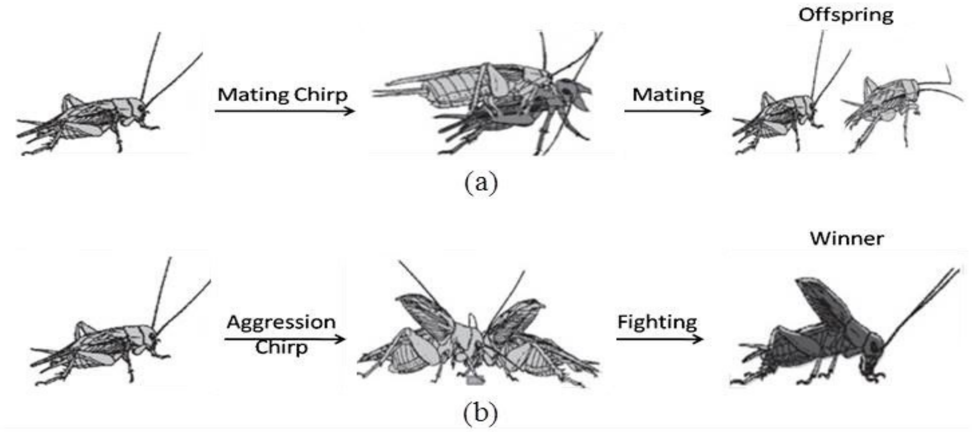


Figure 4.1: Cricket's behavior: (a) Calling Chirp and (b) Aggressive Chirp

$$Tc = 10 + \frac{Nc - 40}{7} \tag{4.1}$$

Or,

$$T_f = 50 + \frac{Nc - 40}{4} \tag{4.2}$$

The chirping rate is derived by using Dolbear's law in a certain temperature T_c or T_f

$$Nc = (Tc - 10) * 7 + 40 \tag{4.3}$$

Chirping rate *(N_c)* is the number of chirps per minutes. The chirping rate represents the frequency of the cricket's chirp. From the frequency, the velocity of each cricket is calculated by using equation 4.4 as follows:

$$v_i = Nc * \lambda \tag{4.4}$$

Here, λ is the wavelength which represents the gap between one chirp to another chirp which is uniformly drawn. From the velocity, the step size (st_i) is calculated by using equation 4.5.

$$st_i = v_i + (x_i - x^*) * \alpha \tag{4.5}$$

Where α=0.01 is a constant value which is used to control the movements of the cricket within a bounded space and x_i is the current position and $x*$ is the best position ever encountered by the cricket. Then the cricket will move to the new position by using the following formula:

$$x_{i+1} = x_i + st_i \tag{4.6}$$

Equations (4.1) to (4.6) are used when the crickets chirp for mating and they change their position according to the chirping rate at a certain temperature.

When crickets chirp for aggression, they have to fight with other crickets. For simplicity, in this design, a probability value named Aggression Rate A_r is considered. Crickets that exceed this probability value are allowed to produce an aggression chirp and move to a new position using the random walk. In the new position, two best positions (crickets) are chosen and a tournament is allowed between them to simulate a fight. The winner is chosen to be the best cricket (position or solution). The flowchart of the algorithm is shown in figure 4.2 and the proposed algorithm is given in table 4.1.

4.3 EXPERIMENTAL RESULTS AND ANALYSIS

The proposed CCA is implemented for various benchmark mathematical functions. In order to analyze the performance of CCA, a full experiment is performed for the proposed algorithm for ten benchmark test functions. The benchmark functions provide a balance between multimodal functions with many local minima and functions with only a few local minima as well as easy and difficult functions.

4.3.1 BENCHMARK TEST FUNCTIONS

There are many benchmark functions to test the performance of the optimization techniques. There is a need for validating and testing any new optimization technique against the benchmark functions. The benchmark test functions that are considered for this experiment are summarized in table 4.2 with the function name, formula, range, variables and their global optimal value.

4.3.2 EXPERIMENTAL RESULTS

In table 4.3, the mean of iterations and time (in seconds) for execution of different benchmark test functions are calculated for varying dimensions like 2, 10, 20, 30 and population size 20 and 40. The algorithm for each function was executed 50 times and the mean of iterations and time is calculated. The aggression rate (probability for aggression A_r) varies from 0 to 1. The implementation is done in Windows 7 operating system computer with Intel(R) Core (TM) i5 processor and 4GB RAM.

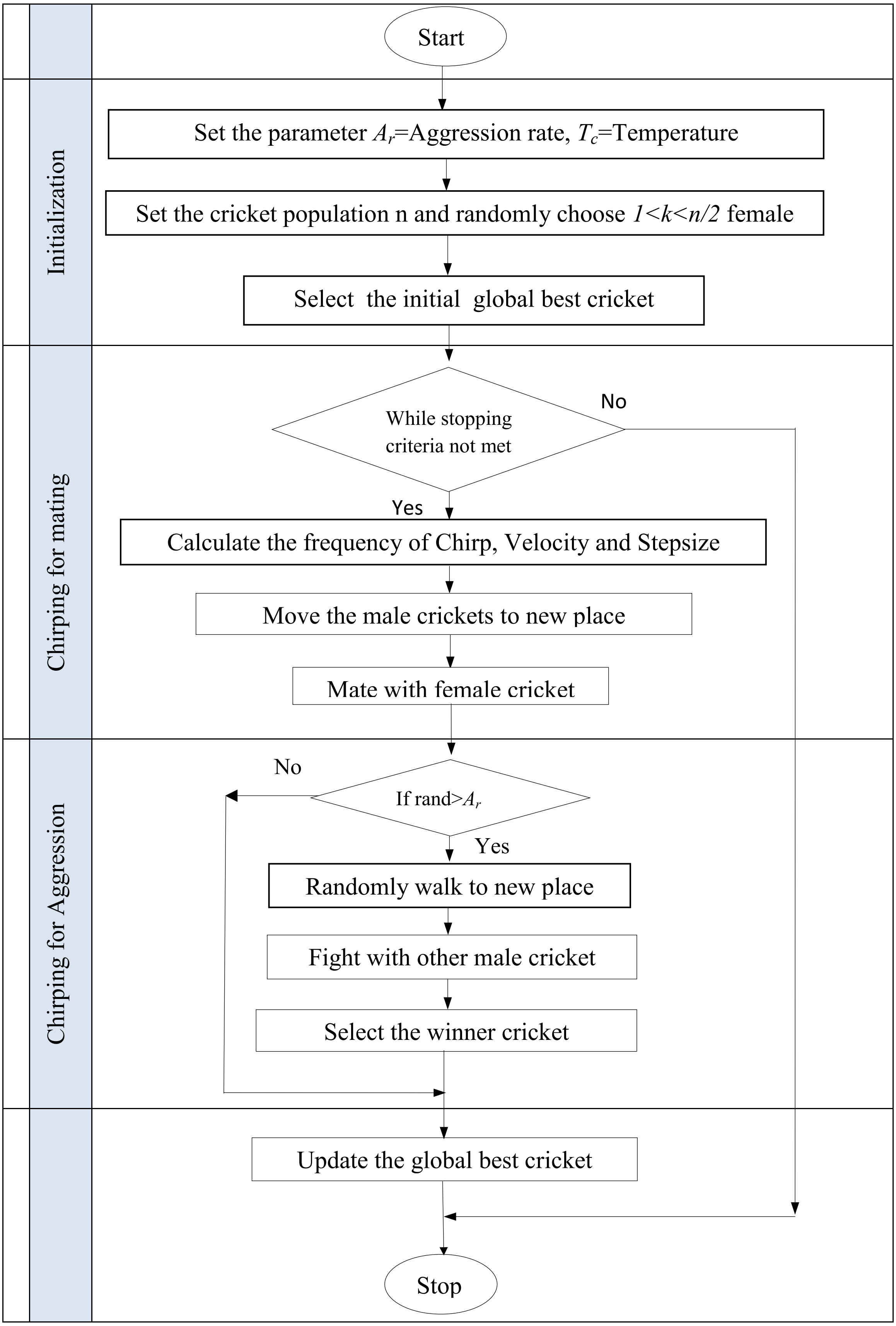

Start
Initialization
Set the parameter A_r=Aggression rate, T_c=Temperature
Set the cricket population n and randomly choose $1<k<n/2$ female
Select the initial global best cricket
Chirping for mating
While stopping criteria not met
No
Yes
Calculate the frequency of Chirp, Velocity and Stepsize
Move the male crickets to new place
Mate with female cricket
Chirping for Aggression
If rand>A_r
No
Yes
Randomly walk to new place
Fight with other male cricket
Select the winner cricket
Update the global best cricket
Stop

Figure 4.2: Flowchart of CCA

Table 4.1: Pseudocode for Cricket Chirping Algorithm (CCA)

```
Algorithm_CCA( )
Begin
Input: $f_c(x)$: Objective function; $n$ : Number of crickets; $T_c$: Temperature; $A_r$: Aggression rate; $k$: No of female crickets, $1<k<n/2$
  1. Randomly Initialize the cricket's position
  2. Randomly choose  $k$ crickets as female crickets
  3. Calculate the fitness of each cricket
  4. Assign best cricket $f_{best\_cricket} \leftarrow$value of the best fit cricket, $P_{best\_cricket} \leftarrow$ position of the best cricket
  5. Set $g_{best\_cricket}$ as the current $f_{best\_cricket}$ //in the initial generation $g_{best\_cricket} = f_{best\_cricket}$//.
  6. While (stopping criteria not met)
     a. Allow male crickets to chirp for mating //Call procedure calling_chirp()//.
     b. Allow male crickets to mate with female crickets //Call procedure mating()//.
     c. With probability $A_r$, allow the male crickets to chirp for aggression //Call procedure aggression_chirp() //.
     d. Compute the fitness of the crickets produced by mating and aggression in the new positions.
     e. Select $f_{best\_cricket}$, from the new positions of the cricket.
     f. If $f_{best\_cricket} > g_{best\_cricket}$, then update $g_{best\_cricket}$ with the current $f_{best\_cricket}$.
7. End while
8. Return the global best cricket at termination.
End
```

```
Procedure calling_chirp( )
  Begin
    for every male cricket,
      1. Calculate the frequency of Chirping  and velocity using equation (4.3) and (4.4)
      2. Calculate the step size using equation (4.5)
      3.  Move each cricket to the new position using equation (4.6)
      4. Return crickets in new position
  End
```

```
Procedure mating( )
// This procedure simulates the mating behavior of the cricket. //
Begin
    1. For every male cricket $M_i$ in their new position, randomly choose a female cricket $F_i$
    2. Randomly choose a cut point in both $M_i$ and $F_i$
    3. Exchange the genetic materials of both $Mi$ and $F_i$ with reference to their cut points
       to produce two new offspring      // Similar to crossover in GA//.
    4. Return the two offspring and the parents as the new cricket positions.
    5. the end for
    6. return
End
```

```
Procedure aggr_chirp()
Begin
    1. if rand>$A_r$
          randomly walk to the new position.
    2. Fight with other male crickets.
    3. Return the winner cricket (position).
End
```

4.3.3 COMPARISON WITH OTHER BIO-INSPIRED ALGORITHMS

The CCA is compared with other popular optimization algorithms like GA, PSO, ABC, BA, and CS for various standard test functions such as Ackley, Easom, Griewank, Matyas, Michalewicz, Ratrigin, Rosenbrock, Schwefel, Shubert, and Sphere. In GA, the standard version with no elitism and population size 1000 and mutation probability of $p_m = 0.05$ and crossover probability of 0.8 is used. For PSO, the standard version is used with learning parameters $\alpha = 2$ and the inertia function $I = 1$. The number of employed and onlooker bees is fixed to 50% of population and scout bee to be one in ABC algorithm. In BA algorithm the loudness and pulse are set to 0.5 and the minimum and maximum frequency is taken as 0 and 2. In CS algorithm, the discovery rate of the alien egg is taken as 0.25.

Table 4.2: The benchmark test functions with their global optimal value

Function Name	Functions	Range	$(\vec{x})$	f_{min}
Ackley	$f(x) = 20 + e - 20\exp\left[-2.0\sqrt{\frac{1}{n\sum_{i=1}^{n} x_i^2}}\right] - \exp\left[1/n\sum_{i=1}^{n}\cos(2\pi x_i)\right]$	$-35 \le x_i \le 35$	$(0,\cdots0)$	0
Easom	$f(x) = -cosx_1 cosx_2 exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$-100 \le x_i \le 100$	$(\pi, ..., \pi)$	-1
Griewank	$f(x) = \sum_{i=1}^{d} \frac{x^2}{4000} - \prod_{i=1}^{n} cos(x_i/\sqrt{i}) + 1$	$-600 \le x_i \le 600$	$(0, ..., 0)$	0
Matyas	$f(x) = 0.26(x_1{}^2 + x_2{}^2) - 0.48x_1x_2$	$-10 \le x_i \le 10$	$(0, ..., 0)$	0
Michaelwicz	$f(x) = -\sum_{i=1}^{d} \sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}, m = 10$	$0 \le x_i \le \pi$	(2.2029,1.5708)	$-1.8013(d=2)$, -9.66015 (d=10)
Rastrigin	$f(x) = 10d + \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i)$	$-5.12 \le x_i \le 5.12$	$(0, ..., 0)$	0
Rosenbrock	$f(x) = \sum_{i=1}^{d-i}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$	$-5 \le x_i \le 10$	$(1, ..., 1)$	0
Schwefel	$f(x) = 418.9829 * d - \sum_{i=1}^{d}\left[x_i sin\left(\sqrt{\lvert x_i \rvert}\right)\right]$	$-500 \le x_i \le 500$	(420.9687, . . . ,420.9687)	0

Table 4.3: The mean of iterations and time taken for different benchmark functions to find global optimal values using CCA

Function	Optimal Value	Mean of	Population=20				Population =40			
			d=2	d=10	d=20	d=30	d=2	d=10	d=20	d=30
Ackley	0	Iteration	6429.2	36018	71908.4	121598.4	13802.4	75523.2	149140	241703.2
		Time	0.00662	0.9026418	1.877198	3.36913	0.303257	1.77108	3.760026	6.374346
Easom	-1	Iteration	4617.6	4638	7215.2	7143.6	15725.6	15899.2	160669.6	160920.8
		Time	0.0808	0.0896	0.149831	0.162417	0.253321	0.28542	0.311915	0.334619
Griewank	0	Iteration	4158.84	29597.82	54473.58	73849.44	8839.6	64642.24	118506.4	157578.6
		Time	0.092566	0.690974	1.39348	2.044292	0.18909	1.472306	2.964198	4.286906
Matyas	0	Iteration	64686	64698	64846	64788	139174	137662	139105.6	137728
		Time	1.10283	1.2239	1.39003	1.50665	2.27423	2.5390	2.7186	2.9097
Rastrigin	0	Iteration	3574.4	21406.56	42750.54	59855.51	7268.48	41734.4	81720	134395.54
		Time	0.2467	1.3050	2.83864	4.28271	0.40299	3.8618	5.4316	10.942278
Rosenbrock	0	Iteration	7025	10208.94	8573.69	8707.2	15350.4	15424.2	15568.52	15204.44
		Time	0.312906	0.5451	0.52933	0.531578	0.6304452	0.712776	0.7785224	0.815933
Shubert	-186.7309	Iteration	7638.54	7974.96	8029.14	8429.82	12800.2	11944.12	12200.78	12468.1
		Time	0.140948	0.151487	0.164864	0.176701	0.210975	0.215981	0.243043	0.25916
Sphere	0	Iteration	70531	354550	666230	955790	148900	710400	1329300	1990400
		Time	3.6360	19.5958	37.3821	56.4690	2.6309	13.93222	26.7191	42.0729

Each algorithm is executed 100 times to carry out meaningful statistical analysis. The algorithm stops when the variations of function values are less than a given tolerance $\tau \leq 10^{-5}$. The results are summarized in table 4.4. It shows the number of function evaluation in the form of an average number (mean) of function evaluations± Standard Deviation (SD) from the success rate of finding the global optima, i.e. mean ± SD (success rate). For example, 670±152 (100%) indicates that the mean of iterations required to coverage is 670 with a standard deviation of 152 and the success rate of finding the global optima for this algorithm is 100%.

Again, the quality of the solution is measured by the Average Error (AE) and Standard Deviation (SD) of 50 independent runs. The AE is computed using equation (4.7) which is obtained from [15]

$$AE = \frac{\sum_{j=1}^{50} \left| \overrightarrow{f(x_{best}^{j})} - f_{min} \right|}{50} \tag{4.7}$$

Where, $\overrightarrow{x_{best}^{j}}$ is the final solution vector corresponding to the j^{th} run and $\overrightarrow{\mathrm{f}(x_{best}^{j})}$ is the value of the benchmark problem corresponding to the final solution vector. The true optimum of a particular benchmark problem is given by f_{min}.

In table 4.5 the AE and SD of GA, PSO, ABC, BA, CS, and CCA for ten benchmark test functions are shown. From the results, it is shown that compared to the other methods CCA has a less average error.

Figures 4.3(a)-4.3(j) show the iterative results for each benchmark problem that is considered for comparison. In every generation, CCA is converging to the optimal value. The graph shows the fitness values obtained from the iteration number for the different optimization algorithm. CCA is converging faster compared to its counterparts. So the convergence of the CCA is higher compared to other algorithms.

Table 4.4: Comparison of Mean , SD and success rate to find optimal value amon GA, PSO, ABC, BA, CS and CCA

Functions	**GA**	**PSO**	**ABC**	**BA**	**CS**	**CCA**
Ackley	12720 ±3127(93%)	10417±2325(95%)	9765±1285(100%)	6711 ±2693 (100%)	4062±413(100%)	**2475±194(100%)**
Easom	14249 ±2397(92%)	15278 ±989(90%)	10204±2011(99%)	8673 ±952(99%)	6633±606(100%)	**4890±461(100%)**
Griewangk	67945 ±8752(90%)	65770±3293(92%)	29655±23407(100%)	10675 ±6459(100%)	6858±2132(100%)	**4227±259(100%)**
Matyas	3701±765(100%)	5628±5210(100%)	3855 ±1974(100%)	630±248(100%)	790±198(100%)	**670±152(100%)**
Michalewicz	7325 ±3614(95%)	7026 ±897(98%)	5181±1120(100%)	3952 ±893(100%)	1701±335(100%)	**1364±182(100%)**
Rastrigin	93573 ±2699(80%)	8058 ±2557(90%)	7777±2593(100%)	11563 ±3782(100%)	3137±593(100%)	**1286±168(100%)**
Rosenbrock	26723 ±6901(90%)	30796 ±6825(95%)	23597±18221(95%)	8923 ±6493(100%)	4846±1589(100%)	**1882±455(100%)**
Schwefel	17629 ±6172(95%)	16550 ±1275(98%)	10722±833(100%)	8929±729(99%)	4596±945(100%)	**1705±195(100%)**
Shubert	65077 ±3987(90%)	20521±1250(95%)	1980±471(100%)	9995 ±5642(100%)	5629±1483(100%)	**2855±957(100%)**
Sphere	20572 ±1307(100%)	9840 ±2423(100%)	4783±955(100%)	1973 ±270(100%)	1599±236(100%)	**804±117(100%)**

Table 4.5: Comparison of Mean and Standard Deviation of Error rate of the benchmark test functions among GA, PSO, ABC, BA, CS and CCA

Algor-	**Mean**	**Functions**									
ithm	**& std**	**Ackley**	**Easom**	**Griewank**	**Matyas**	**Michalewics**	**Restrigin**	**Rosenbrock**	**Schwefel**	**Shubert**	**Sphere**
GA	AE	1.37E+00	3.77E+00	7.678E-03	2.35E-05	5.24E-01	1.396E+00	7.468E-03	1.722E-01	5.11E+04	1.35E-04
	SD	6.420E-01	2.987E-01	4.1207E-03	3.637E-05	1.33E-02	3.2124E-01	4.162E+00	5.381E-02	1.567E+03	3.20E-05
PSO	AE	1.675E-01	9.72E-01	7.10E-04	1.23E-04	2.26E-01	7.96E-06	2.68E-02	9.58E+01	1.01E+01	2.06E-03
	SD	1.22E-01	5.45E-04	8.78E-04	1.41E-04	2.30E-01	3.83E-05	3.60E-02	1.18E+02	1.37E+01	2.68E-03
ABC	AE	3.2276E-004	9.93E-01	1.6132E-04	9.28E-07	0	4.4139E-13	5.5050E-04	9.39E+01	2.99E-01	6.27E-18
	SD	1.40E-03	2.84E-03	1.00E-03	1.93774E-06	0	3.1180E-12	9.45E-02	7.51E+01	1.16E-01	6.2305E-18
BA	AE	1.6250E-005	0.4000	0.0118	3.8588E-012	-0.1678	6.1702E-09	5.0191E-10	7.51E+03	-6.0309E-6	5.7198E-11
	SD	8.32455E-06	0.4949	0.011653	3.58031E-12	0.0258	5.5071E-09	5.5556E-10	1.351E+03	0	2.4495E-10
CS	AE	8.8818E-016	0	1.5169E-04	3.7828E-127	0	0	0	1.43E-02	6.2885E-5	0
	SD	0	0	8.9350E-04	1.8820E-126	0	0	0	7.77E-02	0	0
CCA	**AE**	**1.8212E-016**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	SD	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

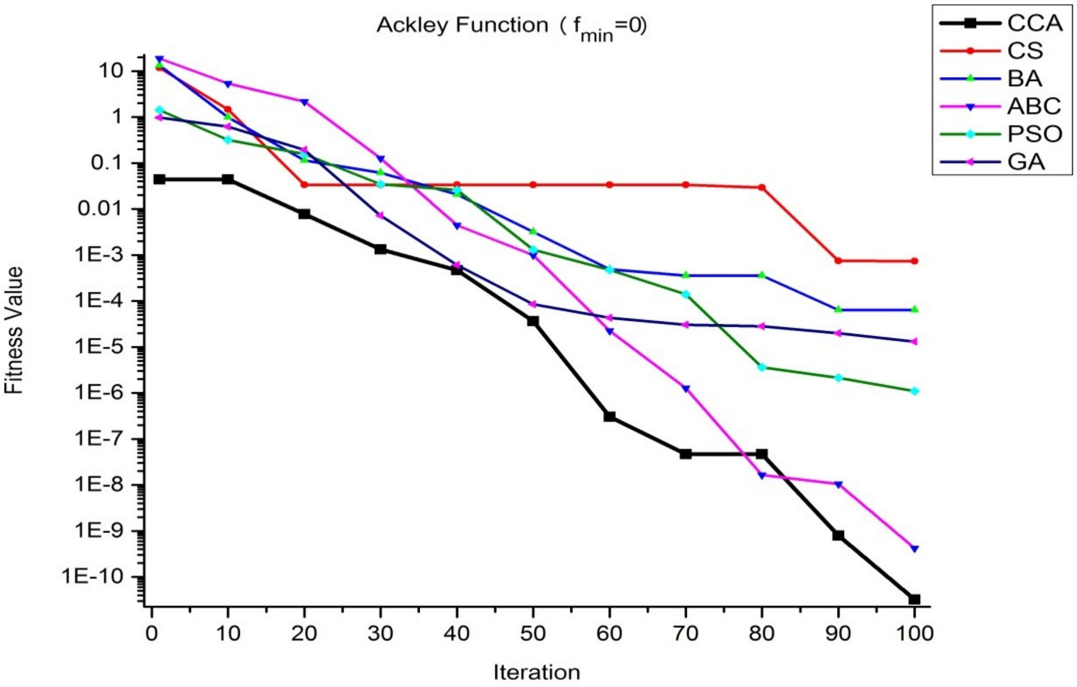


Figure 4.3 (a): Comparison of the convergence in 100 iterations of Ackley function

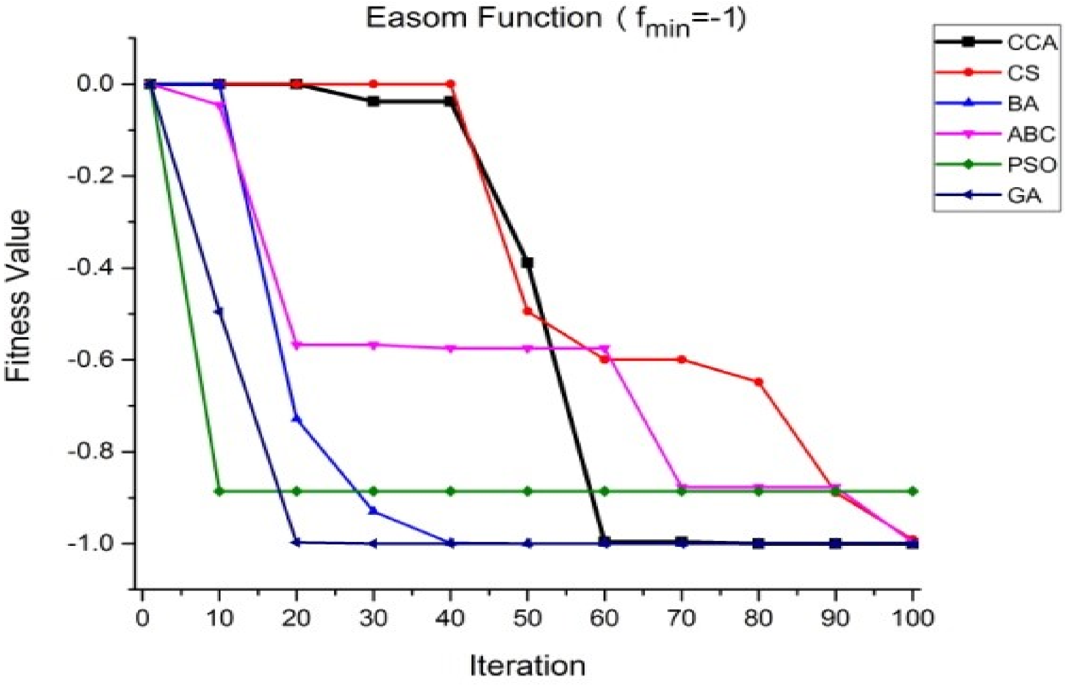


Figure 4.3 (b): Comparison of the convergence in 100 iterations of Easom function

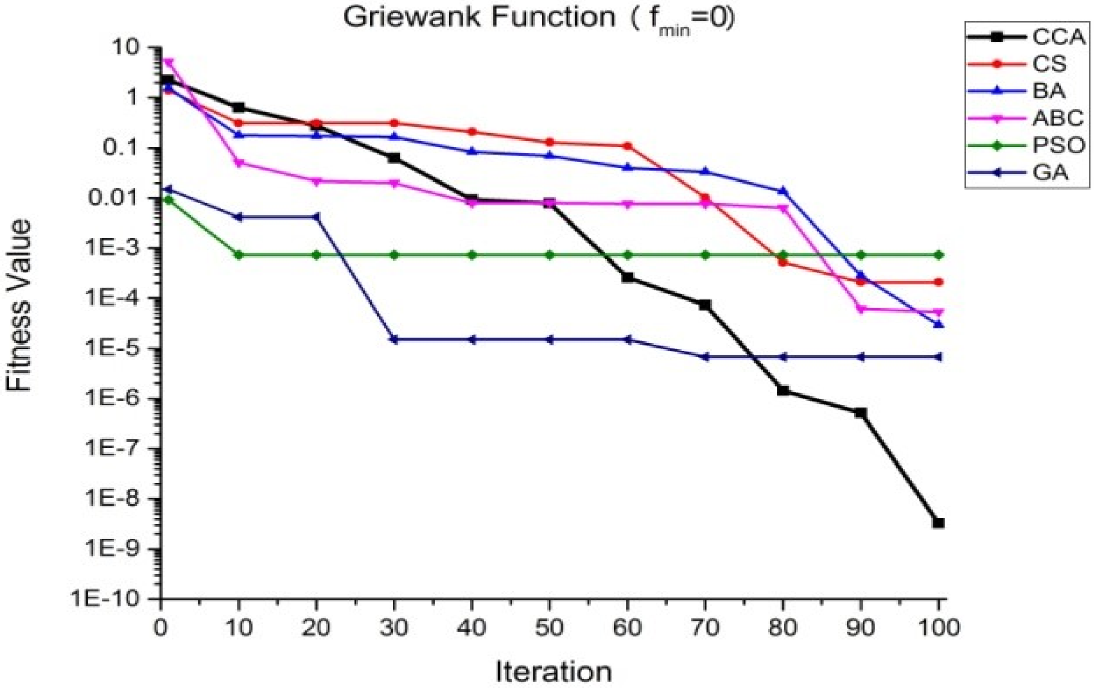


Figure 4.3 (c): Comparison of the convergence in 100 iterations of Griewank function

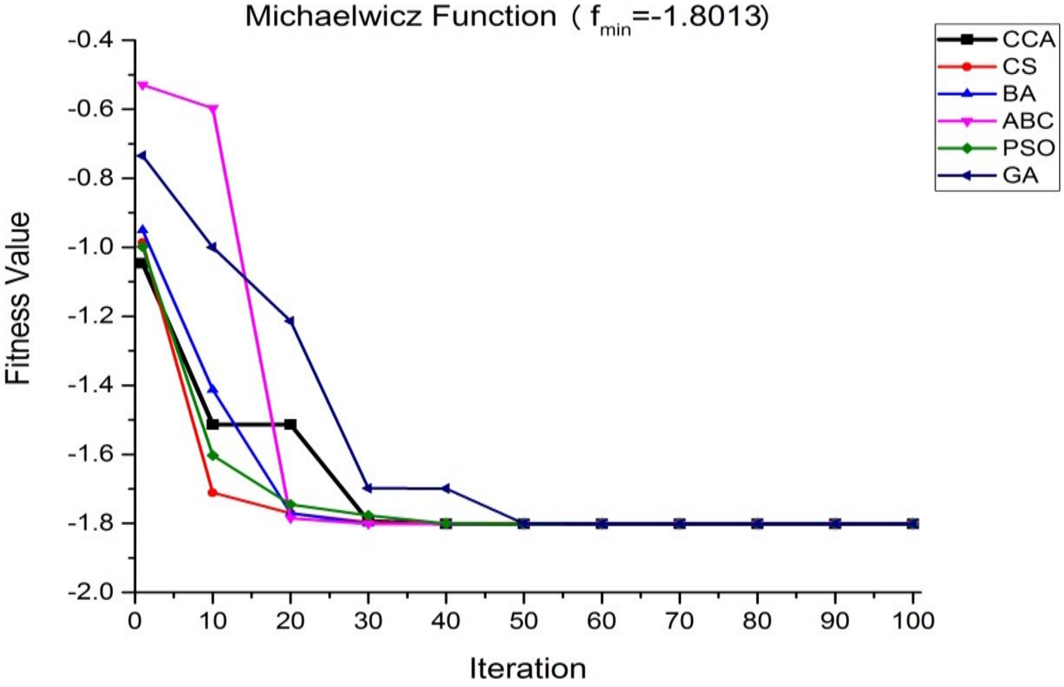


Figure 4.3 (d): Comparison of the convergence in 100 iterations of Michaelwicz function

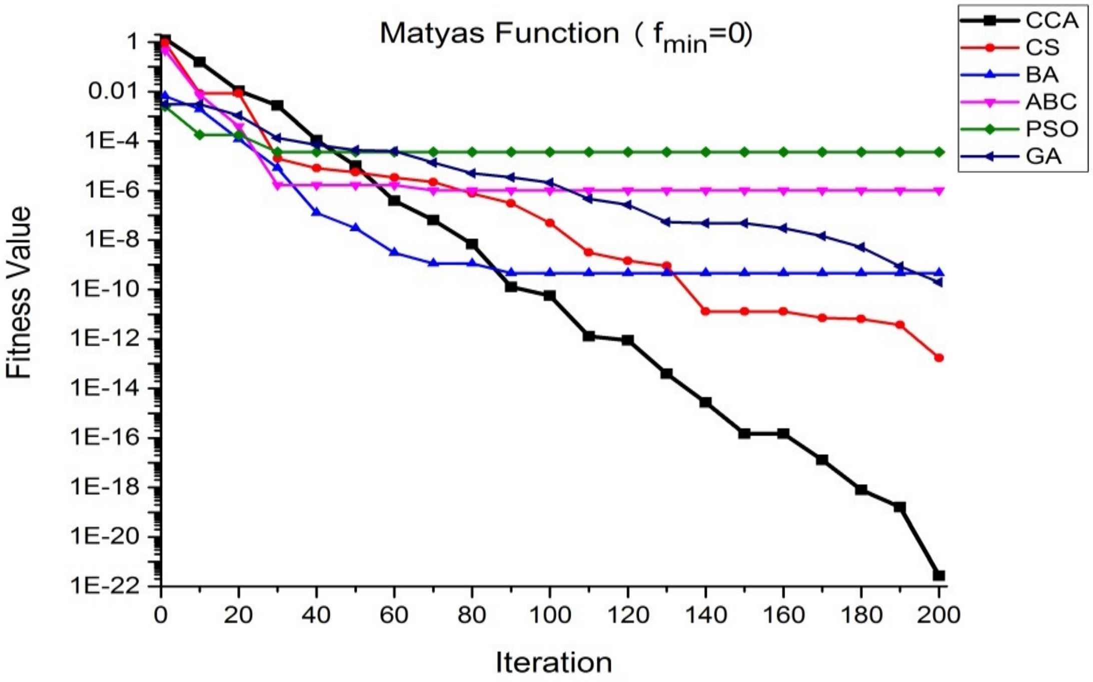


Figure 4.3 (e): Comparison of the convergence in 100 iterations of Matyas function

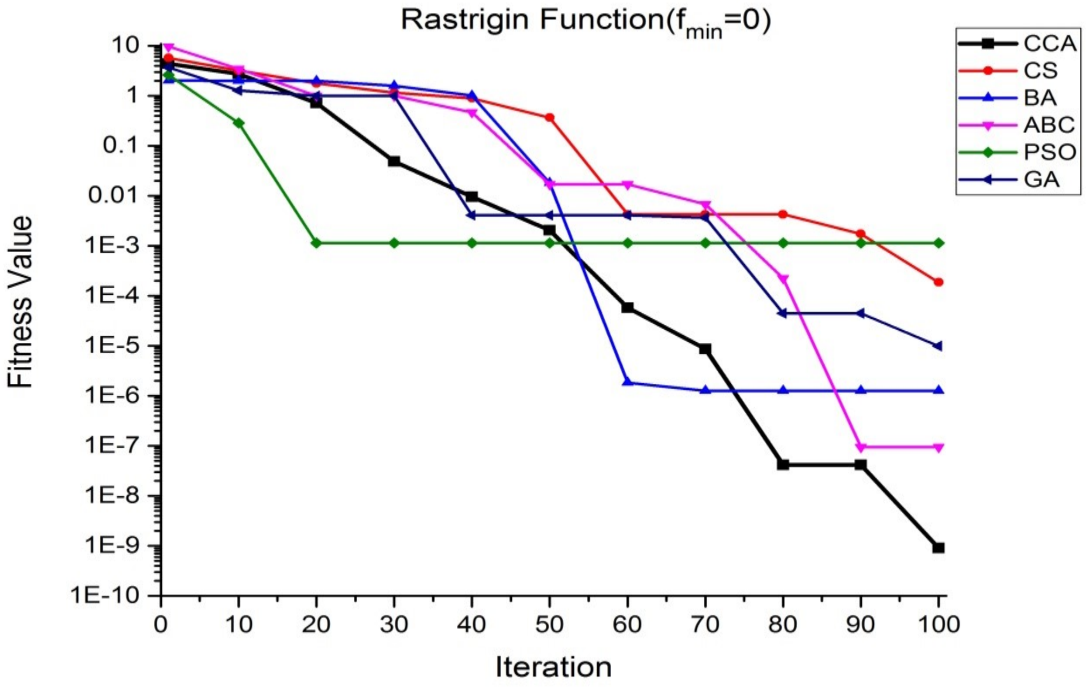


Figure 4.3 (f): Comparison of the convergence in 100 iterations of Rastrigin function

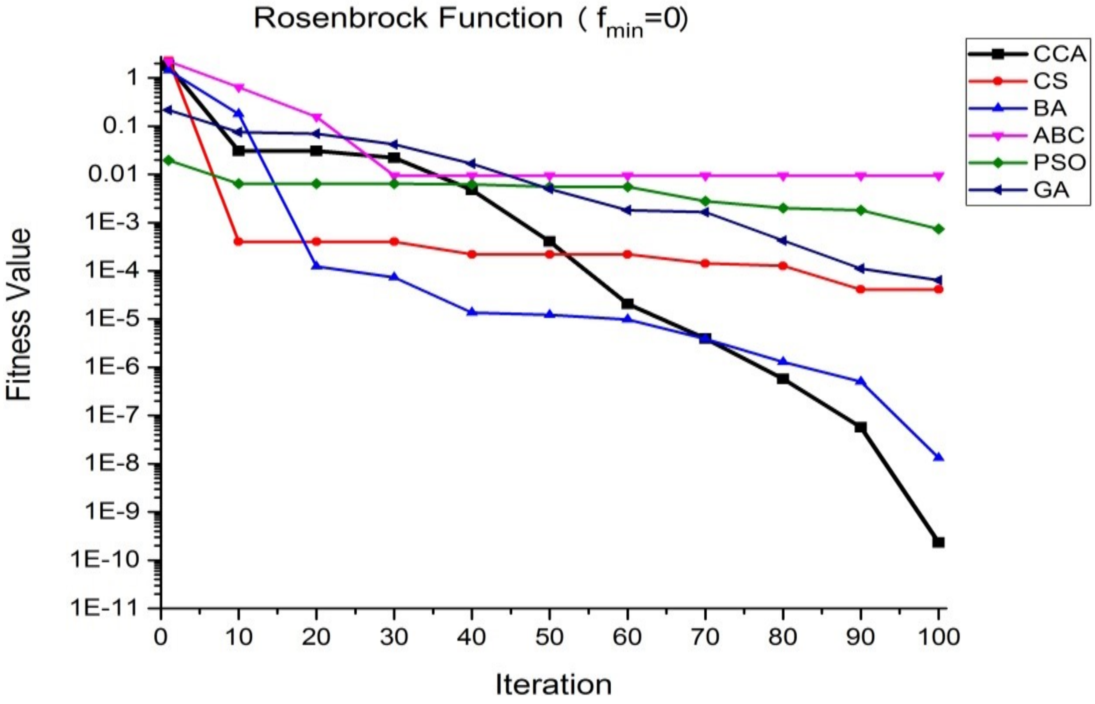


Figure 4.3 (g): Comparison of the convergence in 100 iterations of Rosenbrock function

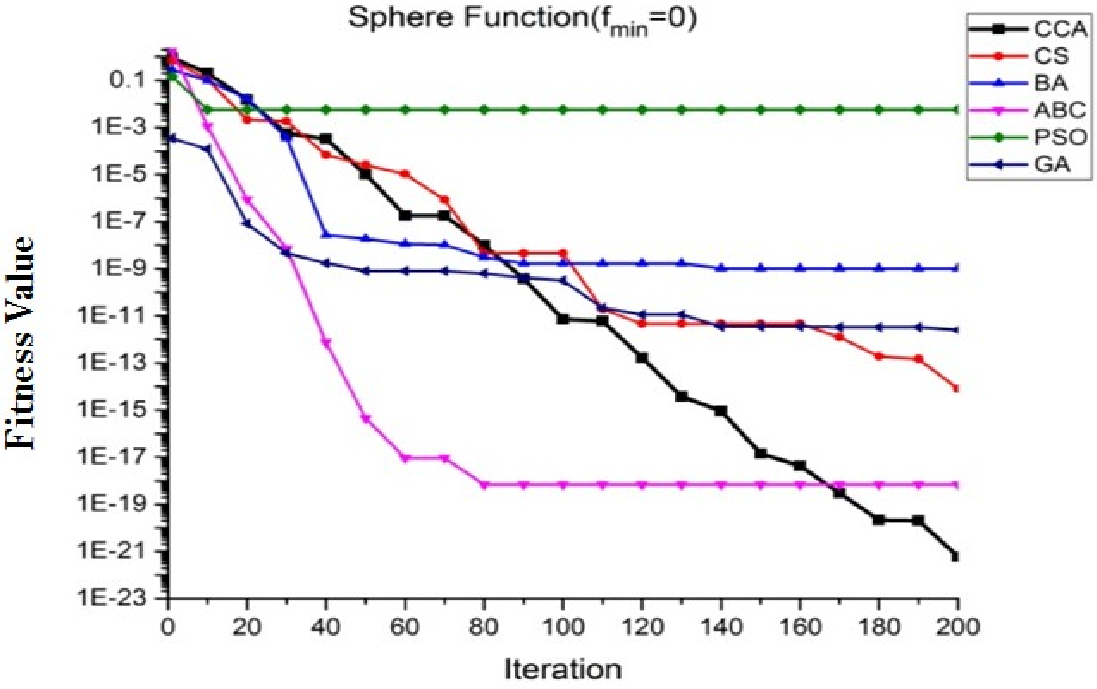


Figure 4.3 (h): Comparison of the convergence in 100 iterations of Sphere function

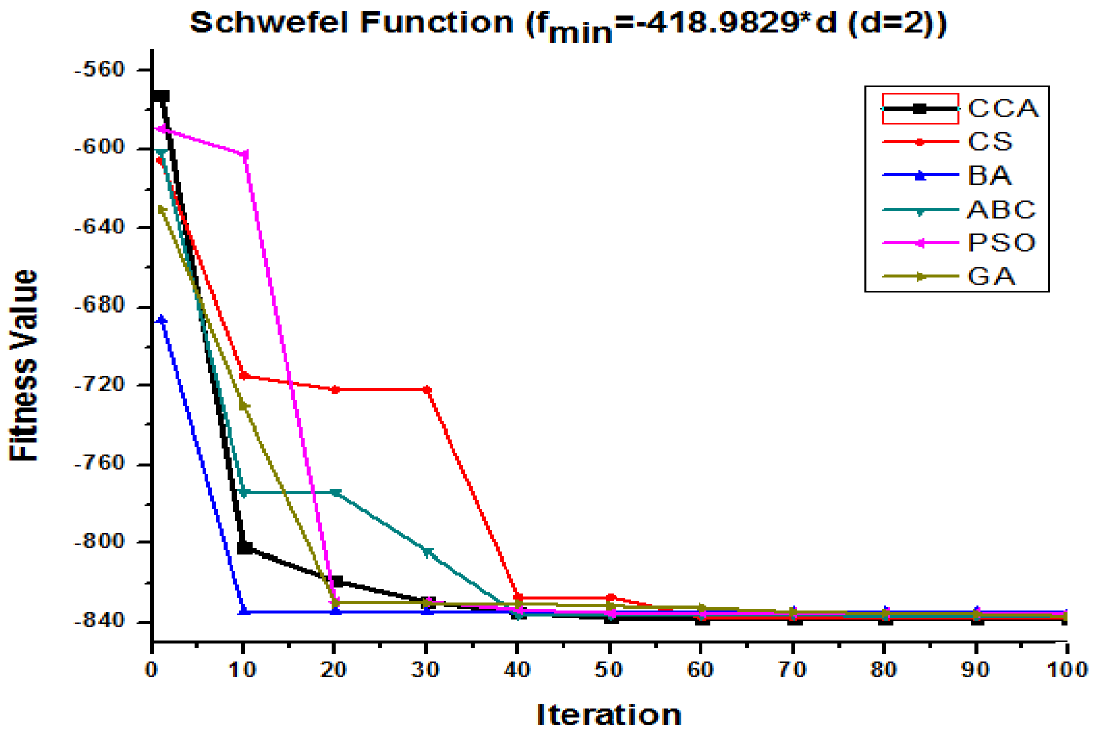


Figure 4.3 (i): Comparison of the convergence in 100 iterations of Schwefel function

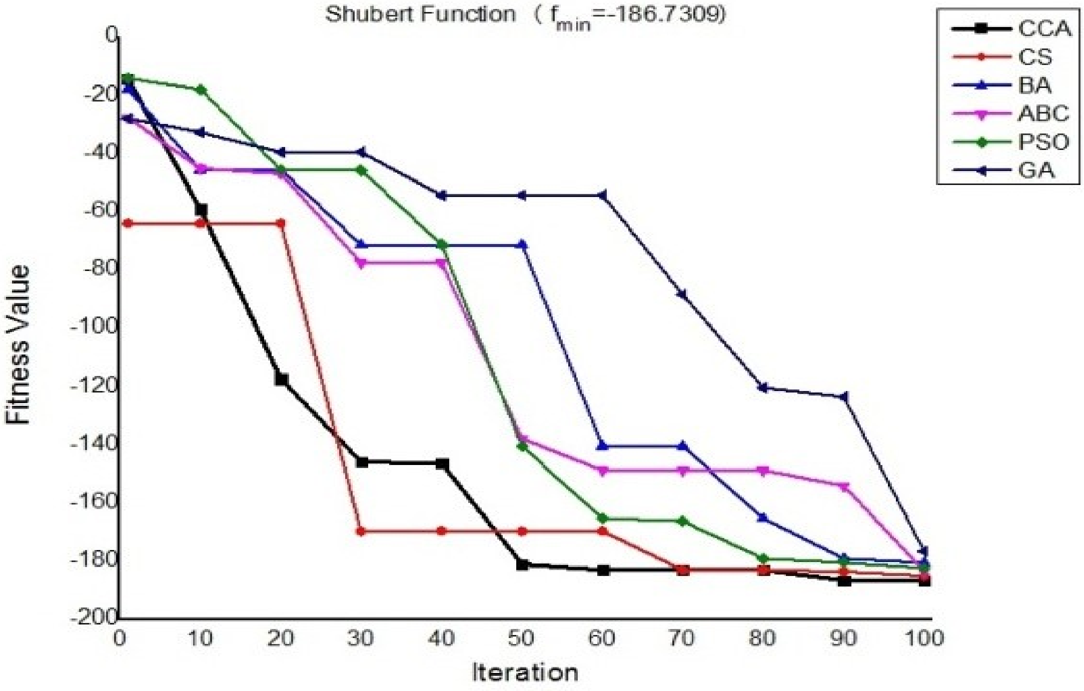


Figure 4.3 (j): Comparison of the convergence in 100 iterations of Shubert function

4.3.4 STATISTICAL ANALYSIS

To test the significance of the results produced by CCA, a statistical analysis using ANOVA has been carried out on the number of iterations taken by various algorithm to converge that is shown in table 4.4. The analysis is performed by considering a 5% significance level over the number of iterations to find the optimal value corresponding to the test functions for six different methods such as GA, PSO, ABC, BA, CS, and CCA. In this analysis the hypothesis is set as follows:

> ***Null hypothesis H_0:*** *There is no significant difference in the number of iterations among the methods GA, PSO, ABC, BA, CS, and CCA.*
>
> ***Alternative hypothesis $H_{1:}$*** *There is a significant difference in the number of iterations among the methods GA, PSO, ABC, BA, CS, and CCA.*

The ANOVA Test is conducted using SPSS tool and the results found in the experiment are shown in table 4.6. If the p-value is less than 0.05, the null hypothesis is rejected. From the ANOVA test shown in table 4.6, the p values (Sig.=0.000) is less than 0.05 (5% significance level). So the null hypothesis is rejected. It is concluded that there is a significant difference in the number of iteration among the methods GA, PSO, ABC, BA, CS, and CCA.

ANOVA

Iteration

	Sum of Squares	**Df**	**Mean Square**	**F**	**Sig.**
Between Groups	6695479700.400	5	1339095940.080	5.790	.000
Within Groups	12488919554.600	54	231276288.048		
Total	19184399255.000	59			

4.4 SUMMARY

In this chapter, the meta-heuristic algorithm, namely Cricket Chirping Algorithm (CCA) has been proposed and implemented. The CCA is inspired by the chirping behavior of crickets, i.e. when they chirp for mating and aggression. The proposed algorithm has been validated and compared with some of the popular algorithms. It generates better solutions as compared to its counterparts and it is also concluded that the CCA performs well both in low and high dimensional problems. A set of benchmark functions have been used to test the CCA in comparison with GA, PSO, ABC, BA, and CS for both lower dimension and higher dimension problems. Experimental results prove the robustness and accuracy of CCA over other search-based approaches, and in every generation, CCA improves its fitness value. The performance of CCA is analyzed by using one way ANOVA test.

Chapter 5

IMPACT OF PARAMETER TUNING ON THE CRICKET CHIRPING ALGORITHM

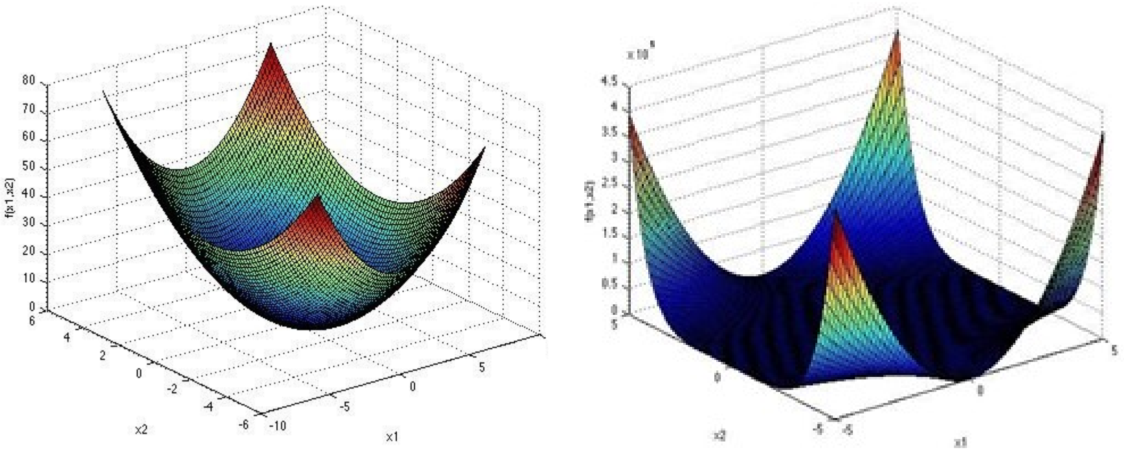
Most of the man-made technologies are nature-inspired including the popular meta-heuristics techniques that solve complex computational optimization problems. Though these methods are easy to implement, they usually require some kind of parameter tuning. This makes them difficult to apply directly because the optimal values of these parameters cannot be recognized earlier and they are often problem-dependent. In most of the metaheuristics algorithms, adjusting the parameters has important significance in obtaining the best performance of the algorithm. While solving the problem, the parameter controlling the algorithm has a potential to improve the efficiency of the algorithm. Cricket Chirping Algorithm (CCA) employs a set of parameters for its smooth functioning. In this chapter the different parameters used in CCA are tuned for better performance of the algorithm and the impact of tuning is experimented and analyzed on a set of sample benchmark test functions, then fine-tuned CCA is compared with other popular meta-heuristic algorithms.

5.1 PARAMETER TUNING IN CCA

When meta-heuristics search algorithms are used to solve a particular problem, we need some techniques to map the original problem context with the problem-solving framework like the specification of the representation and evaluation of the fitness function. Since there is not much knowledge about the effects of parameters on the algorithm performance, determining the best parameter value is a tough and challenging task. In most of the metaheuristics algorithms, manual parameter tuning is a common practice. Generally, one parameter is tuned at a time and repeated for simultaneous tuning of more parameters. However, it leads to a huge amount of experiments and may cause some sub-optimal choices. Obtaining a near optimal or an optimal solution of an algorithm depends on the parameter values. Therefore, zeroing suitable parameter values is important, even if the process requires a lot of added resources. Parameter setting can be done in two methods: parameter tuning (before the run) and parameter control (during the run). In parameter tuning, parameter values are defined first and do not change during the execution of the algorithm. In parameter control, parameter values are changed along with the algorithm run and can be deterministic, adaptive or self-adaptive.

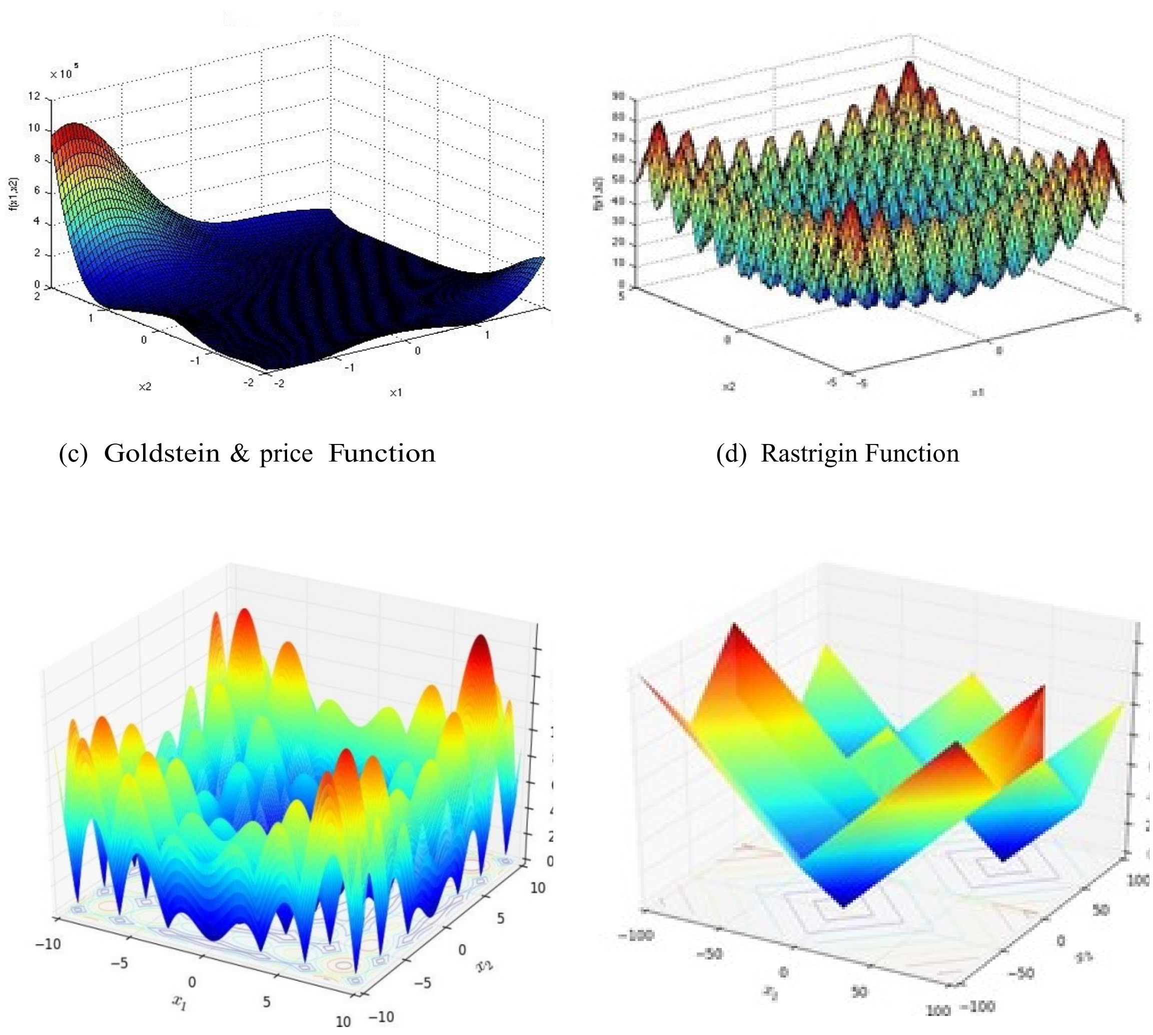
Adjustment of different parameters of the meta-heuristics algorithm is usually a time-consuming task which is mostly done by trial and error method. Here, the performance of CCA is tested with different values of the CCA parameters like temperature, aggression rate, crossover rate and female selection.

Generally, meta-heuristic algorithms with several parameters have to be fine-tuned. The parameters of CCA are Temperature (T_c), Aggression rate (A_r), Crossover rate (C_r) and Female selection (F_s). This section analyses the impact of these parameters on the performance of CCA. The values of each parameter are varied by keeping other parameter fixed. To fix the parameters for CCA the performance of different parameter values are studied on the benchmark mathematical function, namely Alpine, Beale, Goldstein and Price, Rastrigin, Sphere, and Tripod function. A brief description of these functions is given in the next sections and the 3D view for representation of each of these functions is shown in figure 5.1.



(a) Sphere Function

(b) Beale Function



(c) Goldstein & price Function

(d) Rastrigin Function

(e) Alpine 1 Function

(f) Tripod Function

Figure 5.1(a)-(f): Two-dimensional graph representation of the test functions

5.1.1 TEST FUNCTIONS

- **ALPINE1 FUNCTIONS**

This is a multimodal minimization problem defined as follows:

$$f(x)=\sum_{i=1}^{d}\left|x_i\sin(x_i)+0.1x_i\right| \tag{5.1}$$

Here, d represents the number of dimensions and $x_i \epsilon[-10, 10]$ for $i = 1, ., d$. The global optimum is $f_{min}= 0$ where $x_i = 0$ and $i = 1, ., n$.

- **BEALE FUNCTION (HEDAR, N.D.)**

It is a continuous, non-separable, non-scalable, differentiable, multimodal function. The function is defined as follows:

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2 \qquad (5.2)$$

The global minimum is $f_{min} = 0$, at (3, 0.5) for $x_i \in [-4.5, 4.5]$, for all i = 1, 2.

- **GOLDSTEIN & PRICE FUNCTION**

The function Goldstein & Price returns the value:

$$\begin{aligned} f(x) = &\left[1 + (x_0 + x_1 + 1)^2 (19 - 14x_0 + 3x_0^2 - 14x_1 + 6x_0x_1 + 3x_1^2)\right] \times \\ &\left[30 + (2x_0 - 3x_1)^2 (18 - 32x_0 + 12x_0^2 + 48x_1 - 36x_0x_1 + 27x_1^2)\right] \end{aligned} \qquad (5.3)$$

With domain $-2\ |x_i| \leq 2$ and the global minimum $f_{min} = 3$ at the point (0,−1).

- **RASTRIGIN FUNCTION**

The Rastrigin function is a highly multimodal but the locations of the minima are distributed regularly. It has several local minima. It is shown in the figure in its two-dimensional form. The function is given below:

$$f(x) = 10n + \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i) \qquad (5.4)$$

The range is $-5.12 \leq x_i \leq 5.12$ and global minimum $f_{min}=0$ at the point (0,.......,0).

- **SPHERE FUNCTION**

It is a continuous, convex and unimodal function. This function has d local minima except for the global one. The Sphere function can be formulated as shown below:

$$f(x) = \sum_{i=1}^{d} x_i^2 \qquad (5.5)$$

Where, $x_i \in [-5.12,\ 5.12]$ for all $i = 1, . , d$. The Global Minimum $f\ (x^* = 0$, at $x^* = (0,..,0)$.

- **TRIPOD FUNCTION**

It is a semi-continuous problem. The global minimum is $f_{min} =0$ on (0, -50). This function too is theoretically easy. But it is difficult for a lot of algorithms that get trapped in the two local minima. Here, d represents the number of dimensions and $x_i \in [-100,100]$ for $i=1,..,.d.$

$$f(x) = \begin{bmatrix} p\left(x_2\right)*\left(1+p(x_1)\right) \\ abs\left(x_1+50*p(x_2)*\left(1-2*p\left(x_1\right)\right)\right) \\ +abs\left(x_2+50*\left(1-2*p\left(x_2\right)\right)\right) \end{bmatrix} \tag{5.6}$$

5.1.2 IMPACT OF TEMPERATURE (T_c)

In CCA the cricket's chirp depends on the outside temperature. Generally, the higher the temperature of the environment, the higher the chirping rate. Here the cricket is allowed to chirp at different temperatures. The temperature is taken as 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. The program is run for 100 times for every temperature value to find the global optimal value and an average number of iterations is calculated. In figure 5.2(a)-(f) the effect of the different temperatures for the chosen benchmark function is shown.

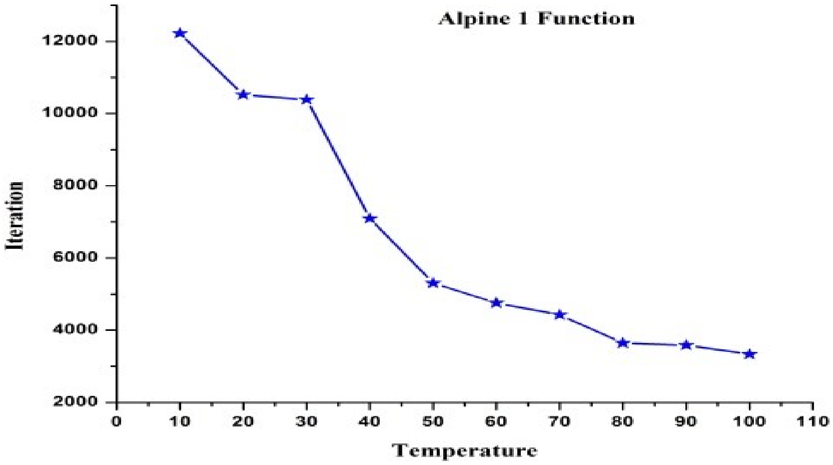


Figure 5.2(a): Alpine 1 function - Number of iterations vs Temperature

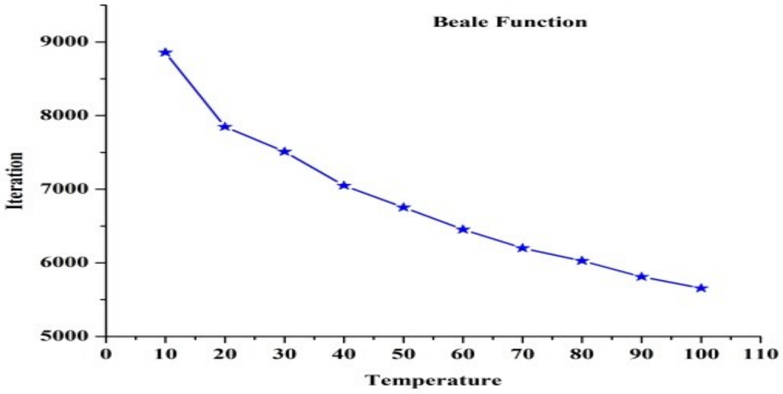


Figure 5.2(b): Beale function - Number of iterations vs Temperature

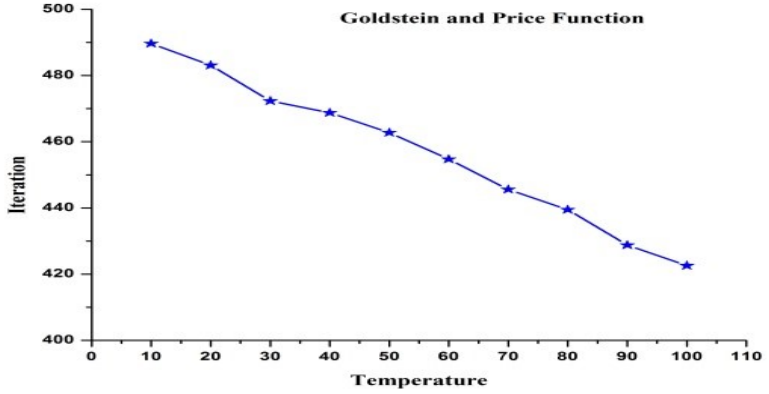


Figure 5.2(c): Goldstein & Price function - Number of iterations vs Temperature

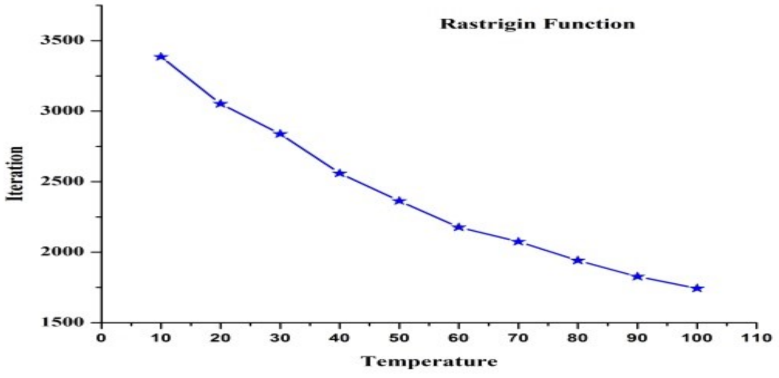


Figure 5.2(d): Rastrigin function - Number of iterations vs Temperature

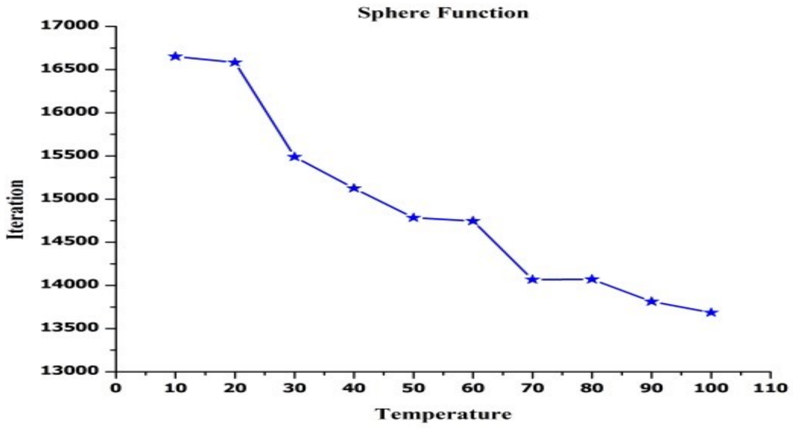


Figure 5.2(e): Sphere function - Number of iterations vs Temperature

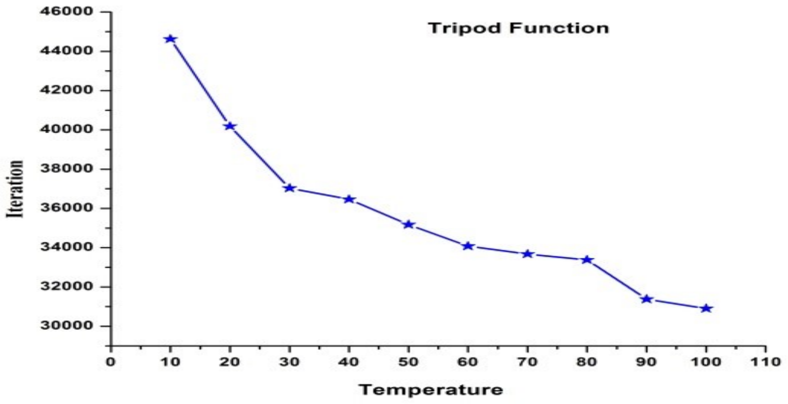


Figure 5.2 (f): Tripod function - Number of iterations vs Temperature

From the graphs shown in figure 5.2 (a)-(f), it is visible that when the temperature is increased the number of iterations essential to find the optimal value is reduced for all functions. So the temperature between 90 and 100 is fixed as the optimal temperature for CCA to perform well.

5.1.3 IMPACT OF AGGRESSION RATE (A_r)

When a cricket wants to fight, it makes an aggressive chirp. Since not all the crickets chirp for aggression, it is needed to choose the aggression rate *(A_r).* Having analyzed, the performance of CCA to be better at higher temperatures, this experiment analyzes the impact of different aggression rate A_r at the temperature of (T_c=100). The probability of aggression rate is varied like 0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95 and 0.99. The program is run 100 times for all test functions with every A_r value and the average iterations to find the global minima are calculated. Figure 5.3(a)-(f) shows how the aggression rate effects the results in different functions.

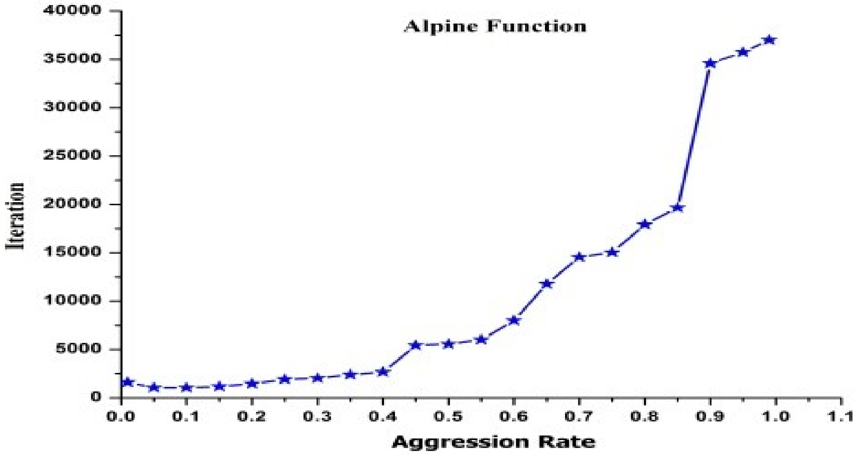


Figure 5.3(a). Alpine 1 function - Number of iterations vs Aggression Rate

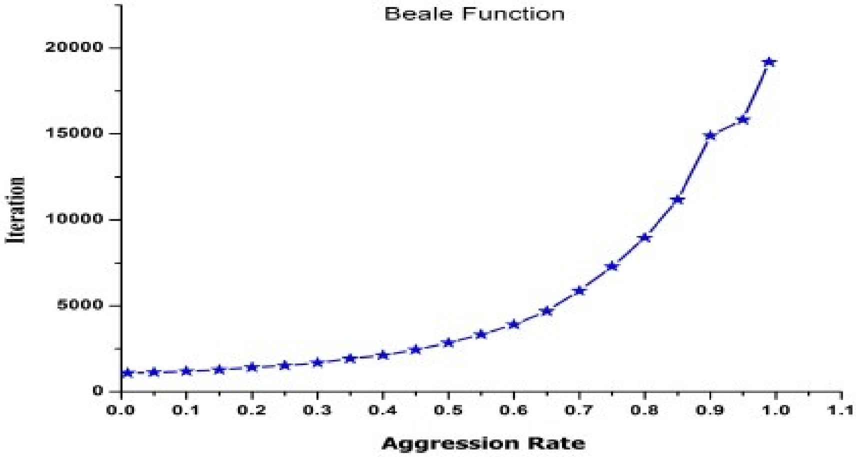


Figure 5.3(b). Beale function - Number of iterations vs Aggression Rate

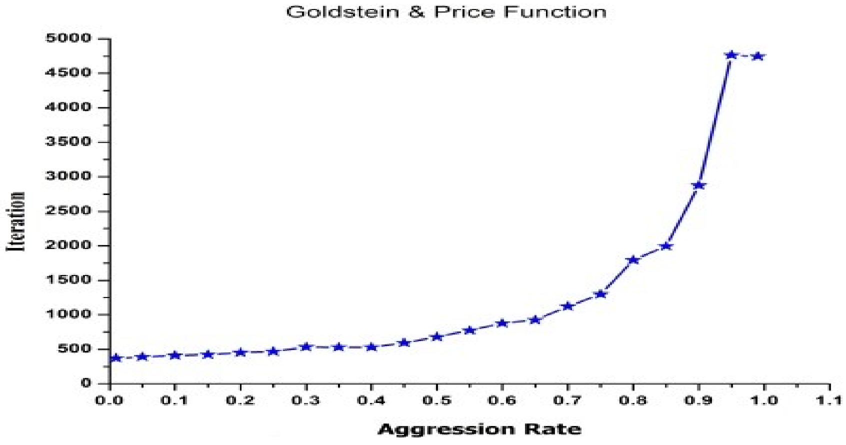


Figure 5.3(c). Goldstein & Price function - Number of iterations vs Aggression Rate

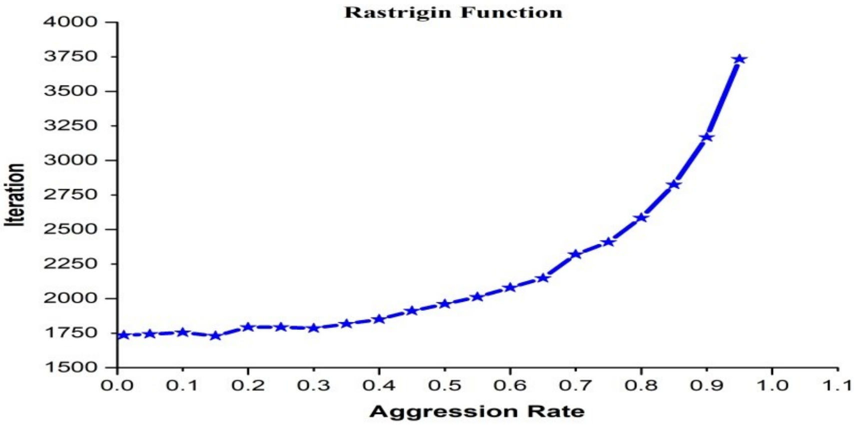


Figure 5.3(d). Rastrigin function - Number of iterations vs Aggression Rate

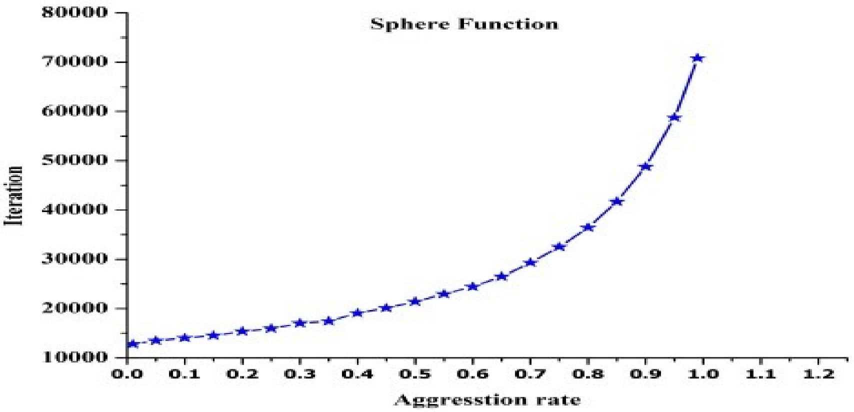


Figure 5.3(e). Sphere function - Number of iterations vs Aggression Rate

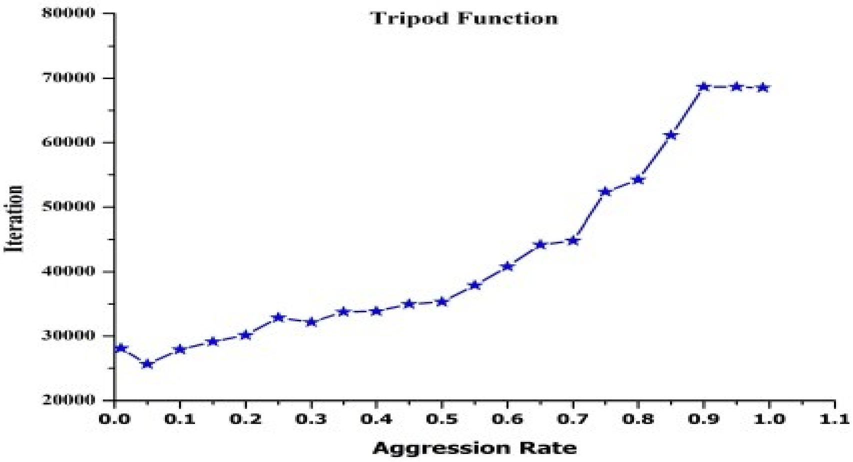


Figure 5.3(f). Tripod function - Number of iterations vs Aggression Rate

From the graphs, it is clear that with a lower value of aggression rate, the number of iterations needed for optimization is less. CCA shows better results at aggression rate 0.05, 0.10, and 0.15. Based on the results, [0.05 to 0.25] is considered as the optimal aggression rate for low dimensional problems.

5.1.4 IMPACT OF CROSSOVER RATE (C_r)

After the calling song, the female and male crickets undergo the mating process. The process is similar to crossover in genetic algorithm and hence it is carried out using different crossover rates *(C_r)*. Generally, crossover rate is high in GA. So the program is tested for crossover rates 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95 and 0.99 at a temperature value of 100 and aggression rate of 0.15. The performances of different crossover rates in each test functions are shown in figure 5.4 (a)-(f).

From the figure, it is shown that higher crossover rates from 0.75 to 0.85 shows good performance. So the crossover rate *(C_r)* is fixed to 0.80.

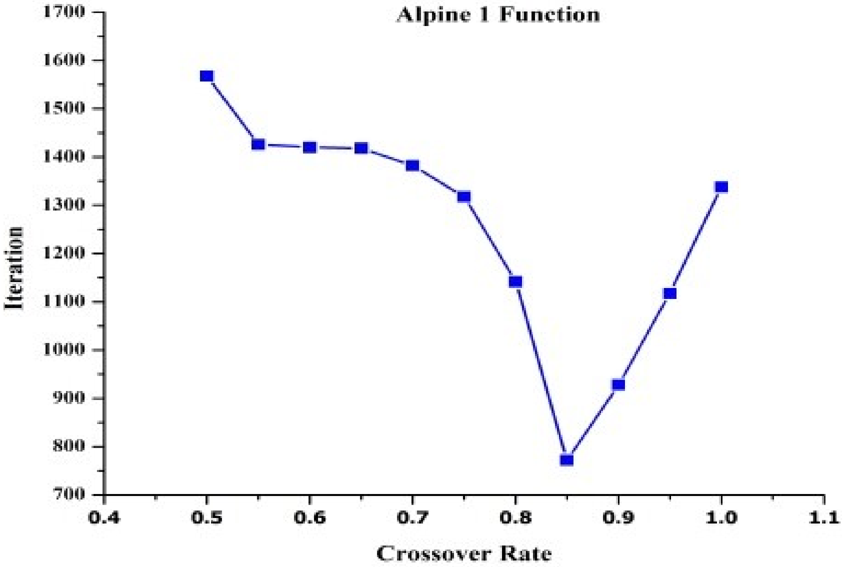


Figure 5.4(a): Alpine 1 function - Number of iterations vs Crossover Rate

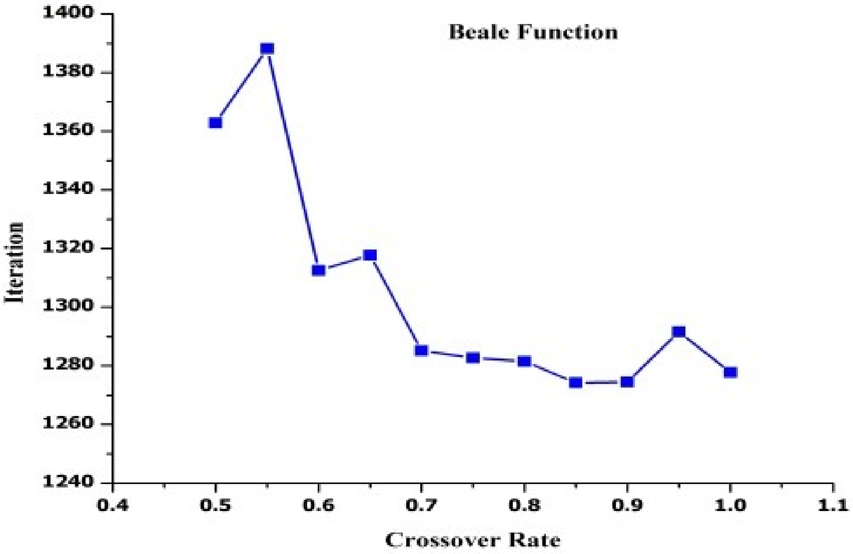


Figure 5.4(b): Beale function - Number of iterations vs Crossover Rate

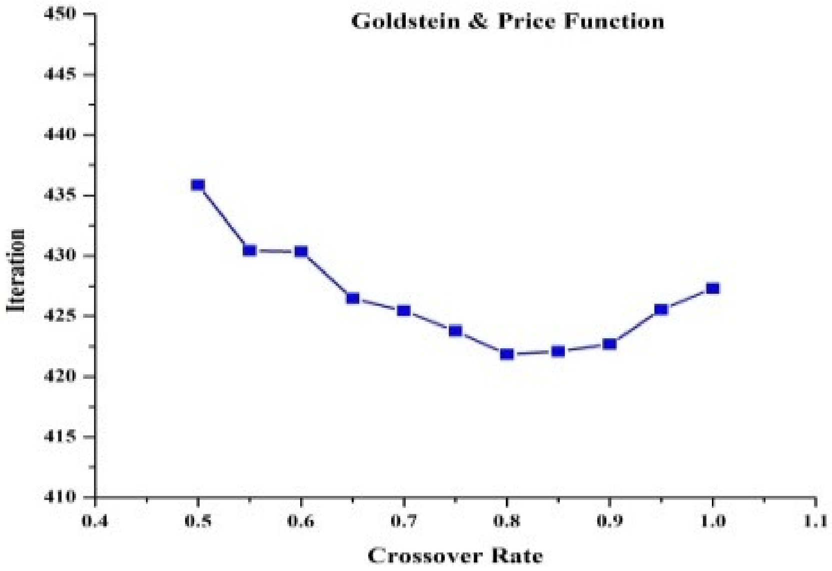


Figure 5.4(c): Goldstein & Price function - Number of iterations vs Crossover Rate

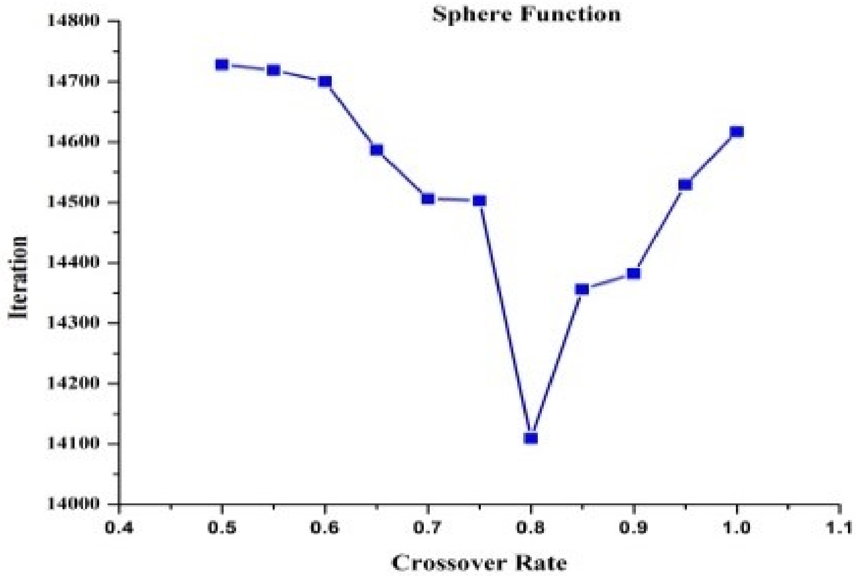


Figure 5.4(a): Sphere function - Number of iterations vs Crossover Rate

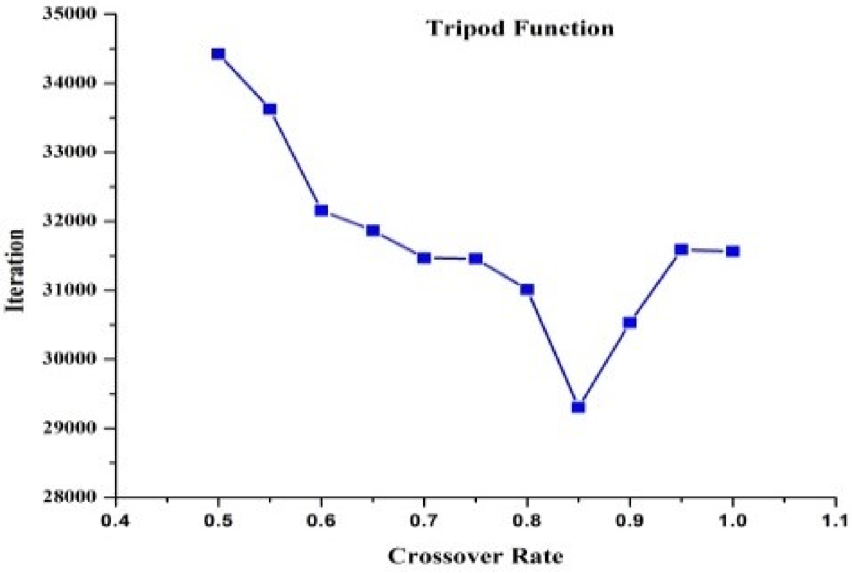


Figure 5.4(f): Tripod function - Number of iterations vs Crossover Rate

5.1.5 IMPACT OF FEMALE SELECTION (F_S)

In CCA, only the male crickets chirp for mating. To perform the mating operation the male crickets have to choose the female crickets. The selection of female (F_s) crickets may be done in different ways. The CCA is tested for the following female cricket selection methods.

- **RANDOM FEMALE SELECTION**

In this method, 50% of the total crickets are randomly chosen as female crickets and allowed to mate with male crickets randomly.

- **BEST FIT FEMALE SELECTION**

In the best fit selection process, the highest fit cricket is selected as female cricket and makes the crossover process in two ways. First, the female cricket is allowed to mate with all the male crickets and second, allow mating only with one male cricket which is randomly chosen.

- **WORST FIT FEMALE SELECTION**

In the worst fit process, the worst cricket is selected as female cricket and allowed to mate. Since this selection does not converge to the optimal fitness value, it is not considered for female selection.

The program is run for 100 iterations for the Random Selection method and Best Fit Selection method (both mating with one cricket and all cricket). The fitness value obtained for the said selection methods for all the test functions are run for 100 iterations and the graph of convergence to the optimal solution is shown in figure 5.5(a)-(f). It is clearly observed from the graph that the Best Fit Selection scheme mating with all male crickets, offers better results in terms of convergence and speed.

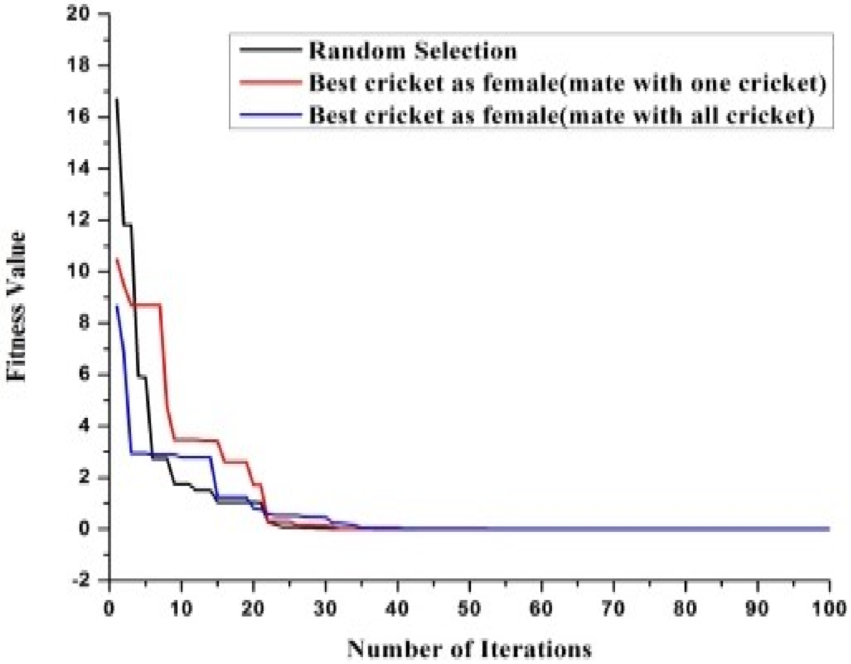


Figure. 5.5(a): Alpine 1 function - Fitness values for three female selection mechanisms

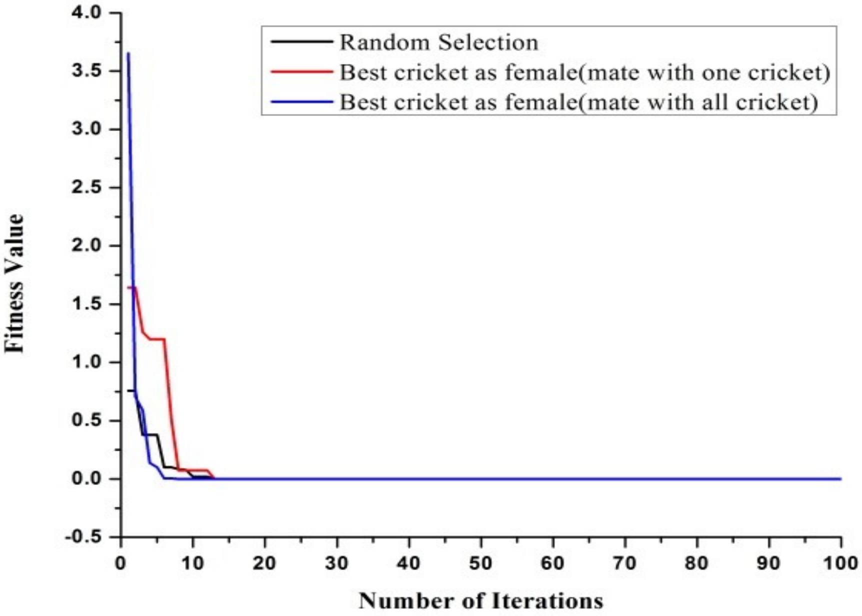


Figure. 5.5(b): Beale function - Fitness values for three female selection mechanisms

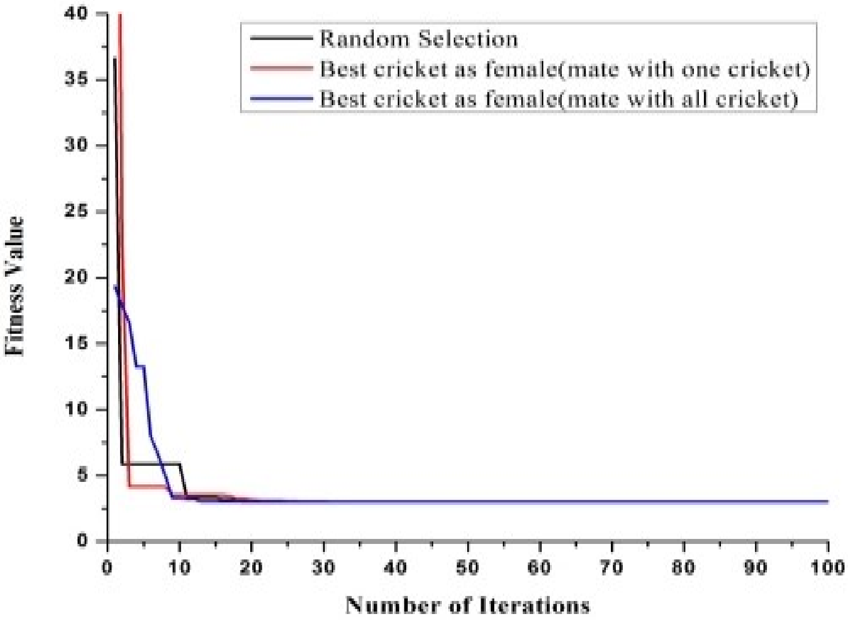


Figure. 5.5(c): Goldstein & Price function - Fitness values for three female selection mechanisms

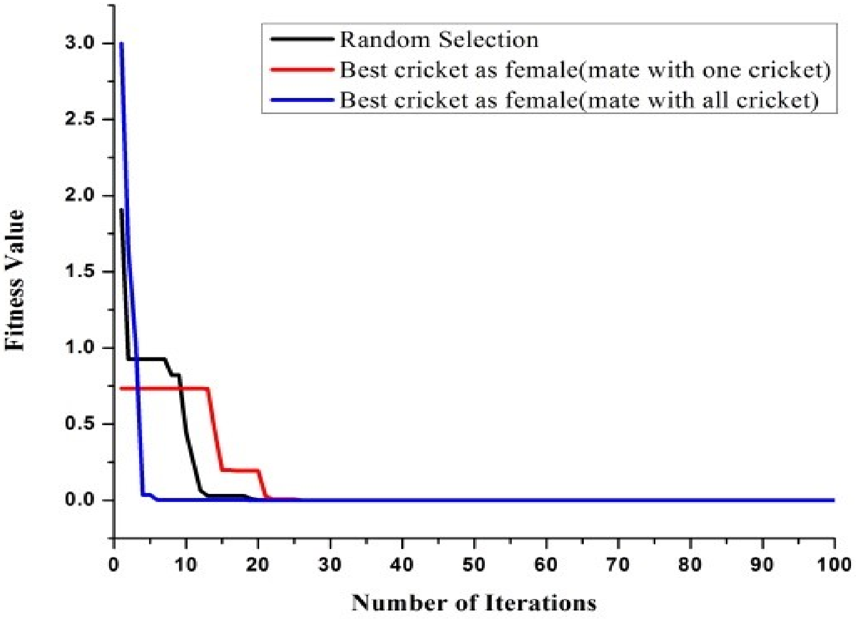


Figure. 5.5(d): Rastrigin function - Fitness values for three female selection mechanisms

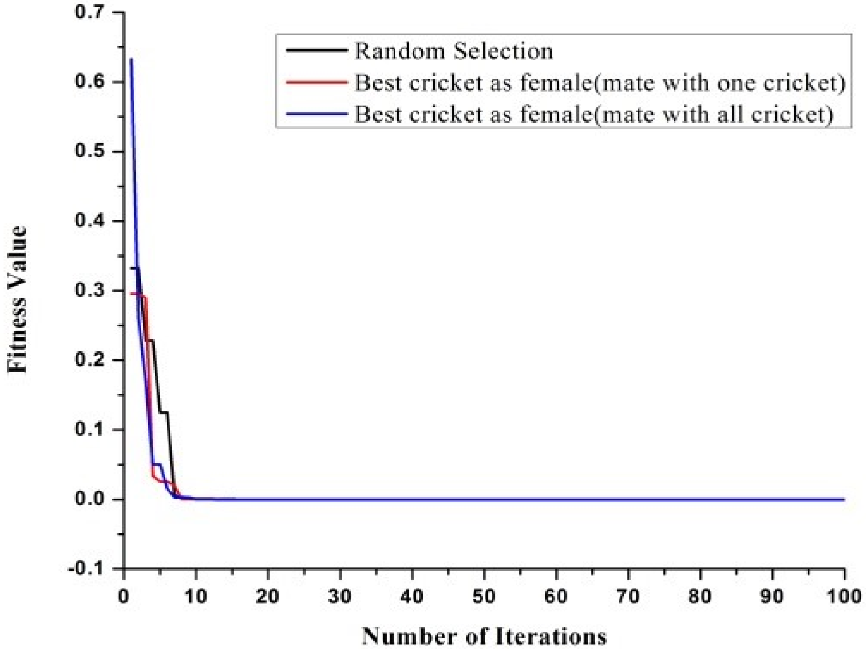


Figure. 5.5(e): Sphere function - Fitness values for three female selection mechanisms

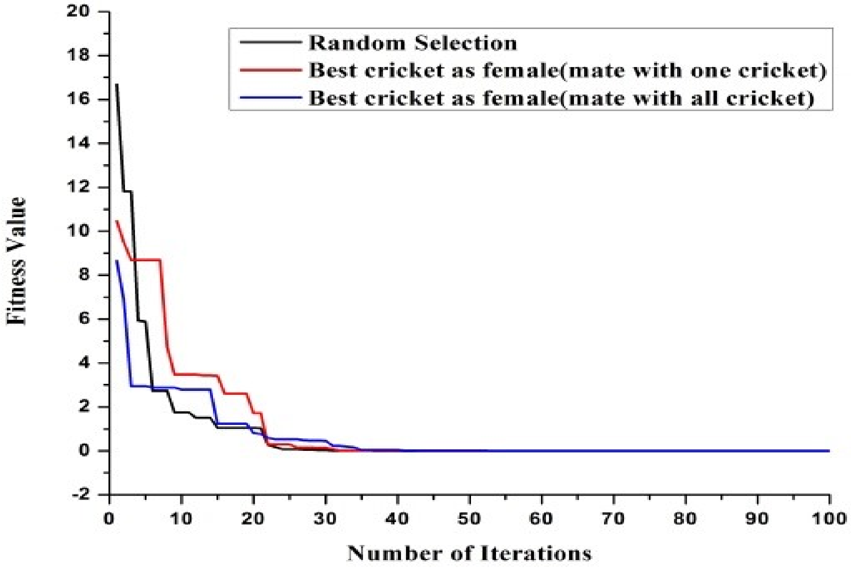


Figure. 5.5(f): Tripod Function - Fitness values for three female selection mechanisms

From the experiments conducted, the best CCA parameters found are listed in table 5.1. Using these CCA parameter values, the CCA Algorithm is compared with another meta-heuristic algorithm in the next section.

Table 5.1: Parameter values of CCA

Parameter Name	Value
Temperature	100
Aggression rate	0.15
Crossover rate	0.80
Female Selection	Best cricket as female and mate with all cricket

5.2 COMPARISON OF CCA WITH OTHER META-HEURISTIC ALGORITHMS

After fine-tuning the optimal parameter values for CCA, the algorithm is compared with the popular meta-heuristic algorithms to show the betterment of CCA. Bat Algorithm (BA) and Cuckoo Search (CS) Algorithm are considered in comparison with CCA as they are the recent and popular algorithms. GA and PSO have not been compared here since the BA and CS have shown better results compared to them. The BA was simulated based on the echolocation behavior of bats[20]. The bat flies in search of its prey with a velocity v_i at position (solution) x_i with varying frequency. It has loudness A_i and pulses emission rate r_i. For BA the loudness and pulse are set to 0.5 and the minimum and maximum frequency is taken as 0 and 2.

The CS algorithm was developed based on the breeding behavior of cuckoo together with Levy flight behavior of some birds [20]. In the CS algorithm, the discovery rate of the alien egg is the only parameter that needs to be set and hence it is set to 0.25.

The CCA, BA, and CS are run for every test function using the above-mentioned parameter values. The convergence towards the optimal value of each test function for each algorithm in 100 generations is shown in figure 5.6(a)-(f). The fitness value of each algorithm for the benchmark test functions in 100 generations is shown in table 5.2. The CCA shows better fitness value compared to the BA and CS with a fixed number of 100 iterations.

Table 5.2: Comparison of fitness values of CCA with BA and CS in 100 generations

Function Name(f(x))	***Fitness Value (f_{min})***		
	BA	**CS**	**CCA**
Alpine 1	1.17E-08	2.02E-08	**2.52E-21**
Beale	1.63E-08	5.66E-07	**8.93E-17**
Goldstein & Price	3	3	3
Rastrigin	2.37E-06	0.000322	**7.11E-15**
Sphere	7.17E-06	4.67E-09	**7.03E-23**
Tripod	0.000146	0.008835	**3.87E-11**

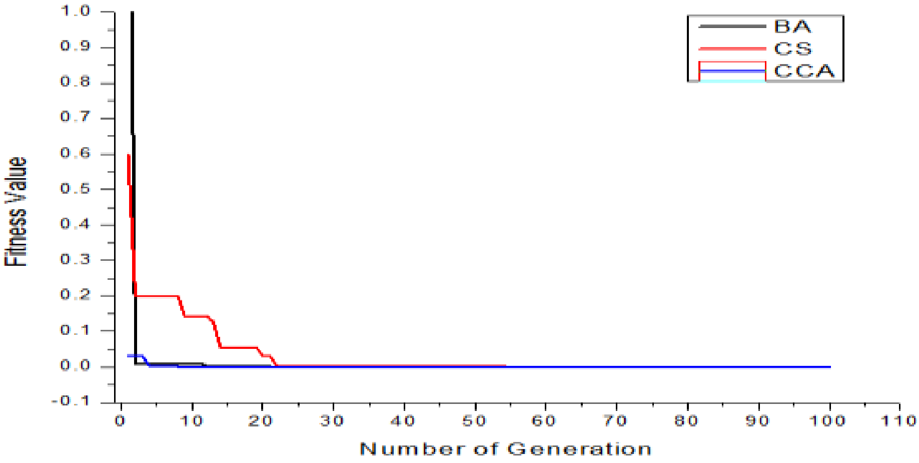


Figure 5.7(a): Fitness value of BA, CS, and CCA for Alpine Function

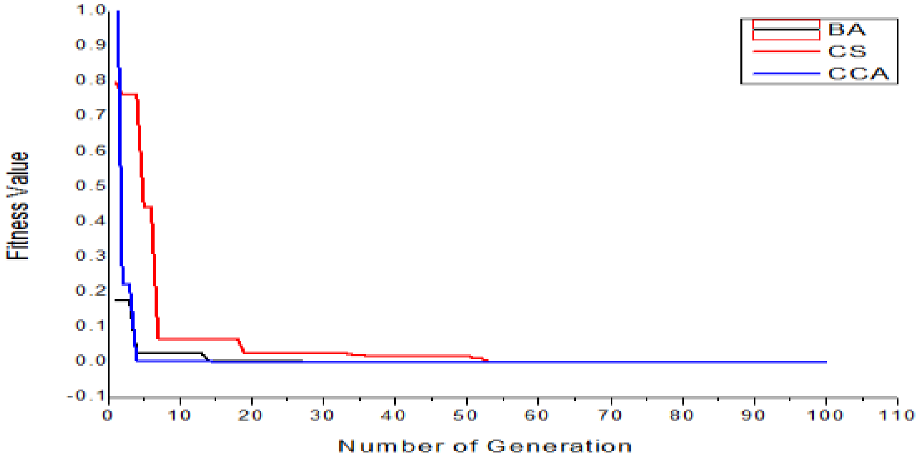


Figure 5.7(b): Fitness value of BA, CS, and CCA for Beale Function

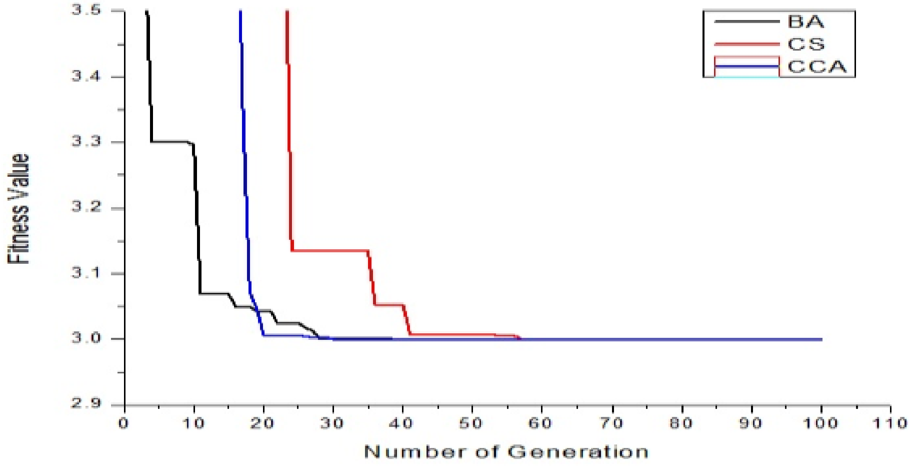


Figure 5.7(c): Fitness value of BA, CS, and CCA for Goldstein & price Function

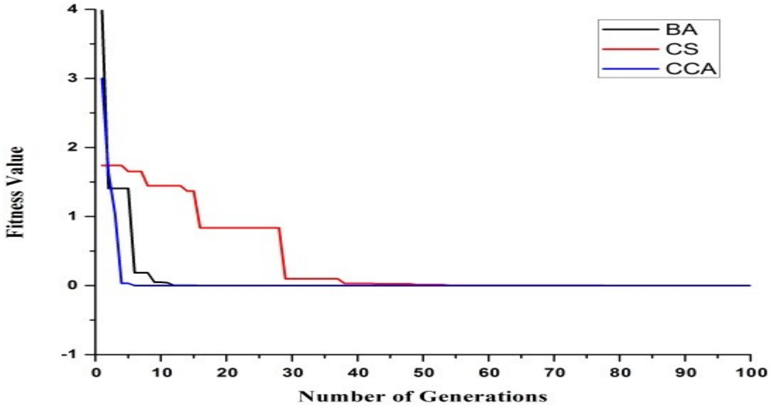


Figure 5.7(d): Fitness value of BA, CS, and CCA for Rastrigin Function

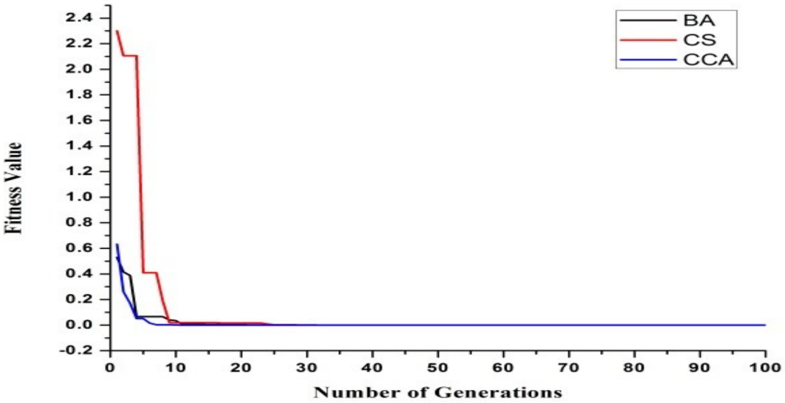


Figure 5.7(e): Fitness value of BA, CS, and CCA for Sphere Function

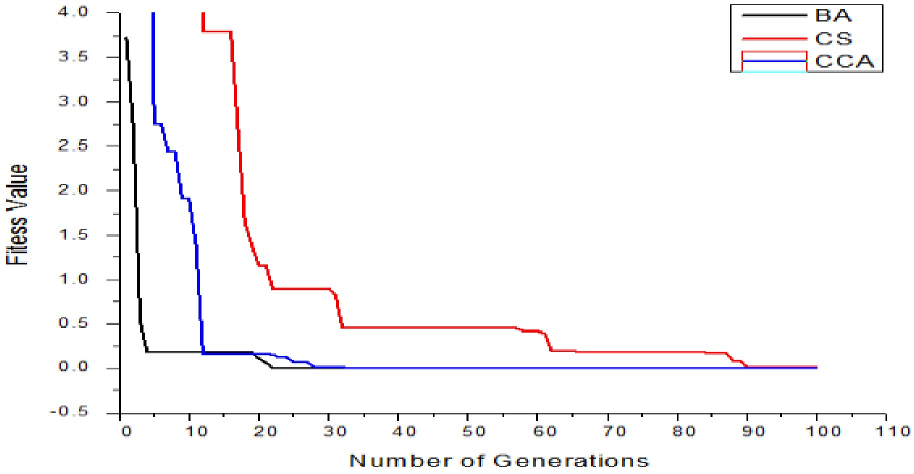


Figure 5.7(f): Fitness value of BA, CS, and CCA for Tripod Function

Table 5.3: Comparison of the mean of number of iterations and time for BA, CS, and CCA

Test Function	BA		CS		CCA	
	Iterations	Time(sec)	Iteration	Time(sec)	Iteration	Time(sec)
Alpine 1	742077.06	19.85605	146212.5	5.41595	**3911.727**	**0.164807**
Beale	391078.12	1.25630	17994.06	0.637054	**5124.63**	**0.234939**
Goldstein	7243.33	0.26624	6295.38	0.233315	**1771.35**	**0.051165**
Rastrigin	529285.08	17.50211	9573.69	0.472673	**1851.15**	**0.075483**
Sphere	241260.54	11.023255	195168.5	9.349882	**34098.54**	**0.963831**
Tripod	263111.22	11.53656	483756	25.9116	**47124.84**	**2.464338**

Table 5.3 shows the algorithm convergence for the benchmark test functions. In comparison with the so far best algorithms namely Bat and Cuckoo search, CCA requires a lesser number of iterations in lesser time to converge for each of the benchmark functions taken for experimental analysis. Each algorithm is executed 100 times and the mean of number of iterations and time taken to find the global optimal value is shown in table 5.3

Table 5.4 Comparison of CCA before parameter tuning and after parameter tuning and improvement

Test Functions	CCA(Before Tuning)	CCA(After Tuning)	Improvement (%)
Ackley	2475±194(100%)	**1116±113(100%)**	54.91
Easom	4890±461(100%)	**2488±194(100%)**	49.12
Griewangk	4227±259(100%)	**1213±216(100%)**	71.30
Matyas	670±152(100%)	**428±90(100%)**	36.12
Michalewicz	1364±182(100%)	**546±90(100%)**	59.97
Rastrigin	1286±168(100%)	**702±98(100%)**	45.41
Rosenbrock	1882±455(100%)	**1485±444(100%)**	21.09
Schwefel	1705±195(100%)	**809±86(100%)**	52.55
Shubert	2855±957(100%)	**1630±766(100%)**	42.91
Sphere	804±117(100%)	**352±78(100%)**	56.22

In table 5.4 the comparison of the mean and standard deviation of the number of iterations before and after tuning of parameters of CCA and its improvement is shown in percentage. The values of CCA before tuning are taken from table 4.4 of Chapter 4. From the experimental results, it is shown that after tuning the parameters the performance of CCA is improved.

5.3 STATISTICAL ANALYSIS

➢ ANALYSIS I

To test the significance of the results of CCA after tuning, a statistical analysis using one way ANOVA has been carried out on the number of iterations, taken by CCA before tuning and after tuning to converge to the optimal value, shown in table 4.4. The analysis is carried out by considering a 5% significance level over the number of iterations to find the optimal value with tolerance value $\tau \leq 10^{-5}$ corresponding to the test functions. Here the hypothesis is set as follows:

> ***Null hypothesis H_0:*** *There is no significant difference in the number of iterations between the CCA before tuning and after tuning.*
>
> ***Alternative hypothesis H_1:*** *There is a significant difference in the number of between the CCA before tuning and after tuning.*

The test is conducted using SPSS tool and the results found in the experiment are shown in table 5.5. From the ANOVA test shown in table 5.5, the p values (Sig.=0.033) is less than 0.05 (5% significance level) that provides an evidence against the null hypothesis. So it is concluded that there is a significant difference between the iteration numbers to obtain the optimal value before parameter tuning and after parameter tuning.

Table 5.5: ANOVA test over the methods CCA before and after tuning

ANOVA

iteration

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	6485466.050	1	6485466.050	5.320	.033
Within Groups	21941586.500	18	1218977.028		
Total	28427052.550	19			

➢ ANALYSIS II

After parameter tuning, the results found by CCA is compared with the other meta-heuristics BA and CS. The differences in performance of all algorithms are statistically analyzed using one way ANOVA test. The hypothesis is set as follows:

Null hypothesis H_0: *There is no significant difference in the number of iterations among the methods BA, CS, and CCA.*

Alternative hypothesis H_1: *There is a significant difference in the number of among the methods BA, CS, and CCA.*

The results found in the experiment in the ANOVA Test are shown in table 5.6. Since the p-value 0.015 is less than 0.05, the null hypothesis is rejected. So it is concluded that there is a significant difference between the iteration numbers to obtain the global optimal value among the techniques.

Table 5.6: ANOVA test over the methods CCA , BA and CS before and after tuning

ANOVA

Iterations

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	368994211215.896	2	184497105607.948	5.575	.015
Within Groups	496423190956.499	15	33094879397.100		
Total	865417402172.394	17			

5.4 SUMMARY

In this chapter, the impact of the different parameters used in CCA is analyzed. The parameters environmental Temperature T_c, Aggression Rate A_r, Crossover Rate C_r and Female Selection F_s have an effective contribution to the performance of CCA. When comparing the initial and final set of parameters, it is found that the final set provides better results compared to the case of initial parameter configuration for the problem under study. As per the analysis of the experiment the higher the temperature, the higher the fitness value of the crickets. The cricket produces high-frequency sound at high temperatures. But, in low aggression rate, it shows better performance for low dimension problems. In female selection, the best fit female selection converges faster compared to other female selection schemes. The values obtained through various experimental settings could be fixed as the standard parameters for the CCA algorithm in future. The comparison with its counterparts also shows that CCA performs better than others and hence it could be used as a good optimization technique.

Chapter 6
CRICKET CHIRPING ALGORITHM FOR MULTI-OBJECTIVE OPTIMIZATION (MOCCA)

In MOO, there are two or more objective functions to be optimized resulting in a set of optimal solutions which is also known as Pareto-optimal (PO) set whereas in SOO only one global optimum value is to be found. The fitness functions are evaluated either using a weighted-sum approach or the Pareto-ranking approach. In this chapter, CCA has been extended to solve MOO problems using two approaches which are discussed here. These two approaches are as follows:

1. Multi-objective CCA using Weighted Sum Approach (MOCCA-W)
2. Multi-objective CCA using Pareto Based Approach (MOCCA-P)

The several MOO algorithms, that falls into either of the two classes. They basically differ in the fitness function evaluation procedure. This research has concentrated on both the types and hence CCA was extended in both directions. The first one, i.e. MOCCA-W is an extension of CCA to solve MOO problems using the weighted sum approach. The second one, i.e. MOCCA-P uses the concept of Pareto dominance for solving MOO problems. Here we get a set of solution that balances the objectives. For MOO, different metrics are used to analyze the performance of Generational Distance, Spacing, and Maximum Spread. These are used to validate the performance of MOCCA in this chapter. The next section discusses MOCCA using weighted sum approach followed by experiment results and analysis.

6.1 MULTI-OBJECTIVE CCA USING WEIGHTED SUM APPROACH (MOCCA-W)

In the weighted sum method, the user needs to assign the weights before the fitness evaluation takes place. Each objective function is given a weight value and the weights are added to give a composite fitness value for the individuals of the MOO problem. In

Pareto-ranking approach, the dominance rule is used to rank the whole population and then each solution based upon its rank is assigned a fitness value, instead of its actual objective function value. Each solution is assumed to be equally important and all of them comprise the global optimum solutions. The problem in MOO is to optimize the set of objective functions simultaneously where the objectives in the problems often may conflict with each other and an improvement of one objective may lead to plummeting of another making it more difficult to solve.

The MOP deals with the minimization or maximization of objectives *F(x)* and can be subjected to a number of bounds or constraints. The normal optimization problem can be e formulated mathematically using equation 6.1 as follows:

Minimize/maximize,

$$F(x) = f_1(x), f_2(x), \dots., f_m(x) \tag{6.1}$$

Subject to, $G_i(x) \leq 0 \qquad i = 1, \dots., p$

$$H_j(x) \leq 0 \qquad i = 1, \dots., q$$

Here, m is the number of objective functions, p is the number of inequality constraints, q is the number of equality constraint and $x = \left[x_{1,}x_2, \dots \dots, x_k\right]^T$ are the decision variables, and G_i, H_j are the constraints function of the problem.

In this section, the weighted sum approach is used along with CCA to solve the multi-objective design problem. It combines all the objectives f_j into a single objective and the sum of assigned weight is always equal to 1 as shown in equation 6.2.

$$\sum_{j=1}^{k} w_j f_j, \qquad \sum_{j=1}^{k} w_j = 1 \tag{6.2}$$

Here, w_j is the weight generated randomly from a uniform distribution. The algorithm for multi-objective Cricket Chirping Algorithm is shown in table 6.1.

Table 6.1: Algorithm for multi-objective CCA with Weighted Sum

```
Algorithm_MOCCA-W( )
Begin
1. Initialize the cricket population $x_i$ $(i = 1, 2, ...,n)$
2. Choose  m crickets randomly as female crickets, such that m<=n/2
3. for i = 1 to N (Number of Pareto fronts), generate $k$ weights $w_j \geq 0$;
        such that    $\sum_{j=1}^{k} w_j = 1$
4. Form a single objective using equation (6.2)
5. Evaluate the crickets using the objective functions and the weights.
6.  While (stopping criteria not met)
        • Allow the cricket for mating chirp // Call procedure calling_chirp()//
        • Mate with female // Call procedure mating()//
        • Allow the cricket for aggression // Call procedure aggression_chirp()//
        • Return the winner
        • Return the  best cricket $x*$
7. End while
End
```

6.1.1 MULTI-OBJECTIVE TEST FUNCTIONS

Two benchmark test functions with convex and non-convex Pareto fronts that are widely used are considered to validate the proposed MOCCA. The functions are listed below:

- **ZDT1**: This is a function of convex front and mathematically stated as equation 6.3 as below:

$$f_1(x) = x_1 \ ,$$

$$f_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)}\right] \tag{6.3}$$

Where,

$$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right) \Big/ (n-1)$$

Here, n is the dimension number. When g=1, it reached to the Pareto-optimality and the true Pareto front for ZDT1 is $f_2 = 1 - \sqrt{f_1}$

- **ZDT2:** This is a function with a non-convex front. It is mathematically formulated by using equation 6.4:

$$f_1(x) = x_1$$

$$f_2(x) = g(x)[1 - x_1/g(x)]^2 \qquad (6.4)$$

Where, $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right) \Big/ (n-1)$ (2)

The true Pareto front for ZDT2 is $f_2 = 1 - (f_1)^2$

To quantify the performance of the algorithm, the error rate *(E_r)* is calculated. This error rate is the distance from the estimated Pareto front $(PF)^e$ to its corresponding true Pareto front$(PF)^t$. It is defined by using equation 6.5 as follows:

$$E_r = \|(PF)^e - PF^t\| = \sum_{i=1}^{N}((PF_i)^e - (PF)^t)^2 \qquad (6.5)$$

Where N defines the number of Pareto points.

6.1.2 EXPERIMENTAL RESULTS AND ANALYSIS

First, the algorithm is validated for some benchmark test functions. The parameters used in MOCCA-W are population size (n), Temperature (T_c), Crossover rate (C_r) and the Aggression rate (A_r) with the values of P_a=0.05 to 0.5, n=20 to 50 and C_r=0.80 to 0.95. The algorithm is run for 1000 iterations and generated 100 Pareto points by MOCCA. In figure 6.1 and figure 6.2, the true Pareto front and the estimated Pareto front by MOCCA for function ZDT1 and ZDT2 are shown respectively.

6.1.3 COMPARISON WITH OTHER METHODS

In order to measure the performance of the proposed MOCCA, it is compared with other MOO algorithms. The basic Particle Swarm Optimization (PSO) and Bat Algorithm (BA) with weighted sum approach are considered for comparison with the MOCCA-W.

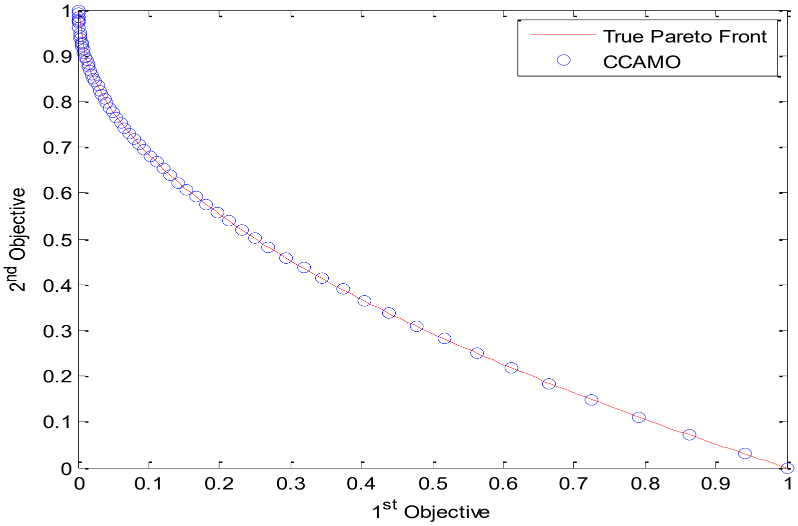


Figure 6.1: Pareto front of MOCCA-W and true Pareto front of ZDT1

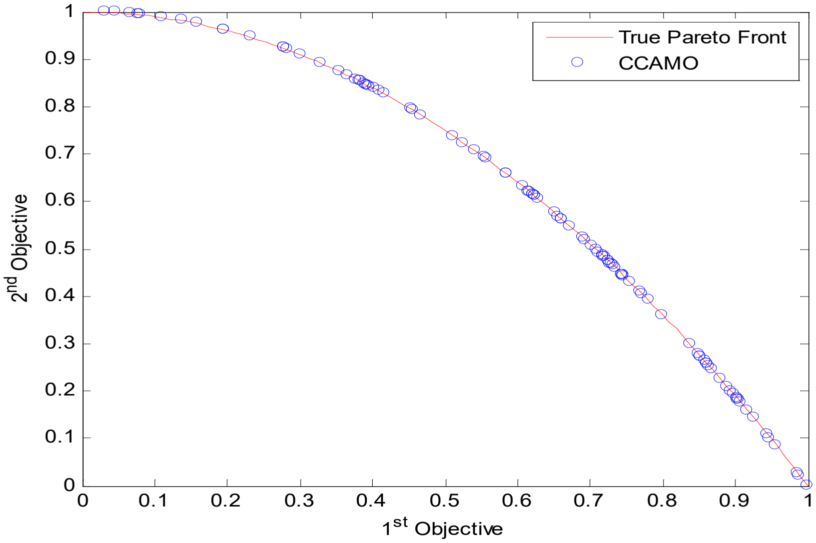


Figure 6.2: Pareto front of MOCCA-W and true Pareto front of ZDT2

The standard PSO is used to solve the problems with weighted sum approach. The parameter swarm size n=50 and inertia weight w=0.9 are used for all test problems. The parameters for BA is set as population size n=50, pulse reduction rate r= 0.7 and loudness A=0.25.

All the three algorithms are run for 1000 iterations and the distance between the true Pareto front and the estimated Pareto front of the algorithm (error rate) is calculated. The error rate is calculated by using equation 6.5.

Table 6.2 shows the error rate of each algorithm for both the test function ZDT1 and ZDT2. The experimental result shows that MOCCA has less error rate compared to its counterparts.

Table 6.2: Comparison of the error rate of the test problems

Functions	**Error (t=1000)**		
	PSO	**BA**	**MOCCA-W**
ZDT1	3.4e-3	2.2e-3	**2.7e-5**
ZDT2	1.9e-2	5.8e-3	**6.4e-4**

6.2 MULTI-OBJECTIVE CCA USING PARETO BASED APPROACH (MOCCA-P)

Having presented the MOCCA-W which uses the weighted sum strategy for MOO, this section describes the design and implementation of MOCCA-P which uses the Pareto approach for extending the CCA to solve MOO problems.

The main differences noticed in CCA for MOP are as follows:

- Instead of choosing female cricket from the population, allow the male cricket to find/search female cricket. Their chirping rate will increase as the temperature increases and based on their chirping rate female cricket gets attracted.
- When the cricket chirps for aggression, it fights with other male crickets and the winner is chosen based on six aggression levels:
  i. **Mutual Avoidance** [20]**:** In this level no aggressive interaction takes place. The winner is decided mutually.

ii. **Pre-Established Dominance** [20]**:** In this level one cricket attacks and the other retreats.

Table 6.3: Algorithm for Multi-objective Cricket Chirping algorithm with Pareto based

Algorithm for Multi-objective Cricket Chirping algorithm (MOCCA-P)

```
1. Inputs: N: Number of cricket population, M_gen: Maximum number of
   generation, T: environment temperature, $P_{agg}$: the probability of aggression, nrep:
   size of the repository, ngrid: number of grids.
2. Problem definition: d: dimension of the cricket in search space, x: the position of
   the cricket, M: number of objectives, ub: upper bound, lb: lower bound.
3. Initialization: initialize cricket population in the search space randomly;
           $x\ (i,d) = lb_{\ (1\ ,d)} + (ub_{\ (1\ ,d)} - lb_{\ (1\ ,d)}\ ) \cdot rand$
           Initialize the repository : rep=[ ]
4. Evaluate the objective function: $f_{(i,\ m)}$
5. Calculate the fitness:
           $fitness_{(i)} \leftarrow calculate\_fitness(x_{(i,d)}, f_{(i,m)})$
6. While stopping criteria is not met:
   a. Allow the cricket for mating chirp:
      i. $female \leftarrow Search\_female(x_{(i,d)})$
      ii. $offsprings \leftarrow mating(female,\ x_{(i,d)})$
      iii. $rep \leftarrow best(offsprings,\ parents)$ //Choose the best one among parents
           and offspring
   b. Allow for aggression chirp on aggression rate $A_r$
         $winner \leftarrow fight()$ //Warn the other male cricket for fighting
7. End while
8. Return the optimal solution
```

iii. **Antenna fencing** [20]**:** In antenna fencing crickets lashes with their antenna. It is an enthusiastically inexpensive signal that carries mostly motivational information about resource value.

iv. **Mandible spreading (Unilateral)** [20], [161]**:** One cricket shows broadly spread mandibles, which indicate that it is superior to the other.

v. **Mandible Spreading (bilateral)** [125]**:** In this level, both crickets display their spread jawbones. Mandible spreading indicates the strength of the cricket.

vi. **Wrestling** [145]**:** In this level a thoroughgoing fight where the crickets may repeatedly disengage, combat and bite other body parts and re-engage mandibles to show their strength.

The fight can be settled at any of the levels (i)-(vi) by an opponent. The looser retreating, upon which the established winner typically produces the rivalry song together with characteristic body tremulous (jerking).

Based on these behaviors of cricket the fitness calculation of two crickets is implemented. In Multi-objective Cricket Chirping Algorithm (MOCCA) an external repository is used to store the non-dominated solutions (Pareto front). The pseudocode is shown in table 6.3 and the detailed stepwise procedure is given in the next section.

6.2.1 GENERAL FRAMEWORK OF MOCCA

- **INPUT**

N: Size of the cricket population, *M_gen*: Maximum number of generations, T: Environment's Temperature, P_{agg}: the probability of aggression, *nrep*: size of the repository, *ngrid*: number of grids.

- **OUTPUT**

rep: An external repository '*rep*' is used to store Pareto front.

- **INITIALIZATION**

Initialize the cricket population in the search space randomly;

$$x\,(i,d) = lb_{(1,d)} + (ub_{(1,d)} - lb_{(1,d)}) \cdot rand \tag{6.6}$$

Where d is the dimension of the decision variables and $i=1,2,\ldots N$, lb is the lower bound and ub is the upper bound of the variable in the search space.

Initialize the repository ‘*rep’* which stores the non-dominated crickets in the initial *x*.

- **EVALUATE THE OBJECTIVE FUNCTION**

Calculate the objective function value of each objective for all the crickets. For cricket *i*, the value of each objective $f_{(i,\ m)},$ where m = 1 , 2, . . . , M, is calculated.

- **FITNESS CALCULATION**

This procedure involves calculating the strength of each cricket. The fitness calculation of the crickets is shown in table 6.4.

Table 6.4: Algorithm for fitness calculation of cricket

```
Calculate_fitness()
  For each cricket ($i=1$ to $N$)
     For each cricket ($j=i+1$ to $N$)
        For each objective ($k=1$ to $M$)
           If ($x_{(i,k)}<x_{(j,k)}$) and ($x_{(i,k)}!=x_{(j,k)}$)      // in case of minimization problems
              greater(i)=greater(i)+1; less(j)= less(j)+1;
           else if ($x_{(i,k)}==x_{(j,k)}$)
              equal(i)= equal(i)+1;  equal(j)= equal(j)+1;
           else
              greater(j)=greater(j)+1;  less(i)= less(i)+1;
           end
           fit(i)=[ greater(i)+ equal(i)+ less(i)];
           fit(j)=[ greater(j)+ equal(j)+ less(j)];
           if (fit(i)>= fit(j))
              strength(i)= strength(i)+1;
           End
     End
  End
```

➢ MATING PHASE

In this phase, the cricket will search for female crickets and mate with them. After successful completion of mating, they produce the offspring.

SEARCH FEMALE CRICKET

In this phase, the cricket makes calling chirps to search for female crickets and based on their chirping rate the female crickets get attracted. The movement of the cricket is calculated by equations 4.1 to 4.6 given in Chapter 4.

MATING

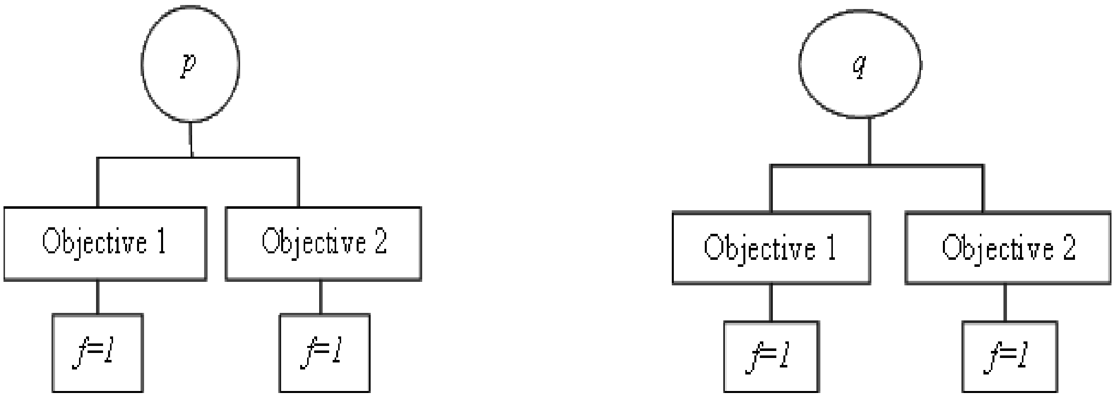
The mating process is done similar to the crossover process of the genetic algorithm. The mating process produces offspring and selects the best one among the offspring and parents. Then the non-dominated cricket is stored in the repository '*rep'*.

➢ AGGRESSION PHASE

The male cricket gets into the aggression phase with probability rate A_r. The two crickets fight with each other and the winner is selected based on the six aggression levels. The levels are described as follows:

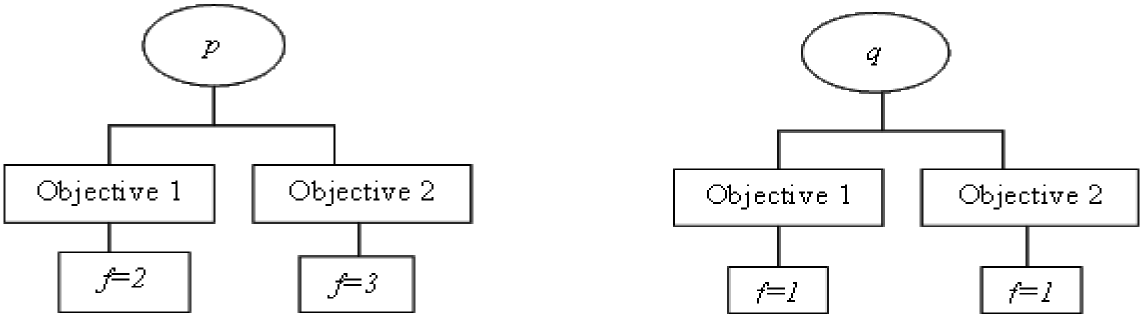
LEVEL 1: MUTUAL AVOIDANCE

When the values of each objective of a solution *p* are equal to the corresponding values of each objective of the solution *q*, then anyone solution (cricket individual) will be selected as the winner. For example, *p* and *q* have two objectives and the fitness *(f)* of both solutions is equal. In this case, any solution *p* or *q* is selected randomly.



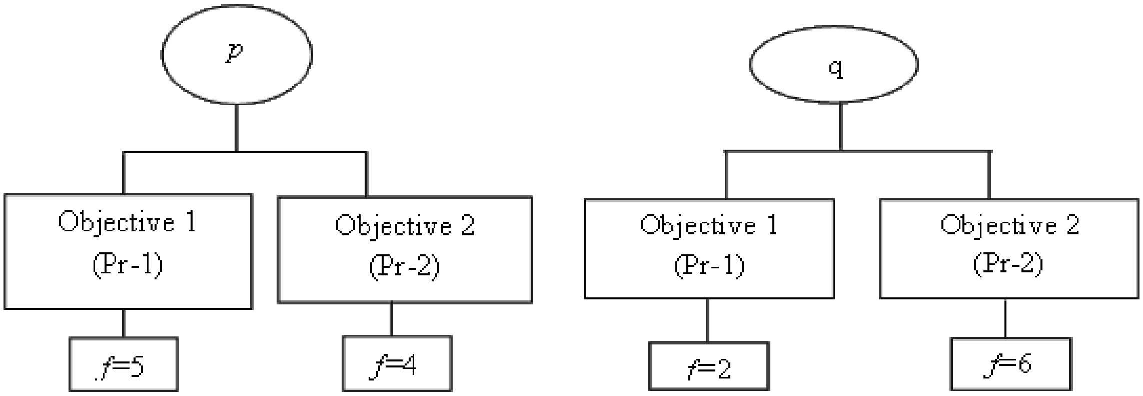
LEVEL 2: ANTENNAL FENCING

When the values of each objective of a solution *p* are greater than the corresponding values of each objective of the solution *q* then *p* wins. For example, in the following figure all the objectives of *p* are greater than *q*, so *p* will win.



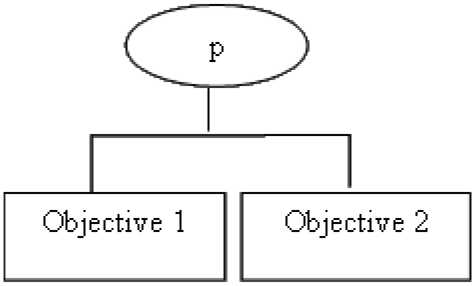
LEVEL 3: PRE-ESTABLISHED DOMINANCE

For this phase fix a priority *(Pr)* for each objective and based on priority and objective value choose the winner. For example, the priority is assigned high in objective 2. Here higher value is having high priority. In this case, the solution q will win since q is having the highest value of objective 2, that is having higher priority.

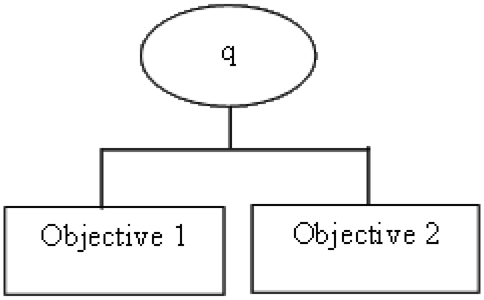


LEVEL 4: MANDIBLE SPREADING (UNILATERAL)

In this phase, One solution *p* that satisfies the constraints, whereas another one *q* is not, then *p* will win.



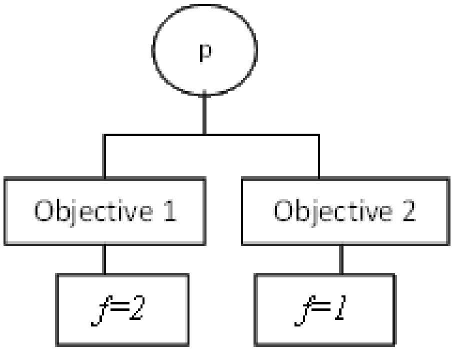
Satisfying Constraints



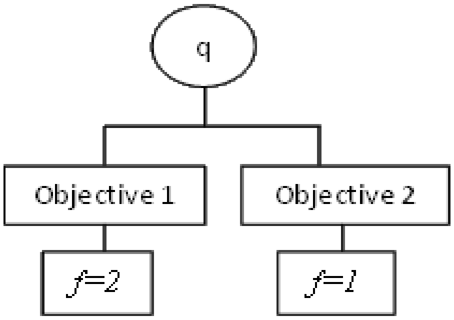
Not satisfying Constraints

LEVEL 5: MANDIBLE SPREADING (BILATERAL)

In this case, both the solutions are satisfying constraints, but a number of constraints satisfying solution will win. Check the number of constraints satisfied by each solution. For example, from the figure number of constraints satisfied by q is more than p. So q will win.



No. of Satisfying Constraints=2



No. of Satisfying Constraints=3

LEVEL 6: WRESTLING

At this level, the cricket will fight with each other but not exploited in this research.

- **RETAINING OF NON-DOMINATED SOLUTION**

An external repository or archive is used to store the records of non-dominated solutions. It consists of an archive controller and an adaptive grid. The function of addition or deletion of a solution to the archive is controlled by the archive controller. The main purpose of the adaptive grid is to produce well-distributed Pareto fronts. The reason for choosing an adaptive grid is its computational cost that is easy and lower than niching.

6.2.2 EXPERIMENTAL RESULTS AND ANALYSIS

This section first describes the standard benchmark test functions that are considered for experiments. The different performance metrics used for performance measurement and the implementation results of MOCCA-P are described in the following sections.

6.2.2.1 MULTI-OBJECTIVE TEST FUNCTIONS

There are a large number of standard test functions that are available for MOO problems. To validate the proposed MOCCA-P, a subset of a few widely used functions is selected that is convex, non-convex and discontinuous. The test functions without constraints are given in table 6.5 and the test problems with constraints are given in table 6.6.

6.2.2.2 PERFORMANCE METRICS

Any algorithm is validated by using a set of performance metrics. The metrics used for validating SOO problems may not correctly evaluate the performance of MOO problem. Hence there exists a separate set of performance metrics exclusively designed for validating MOO problems. The important ones that are used to evaluate the performance of the proposed MOCCA are given below.

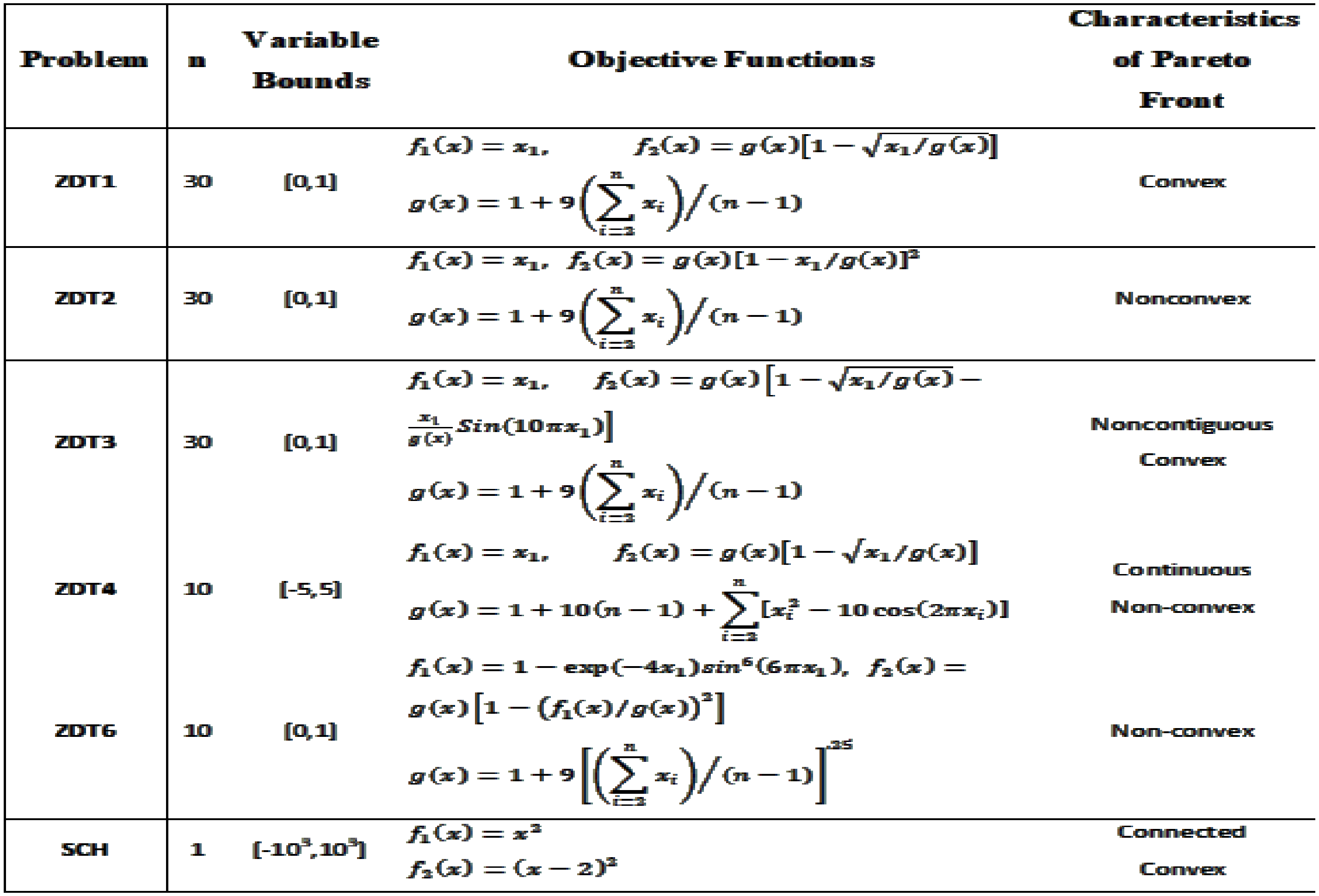
- **GENERATIONAL DISTANCE (GD)**

The most commonly used performance metric is Generational distance. It is measured as the extent to which the actual Pareto Front and the obtained Pareto Front are distant from each other. It is mathematically computed as shown in Equation (6.7)

$$GD = \frac{1}{n}\sqrt{\sum_{i=1}^{n} dist_i^2} \qquad (6.7)$$

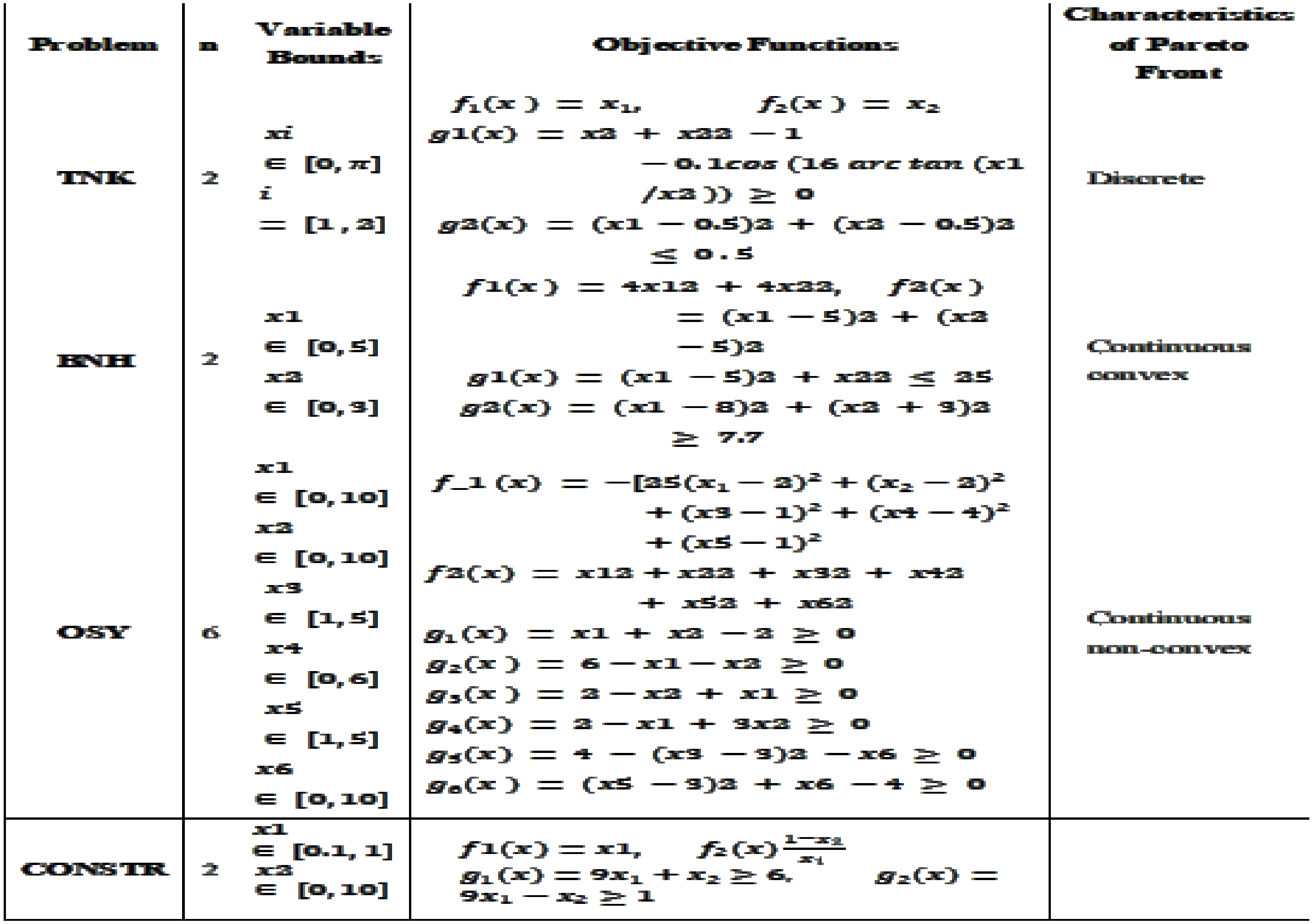
In equation 6.7, n indicates the cardinality of solutions in the generated Pareto-Front. $dist_i$ signifies the Euclidean distance between solution i in the actual Pareto front and its closest neighbor in the generated Pareto front. The lesser is the value of *GD,* the better will be the convergence.

Table 6.5: Test problem without Constraint



Problem	n	Variable Bounds	Objective Functions	Characteristics of Pareto Front
ZDT1	30	[0,1]	$f_1(x) = x_1, \quad f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	Convex
ZDT2	30	[0,1]	$f_1(x) = x_1, \; f_2(x) = g(x)[1 - x_1/g(x)]^2$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	Nonconvex
ZDT3	30	[0,1]	$f_1(x) = x_1, \quad f_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)} - \right.$ $\left.\frac{x_1}{g(x)} Sin(10\pi x_1)\right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	Noncontiguous Convex
ZDT4	10	[-5,5]	$f_1(x) = x_1, \quad f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	Continuous Non-convex
ZDT6	10	[0,1]	$f_1(x) = 1 - \exp(-4x_1) sin^6(6\pi x_1), \; f_2(x) =$ $g(x)\left[1 - (f_1(x)/g(x))^2\right]$ $g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{.25}$	Non-convex
SCH	1	$[-10^3, 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	Connected Convex

Table 6.6: Test problem with Constraints



Problem	n	Variable Bounds	Objective Functions	Characteristics of Pareto Front
TNK	2	$x_i \in [0, \pi]$ $i = [1, 2]$	$f_1(x) = x_1, \quad f_2(x) = x_2$ $g1(x) = x2 + x22 - 1 - 0.1cos\,(16\, arc\, tan\, (x1 / x2)) \geq 0$ $g2(x) = (x1 - 0.5)2 + (x2 - 0.5)2 \leq 0.5$	Discrete
BNH	2	$x1 \in [0, 5]$ $x2 \in [0, 3]$	$f1(x) = 4x12 + 4x22, \quad f2(x) = (x1 - 5)2 + (x2 - 5)2$ $g1(x) = (x1 - 5)2 + x22 \leq 25$ $g2(x) = (x1 - 8)2 + (x2 + 3)2 \geq 7.7$	Continuous convex
OSY	6	$x1 \in [0, 10]$ $x2 \in [0, 10]$ $x3 \in [1, 5]$ $x4 \in [0, 6]$ $x5 \in [1, 5]$ $x6 \in [0, 10]$	$f\_1\,(x) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x3 - 1)^2 + (x4 - 4)^2 + (x5 - 1)^2$ $f2(x) = x12 + x22 + x32 + x42 + x52 + x62$ $g_1(x) = x1 + x2 - 2 \geq 0$ $g_2(x) = 6 - x1 - x2 \geq 0$ $g_3(x) = 2 - x2 + x1 \geq 0$ $g_4(x) = 2 - x1 + 3x2 \geq 0$ $g_5(x) = 4 - (x3 - 3)2 - x6 \geq 0$ $g_6(x) = (x5 - 3)2 + x6 - 4 \geq 0$	Continuous non-convex
CONSTR	2	$x1 \in [0.1, 1]$ $x2 \in [0, 10]$	$f1(x) = x1, \quad f_2(x)\frac{1-x_2}{x_1}$ $g_1(x) = 9x_1 + x_2 \geq 6, \quad g_2(x) = 9x_1 - x_2 \geq 1$	

- **SPACING (SP)**

The primary intention of spacing metrics is to determine the extent to which the solutions are equally spread along the generated Pareto front. It can be mathematically defined as in equation 6.8 [175]:

$$SP = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(dist_i - \bar{d}\right)^2} \qquad (6.8)$$

Where, $\bar{d} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(dist_i)}$

In equation 6.8, n and $dist_i$ holds similar meaning as that of equation 6.7. When this metric holds a small value, it signifies a more uniform spread of the solutions.

- **MAXIMUM SPREAD (MS)**

The maximum spread (MS) metric exhibits the extent to which the actual Pareto Front encloses the generated Pareto-optimal front. This is identified in accordance with the hyper boxes constructed by the optimal function values from the actual Pareto-optimal front and the generated Pareto-optimal front. It is mathematically expressed as in Equation 6.9.

$$MS = \left[\frac{1}{m}\sum_{i=1}^{m}\left[\frac{min(f_i^{max}, F_i^{max}) - min(f_i^{min}, F_i^{min})}{F_i^{max} - F_i^{min}}\right]^2\right]^{1/2} \qquad (6.9)$$

In Equation 6.9, m indicates the number of objectives. $f_i^{max}$and f_i^{min} signify the respective maximum and minimum of associated i^{th} objective in the generated Pareto-front, respectively, and $F_i^{max}$and $F_i^{min}$are the maximum and minimum of the i^{th} objective in the true Pareto-front. When maximum spread metric characterizes a larger value, it depicts that the spread of the solutions is better.

6.2.2.3 PARAMETER SETTINGS

Generally, meta-heuristics approaches need parameter settings for better performance. The different parameters used in MOCCA, MOPSO, SPEA2, and NSGA2 are given in table 6.7

Table 6.7: Parameters used in MOCCA and the other algorithms

Parameters	MOPSO	SPEA2	NSGA2	MOCCA
Population Size	100	100	100	100
External Archive Size	100			100
No. of adaptive grid	7			7
Inertia weight	0.4			
c_1 and c_2	[0,1]	-	-	-
Aggression rate	-	-	-	0.50

6.2.2.4 EXPERIMENTAL RESULTS

To illustrate the validity of the proposed MOCCA, a number of experiments are conducted over test functions for both multi-objectives optimizations with constraint and without constraints. Figure 6.3 (a)-(g) shows the non-dominated Pareto front produce by MOCCA-P for MOO without constraints and figure 6.4 (a)-(d) shows the Pareto front produce by MOCCA for MOO with constraints.

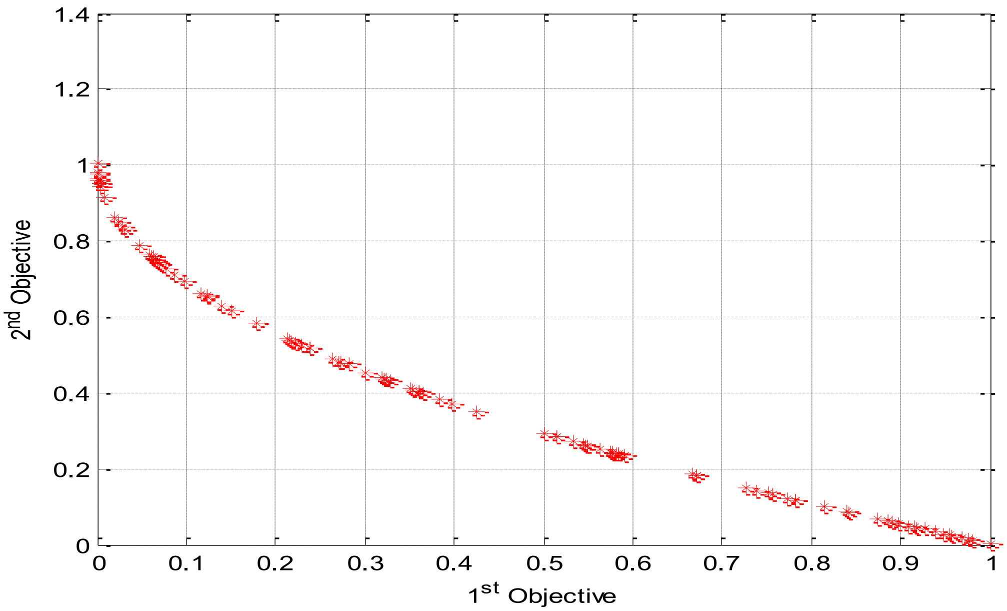


Figure 6.3(a): Pareto front produced by MOCCA-P for ZDT1

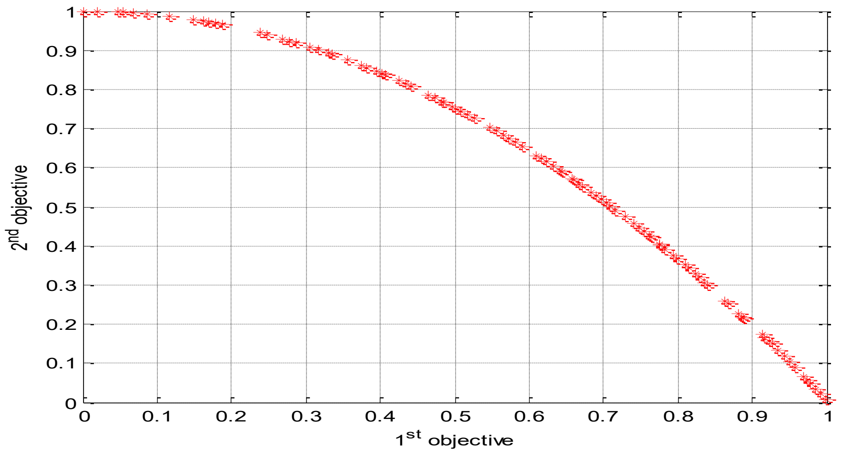


Figure 6.3(b): Pareto front produced by MOCCA-P for ZDT2

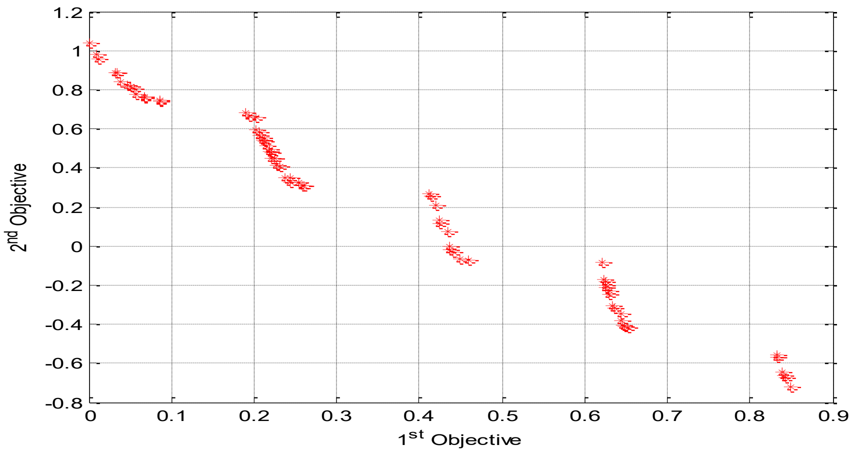


Figure 6.3(c): Pareto front produced by MOCCA-P for ZDT3

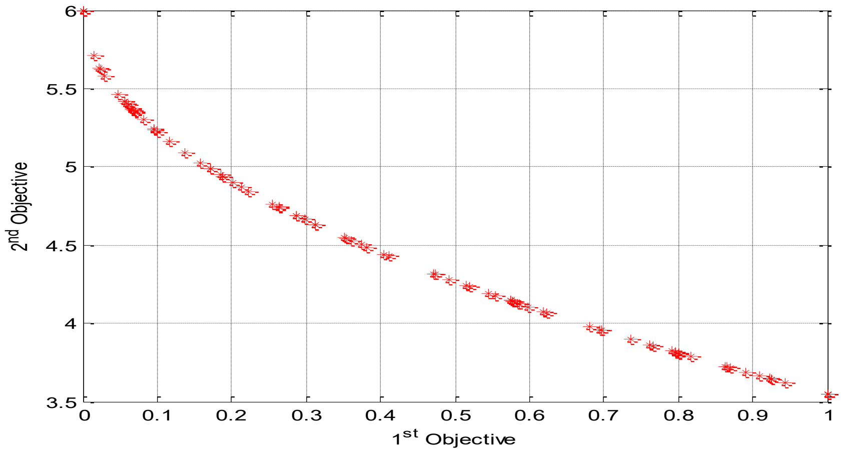


Figure 6.3(d): Pareto front produced by MOCCA-P for ZDT4

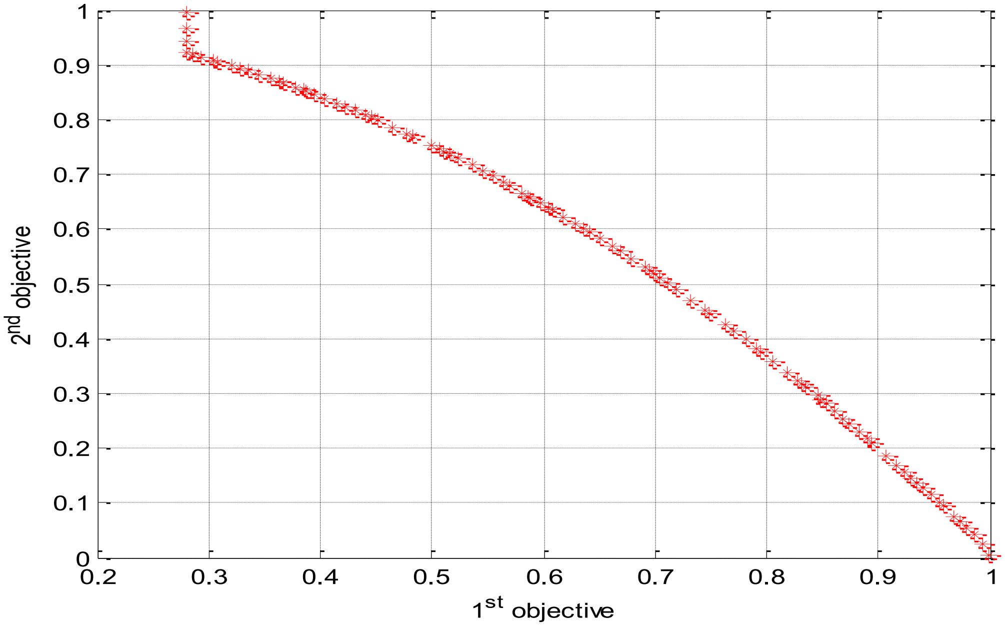


Figure 6.3(e): Pareto front produced by MOCCA-P for ZDT6

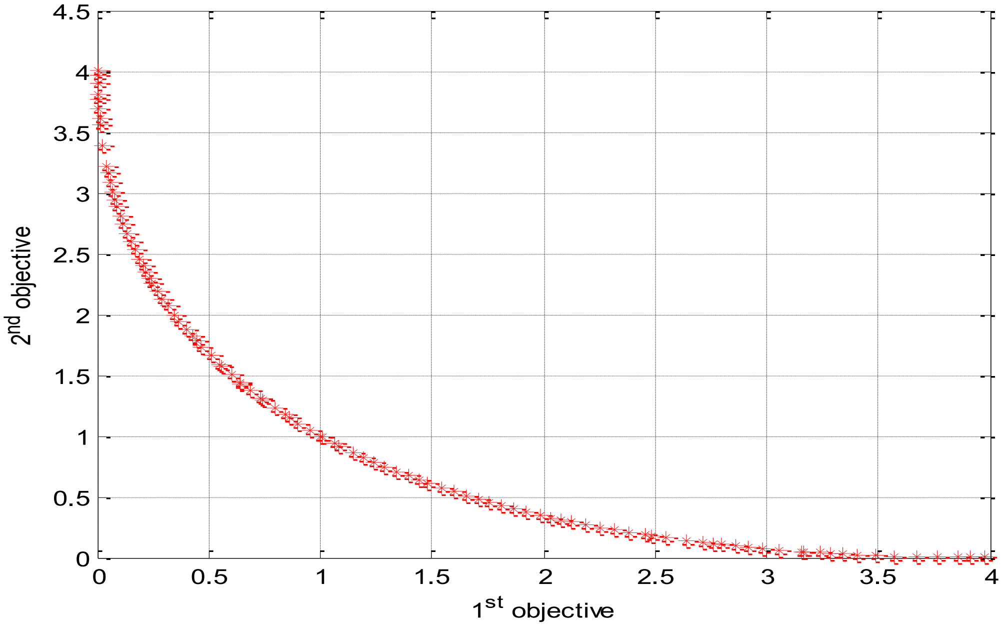


Figure 6.3(f): Pareto front produced by MOCCA-P for SCH

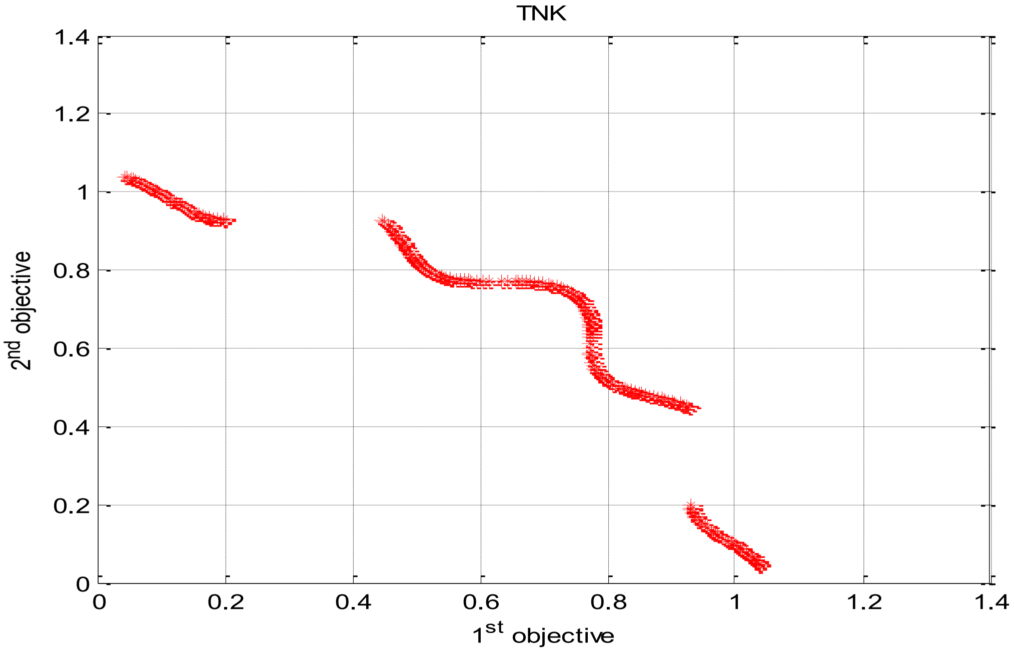


Figure 6.4(a): Pareto front produced by MOCCA-P for TNK

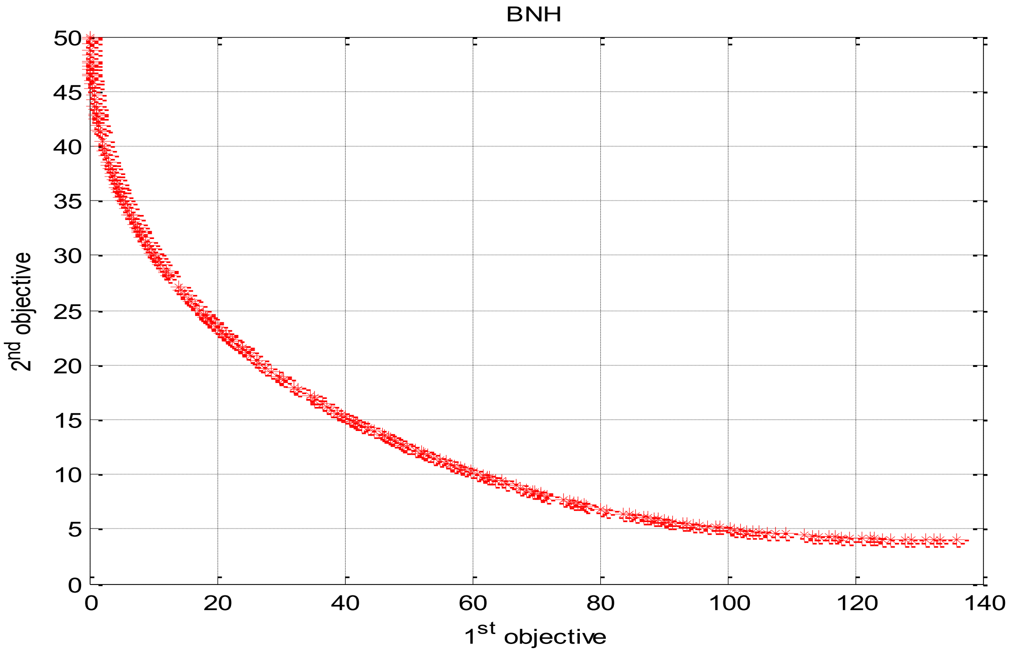


Figure 6.4(b): Pareto front produced by MOCCA-P for BNH

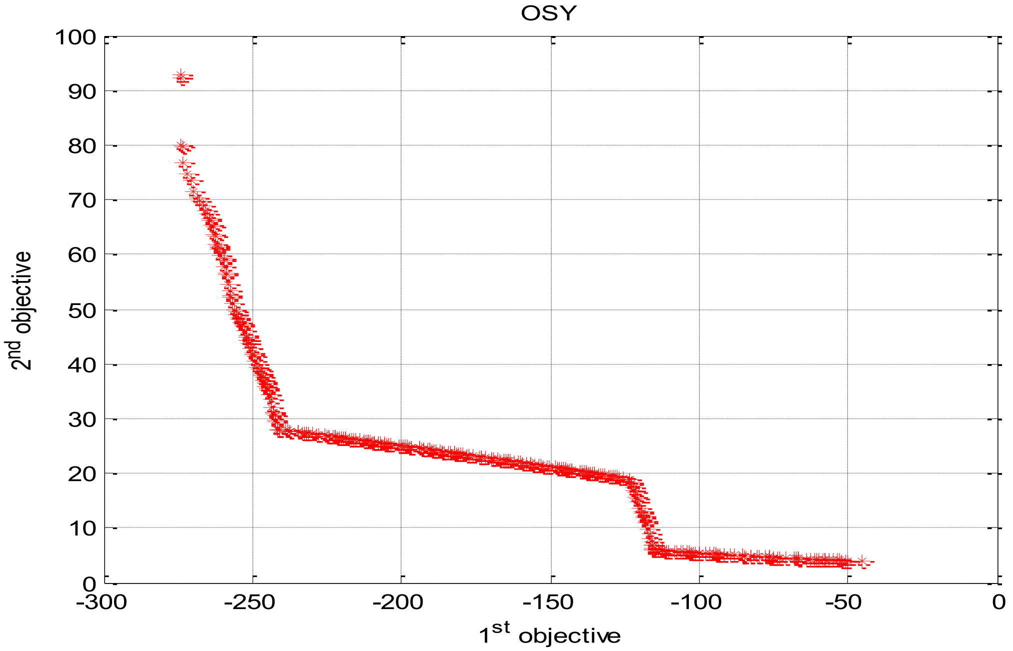


Figure 6.4(c): Pareto front produced by MOCCA-P for OSY

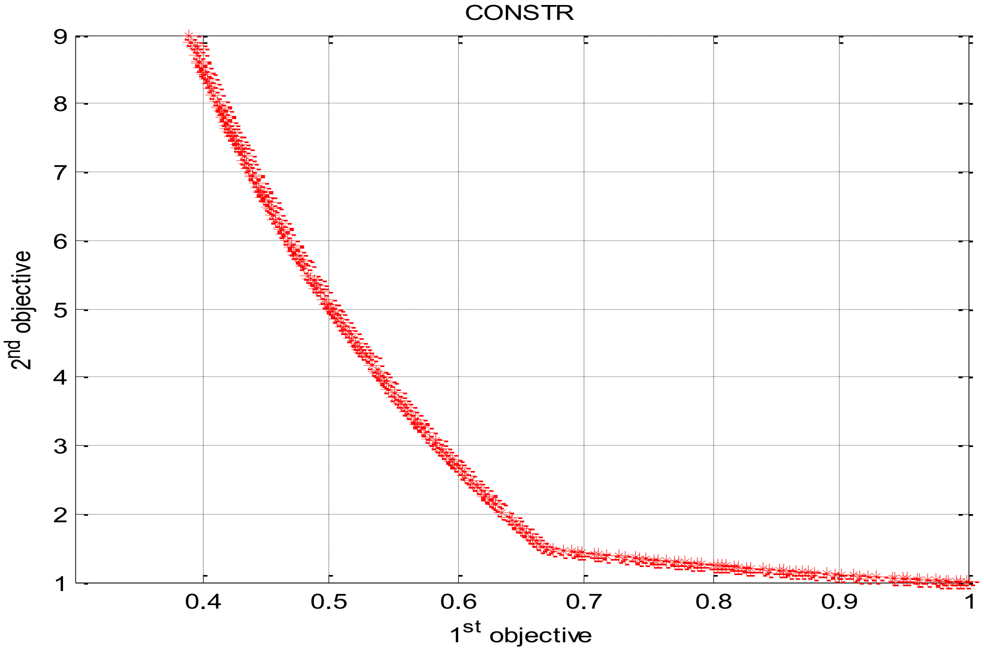


Figure 6.4(d): Pareto front produced by MOCCA-P for CONSTR

6.2.3 COMPARISON WITH OTHER ALGORITHMS

Having obtained the Pareto solutions for the various benchmark functions using MOCCA-P, it has to be compared with another such algorithm for its performance efficiency. Hence MOCCA-P is compared with a set of standard MOO algorithms such as SPEA2, NSGA-II, and MOPSO. The parameter values used in all MOO methods are given in table 6.7 and three performance metrics are considered for evaluating the experimental results and these are briefly explained in Section 6.2.2.2. Each algorithm is executed 50 times for every method and the statistical results of the performance metrics SP, MS and GD are reported in table 6.8, table 6.9 and table 6.10 respectively. From the statistical analysis, it is shown that the MOCCA performs better compared to its counterpart considering the performance metrics GD, SP, and MS.

Table 6.8 Comparison of MOCCA with other algorithms regarding the mean of SP

Problem	**MOPSO**	**SPEA2**	**NSGA-II**	**MOCCA**
ZDT1	0.312	0.02671	0.03128	**0.0011**
ZDT2	0.3167	0.1067	0.0187	**0.00406**
ZDT3	0.03299	0.0129	0.00856	**0.00299**
ZDT4	0.5776	0.03109	0.0189	**0.0058**
ZDT6	0.29	0.04899	0.01589	**0.00369**

Table 6.9 Comparison of MOCCA with other algorithms regarding the mean of MS

Problem	**MOPSO**	**SPEA2**	**NSGA-II**	**MOCCA**
ZDT1	0.9982	0.89986	0.9992	**1**
ZDT2	0.9862	0.8906	**1**	**1**
ZDT3	0.88927	0.98902	0.99018	**1**
ZDT4	0.9852	0.93058	0.9988	**0.99998**
ZDT6	0.8485	0.98905	**1**	**1**

Table 6.10 Comparison of MOCCA with other algorithms regarding the mean of GD

Problem	MOPSO	SPEA2	NSGA-II	MOCCA
ZDT1	0.0645	0.01809	0.01925	**0.001**
ZDT2	0.0523	0.1523	0.0053	**0.0013**
ZDT3	0.07853	0.03553	0.00607	**0.00238**
ZDT4	0.2609	0.23284	0.20244	**0.03001**
ZDT6	0.0513	0.0765	0.04394	**0.0017**

6.2.4 STATISTICAL ANALYSIS

To test the significance of the results produced by MOCCA-P, a statistical analysis using ANOVA has been carried out on the basis of SP, MS, and GD produced by various algorithms that are shown in table 6.8, 6.9 and 6.10 respectively. The analysis is done by considering a 5% significance level over the performance metrics produced by the algorithms corresponding to the test problems for the three different methods i.e. MOPSO, SPEA2 NSGA2 and MOCCA-P.

- **ANALYSIS I**

In this analysis the hypothesis is set as follows:

> ***Null hypothesis H_0:*** *There is no significant difference in the SP among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.*
>
> ***Alternative hypothesis H_1:*** *There is a significant difference in the SP among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.*

The one way ANOVA Test is done and the results found in the experiment are shown in table 6.11. Here, p values (Sig.= 0.000) is less than 0.05 that strongly oppose the null hypothesis. So it is concluded that there is a significant difference of the SP among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.

Table 6.11 ANOVA test over the methods MOCCA-P with MOPSO, SPEA2, NSGA2 based on SP

ANOVA

SP

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	.306	3	.102	10.558	.000
Within Groups	.154	16	.010		
Total	.460	19			

- **ANALYSIS II**

In this analysis the hypothesis is set as follows:

> ***Null hypothesis H_0:*** *There is no significant difference in the MS among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.*
>
> ***Alternative hypothesis H_1:*** *There is a significant difference in the MS among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.*

Table 6.12 ANOVA test over the methods MOCCA-P with MOPSO, SPEA2, NSGA2 based on MS

ANOVA

MS

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.017	3	.006	3.283	.048
Within Groups	.028	16	.002		
Total	.044	19			

The one way ANOVA test is performed and the results found in the experiment are shown in the table 6.12. From the ANOVA test shown in table 6.12, it is found that the p values (Sig.=0.048) are less than the significance value 0.05. So it is concluded that there is a significant difference in the MS among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.

- **ANALYSIS III**

In this analysis the hypothesis is set as follows:

Null hypothesis H_0: *There is no significant difference in the GD among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.*

Alternative hypothesis H_1: *There is a significant difference in the GD among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.*

Table 6.13 ANOVA test over the methods MOCCA-P with MOPSO, SPEA2, NSGA2 based on GD

ANOVA

GD

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.031	3	.010	1.782	.191
Within Groups	.093	16	.006		
Total	.124	19			

The one way ANOVA test is done and the results found in the experiment are shown in the table 6.13. From the results, it is shown that p values (Sig.=0.191) are greater than 0.05 (5% significance level). Therefore, the null hypothesis H_0 is accepted. So it is concluded that there is no significant difference of the GD among the methods MOPSO, SPEA2, NSGA2, and MOCCA-P.

6.3 SUMMARY

In this chapter, the CCA is extended for solving MOO problems in two ways i.e. MOCCA-W and MOCCA-P. The MOCCA is differing from the basic CCA in two terms. First, the male cricket is allowed to search the female cricket in the search space and secondly, when the male cricket chirps for aggression the winner is selected depends on the seven aggression levels. A different fitness calculation method is also developed and an external archive is used to retain the non-dominated solutions. The MOCCA is implemented and experimented with some of the standard benchmark test problems with constraint and without constraints and compared with three popular techniques i.e. MOPSO, SPEA2, and NSGA2. The experiment result shows better results compared to its counterparts in term of generational distance, spacing and maximum spread. The performance of the methods is statistically analyzed by using one way ANOVA test based on the SP, MS, and GD. Though the MOCCA-P shows better result corresponding to GD, there is no significant difference among the methods over the performance metrics GD, but shows significant difference among the methods over the performance metrics SP and MS.

Chapter 7

CASE STUDY

In this chapter, several case studies have been taken up to test the performance of CCA and MOCCA for real-world problems that involve single and multi-objective optimization. The Cricket Chirping Algorithm is applied in varied fields to test its performance and efficiency. The following case studies have been taken up in each category and tested.

A. For Single Objective Optimization
    - Engineering Design Optimization Problem
    - Multilevel Thresholding for Image Segmentation

B. For Multi-objective Optimization
    - Engineering Design Optimization Problem.

7.1 ENGINEERING DESIGN OPTIMIZATION USING CCA

Among the several real-world engineering problems, two standard optimization problems namely tension/compression spring design and welded beam design are considered in this research since it has been extensively used by other meta-heuristics algorithms like a benchmark problem.

7.1.1 SPRING DESIGN OPTIMIZATION PROBLEM (SDOP)

A mechanical engineering design problem namely spring design optimization is an important problem in engineering optimization. There are different types of springs based on their load force, shape etc. Two mostly used springs in Engineering are Tension and Compression spring where the first one is designed to wield with a tension load and the second one is designed to operate compression load. For this optimization problem, the objective is to minimize the tension or compression spring weight. It constitutes with the constraints of surge frequency, minimum deflection, shear stress, as well as limits on outside diameter and on design variables. It consists of three design variables namely the wire diameter d_1, the mean coil diameter d_2, and the number of active coils d_3. The mathematical statement of this problem given below is as described in [176],[143].

Minimize: $$f(x) = (d_3 + 2)d_2{d_1}^2 \tag{7.1}$$

Subject to,

$$g_1(x) = 1 - \frac{d_2^3 d_3}{7178 d_1^4} \leq 0$$

$$g_2(x) = 1 - \frac{140.45 d_1}{d_2^2 d_3} \leq 0$$

$$g_3(x) = \frac{d_2 + d_1}{1.5} - 1 \leq 0$$

$$g_4(x) = \frac{4d_2^2 - d_1 d_2}{12566(d_2 d_1^3) - d_1^4} + \frac{1}{5108 d_1^2} - 1 \leq 0$$

With the limits *$0.05 \leq d_1 \leq 2.0$, $0.25 \leq d_2 \leq 1.3$, $2.0 \leq d_3 \leq 15.0$.*

7.1.2 WELDED BEAM DESIGN OPTIMIZATION PROBLEM (WBDOP)

Another mechanical engineering design problem that is considered in this research is welded beam design problem. A rectangular beam is designed as a cantilever beam to carry a certain load with the minimum overall cost of fabrication [145], [143]. The objective of this problem is to minimize the cost, subject to a set of constraints on bending stress in the beam (σ), shear stress (τ), end reflection of the beam (Ω), buckling load on the bar P_C, and side constraints. The problem consists of four design variables: the width d_1 and length d_2 of the welded area, the depth d_3 and thickness d_4 of the main beam.

This problem can be mathematically formulated as equation 7.2.

Minimize

$$f(x) = 1.10471 d_1^2 d_2 + 0.04811 d_3 d_4 (14 + d_2) \tag{7.2}$$

Subject to

$$g_1(x) = d_1 - d_4 \leq 0,$$

$$g_2(x) = \delta(x) - 0.25 \leq 0,$$

$$g_3(x) = \tau(x) - 13600 \leq 0,$$

$$g_4(x) = \sigma(x) - 30000 \leq 0,$$

$$g_6(x) = 0.125 - d_1 \leq 0,$$

$$g_7(x) = 6000 - P_c(x) \leq 0,$$

$$g_5(x) = 0.10471 d_1^2 + 0.04811 d_3 d_4 \,(14 + d_2) - 5 \leq 0$$

Where,

$$\tau(x) = \sqrt{A^2 + (2AB)\frac{d_2}{2R} + B^2}$$

$$A = \frac{6000}{\sqrt{2d_1 d_2}}\ , B = \frac{MR}{J}\,,$$

$$M = 6000\left(14 + \frac{d_2}{2}\right),$$

$$R = \sqrt{\frac{d_2^2}{4} + \left(\frac{d_1+d_2}{2}\right)^2}\ \ ,$$

$$J = 2\left\{d_1 d_2 \sqrt{2}\left[\frac{d_2^2}{12} + \left(\frac{d_1+d_2}{2}\right)^2\right]\right\}$$

$$\sigma(x) = \frac{504000}{d_4 d_3^2}$$

$$\delta(x) = \frac{65{,}856{,}000}{(30\times10^6) d_4 d_3^3}$$

$$p_c(x) = \frac{4013(30\times10^6)\sqrt{\frac{d_3^2 d_2^6}{36}}}{196}\left(1 - \frac{d_3\sqrt{\frac{30\times10^6}{4(12\times10^6)}}}{28}\right)$$

With the range $0.1 \leq d_1 \leq 2.0,\ \ 0.1 \leq d_2 \leq 10,\ \ 0.1 \leq d_3 \leq 10,\ \ 0.1 \leq d_4 \leq 20.$

7.1.3 EXPERIMENTAL RESULTS AND ANALYSIS

The Cricket Chirping Algorithm is applied to solve the above mentioned two Engineering Design Optimization Problems. Results of CCA are compared with respect to the best results reported in literature i.e. Simple Constrained PSO (SiC-PSO) from [177] and

Cuckoo Search (CS) algorithm from [67]. The parameters used in our algorithm are cricket population n=20, temperature T_c=100 and aggression rate A_r=0.20. For cuckoo search algorithm the parameters are used as n=20 and p_a=0.25 and the results for SiC-PSO is taken from [67]

The problem SDOP gives optimal value within 16000 objective function evaluations and WBDOP required less than or equal to 8000 objective function evaluations. The table 7.1 and table 7.2 shows the best solution found by the proposed algorithm and its counterparts for both problems including a number of evaluations.

Table 7.1 Best solution for SDOP

	Best Solution		
	SiC-PSO	**CS**	**CCA**
d_1	0.051583	0.0518764	0.05179146
d_2	0.354190	0.361241	0.35918632
d_3	11.438675	11.0287	11.145769823
$g_1(x)$	-2.000E-16	-0.000005302	0.000006543
$g_2(x)$	-1.000E-16	-0.000002880	-0.000004109
$g_3(x)$	-4.048765	-4.06389202	-4.06129878
$g_4(x)$	-0.729483	-0.72411871	-0.725068250
f(x)	0.012665	0.012665	0.012665
Eval.	24,000	30,000	16,000

From the test results, it is found that the proposed algorithm is more powerful in terms of speed and accuracy. The SiC-PSO found the best result after 24,000 objective function evaluations for both problems and CS found the best solution after 30,000 evaluations for problem SDOP and 10,000 objective function evaluations for WBDOP. But the CCA required 16,000 evaluations for SDOP and 8000 evaluations for WBDOP to get the best solutions.

Table 7.2: Best solution for WBDOP

	Best Solution		
	SiC-PSO	**CS**	**CCA**
d_1	0.205729	0.17814	0.24027
d_2	3.470488	3.6573	3.9728
d_3	9.036624	8.7405	6.4223
d_4	0.205729	0.20385	0.26197
$g_1(x)$	-1.819E-12	-0.00000017	-0.0000001155
$g_2(x)$	-0.003721	-0.00000018	-0.00000032648
$g_3(x)$	0.000000	-0.00156325	0.00014037122
$g_4(x)$	-3.432983	-0.28420437	-2.5009783685
$g_5(x)$	-0.080729	-0.00000147	-0.0000343018
$g_6(x)$	-0.235540	-0.00000039	-0.0000019851
$g_7(x)$	0.000000	-1.89356441	-1.7266043670
f(x)	1.724852	1.72485	1.724801
Eval.	24,000	10,000	8,000

The algorithm is executed for 30 times and the mean and Standard Deviation (SD) of the best value is found. This value produced by the proposed algorithm is compared with SiC-PSO and CS. Table 7.3 shows the mean and SD with their respective evaluation numbers. The CCA obtained the optimal values for both the test problems in a lesser number of evaluations.

Table 7.3 Comparison of SiC-PSO, CS and CCA

Problems		**SiC-PSO**	**CS**	**CCA**
SDOP	**Mean**	0.0131	0.014742	**0.0126722**
	SD	4.1E-04	1.98E-03	**1.62E-05**
	Evaluation	24,000	30,000	**16,000**
WBDOP	**Mean**	2.0574	1.80649	**1.7482081**
	SD	0.2154	0.333653	**0.2880167**
	Evaluation	24,000	10,000	**8,000**

7.2 MULTILEVEL THRESHOLDING FOR IMAGE SEGMENTATION USING CCA

Image processing plays a crucial role in different fields such as medical discipline, industry, agriculture, navigation, environment modeling, automatic event detection, surveillance, texture and pattern recognition, damage detection etc. It is motivated by three major applications like pictorial information improvement for human perception, image processing in the case of autonomous machine application and efficient storage and transmission. One of the major and primary tasks in image processing is image segmentation. Image segmentation is the partitioning of an image into multiple sets of pixels or segments or regions that share some common characteristics such as color or intensity or similarity or discontinuity etc.

In this section, Cricket Chirping Algorithm is combined with Kapur's Entropy Criterion method and Otsu's between-class variance method and applied in multi-level thresholding for image segmentation. In this process, a random solution is taken place from the feasible search space inside the image histogram. The fitness of the solution is evaluated by considering the objective functions, Kapur's and Otsu's method. Directed by this objective value, the set of candidate solutions are adapted using the CCA operators and search for the optimal solution in the search space proceeds.

7.2.1 MULTILEVEL THRESHOLDING (MT)

In thresholding process [147], the pixel of the grayscale image is divided into groups based on the intensity level $l.$ To make the grouping it is necessary to choose a threshold value (θ) following some simple rules as follows:

$$\begin{aligned} G_0 &\leftarrow p, \qquad if\ 0 \leq p \leq \theta, \\ G_1 &\leftarrow p, \qquad if \theta \leq p \leq l-1, \end{aligned} \tag{7.3}$$

where p is one of the $m \times n$ pixels of the grayscale image I_g and it can be represented in 'l' grayscale levels $l = \{0, 1, 2, \dots, l-1\}$ and G_1 and G_2 are the groups in which the pixel namely p, can be located. This can be extended for multilevel thresholding as follows [67]:

$$\left.\begin{array}{ll} G_0 \leftarrow p, & if\ 0 \leq p \leq \theta_1, \\ G_1 \leftarrow p, & if \theta_1 \leq p \leq \theta_2, \\ \ldots\ldots & \ldots\ldots\ \ldots\ldots \\ G_i \leftarrow p, & if \theta_i \leq p \leq \theta_{i+1}, \\ \ldots\ldots & \ldots\ldots\ \ldots\ldots \\ G_n \leftarrow p, & if \theta_n \leq p \leq l-1, \end{array}\right\} \quad (7.4)$$

where $\{\theta_1, \theta_{2,} \ldots\ldots., \theta_{i+1}, .., \theta_n\}$ are different thresholds. Here, the main objective is to select the *θ* values that identify the classes correctly for both bi-level and multi-level thresholding. The popular thresholding method, Otsu's and Kapur's methods are generally applied for identifying such values where the objective function is to maximize to find the optimal threshold values. The details of the objective functions proposed by them are given in the subsequent sections.

➢ KAPUR'S METHOD (ENTROPY CRITERION METHOD)

One of the most popular thresholding methods, the Kapur's method, was developed based on the entropy and the probability distribution of the image histogram. It is also known as the Entropy Criterion Method [177], [178]**.** This method is intended to obtain the optimal threshold value *θ* that exploits the overall entropy of an image that processes the density and separability among classes or groups. Incidentally, when the optimal *θ* value appropriately separates the classes the entropy has the maximum value. The objective function of the Kapur's problem for bi-level example can be defined by using equation 7.5.

$$\left.\begin{array}{l} \mathrm{I}(\theta) = \mathrm{H}_1^{\mathrm{c}} + \mathrm{H}_2^{\mathrm{c}}\ , \\ \mathrm{c} = \begin{cases} 1,2,3 & \text{if RGB image} \\ 1, & \text{if Grayscale image} \end{cases} \end{array}\right\} \quad (7.5)$$

Where, $\mathrm{H}_1^{\mathrm{c}}$ and $\mathrm{H}_2^{\mathrm{c}}$ are entropies and computed using the following model

$$\left.\begin{array}{l} \mathrm{H}_1^{\mathrm{c}} = \sum_{i=1}^{thr} \frac{P_i^c}{w_0^c} \ln\left(\frac{P_i^c}{w_0^c}\right), \end{array}\right\} \quad (7.6)$$

$$\mathrm{H}_2^{\mathrm{c}} = \sum_{thr=1}^{l} \frac{P_i^c}{w_1^c} \ln\left(\frac{P_i^c}{w_1^c}\right)$$

where, P_i^c is the probability distribution of the intensity levels which is gained using equation 7.9, ${w_0}^c$ and ${w_1}^c$ are probability distributions for c_1 and c_2 .

For multiple threshold values, the image is divided into 'k' classes using a similar number of thresholds. Then a new objective function can be defined as follow:

$$\left.\begin{aligned} &F_{kapur}(\mathrm{TH}) = \max\left(\textstyle\sum_{\mathrm{i}}^{\mathrm{k}} \mathrm{H}_{\mathrm{i}}^{\mathrm{c}}\right) \ , \\ &\mathrm{c} = \begin{cases} 1,2,3 & \text{if RGB image} \\ 1, & \text{if Grayscale image} \end{cases} \end{aligned}\right\} \tag{7.7}$$

Where TH = $[\theta_1,\theta_2,...,\theta_{k-1}]$ is a vector that has multiple thresholds. Each entropy is computed separately with its corresponding θ value. It is expanded for k entropies as follows:

$$\left.\begin{aligned} &\mathrm{H}_1^{\mathrm{c}} = \textstyle\sum_{i=1}^{\theta_1} \frac{P_i^c}{w_0^c} \ln\left(\frac{P_i^c}{w_0^c}\right), \\ &\mathrm{H}_2^{\mathrm{c}} = \sum_{i=\theta_1+1}^{\theta_2} \frac{P_i^c}{w_1^c} \ln\left(\frac{P_i^c}{w_1^c}\right), \\ &\qquad \cdots \qquad \cdots \\ &\qquad \cdots \qquad \cdots \\ &\mathrm{H}_{\mathrm{k}}^{\mathrm{c}} = \sum_{i=\theta_{k+1}+1}^{\theta} \frac{P_i^c}{w_{k-1}^c} \ln\left(\frac{P_i^c}{w_{k-1}^c}\right) \end{aligned}\right\} \tag{7.8}$$

Where, $\omega_0(\theta) = \sum_{i=1}^{\theta_1} P_i,\ \omega_1(\theta) = \sum_{i=\theta_1+1}^{\theta_2} P_i . . . .\omega_{k-1}(\theta) = \sum_{i=\theta_{k+1}}^{l} P_i$

➢ OTSU'S METHOD (BETWEEN-CLASS VARIANCE)

Another thresholding method, that incorporates between-class variance, has been propounded by Otsu [178]. It is non-parametric and unsupervised. Maximum variance is estimated for all the classes. Based on this value, image segmentation is carried out.

Considering l intensity levels from a grayscale image or from each RGB component (red, green, and blue) image, the probability distribution of the intensity values is computed as follows:

$$\left.\begin{array}{l} P_i^c = \frac{h_i^c}{NP} \quad , \quad \sum_{i=1}^{NP} P_i^c = 1 \\ c = \begin{cases} 1, 2, 3 & ifRGBimage \\ 1, & ifGrayscaleimage \end{cases} \end{array}\right\} \tag{7.9}$$

where,

i is a specific intensity level $(0 \leq i \leq l - 1)$,

c is the component of the image which depends on the image type (grayscale or RGB),

NP is the total number of pixels in the image.

h_i^c (Histogram) is the number of pixels that corresponds to the *i* intensity level in c.

The histogram is normalized within a probability distribution P_i^c

For bi-level segment, that is the simplest segmentation, two classes are defined as:

$$\left.\begin{array}{l} G_1 = \frac{P_1^c}{\omega_0^c}, \dots \dots \dots \dots \dots, \frac{P_\theta^c}{\omega_0^c} \\ G_2 = \frac{P_{\theta+1}^c}{\omega_1^c}, \dots \dots \dots, \frac{P_l^c}{\omega_1^c} \end{array}\right\} \tag{7.10}$$

Where, w_0^c and w_1^c are probabilities distributions for G_1 and G_2 , as it is shown below

$$\omega_0^c = \sum_{i=1}^{\theta} P_i^c, \quad \omega_1^c = \sum_{i=\theta+1}^{l} P_i^c \tag{7.11}$$

The mean level is calculated as follows:

$$\mu_0 = \sum_{i=1}^{\theta} \frac{iP_i^c}{{\omega_0}^c} \mu_1 = \sum_{i=1}^{\theta} \frac{iP_i^c}{{\omega_1}^c} \tag{7.12}$$

Then Otsu variance between classes $\sigma^2$is calculated as follows:

$$\sigma^2 = \sigma_1 + \sigma_2 \tag{7.13}$$

Where, σ_1 and σ_2 are variances of G_1 and G_2 that is calculated as follows:

$$\sigma_1 = \omega_0^c(\mu_0 + \mu_T)^2 \sigma_2 = \omega_1^c(\mu_1 + \mu_T)^2 \tag{7.14}$$

Where, $\mu_T = \omega_0^c\mu_0 + \omega_1^c\mu_1$ and $\omega_0^c + \omega_1^c$=1

Based on the values σ_1 and σ_2 the objective function is shown as below:

$$F_{Otsu}(\theta) = \max(\sigma^2(\theta)), 0 \leq \theta \leq l - 1 \tag{7.15}$$

Where σ^2 is the Otsu's variance for a given θ value. The optimization problem is reduced to find the intensity levels *(θ)* that maximizes equation (7.15). This equation can be rewritten for multiple threshold value as follows:

$$\left.\begin{array}{l} F_{Otsu}(X) = \max(\sigma^2(X)), \\ 0 \leq \theta_i \leq l - 1, \quad i = 1,2, \dots k \end{array}\right\} \tag{7.16}$$

Where X= $[\theta_0, \theta_1, \theta_2, \dots\dots, \theta_{k-1}]$ is a vector containing thresholds and the variances are computed as:

$$\sigma^2 = \sum_{i=1}^{k} \sigma_i = \sum_{i}^{k} \omega_i(\mu_i + \mu_T)^2$$

Here *i* represent a specific class. ω_i and μ_i are the probability of occurrence and the mean of a class, respectively.

For MT such values are obtained as follows:

$$\left.\begin{array}{l} \omega_0(\theta) = \sum_{i=1}^{\theta_1} P_i \\ \omega_1(\theta) = \sum_{i=\theta_1+1}^{\theta_2} P_i \\ \quad\quad \dots. \\ .\omega_{k-1}(\theta) = \sum_{i=\theta_{k+1}}^{l} P_i \end{array}\right\} \tag{7.17}$$

And for mean values,

$$\mu_0 = \sum_{i=1}^{\theta_1} \frac{iP_i}{\omega_0(\theta_1)}$$

$$\mu_1 = \sum_{i=\theta_1+1}^{\theta_2} \frac{iP_i}{\omega_0(\theta_2)}$$

$$\ldots\ldots\ldots..$$

$$\mu_{k-1} = \sum_{i=\theta_k+1}^{l} \frac{iP_i}{\omega_1(\theta_k)}$$

7.2.2 MULTI-LEVEL THRESHOLDING USING CRICKET CHIRPING ALGORITHM

In this section, the cricket chirping algorithm is utilized to find the optimal threshold values for multilevel thresholding problem in image segmentation. The problem is viewed as an optimization problem and the methodology to apply CCA for optimizing threshold values is presented. Here the algorithm is implemented considering the two objective functions namely Otsu's between class variance and Kapur's entropy criteria. The segmentation problem is represented as an optimization problem as given below:

$$\begin{array}{ll} \text{Maximize,} & F(x), \\ & x = \left[\int_{kapur}(TH)\, or \int_{Otsu}(TH)\right] \\ & TH = [\theta_1, \theta_2, \ldots\ldots, \theta_k] \\ \text{Subject to,} & TH \in X \end{array} \tag{7.18}$$

Where, $X = \{TH \in \theta_i | 0 \leq \theta_i \leq 255,\ i = 1, .. k\}$ refers to the bounded feasible region constrained within the interval [0-255].

In cricket population, k various decision variables are adopted by each individual. Each represents a different threshold point θ, which will be employed for segmenting the image. The entire population of cricket is expressed as $S = [TH_1, TH_2 \ldots\ldots.., TH_N]$, $TH_i = [\theta_{1,}\theta_{2,} \ldots\ldots, \theta_k]^T$ with boundary search space lb=0 and ub=255. N signifies the cardinality of the entire cricket population. The implementation of the proposed method is summarized in table 7.4.

The performance of CCA depends on the parameter settings. In this implementation, we set the aggression rate to 0.45, the mating rate to 0.80 and population size to 50 as shown in table 7.5. The temperature of outside environment is fixed to 100-degree Celsius. The female population *k* is chosen randomly such that *k≤n/2* and the one point crossover for mating.

Table 7.4: Algorithm for multilevel image segmentation using CCA

```
Image_seg_cca( )
Begin
    1. Read the image $I_g$ or$I_{rgb.}$
    2. Get the histograms, for RGB images $h_r$, $h_g$, $h_b$ and for grayscale images $h_{gr}$.
    3. Calculate the probability distribution with equation 7.9 and obtain the histograms
    4. Initialize the CCA parameters: T, k, $C_r$, $P_{aggr}$
    5. Initialize the cricket.
    6. Evaluate the fitness using Kapur's or Otsu's methods.
    7. While (i<max iter or stopping criteria not met)
        a. Allow the cricket to chirp for mating.
            i. Compute $C_n$, $v_i$ and $st_i$ using equation (4.3), (4.4) and (4.5).
            ii. Mate with female cricket
        b. Allow the cricket to chirp for aggression
            i. Fight with other male cricket
        c. Calculate the fitness using Kapur's or Otsu's methods.
        d. Select the cricket with the best fit objective value.
    8. While end
    9. Apply the thresholds values contained in best to the image Ig  or Irgb.
End
```

Table 7.5: Control parameters of CCA

Population Size(n)	Female pop. Size(k)	Aggression rate	Mating rate
50	25	0.45	0.80

7.2.3 EXPERIMENTAL RESULTS AND ANALYSIS

The proposed CCA algorithm for multi-level thresholding has been tested for 8 benchmark test images which are in JPEG format and provided by the Berkeley segmentation dataset [179], [180]. The experiments are carried out with an Intel® Core(TM) i5 CPU and 4 GB RAM with Windows 7 Operating system. The algorithms were implemented using MATLAB. The required parameters are set as in table 7.5 for all the test images. The program was run 50 times for each image separately and the performance of the proposed method is evaluated using the well-known parameters such as peak signal to noise ratio (PSNR) and structural similarity indices (SSIM). PSNR is used to assess the similarity of the segmented image against a reference image (original image) based on θ. The PSNR is defined as equation 7.19.

$$\left.\begin{aligned} PSNR &= 20\log_{10}\left(\frac{255}{\sqrt{MSE}}\right), \\ MSE &= \frac{\sum_{i=1}^{r}\sum_{j=1}^{c}[I_o(i,j)-I_s(i,j)]^2}{r\times c} \end{aligned}\right\} \tag{7.19}$$

Where, I_o is the original image and I_s is the segmented image, r and c are the total number of rows and columns of the image respectively.

The Structural SIMilarity (SSIM) index is a full reference metric used for measuring the similarity between two images [67]. It is the measurement or prediction of image quality based on an initial uncompressed or distortion-free image as reference [145]. SSIM evaluates the visual similarity between the original image x and the segmented image y and of common size $N\times N$ is

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \tag{7.20}$$

Where μ_x is the average of x , μ_y is the average of y , σ_x^2 the variance of x , σ_y^2 the variance of y , σ_{xy} the covariance of x and y , $C_1 = (k_1 l)^2$, $C_2 = (k_2 l)^2$ two variables to stabilize the division with the weak denominator, l the dynamic range of the pixel-values (typically this is $2^{\#bitsperpixel} - 1$); $k_1 = 0.01$ and $k_2 = 0.03$ by default.

7.2.3.1 IMAGE SEGMENTATION USING CCA AND KAPUR'S METHOD

In this section, the CCA is executed considering Kapur's Entropy method as the objective function that is given in equation 7.7. For the complete set of benchmark images, this method is applied by considering four different threshold points 2,3,4,5 and the results like PSNR, SSIM, execution time etc. are calculated. The original image and segmented image with different threshold values and their histogram and fitness graph are shown in table 7.6. From the results, it is shown that the PSNR and SSIM values increase their magnitude as the number of threshold points increases. The best results found using CCA with Kapur's function in the 50 runs for each image with the different threshold value is shown in table 7.7.

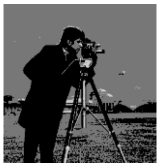
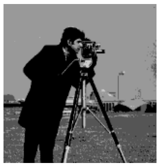
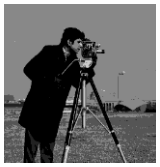
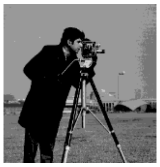
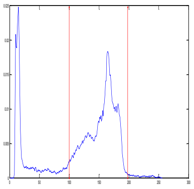
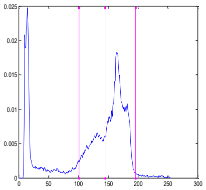
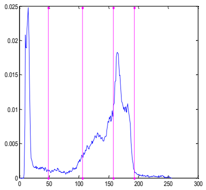
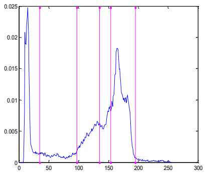
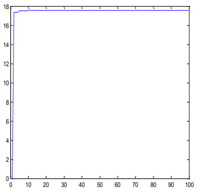
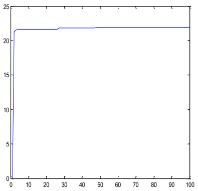
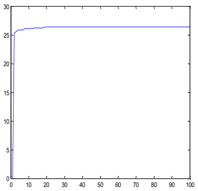
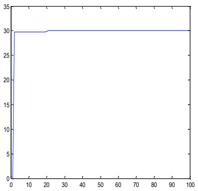
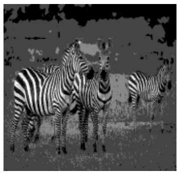
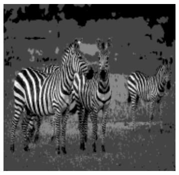
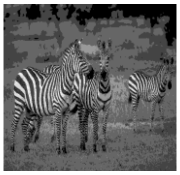
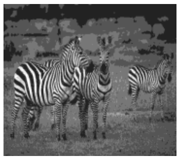
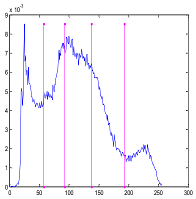
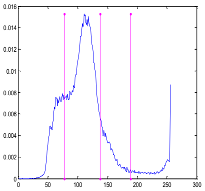
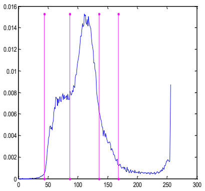
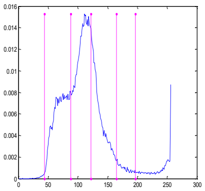
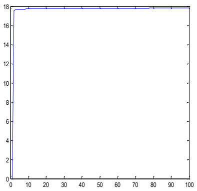
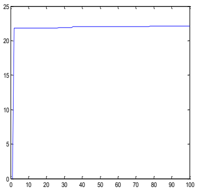
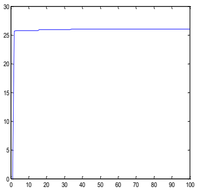
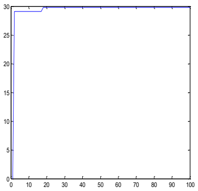
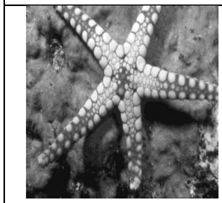
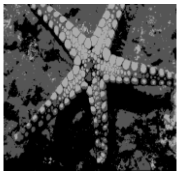
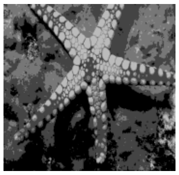
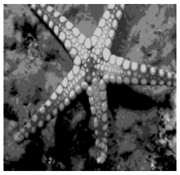
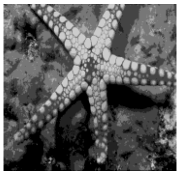
7.2.3.2 IMAGE SEGMENTATION USING CCA AND OTSU'S METHOD

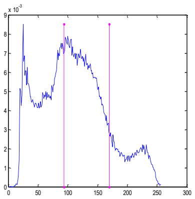
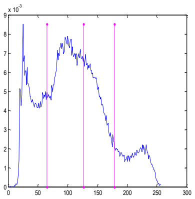
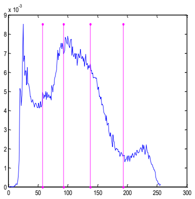
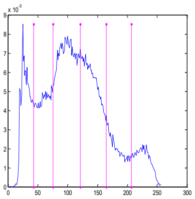
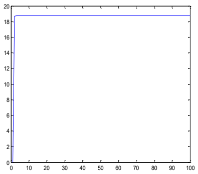
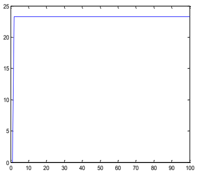
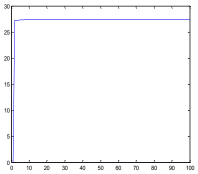
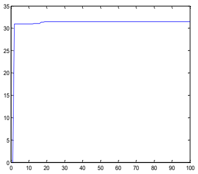
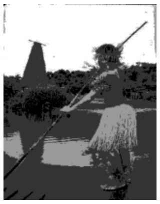
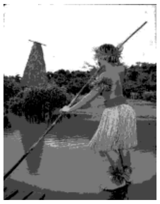
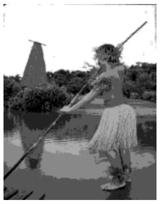
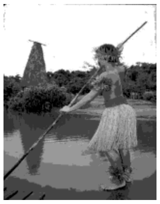
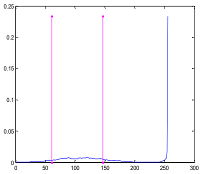
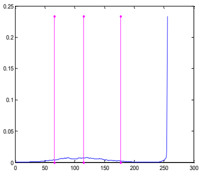
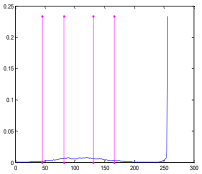
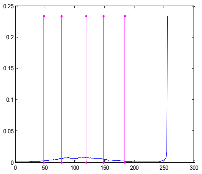
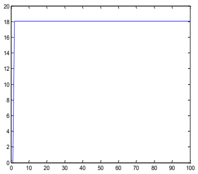
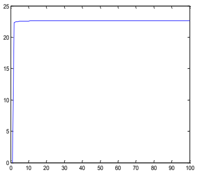
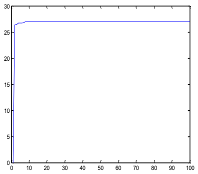
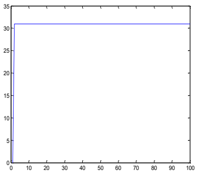
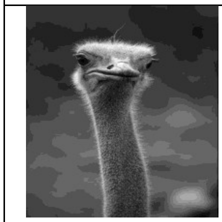
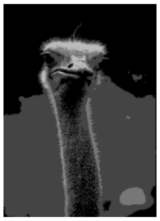
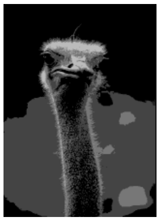
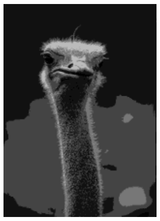
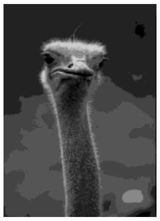
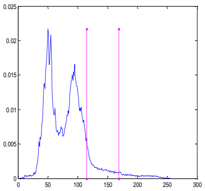
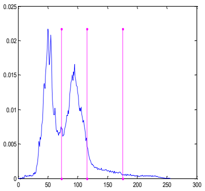
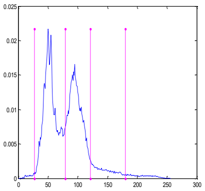
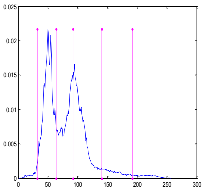
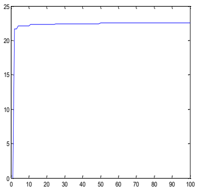
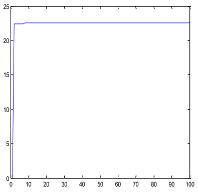
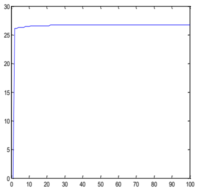
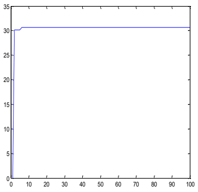
In this section, the CCA is executed considering Otsu's between class variances method as the objective function that is given in equation 7.15. This is applied over the whole set of benchmark images considering four different threshold points 2,3,4,5 and the best results like PSNR, SSIM, time etc. are stated in table 7.9 and the segmented image, histogram, and fitness graph is shown in table 7.8. From the results, it is shown that the PSNR and SSIM values increment their magnitude when there is an increment in the number of threshold points.

7.2.4 COMPARISONS AND STATISTICAL ANALYSIS

In this section, three different comparisons are done to analyze the performance of the CCA. The first comparison is executed between the two versions of CCA, via the Kapur function and Otsu's criterion. Secondly, examines the comparison among the CCA with the other meta-heuristics algorithm. The third one is the statistical analysis of the obtained results of CCA and MTEMO to validate its performance and computational effort. All the algorithms are executed 50 times over each selected image. For each image, the PSNR, SSIM, Time and the mean of the objective function values are calculated. The complete test is performed using both Otsu`s and Kapur`s objective functions.

Table 7.6: Resultant images after applying the CCA using Kapur's Function

Original image	Θ =2	Θ=3	Θ=4	Θ=5
 Cameraman				
Histogram				
Fitness Graph				
 Zebra				
Histogram				
Fitness Graph				
 Sea Fish				

Histogram				
Fitness Graph				
 Boat man				
Histogram				
Fitness Graph				
 Ostrish				
Histogram				
Fitness Graph				

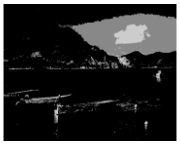
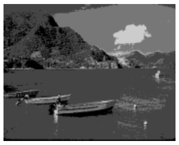
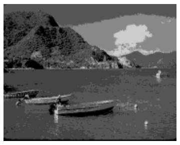
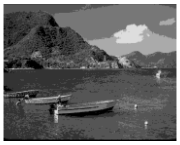
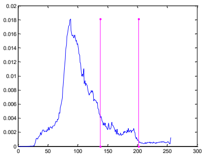
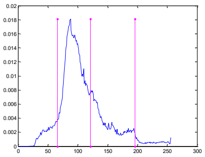
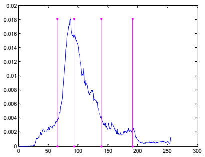
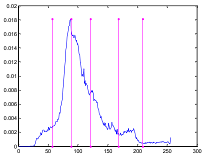
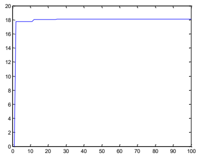
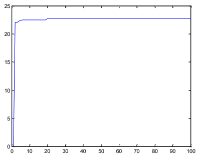
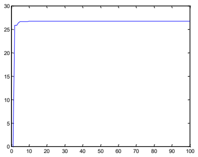
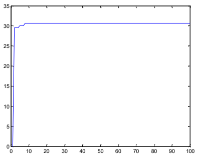
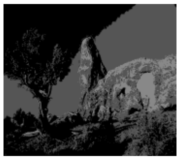
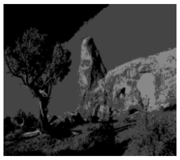
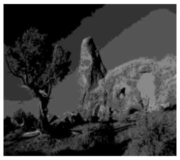
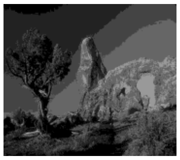
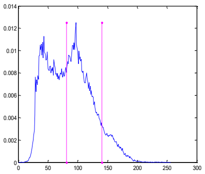
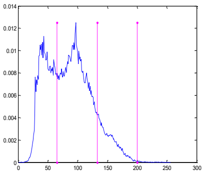
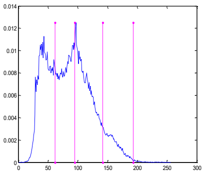
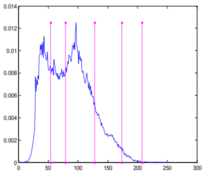
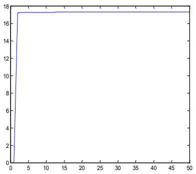
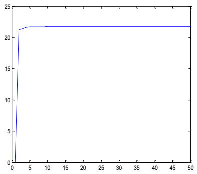
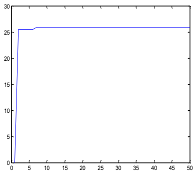
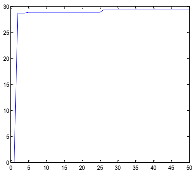
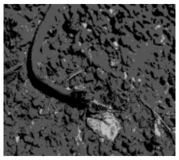
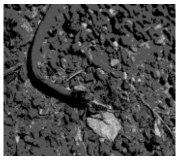
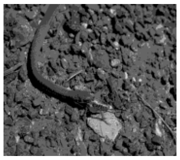
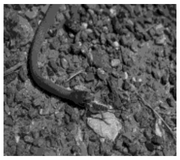
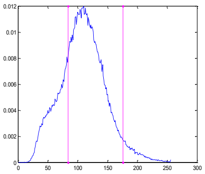
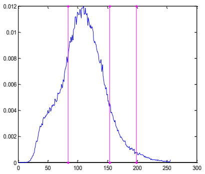
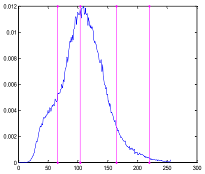
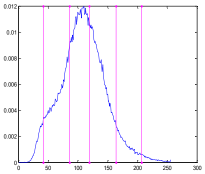
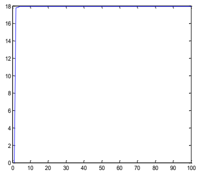
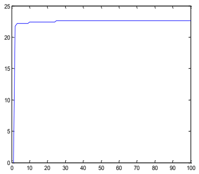
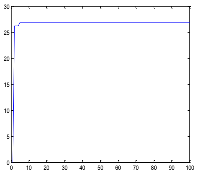
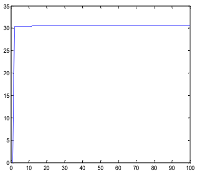
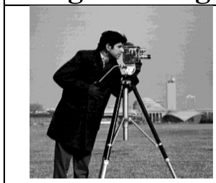
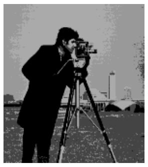
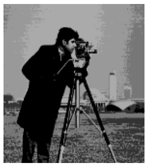
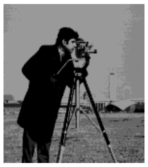
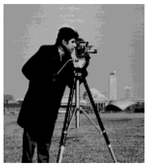
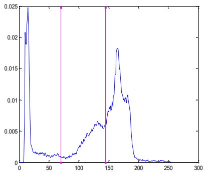
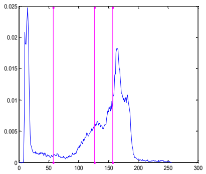
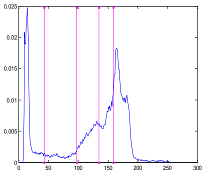
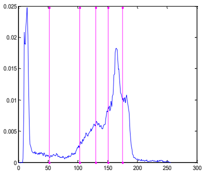
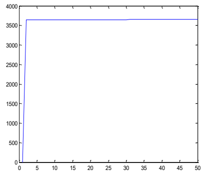
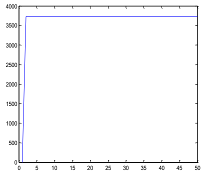
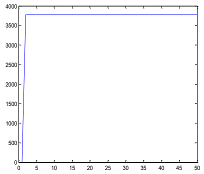
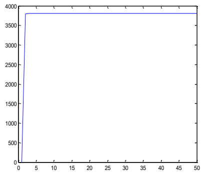
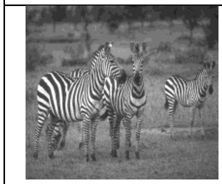
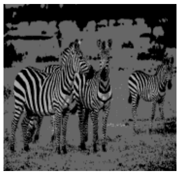
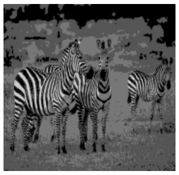
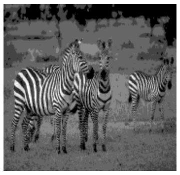
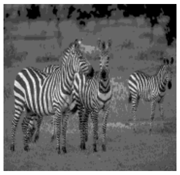
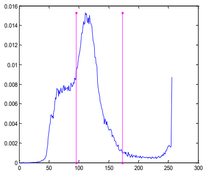
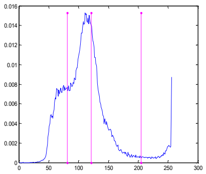
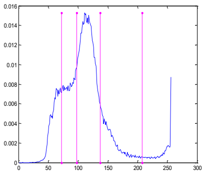
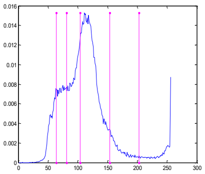
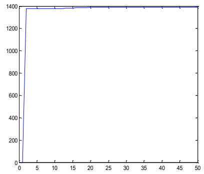
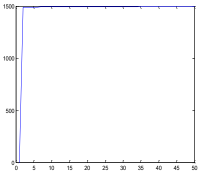
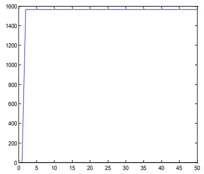
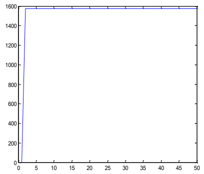
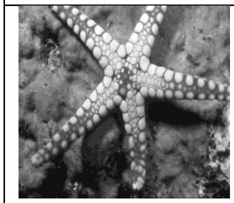
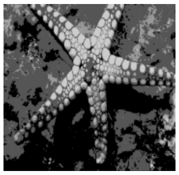
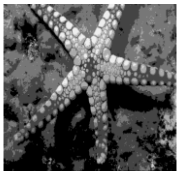
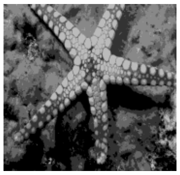
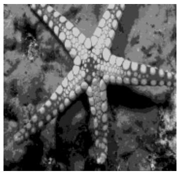
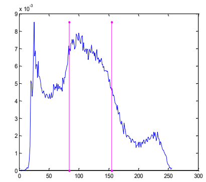
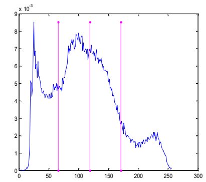
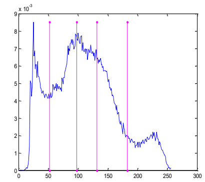
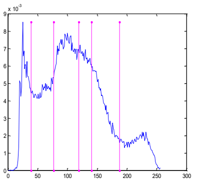
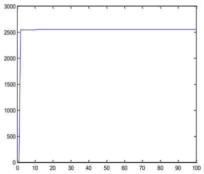
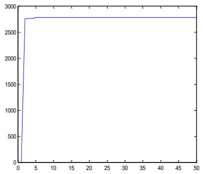
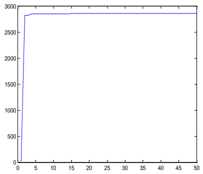
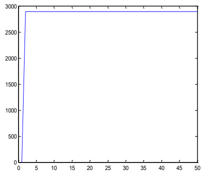
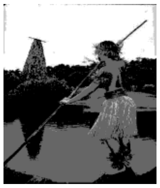
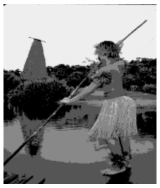
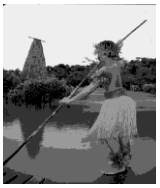
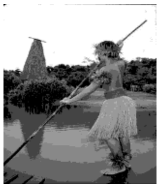
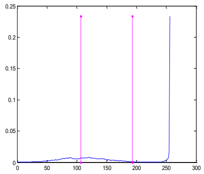
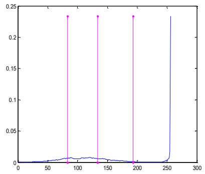
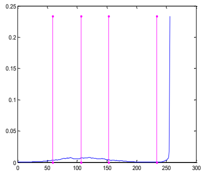
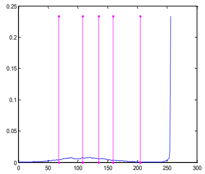
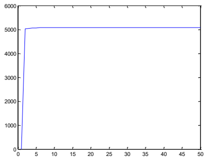
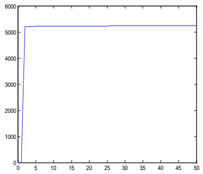
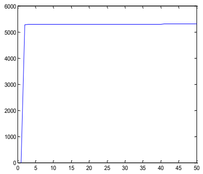
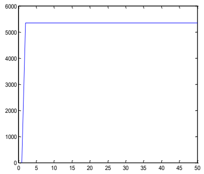
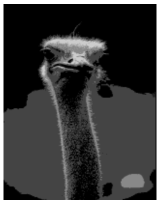
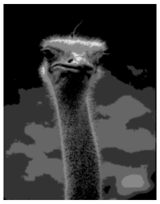
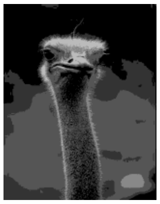
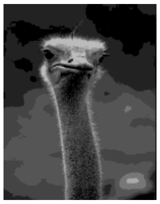
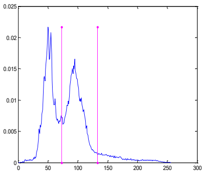
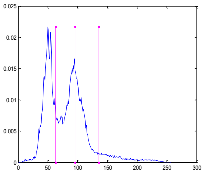
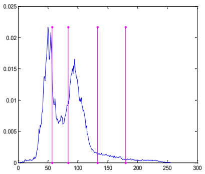
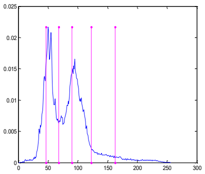
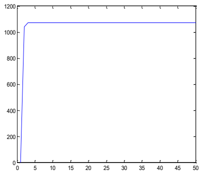
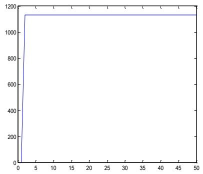
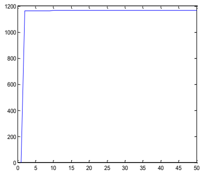
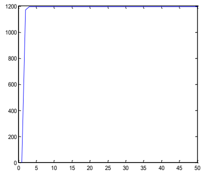
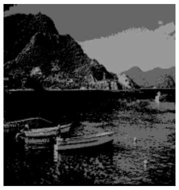
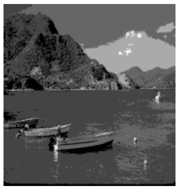
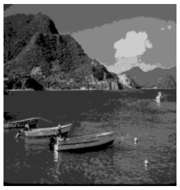
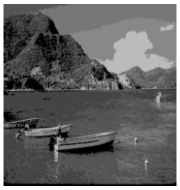
 Boat				
Histogram				
Fitness Graph				
 Tree				
Histogram				
Fitness Graph				
 Snake				
Histogram				
Fitness Graph				

Table 7.7: Best Results after applying the CCA using Kapur's function

Image	Ɵ	Best fitness	PSNR	SSIM	Time(sec)	Threshold value
Cameramen	2	17.5526	14.1796	0.5394	0.4604	120 193
	3	20.8251	17.9649	0.6351	0.5469	100 137 216
	4	25.8039	18.0425	0.6310	0.6300	36 82 155 199
	5	30.2333	21.8695	0.7037	0.7115	31 91 126 149 194
Star Fish	2	18.7536	14.4390	0.4081	0.4665	90 170
	3	23.2861	17.0826	0.5248	0.5484	65 127 179
	4	27.4114	19.2181	0.6367	0.6410	58 93 138 193
	5	31.4256	20.4641	0.7104	0.7221	43 76 121 165 207
Zebra	2	17.8065	14.4871	0.4552	0.4592	92 160
	3	22.1330	16.0182	0.5352	0.5602	78 138 189
	4	25.8618	19.1451	0.5639	0.6451	44 87 136 168
	5	29.8213	21.2276	0.6913	0.7214	44 88 122 165 197
Boat man	2	18.0242	16.0707	0.6066	0.4620	61 147
	3	22.6285	19.2583	0.7321	0.5506	66 115 177
	4	26.9793	21.1183	0.7657	0.6339	45 82 131 166
	5	30.9901	22.6232	0.7820	0.7164	48 78 119 148 184
Ostrich	2	22.5240	16.2691	0.0771	0.4633	64 125 186
	3	22.5229	16.1868	0.4785	0.5496	73 116 176
	4	26.7208	19.9436	0.7018	0.6325	27 79 121 180
	5	30.4907	22.7053	0.7524	0.7199	32 64 92 141 192
Boat	2	18.0903	9.2283	0.1788	0.4734	138 202
	3	22.7516	17.5315	0.5702	0.5494	66 121 196
	4	26.7387	19.8912	0.6812	0.6466	65 94 139 192
	5	30.8757	21.2346	0.7143	0.7208	57 89 121 168 209
Tree	2	17.2734	15.5238	17.2734	0.2463	81 141
	3	21.7707	16.5785	21.7707	0.2905	68 127 196
	4	25.6561	18.6202	25.6561	0.3299	57 102 141 187
	5	29.2947	21.4849	29.2947	0.3922	40 79 104 149 201
Snake	2	17.9342	14.6194	0.5222	0.4603	84 176
	3	22.5789	15.4028	0.5948	0.5548	84 154 199
	4	26.7863	18.4103	0.7405	0.6347	66 104 165 220
	5	30.5564	21.0050	0.8358	0.7187	42 86 119 164 207

Table 7.8: Resultant images after applying the CCA using Otsu's function

Original image	Θ=2	Θ=3	Θ=4	Θ=5
 Cameraman				
Histogram				
Fitness Graph				
 Zebra				
Histogram				
Fitness Graph				
 Sea Star				
Histogram				

Fitness Graph				
 Boatman				
Histogram				
Fitness Graph				
 Ostrish				
Histogram				
Fitness Graph				
 Boat				

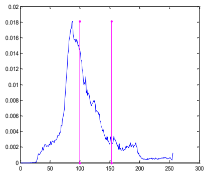
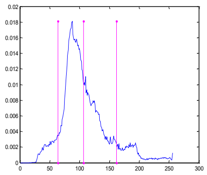
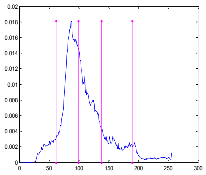
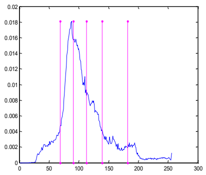
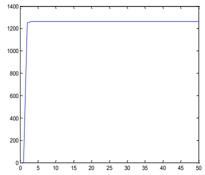
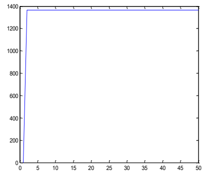
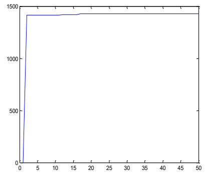
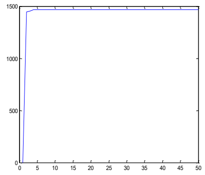
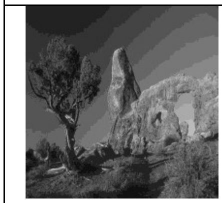
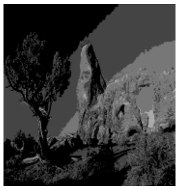
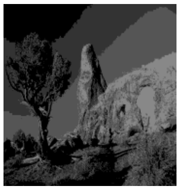
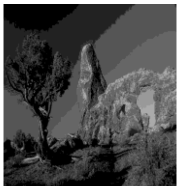
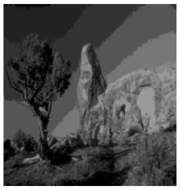
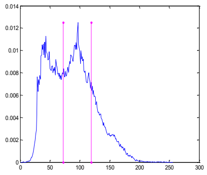
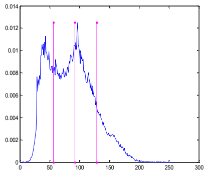
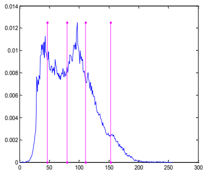
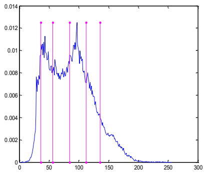
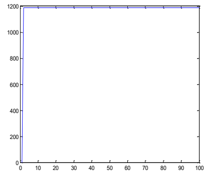
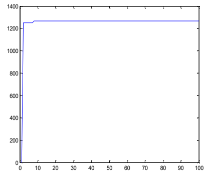
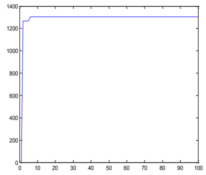
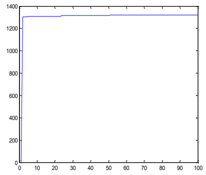
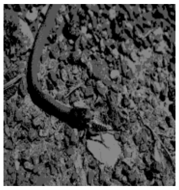
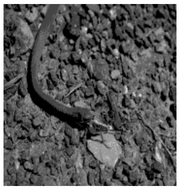
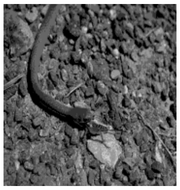
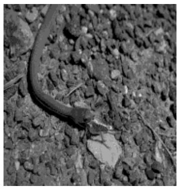
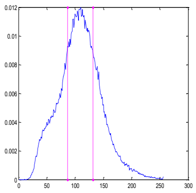
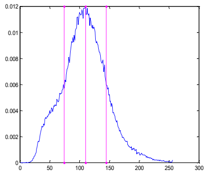
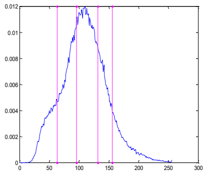
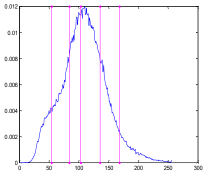
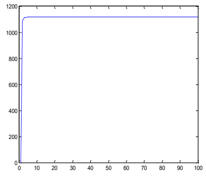
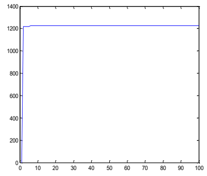
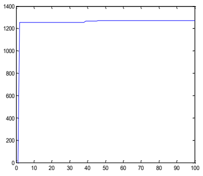
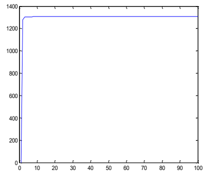
Histogram				
Fitness Graph				
 **Tree**				
Histogram				
Fitness Graph				
 Snake				
Histogram				
Fitness Graph				

Table 7.9: Best Results after applying the CCA using Otsu's function

Image	Ɵ	Best fitness	PSNR	MSSIM	Time(sec)	Threshold value
Cameramen	2	3.6517e+003	18.5827	0.5738	0.3777	70 145
	3	3.7269e+003	21.1954	0.6444	0.4221	58 127 157
	4	3.7733e+003	22.6114	0.686	0.4691	43 98 135 159
	5	3.8033e+003	23.6897	0.6981	0.5438	52 103 130 151 175
Sea star	2	2.5466e+003	15.5738	0.4545	0.3466	84 155
	3	2.7757e+003	17.8468	0.5645	0.3857	66 119 171
	4	2.8594e+003	19.2581	0.6483	0.4147	52 98 132 183
	5	2.8942e+003	21.2615	0.7152	0.4520	39 77 119 141 188
Zebra	2	1.3940e+003	13.9618	0.4453	0.3568	98 171
	3	1.5184e+003	16.3846	0.5785	0.5267	81 121 205
	4	1.5648e+003	18.6222	0.6714	0.6239	72 98 137 208
	5	1.5780e+003	20.1544	0.7101	0.6898	64 81 104 154 203
Boat man	2	5.0744e+003	14.2071	0.5385	0.3552	107 193
	3	5.2342e+003	17.5607	0.6743	0.3840	83 134 193
	4	3080e+003	20.2656	0.7661	0.4268	59 107 153 234
	5	5.3443e+003	21.3697	0.7818	0.4473	68 108 135 159 205
Ostrich	2	1.0729e+003	16.2329	0.4439	0.3614	73 133
	3	1.1352e+003	17.3580	0.4815	0.3907	63 96 136
	4	1.1716e+003	18.3364	0.5447	0.4205	57 84 133 180
	5	1.1958e+003	22.7184	0.7019	0.4480	47 68 90 123 163
Boat	2	1.2643e+003	12.3241	0.3490	0.3654	100 153
	3	1.3699e+003	18.6936	0.6477	0.3915	64 107 162
	4	1.4299e+003	18.3164	0.6735	0.4215	62 99 138 190
	5	1.4689e+003	21.0115	0.7260	0.2464	69 91 113 139 182
Tree	2	1.1887e+003	16.7795	0.5392	0.3101	72 119
	3	1.2672e+003	19.1352	0.6553	0.3674	56 92 129
	4	1.3031e+003	20.9931	0.7367	0.4197	47 80 111 153
	5	1.3220e+003	24.4303	0.8341	0.4828	36 56 85 112 136
Snake	2	1.1178e+003	15.7850	0.6250	0.3233	87 132
	3	1.2271e+003	18.7065	0.7594	0.3748	74 110 145
	4	1.2727e+003	20.5848	0.8205	0.46164	63 96 132 156
	5	1.3070e+003	22.0542	0.8619	0.5188	54 84 103 136 168

The MTEMO algorithm is considered for comparison as presented by Oliva et. al. [181] [180]. The other methods like Genetic Algorithms (GA), Particle Swarm Optimization (PSO) etc. are not considered for comparison as they were already compared with MTEMO [67] and found inferior to its performance.

7.2.4.1 COMPARISON BETWEEN OTSU AND KAPUR IN CCA

A non-parametric statistical test known as the Wilcoxon's rank test [182], [164] is used to compare the results of Otsu and Kapur and it has been done for 50 independent samples. By using this test method, the differences between two related methods can be measured. In this analysis, a 5% significance level is considered over the PSNR data corresponding to the test images with two to five threshold points. The hypothesis is considered as follows:

- ***The Null Hypothesis***: There is no difference between the values of the two objective functions Otsu and Kapur functions
- ***The alternative Hypothesis:*** is considered that there is a significant difference between the values of the two objective functions.

In table 7.10 the *p*-values produced by Wilcoxon's rank test for a pairwise comparison of the PSNR values between the Otsu and Kapur objective functions are shown. Here, *h* represents the hypothesis. All p-values stated in table 7.10 are less than the significance value 0.05. Thus it is strongly evidenced against the null hypothesis. So it is concluded that there is a significant difference between the values of two methods, i.e. Otsu's objective method's performance is statistically better than Kapur's objective method.

7.2.4.2 COMPARISON OF CCA WITH MTEMO

The proposed method is compared with another meta-heuristic method MTEMO (Multilevel Thresholding using Electro-Magnetism Optimization) as it has proved itself to be better than another popular metaheuristic algorithm. Both the algorithms are run 50 times for each image. The mean of the objective function values, PSNR, SSIM, and time for each image are reported in table 7.7 and 7.9. Table 7.11 and table 7.12 shows the comparison of CCA and MTEMO for Kapur's and Otsu's methods respectively.

Table 7.10: Wilcoxon's rank test comparing Otsu vs. Kapur over PSNR

Image	**ɵ**	**P-value: Otsu vs. Kapur**	**h**		**Image**	**ɵ**	**P-value: Otsu vs. Kapur**	**H**
Cameramen	2	0.00	1		**Ostrich**	2	0.00	1
	3	0.00	1			3	0.00	1
	4	0.00	1			4	0.00	1
	5	0.00	1			5	0.00	1
Sea star	2	0.00	1		**Boat**	2	0.00	1
	3	0.00	1			3	0.00	1
	4	0.00	1			4	0.00	1
	5	0.00	1			5	0.00	1
Zebra	2	0.00	1		**Tree**	2	0.00	1
	3	0.00	1			3	0.00	1
	4	0.00	1			4	0.00	1
	5	0.00	1			5	0.00	1
Boat man	2	0.00	1		**Snake**	2	0.00	1
	3	0.00	1			3	0.00	1
	4	0.00	1			4	0.00	1
	5	0.00	1			5	0.00	1

Table 7.11: Comparison of CCA and MTEMO using Kapur's method

Image	**ɵ**	**MTEMO**				**CCA**			
		PSNR	**mean**	**SSIM**	**Time**	**PSNR**	Mean	**SSIM**	**Time**
Cameramen	2	13.626	17.5842	0.5202	3.1307	**14.1255**	17.5663	0.5394	**0.4604**
	3	14.4602	21.976	0.5966	4.6054	**17.4939**	21.9109	0.6351	**0.5469**
	4	20.1531	26.586	0.6672	6.2591	**20.1656**	26.2674	0.6310	**0.6300**
	5	20.661	30.506	0.6850	8.0842	**21.8695**	30.2333	0.7037	**0.7115**
Sea star	2	14.3982	18.7542	0.4008	3.1986	**14.5611**	18.7503	0.4081	**0.4665**
	3	16.987	23.3233	0.5257	4.6772	**17.0149**	23.2790	0.5248	**0.5484**
	4	18.304	27.5817	0.5901	6.2660	**18.9300**	27.4544	0.6367	**0.6410**
	5	20.165	31.5626	0.6697	8.1592	**20.7114**	31.1217	0.7104	**0.7221**
Zebra	2	13.8051	17.8802	0.4455	3.2240	**14.2625**	17.7929	0.4552	**0.4592**
	3	15.0013	22.3129	0.5162	4.8551	**15.8788**	22.1162	0.5352	**0.5602**
	4	15.5085	26.5212	0.5440	6.7635	**16.2427**	26.0255	0.5639	**0.6451**
	5	20.2723	30.2664	0.7052	8.2603	**20.5283**	29.5575	0.6913	**0.7214**
Boat man	2	15.8962	18.0625	0.6238	3.3536	**16.0127**	18.0066	0.6258	**0.4620**
	3	18.1776	22.8079	0.7079	4.7672	**19.1801**	22.6660	0.7498	**0.5708**
	4	20.1480	27.2968	0.7738	6.4385	**21.0354**	26.8740	0.7852	**0.6469**
	5	22.2361	31.3337	0.8137	8.1236	**22.6732**	30.9901	0.8355	**0.7264**
Ostrich	2	10.8341	18.1959	0.0804	3.3250	**10.9067**	18.1839	0.0840	**0.4833**
	3	16.0263	22.5932	0.4415	4.8669	**16.3296**	22.5554	0.4785	**0.5496**
	4	16.2206	26.8341	0.4528	6.7516	**19.1780**	26.6143	0.7018	**0.6325**
	5	20.7745	31.0419	0.7763	8.6439	**21.5161**	30.4390	0.7825	**0.7220**
Boat	2	9.3556	18.0709	0.1629	3.2673	**9.7614**	18.0781	0.1963	**0.4705**
	3	17.1017	22.9471	0.5558	4.7784	**17.4877**	22.8050	0.5702	**0.5494**
	4	18.9648	27.1648	0.6253	6.4803	**19.7622**	26.8680	0.6812	**0.6466**
	5	19.8263	31.0884	0.6556	8.4225	**20.5056**	30.5153	0.7143	**0.7208**
Tree	2	15.6147	17.2739	0.4821	3.3594	**15.8236**	17.2706	0.4962	**0.4776**
	3	15.7403	21.8542	0.4905	4.8322	**16.3621**	21.7709	0.5507	**0.5578**
	4	16.9652	26.0243	0.5553	6.6374	**18.4098**	25.8224	0.6246	**0.6436**
	5	18.5409	29.9408	0.6343	8.5643	**21.4849**	29.4536	0.7698	**0.7108**
Snake	2	14.7184	17.9398	0.5357	3.2498	**14.8246**	17.9321	0.5431	**0.4748**
	3	15.4042	22.6388	0.5879	4.7376	**16.2885**	22.5853	0.5604	**0.6482**
	4	17.5795	26.9963	0.7205	6.7119	**18.4103**	26.7863	0.7405	**0.6347**
	5	19.7672	31.0221	0.8011	8.8380	**21.0050**	30.5564	0.8358	**0.7187**

Table 7.12: Comparison of CCA with MTEMO using Otsu’s method

Image	ϴ	**MTEMO**				**CCA**			
		PSNR	**Mean**	**SSIM**	**Time(s)**	**PSNR**	**Mean**	**SSIM**	**Time(s)**
Cameramen	2	17.2474	3.6519e+003	0.5964	2.9145	**18.1827**	3.6519e+003	0.6022	**0.3777**
	3	20.2114	3.7270e+003	0.6415	4.5650	**21.0693**	3.7227e+003	0.6512	**0.4221**
	4	21.5328	3.7824e+003	0.6648	6.5646	**22.6114**	3.7733e+003	0.6599	**0.4691**
	5	23.2235	3.8119e+003	0.6951	8.6998	**23.5897**	3.8033e+003	0.7058	**0.5438**
Star Fish	2	14.8158	2.5469e+003	0.4230	3.6676	**15.5738**	2.5466e+003	0.4545	**0.3466**
	3	17.3301	2.7799e+003	0.5485	5.8401	**17.8468**	2.7757e+003	0.5645	**0.3857**
	4	19.1259	2.8657e+003	0.6386	8.1728	**19.2581**	2.8594e+003	0.6483	**0.4147**
	5	20.7674	2.9128e+003	0.7092	10.7722	**21.2615**	2.8942e+003	0.7152	**0.4520**
Zebra	2	13.4728	1.3947e+003	0.4310	3.8425	**13.9618**	1.3940e+003	0.4453	**0.3568**
	3	15.2286	1.5263e+003	0.5444	5.9443	**16.3846**	1.5184e+003	0.5785	**0.5267**
	4	16.8718	1.5825e+003	0.6509	8.3692	**18.6222**	1.5648e+003	0.6714	**0.6239**
	5	18.2373	1.6105e+003	0.7101	10.8720	**20.1544**	1.5780e+003	0.7101	**0.6898**
Boat man	2	12.6309	5.0750e+003	0.5422	3.5575	**14.0071**	5.0744e+003	0.5385	**0.3552**
	3	15.0155	5.2399e+003	0.6383	5.4001	**17.3007**	5.2342e+003	0.6743	**0.3840**
	4	17.6208	5.3169e+003	0.7547	7.4238	**20.0656**	5.3080e+003	0.7661	**0.4268**
	5	18.8359	5.3559e+003	0.7973	9.5893	**21.0697**	5.3443e+003	0.7818	**0.4473**
Ostrich	2	15.6925	1.0735e+003	0.4219	3.7034	**16.0329**	1.0729e+003	0.4439	**0.3614**
	3	16.8151	1.1392e+003	0.4534	5.7863	**17.3580**	1.1352e+003	0.4815	**0.3907**
	4	17.4478	1.1786e+003	0.4870	8.2029	**18.3164**	1.1716e+003	0.5447	**0.4205**
	5	18.7970	1.2037e+003	0.5584	10.2127	**22.1184**	1.1958e+003	0.7019	**0.4480**
Boat	2	12.3263	1.2645e+003	0.3492	3.8212	**12.3241**	1.2643e+003	0.3490	**0.3654**
	3	17.8963	1.3749e+003	0.5972	6.7030	**18.6936**	1.3699e+003	0.6477	**0.3915**
	4	19.1961	1.4373e+003	0.6504	9.0585	**18.3164**	1.4299e+003	0.6735	**0.4215**
	5	20.9881	1.4794e+003	0.7201	11.8090	**20.9115**	1.4689e+003	0.7260	**0.2464**
Tree	2	16.7057	1.1890e+003	0.5241	4.1409	**16.7595**	1.1886e+003	0.5392	**0.3101**
	3	18.6252	1.2693e+003	0.6271	6.4711	**19.1352**	1.2663e+003	0.6553	**0.3674**
	4	20.2430	1.3135e+003	0.7005	8.5807	**20.9931**	1.3038e+003	0.7367	**0.4197**
	5	21.7758	1.3344e+003	0.7497	11.1699	**24.0303**	1.3225e+003	0.8341	**0.4828**
Snake	2	15.6662	1.1186e+003	0.6226	3.8269	**15.7050**	1.1182e+003	0.6250	**0.3233**
	3	17.9818	1.2313e+003	0.7354	5.9151	**18.4765**	1.2275e+003	0.7594	**0.3748**
	4	19.6907	1.2865e+003	0.8005	8.0355	**20.3848**	1.2770e+003	0.8205	**0.46164**
	5	20.8990	1.3170e+003	0.8395	10.8160	**22.0442**	1.3032e+003	0.8619	**0.5188**

7.2.4.3 STATISTICAL ANALYSIS

For statistical analysis of the result shown in the table 7.11 and 7.12, one way ANOVA test has been done. To perform the analysis a 5% significance level is considered over the execution time corresponding to the test images with two to five threshold points. In this analysis, the hypothesis is set as follows.

Null hypothesis H_0: *There is no significant difference in the execution time between the two methods MTEMO and CCA.*

Alternative hypothesis H_1: *There is a significant difference in the execution time between the two approaches MTEMO and CCA.*

Table 7.13: ANOVA test based on Kapur's method for the CCA and MTEMO.

ANOVA

Time

	Sum of Squares	Df	Mean Square	F	Sig.
Between Groups	422.678	1	422.678	220.149	.000
Within Groups	119.037	62	1.920		
Total	541.715	63			

Table 7.14: ANOVA test for the CCA and MTEMO based on Otsu's method

ANOVA

Time

	Sum of Squares	Df	Mean Square	F	Sig.
Between Groups	695.128	1	695.128	195.789	.000
Within Groups	220.124	62	3.550		
Total	915.253	63			

The one way ANOVA Test is conducted using SPSS tool and the results found in the experiment is shown in table 7.13 and table 7.14. Table 7.13 presents the result of ANOVA test regarding execution time that is obtained from table 7.11. Table 7.14 presents the result of ANOVA test regarding execution time that is obtained from table 7.12. Here the null hypothesis is rejected since the p-value (0.00) is less than the significance value 0.05. Therefore it can be concluded that there is a significant difference in the execution time between MTEMO and CCA.

7.3 MULTI-OBJECTIVE ENGINEERING DESIGN OPTIMIZATION USING MOCCA-W

The design optimization using meta-heuristics algorithm has many applications in engineering and industry [128], [125], [183]. There are plenty of benchmarks design optimization problem, among them the disc brake design and design of welded beam are most popular. In this section, these two design benchmark problems are solved using MOCCA-W.

7.3.1 DESIGN OF A WELDED BEAM

The design of welded beam problem is a multi-objective design problem which has been solved by many researchers using different methods [19] [179], [184] . This design problem consists of four design variables such as length of the welded area (l), the thickness of the main beam (h), width (w) and the depth (d). It has two objectives to optimize. The objective is to minimize both the end deflection δ and the overall fabrication cost. The mathematical formulation of the problem is given below:

Minimize,

$$f_1(x) = \delta \qquad (7.21)$$

$$f_2(x) = 1.10471 w^2 l + 0.04811 dh(14.0 + l)$$

Subject to

$$g_1(x) = w - h \leq 0,$$

$$g_2(x) = \delta(x) - 0.25 \leq 0,$$

$$g_3(x) = \tau(x) - 13{,}600 \leq 0,$$

$$g_4(x) = \sigma(x) - 30{,}000 \leq 0,$$

$$g_6(x) = 0.125 - w \le 0,$$

$$g_7(x) = 6000 - P \le 0,$$

$$g_5(x) = 0.10471w^2 + 0.04811wh(14 + l) - 5.0 \le 0$$

Where,

$$\sigma(x) = \frac{504000}{hd^2}$$

$$\tau(x) = \sqrt{A^2 + \frac{ABl}{R} + B^2}$$

$$A = \frac{6000}{\sqrt{2}wh}, B = \frac{MR}{J}$$

$$M = 6000\left(14 + \frac{l}{2}\right)$$

$$R = \sqrt{\frac{l^2}{4} + \left(\frac{w + d}{2}\right)^2}$$

$$J = \sqrt{2}wl\left[\frac{l^2}{6} + \frac{(w + d)^2}{2}\right]$$

$$\delta(x) = \frac{65,856}{30,000hd^3}$$

$$P = 0.61423 \times 10^6 \frac{dh^3}{6}\left(1 - \frac{d\sqrt{30/48}}{28}\right)$$

With the range 0.1≤l≤2.0, 0.1 ≤d≤10, 0.125≤w≤10, 0.1≤h≤2.0.

7.3.2 DESIGN OF A DISC BRAKE

The disc brake design is an optimization problem having two objectives. The objectives of designing multiple disc brakes are to minimize the braking time and the overall mass by choosing optimal design variables. This problem has also four design variables such as outer radius (R) of the discs, the inner radius(r), the number of the friction surface (S) and the engaging force (F). It consists of some design constraints such as temperature, pressure, the length of the brake and torque [185] [54]. It is mathematically formulated as:

Minimize, $f_1(x) = 4.9 \times 10^{-5}(R^2 - r^2)(s - 1)$ (7.22)

$$f_2(x)\frac{9.82\times10^6\left(R^2-r^2\right)}{Fs\left(R^3-r^3\right)}$$

Subject to,

$$g_1(x)=20-(R-r)\leq0,$$

$$g_2(x)=2.5(s+1)-30\leq0,$$

$$g_3(x)=\frac{}{3.14\left(R^2-r^2\right)}-0.4\leq0,$$

$$g_4(x)=\frac{2.22\times10^{-3}F\left(R^3-r^3\right)}{\left(R^2-r^2\right)}-1\leq0,$$

$$g_5(x)=900-\frac{0.0266Fs(R^3-r^3)}{(R^2-r^2)}\leq0$$

$g_4(x)=\frac{2.22\times10^{-3}F(R^3\text{-}r^3)}{(R^2\text{-}r^2)}$-1 $\leq$ 0,With the range 1000≤F≤3000, 75≤R≤110, 55≤r≤80, 2≤s≤20.

7.3.3 EXPERIMENTAL RESULTS

The algorithm is run for 1000 iterations for both the welded beam design and the design of disc brake problem. The optimal Pareto fronts found in the experiments for both the problem are shown in figure 7.1 and figure 7.2.

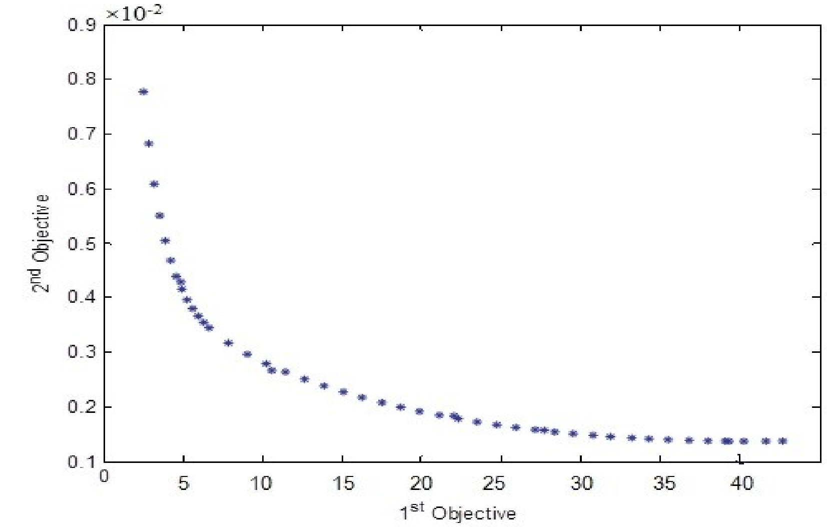


Figure 7.1 Pareto front of the Welded Beam Design using CCA

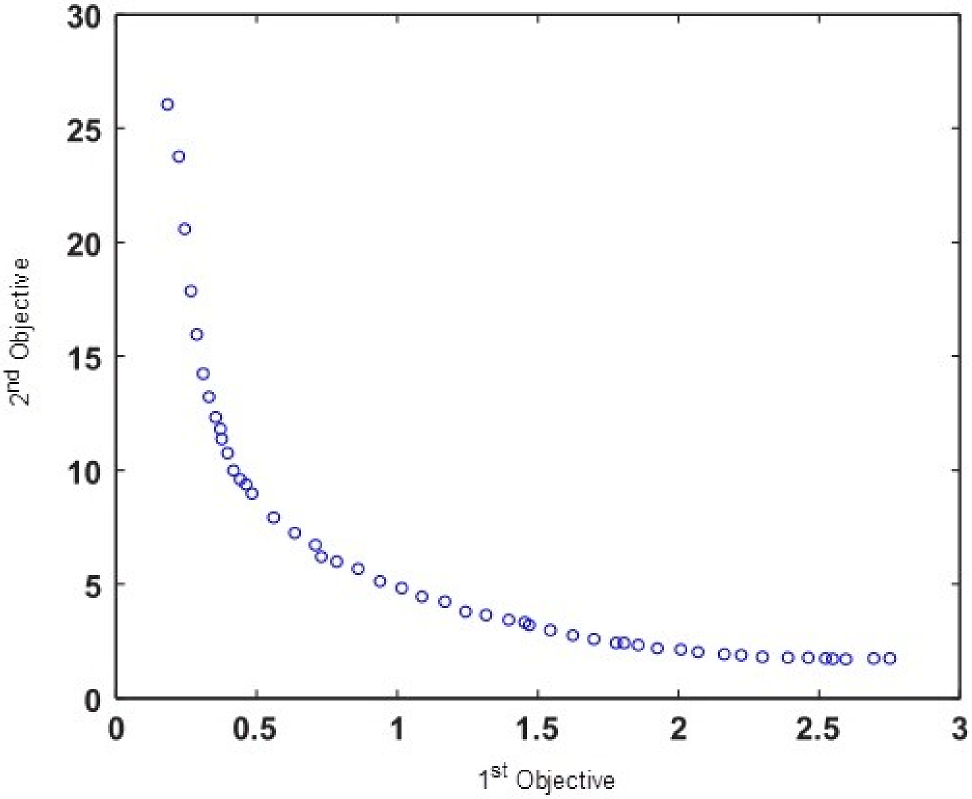


Figure 7.2 Pareto front of Disc Brake Design using CCA

7.4 MULTI-OBJECTIVE ENGINEERING DESIGN OPTIMIZATION USING MOCCA-P

In this section, the engineering design problems Welded Beam Design and Disc brake problem are solved using MOCCA-P. The details about both the problems are explained in the previous section 7.3.1 and 7.3.2.

The algorithm MOCCA-P is run for 1000 iterations for both the welded beam design and the design of disc brake problem. The optimal Pareto front found in the experiment is for both problems welded beam design and disc brake are shown in figure 7.3 and figure 7.4 respectively.

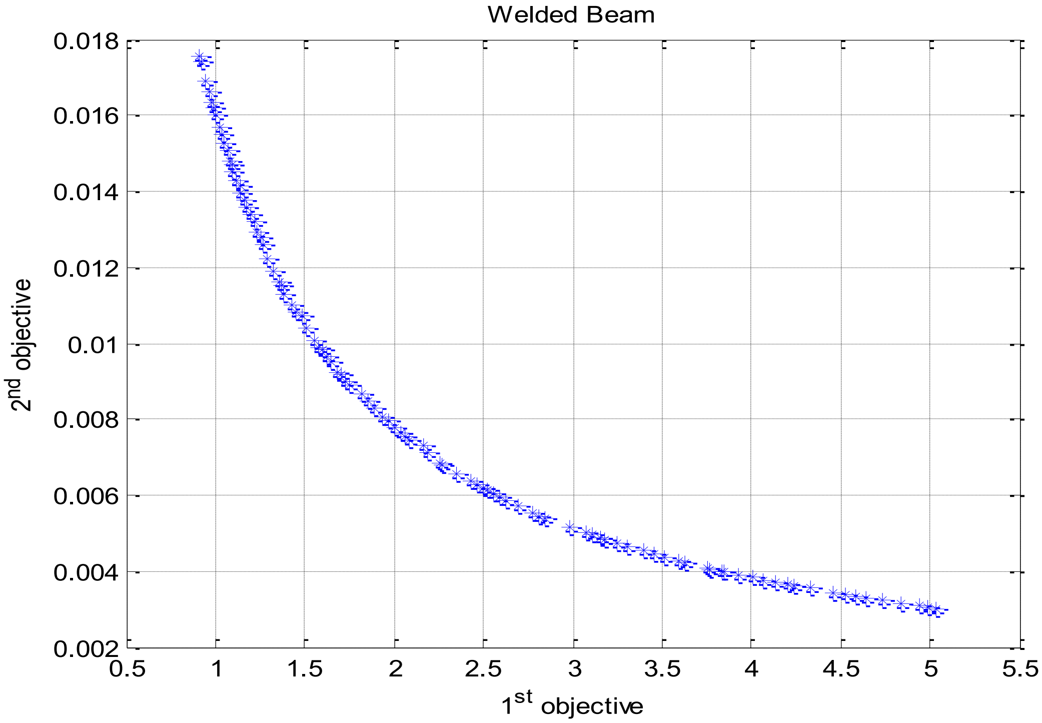


Figure 7.3: Pareto front of welded beam design using MOCCA-P

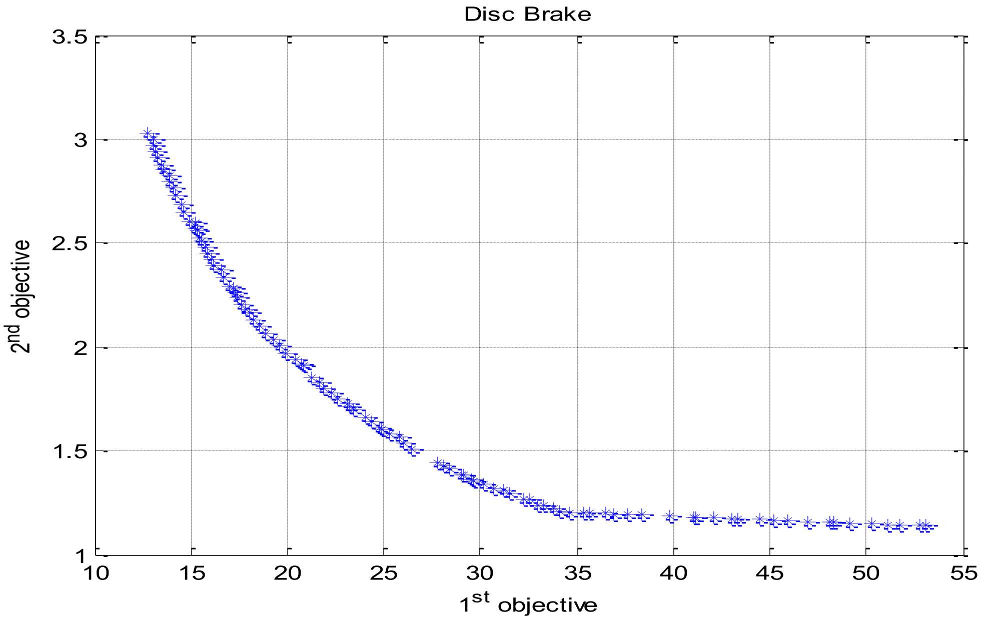


Figure 7.4: Pareto front of Disc brake design using MOCCA-P

7.5 SUMMARY

In this Chapter, different case studies for both SOO and MOO have been taken up and optimized using CCA and its variants. CCA is applied to mechanical engineering design problems and multilevel thresholding (MT) for image segmentation. In engineering design, two problems namely spring design and welded beam design are solved using CCA and compared the performance with PSO and BA algorithm. Then the CCA is applied to solve multilevel thresholding in image segmentation. Two popular MT methods Otsu's and Kapur's method are used as objective function and combined with the searching capabilities of CCA. With the purpose of measuring the performance, the PSNR that measures the quality of segmentation by complying the oddity between the segmented and the original images, SSIM (Structural Similarity) and Computational time is used. To analyze the performance of Otsu's and Kapur's methods with CCA for multilevel thresholding of image segmentation, the Wilcoxon's rank test is used. The proposed algorithm is then compared with MTEMO and statistically analyzed using one way ANOVA test. Finally, it is concluded that CCA is better than its counterparts in term of fitness value, PSNR, SSIM and computational times. The algorithm MOCCA-W and MOCCA-P are applied to solve the engineering design problem having multiple objectives like welded beam design and disc brake design problem.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

This chapter provides the conclusions derived from this work and discusses the possible enhancements that could be done in future.

8.1 CONCLUSION

The NP-hard combinatorial optimization problems are the most challenging problems nowadays. Compared to other methods for solving the optimization problem, meta-heuristics techniques are getting more attention in the research field because of their efficiency, optimality, and speed. There has previously been plenty of research on meta-heuristics techniques. In this research work, the research motivations have been thoroughly analyzed from the literature review and the research objectives have been framed.

Based on the research motivation and objectives a new meta-heuristics bio-inspired algorithm called Cricket Chirping Algorithm (CCA) is developed for SOO. A set of benchmark functions have been used to test and validate the CCA and compare its performance with some of the popular algorithms such as GA, PSO, ABC, BA, and CS for both lower dimension and higher dimension problems.

Since the tuning of parameter plays a vigorous role in the performance of an algorithm, the impact of various parameters used in CCA is analyzed. The parameters viz., environmental Temperature T_c, Aggression Rate A_r, Crossover Rate C_r and Female Selection F_s have an effective contribution to the performance of CCA. When comparing the initial and final set of parameters, it is found that the final set provides better results than the initial parameter configuration for the problem under study. As per the analysis of the experiment the higher the temperature, the higher the fitness value of the crickets. The cricket produces high-frequency sound at high temperatures. But, in low aggression rate, it shows better performance for low dimension problems. In female selection, the best fit female selection converges faster compared to other female selection schemes.

The values obtained through various experimental settings could be fixed as the standard parameters for the CCA algorithm in future.

The CCA is extended for solving MOO problems in two ways i.e. using weighted sum approach (MOCCA-W) and Pareto based technique, mimicking some interesting behavior of crickets (MOCCA-P). The MOCCA-P differs from the basic CCA in two terms. First, the male cricket is allowed to search the female cricket in the search space and secondly, when the male cricket chirps for aggression the winner is selected depending on the seven aggression levels. A different fitness calculation method is also developed and an external archive is used to retain the non-dominated solutions. The MOCCA-P is implemented and experimented with some standard benchmark test problems with constraints and without constraints and compared with three popular techniques i.e. MOPSO, SPEA2, and NSGA2. The experimental results show better results compared to its counterparts in term of generational distance, spacing and maximum spread, the popular metrics used to validate Multiobjective optimization algorithm.

In the last, the different case studies for both single and MOO has been taken up and optimized using CCA and its variants. CCA is applied to mechanical engineering design problems and multilevel thresholding (MT) for image segmentation problem. In engineering design, two problems, namely spring design, and welded beam design are solved using CCA and compared the performance with PSO and BA algorithm. Then the CCA is applied to solve multilevel thresholding in image segmentation. Two popular MT methods Otsu’s and Kapur’s method are used as objective function and combined with the searching capabilities of CCA. With the purpose of measuring the performance, the PSNR that measures the quality of segmentation by complying the oddity between the segmented and the original images, SSIM (Structural Similarity) and Computational time is used. To see the effect of Otsu’s and Kapur's methods with CCA for multilevel thresholding of image segmentation, the Wilcoxon’s rank test is used. The performance of the algorithm is then compared with MTEMO and statistically analyzed using one way ANOVA test. Then, it is concluded that CCA is better than the popular meta-heuristics algorithm in terms of fitness value, PSNR, SSIM and computational times.

Finally, the extended version of CCA, i.e., MOCCA-W and MOCCA-P are applied to solve the engineering design problem having multiple objectives like welded beam design and disc brake design problem and found to produce promising results.

8.2 FUTURE ENHANCEMENTS

In this research work, CCA is applied only in a few areas. The CCA can be extensively used and modified or improved for various fields of real-world SOO and MOO problems. In future, the MOCCA-P could be tested for many objective optimization problems and applied to various fields. Moreover, the algorithm could be tested for many objective functions with and without constraints by exploiting other characteristics of crickets.

REFERENCES

[1] M. Caramia and P. Dell'Olmo, *Multi-objective Optimization*. 2008.

[2] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2007.

[3] I. Tome, "Cours d ' Économie politique," 1919.

[4] M. Management, "Multi-objective Optimization," 2008, pp. 11–37.

[5] P. Festa, "A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems," *Int. Conf. Transparent Opt. Networks*, pp. 1–20, 2014.

[6] K. Sörensen and S. Kenneth, "Metaheuristics -- the metaphor exposed Metaheuristics – the Metaphor Exposed," no. March, 2017.

[7] C. A. C. Coello, "Metaheuristics for Multiobjective Optimization," no. 2508, pp. 1–15, 2015.

[8] X. Yang and L. Press, *Nature-Inspired Metaheuristic Algorithms Second Edition*. .

[9] P. Agarwal and S. Mehta, "Nature-Inspired Algorithms: State-of-Art, Problems and Prospects," *Int. J. Comput. Appl.*, vol. 100, no. 14, pp. 14–21, 2014.

[10] D. Whitley, "An overview of evolutionary algorithm: practical Issues and common pitfalls," *Inf. Softw. Technol.*, vol. 43, pp. 817–831, 2001.

[11] N. Theorems and X. Yang, "Swarm-Based Metaheuristic Algorithms and," 2011.

[12] M. Settles, "An Introduction to Particle Swarm Optimization," pp. 1–8, 2005.

[13] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," *Proc. 1999 Congr. Evol. Comput. (Cat. No. 99TH8406)*, vol. 2, pp. 1470–1477, 1999.

[14] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, 2007.

[15] X. Yang and S. Deb, "Cuckoo Search via Lévy Flights," in *India. IEEE Publications*, 2009, pp. 210–214.

[16] X. Yang, "A New Metaheuristic Bat-Inspired Algorithmin," *Stud. Comput. Intell. Springer Berlin,* vol. 284, pp. 65–74, 2010.

[17] X. Yang, "Firefly Algorithm, L´ evy Flights and Global Optimization," *Res. Dev.*

Intell. Syst. XXVI, pp. 209–218, 2010.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[19] Ş. I. Birbil and S. C. Fang, “An electromagnetism-like mechanism for global optimization,” *J. Glob. Optim.*, vol. 25, no. 3, pp. 263–282, 2003.

[20] K. Lorenz and N. Tinbergen, “Behavior of the House Cricket , Acheta domesticus,” no. 1, 2010.

[21] D. Whitley, “A genetic algorithm tutorial,” *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, 1994.

[22] C. R. Reeves, “A genetic algorithm for flowshop sequencing,” *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, 1995.

[23] P. C. Chu and J. E. Beasley, “A genetic algorithm for the multidimensional knapsack problem,” *J. Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[24] D. M. Deaven and K. M. Ho, “Molecular geometry optimization with a genetic algorithm,” *Phys. Rev. Lett.*, vol. 75, no. 2, pp. 288–291, 1995.

[25] F. Pezzella, G. Morganti, and G. Ciaschetti, “A genetic algorithm for the Flexible Job-shop Scheduling Problem,” *Comput. Oper. Res.*, vol. 35, no. 10, pp. 3202–3212, 2008.

[26] M. Kumar, M. Husian, N. Upreti, and D. Gupta, “Genetic Algorithm: Review and Application,” *Int. J. Inf. Technol. Knowl. Manag.*, vol. 2, no. 2, pp. 451–454, 2010.

[27] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948.

[28] M. Pant, R. Thangaraj, and A. Abraham, “Particle Swarm Optimization : Performance Tuning and Empirical Analysis,” vol. 3, pp. 101–128, 2009.

[29] Q. Bai, “Analysis of Particle Swarm Optimization Algorithm,” *Comput. Inf. Sci.*, vol. 3, no. 1, 2010.

[30] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: Optimization by a colony of cooperating agents,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[31] A. Rabanimotlagh, “An Efficient Ant Colony Optimization Algorithm for

Multiobjective Flow Shop Scheduling Problem," vol. 5, no. 3, pp. 127–133, 2011.

[32] M. Dorigo and T. Stützle, *Ant Colony Optimization*. 2004.

[33] B. Chandra Mohan and R. Baskaran, "A survey: Ant Colony Optimization based recent research and implementation on several engineering domain," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4618–4627, 2012.

[34] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, Nov. 2005.

[35] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 651–665, 2007.

[36] Q. Yang *et al.*, "Adaptive Multimodal Continuous Ant Colony Optimization," *IEEE Trans. Evol. Comput.*, no. c, pp. 1–1, 2016.

[37] D. KARABOGA, "An Idea based on honey bee swarm for numerical optimization," 2005.

[38] D. Karaboga and B. Basturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization," *Lnai 4529*, pp. 789–798, 2007.

[39] D. Karaboga, S. Okdem, and C. Ozturk, "Cluster based wireless sensor network routing using artificial bee colony algorithm," *Wirel. Networks*, vol. 18, no. 7, pp. 847–860, 2012.

[40] J. Bansal, H. Sharma, K. V. Arya, and A. Nagar, "Memetic search in artificial bee colony algorithm," *Soft Comput.*, vol. 5, no. 2, pp. 1–18, 2013.

[41] A. Layeb, "A Multi-objective Binary Cuckoo Search for Bi- criteria Knapsack Problem," no. October, pp. 8–15, 2013.

[42] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Comput. Appl.*, vol. 24, no. 7–8, pp. 1659–1669, Jun. 2014.

[43] A. Bouaziz, A. Draa, and S. Chikhi, "A Cuckoo search algorithm for fingerprint image contrast enhancement," *Complex Syst. (WCCS), 2014 Second World Conf.*, pp. 678–685, 2014.

[44] M. Sreenivasa Rao and N. Venkaiah, "A modified cuckoo search algorithm to optimize Wire-EDM process while machining Inconel-690," *J. Brazilian Soc.*

Mech. Sci. Eng., vol. 39, no. 5, pp. 1647–1661, 2017.

[45] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, “Modified cuckoo search: A new gradient free optimisation algorithm,” *Chaos, Solitons and Fractals*, vol. 44, no. 9, pp. 710–718, 2011.

[46] B. H. Dinh, T. T. Nguyen, and D. N. Vo, “Adaptive cuckoo search algorithm for short-term fixed-head hydrothermal scheduling problem with reservoir volume constraints,” *Int. J. Grid Distrib. Comput.*, vol. 9, no. 5, pp. 191–204, 2016.

[47] A. H. Gandomi and X. S. Yang, “Chaotic bat algorithm,” *J. Comput. Sci.*, vol. 5, no. 2, pp. 224–232, 2014.

[48] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa, and X. S. Yang, “BBA: A binary bat algorithm for feature selection,” in *Brazilian Symposium of Computer Graphic and Image Processing*, 2012, pp. 291–297.

[49] M. Jamil and X.-S. Yang, “A Literature Survey of Benchmark Functions For Global Optimization Problems Citation details: Momin Jamil and Xin-She Yang, A literature survey of benchmark functions for global optimization problems,” *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.

[50] X.-S. Yang, “Firefly Algorithm, Stochastic Test Functions and Design Optimisation,” Mar. 2010.

[51] B. Crawford *et al.*, “A binary coded firefly algorithm that solves the set covering problem,” *Rom. J. Inf. Sci. Technol.*, vol. 17, no. 3, pp. 252–264, 2014.

[52] S. Palit, S. N. Sinha, M. A. Molla, A. Khanra, and M. Kule, “A cryptanalytic attack on the knapsack cryptosystem using binary Firefly algorithm,” in *2011 2nd International Conference on Computer and Communication Technology, ICCCT-2011*, 2011, pp. 428–432.

[53] S. M. Farahani, A. A. Abshouri, B. Nasiri, and M. R. Meybodi, “A Gaussian Firefly Algorithm,” vol. 1, no. 5, 2011.

[54] X.-S. Yang, “Firefly algorithm, Lévy flights and global optimization,” in *Research and Development in Intelligent Systems XXVI: Incorporating Applications and Innovations in Intelligent Systems XVII*, 2010, pp. 209–218.

[55] S. H. Strogatz, “Nonlinear Dynamics and Chaos,” *Library*, vol. 48, no. 3. p. 498, 1994.

[56] M. Subutic, M. Tuba, and N. Stanarevic, “Parallelization of the firefly algorithm for unconstrained optimization problems,” *Latest Adv. Inf. …*, pp. 264–269, 2012.

[57] and M. R. M. Sh. Mashhadi Farahani, A. Amin Abshouri, B. Nasiri, “Some hybrid models to improve Firefly algorithm performance,” *Int. J. Artif. Intell.*, vol. 8, no. 12 S, pp. 97–117, 2012.

[58] M. Younes and R. L. Kherfane, “Hybrid FA and GA for generation allocation problem optimization,” *J. Electr. Eng.*, vol. 14, no. 1, pp. 108–113, 2014.

[59] S. Das and P. N. Suganthan, “Differential evolution: A survey of the state-of-the-art,” *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[60] T. Hassanzadeh and M. R. Meybodi, “A new hybrid algorithm based on firefly algorithm and cellular learning automata,” in *ICEE 2012 - 20th Iranian Conference on Electrical Engineering*, 2012, pp. 628–633.

[61] A. Rajini and V. David, “A Comparative Performance Study on Hybrid Swarm Model for Micro array Data,” *Int. J. Comput. Appl.*, vol. 30, no. 6, pp. 10–14, 2011.

[62] X. S. Yang and X. He, “Firefly algorithm: recent advances and applications,” *Int. J. Swarm Intell.*, vol. 1, no. 1Yang, Xin She, and Xingshi He. “Firefly Algorithm: Recent Advances and Applications.” International Journal of Swarm Intelligence 1, 1 (2013): 36. doi:10.1504/IJSI.2013.055801., p. 36, 2013.

[63] F. Shabbir and P. Omenzetter, “Particle swarm optimization with sequential niche technique for dynamic finite element model updating,” *Comput. Civ. Infrastruct. Eng.*, vol. 30, no. 5, pp. 359–375, 2015.

[64] C. Li and J. Zhou, “Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm,” in *Energy Conversion and Management*, 2011, vol. 52, no. 1, pp. 374–381.

[65] M. Khajehzadeh, M. R. Taha, A. El-Shafie, and M. Eslami, “A modified gravitational search algorithm for slope stability analysis,” *Eng. Appl. Artif. Intell.*, vol. 25, no. 8, pp. 1589–1597, 2012.

[66] E. Cuevas, D. Oliva, D. Zaldivar, M. Pérez-Cisneros, and H. Sossa, “Circle detection using electro-magnetism optimization,” *Inf. Sci. (Ny).*, vol. 182, no. 1, pp. 40–55, 2012.

[67] D. Oliva, E. Cuevas, G. Pajares, D. Zaldivar, and V. Osuna, “A Multilevel thresholding algorithm using electromagnetism optimization,” *Neurocomputing*, vol. 139, pp. 357–381, 2014.

[68] A. Abraham, L. Jain, and G. (Eds) Robert, *Evolutionary Multiobjective Optimization*. 2005.

[69] N. Srinivas and K. Deb, “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms 1 Introduction,” vol. 2, no. 3.

[70] W. Stadler, “A survey of multicriteria optimization or the vector maximum problem, part I: 1776-1960,” *J. Optim. Theory Appl.*, vol. 29, no. 1, pp. 1–52, 1979.

[71] D. a. Van Veldhuizen, “Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations,” *IRE Trans. Educ.*, 1999.

[72] E. Zitzler and K. Simon, “Indicator-Based Selection in Multiobjective Search,” *8th Int. Conf. Parallel Probl. Solving from Nat. (PPSN VIII)*, vol. 3242, no. i, pp. 832–842, 2004.

[73] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, “Multiobjective evolutionary algorithms: A survey of the state of the art,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[74] J. D. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms,” *1st Int. Conf. Genet. Algorithms*, no. JANUARY 1985, pp. 93–100, 1985.

[75] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

[76] K. Deb, S. Agrawal, S. Pratab, and T. Meyarivan, “A fast elitist non- dominated sorting genetic algorithm for multi-objective optimization: NSGA-II.,” *Parallel Probl. Solving from Nat. VI Conf.*, 2000.

[77] J. Horn, N. Nafpliotis, and D. E. Goldberg, “A niched Pareto genetic algorithm for multiobjective optimization,” *Evol. Comput. 1994. IEEE World Congr. Comput. Intell. Proc. First IEEE Conf.*, vol. 1, pp. 82–87, 1994.

[78] Open Source Engineering, “M Ultiobjective Optimization and Genetic Algorithms.”

[79] Q. Zhang and H. Li, “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition,” *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.

[80] J. D. Knowles and D. W. Corne, “Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy,” *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, 2000.

[81] E. Zitzler and L. Thiele, “An Evolutionary Algorithm for Multiobjective Optimization : The Strength Pareto Approach,” no. 43, 1998.

[82] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2 : Improving the Strength Pareto Evolutionary Algorithm,” pp. 1–21, 2001.

[83] J. Moore and R. Chapman, “Application of particle swarm to multiobjective optimization,” *Dep. Comput. Sci. Softw. Eng. Dep. Auburn Univ.*, pp. 1–4, 1999.

[84] C. a. Coello Coello and M. Reyes-Sierra, “Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art,” *Int. J. Comput. Intell. Res.*, vol. 2, no. 3, 2006.

[85] C. a C. Coello, G. T. Pulido, and M. S. Lechuga, “Handling multiple objectives with particle swarm optimization,” *Evol. Comput. IEEE Trans.*, vol. 8, no. 3, pp. 256–279, 2004.

[86] X. Hu and R. Eberhart, “Multiobjective optimization using dynamic neighborhood particle swarm optimization,” *Congress on Evolutionary Computation*, vol. 2. pp. 1677–1681, 2002.

[87] X. H. X. Hu, R. C. Eberhart, and Y. S. Y. Shi, “Particle swarm with extended memory for multiobjective optimization,” *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS’03 (Cat. No.03EX706)*. 2003.

[88] C. A. Coello Coello and M. S. Lechuga, “MOPSO: A proposal for multiple objective particle swarm optimization,” *Proc. 2002 Congr. Evol. Comput. CEC 2002*, vol. 2, pp. 1051–1056, 2002.

[89] S. Janson, D. Merkle, and M. Middendorf, “Molecular docking with multi-objective Particle Swarm Optimization,” vol. 8, pp. 666–675, 2008.

[90] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, “Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients,” *Inf. Sci. (Ny).*, vol. 177, no. 22, pp. 5033–5049, 2007.

[91] N. Tian and Z. Ji, "Pareto-Ranking Based Quantum-Behaved Particle Swarm Optimization for Multiobjective Optimization," vol. 2015, 2015.

[92] Y. Wang and Y. Yang, "Particle swarm optimization with preference order ranking for multi-objective optimization," *Inf. Sci. (Ny).*, vol. 179, no. 12, pp. 1944–1959, 2009.

[93] V. L. Huang, P. N. Suganthan, and J. J. Liang, "Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems," *Int. J. Intell. Syst.*, vol. 21, no. 2, pp. 209–226, 2006.

[94] S. Z. Zhao and P. N. Suganthan, "Two-lbests based multi-objective particle swarm optimizer," *Eng. Optim.*, vol. 43, no. 1, pp. 1–17, 2011.

[95] X. Yu and X. Zhang, "Multiswarm comprehensive learning particle swarm optimization for solving multiobjective optimization problems," *PLoS One*, vol. 12, no. 2, pp. 1–21, 2017.

[96] D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh, and W. K. Ho, "On solving multiobjective bin packing problems using evolutionary particle swarm optimization," *Eur. J. Oper. Res.*, vol. 190, no. 2, pp. 357–382, 2008.

[97] J. Rada-Vilela, M. Chica, O. Cordon, and S. Damas, "A comparative study of Multi-Objective Ant Colony Optimization algorithms for the Time and Space Assembly Line Balancing Problem," *Appl. Soft Comput.*, vol. 13, no. 11, pp. 4370–4382, 2013.

[98] M. Dorigo and M. Birattari, "Multi-Objective Ant Colony Optimization," *Encycl. Mach. Learn.*, no. October 2003, 2010.

[99] Z. Zhang, C. Gao, Y. Lu, Y. Liu, and M. Liang, "Multi-objective ant colony optimization based on the physarum-inspired mathematical model for bi-objective traveling salesman problems," *PLoS One*, vol. 11, no. 1, pp. 1–23, 2016.

[100] D. Angus, "Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem," *Proc. 2007 IEEE Symp. Comput. Intell. Multicriteria Decis. Making, MCDM 2007*, no. Mcdm, pp. 333–340, 2007.

[101] J. Cheng, G. Zhang, Z. Li, and Y. Li, "Multi-objective ant colony optimization based on decomposition for bi-objective traveling salesman problems," *Soft Comput.*, vol. 16, no. 4, pp. 597–614, 2012.

[102] B. Barán and M. Schaerer, “A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows,” in *21st IASTED International Conference on Applied Informatics*, 2003, pp. 97–102.

[103] J. M. Pasia, R. F. Hartl, and K. F. Doerner, *Solving a bi-objective flowshop scheduling problem by pareto-ant colony optimization*, vol. 4150. 2006.

[104] T. B. Kurniawan, Z. Ibrahim, N. K. Khalid, and M. Khalid, “A Population-Based Ant Colony Optimization Approach for DNA Sequence Optimization,” *2009 Third Asia Int. Conf. Model. Simul.*, pp. 246–251, 2009.

[105] J. A. Sabino, J. E. Leal, T. Stützle, and M. Birattari, “A multi-objective ant colony optimization method applied to switch engine scheduling in railroad yards,” *Pesqui. Operacional*, vol. 30, no. 2, pp. 486–514, 2010.

[106] L. A. Moncayo-Martínez and D. Z. Zhang, “Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design,” *Int. J. Prod. Econ.*, vol. 131, no. 1, pp. 407–420, 2011.

[107] M. López-Ibánez and T. Stützle, “The automatic design of multiobjective ant colony optimization algorithms,” *Evol. Comput. IEEE …*, no. February, 2012.

[108] S. N. Omkar, J. Senthilnath, R. Khandelwal, G. Narayana Naik, and S. Gopalakrishnan, “Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures,” *Appl. Soft Comput.*, vol. 11, no. 1, pp. 489–499, 2011.

[109] S. A. R. Mohammadi, M. R. F. Derakhshi, and R. Akbari, “An Adaptive Multi-Objective Artificial Bee Colony with crowding distance mechanism,” *Iran. J. Sci. Technol. - Trans. Electr. Eng.*, vol. 37, no. E1, pp. 79–92, 2013.

[110] B. Akay, “Synchronous and asynchronous Pareto-based multi-objective Artificial Bee Colony algorithms,” *J. Glob. Optim.*, vol. 57, no. 2, pp. 415–445, 2013.

[111] P. Rakshit, A. Konar, and A. K. Nagar, “Artificial Bee Colony induced multi-objective optimization in presence of noise,” *Proc. 2014 IEEE Congr. Evol. Comput. CEC 2014*, pp. 3176–3183, 2014.

[112] Y. Xiang, Y. Zhou, and H. Liu, “An elitism based multi-objective artificial bee colony algorithm,” *Eur. J. Oper. Res.*, vol. 245, no. 1, pp. 168–193, 2015.

[113] Y. Xiang and Y. Zhou, “A dynamic multi-colony artificial bee colony algorithm

for multi-objective optimization," *Appl. Soft Comput. J.*, vol. 35, pp. 766–785, 2015.

[114] Y. Huo, Y. Zhuang, J. Gu, and S. Ni, "Elite-guided multi-objective artificial bee colony algorithm," *Appl. Soft Comput.*, vol. 32, pp. 199–210, 2015.

[115] W. Y. Wu Chunming, Li Tingting, "A Novel Multiobjective Optimization Method Based on Improved Artificial Bee Colony Algorithm," *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 9, no. 3, pp. 231–238, 2016.

[116] W. Zou, Y. Zhu, H. Chen, and H. Shen, "A novel multi-objective optimization algorithm based on artificial bee colony," *Proc. 13th Annu. Conf. companion Genet. Evol. Comput. - GECCO '11*, p. 103, 2011.

[117] L. Ma, K. Hu, Y. Zhu, and H. Chen, "Cooperative artificial bee colony algorithm for multi-objective RFID network planning," *J. Netw. Comput. Appl.*, vol. 42, pp. 143–162, 2014.

[118] D. L. González-Álvarez, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez, "Finding motifs in DNA sequences applying a Multiobjective Artificial Bee Colony (MOABC) algorithm," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6623 LNCS, pp. 89–100, 2011.

[119] K. Atashkari, N. NarimanZadeh, a. R. Ghavimi, M. J. Mahmoodabadi, and F. Aghaienezhad, "Multi-objective optimization of power and heating system based on artificial bee colony," *2011 Int. Symp. Innov. Intell. Syst. Appl.*, pp. 64–68, 2011.

[120] M. Silva Maximiano, M. a. Vega-Rodríguez, J. a. Gómez-Pulido, and J. M. Sánchez-Pérez, "A new Multiobjective Artificial Bee Colony algorithm to solve a real-world frequency assignment problem," *Neural Comput. Appl.*, pp. 1447–1459, 2012.

[121] R. K. Jena, "Artificial Bee Colony Algorithm based Multi-Objective Node Placement for Wireless Sensor Network," *Int. J. Inf. Technol. Comput. Sci.*, vol. 6, no. 6, pp. 25–32, 2014.

[122] T. Sag and M. Cunkas, "Color image segmentation based on multiobjective artificial bee colony optimization," vol. 34, pp. 389–401, 2015.

[123] Z. Wang, M. Li, L. Dou, Y. Li, Q. Zhao, and J. Li, “A novel multi-objective artificial bee colony algorithm for multi-robot path planning,” *2015 IEEE Int. Conf. Inf. Autom. ICIA 2015 - conjunction with 2015 IEEE Int. Conf. Autom. Logist.*, no. August, pp. 481–486, 2015.

[124] A. K. Dwivedi, S. Ghosh, and N. D. Londhe, “Low power FIR filter design using modified multi-objective artificial bee colony algorithm,” *Eng. Appl. Artif. Intell.*, vol. 55, pp. 58–69, 2016.

[125] X. S. Yang and S. Deb, “Multiobjective cuckoo search for design optimization,” *Comput. Oper. Res.*, vol. 40, no. 6, pp. 1616–1624, 2013.

[126] H. V. H. Ayala and L. Dos Santos Coelho, “Multiobjective cuckoo search applied to radial basis function neural networks training for system identification,” *IFAC Proc. Vol.*, vol. 19, pp. 2539–2544, 2014.

[127] M. Akbari and H. Rashidi, “A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems,” *Expert Syst. Appl.*, vol. 60, pp. 234–248, 2016.

[128] X. Yang, “Bat Algorithm for Multi-objective Optimisation,” pp. 1–12, 2011.

[129] L. M. Amine and K. Nadjet, “A Multi-objective Binary Bat Algorithm,” *Proc. Int. Conf. Intell. Inf. Process. Secur. Adv. Commun. - IPAC ’15*, pp. 1–5, 2015.

[130] N.-C. Yang and M.-D. Le, “Multi-objective bat algorithm with time-varying inertia weights for optimal design of passive power filters set,” *IET Gener. Transm. Distrib.*, vol. 9, no. 7, pp. 644–654, 2015.

[131] T. K. Tharakeshwar, K. N. Seetharamu, and B. Durga Prasad, “Multi-objective optimization using bat algorithm for shell and tube heat exchangers,” *Appl. Therm. Eng.*, vol. 110, pp. 1029–1038, 2017.

[132] N.-C. Yang and M.-D. Le, “Optimal design of passive power filters based on multi-objective bat algorithm and pareto front,” *Appl. Soft Comput.*, vol. 35, pp. 257–266, 2015.

[133] A. M. Elewe, K. B. Hasnan, and A. B. Nawawi, “Hybridized firefly algorithm for multi-objective Radio Frequency Identification (RFID) Network planning,” *ARPN J. Eng. Appl. Sci.*, vol. 12, no. 3, 2017.

[134] C. Yammani, S. Maheswarapu, and S. K. Matam, “A Multi-objective Shuffled Bat

algorithm for optimal placement and sizing of multi distributed generations with different load models," *Int. Trans. Electr. Energy Syst.*, vol. 26, no. 2, pp. 274–292, 2016.

[135] X.-S. Yang, "Multiobjective Firefly Algorithm for Continuous Optimization," vol. 29, no. 2, pp. 175–184, 2013.

[136] X. S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect," *Appl. Soft Comput. J.*, vol. 12, no. 3, pp. 1180–1186, 2012.

[137] S. Santander-Jiménez and M. A. Vega-Rodríguez, "A multiobjective proposal based on the firefly algorithm for inferring phylogenies," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7833 LNCS, pp. 141–152.

[138] C.-W. Tsai, Y.-T. Huang, and M.-C. Chiang, "A non-dominated sorting firefly algorithm for multi-objective optimization," in *2014 14th International Conference on Intelligent Systems Design and Applications*, 2014, pp. 62–67.

[139] C. Zhao *et al.*, "Decomposition-based multi-objective firefly algorithm for RFID network planning with uncertainty," *Appl. Soft Comput. J.*, vol. 55, pp. 549–564, 2017.

[140] X. Yang, "Engineering Optimization and Metaheuristics Metaheuristics and Engineering Optimization Introduction Design Optimization Conventional Approach Metaheuristics Genetic Algorithms & Simulated Annealing Particle Swarm Optimization Firefly Algorithm & Cuckoo S."

[141] S. Akhtar, K. Tai, and T. Ray, "A Socio-Behavioural Simulation Model for Engineering Design Optimization," *Eng. Optim.*, vol. 34, no. 4, pp. 341–354, 2002.

[142] A. H. Aguirre, A. E. M. Zavala, E. Villa, A. Hern, and A. E. Mu, "COPSO : Constrained Optimization via PSO algorithm," vol. 2007.

[143] L. C. Cagnina, S. C. Esquivel, U. Nacional, D. S. Luis, S. Luis, and C. A. C. Coello, "Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer : SiC-PSO," *Informatica*, no. 32, pp. 319–326, 2008.

[144] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *J. Intell. Manuf.*, vol. 23, no. 4,

pp. 1001–1014, 2012.

[145] X. Yang and S. Deb, "Engineering Optimisation by Cuckoo Search," pp. 1–17, 2009.

[146] V. Nannen, S. K. Smit, and A. E. Eiben, "Costs and benefits of tuning parameters of evolutionary algorithms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5199 LNCS, pp. 528–538.

[147] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram," *Computer Vision Graphics and Image Processing*, vol. 29. pp. 273–285, 1985.

[148] M. Waseem Khan, "A Survey: Image Segmentation Techniques," *Int. J. Futur. Comput. Commun.*, vol. 3, no. 2, pp. 89–93, 2014.

[149] M. Molga and C. Smutnicki, "Test functions for optimization needs," *Test Funct. Optim. needs*, no. c, pp. 1–43, 2005.

[150] C. Tsallis, "Entropic nonextensivity: A possible measure of complexity," *Chaos, Solitons and Fractals*, vol. 13, no. 3, pp. 371–391, 2002.

[151] M. Portes de Albuquerque, I. A. Esquef, A. R. Gesualdi Mello, and M. Portes de Albuquerque, "Image thresholding using Tsallis entropy," *Pattern Recognit. Lett.*, vol. 25, no. 9, pp. 1059–1065, 2004.

[152] H. Laurent, S. Chabrier, C. Rosenberger, and B. Emile, "Optimization-based image segmentation by genetic algorithms," *Eurasip J. Image Video Process.*, vol. 2008, 2008.

[153] K. E. Melkemi, M. Batouche, and S. Foufou, "A multiagent system approach for image segmentation using genetic algorithms and extremal optimization heuristics," *Pattern Recognit. Lett.*, vol. 27, no. 11, pp. 1230–1238, 2006.

[154] S. Yilmaz, E. U. Kucuksille, and Y. Cengiz, "Modified bat algorithm," *Elektron. ir Elektrotechnika*, vol. 20, no. 2, pp. 71–78, 2014.

[155] K. Hammouche, M. Diaf, and P. Siarry, "A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation," *Comput. Vis. Image Underst.*, vol. 109, no. 2, pp. 163–175, 2008.

[156] A. Chander, A. Chatterjee, and P. Siarry, "A new social and momentum

component adaptive PSO algorithm for image segmentation," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 4998–5004, 2011.

[157] M. Maitra and A. Chatterjee, "A hybrid cooperative-comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 1341–1350, 2008.

[158] X. S. Yang and S. Deb, "Cuckoo search via L??vy flights," in *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 2009, pp. 210–214.

[159] Y. Liu, C. Mu, W. Kou, and J. Liu, "Modified particle swarm optimization-based multilevel thresholding for image segmentation," *Soft Comput.*, pp. 1311–1327, 2014.

[160] A. D. M. Wilson, E. M. Whattam, R. Bennett, L. Visanuvimol, C. Lauzon, and S. M. Bertram, "Behavioral correlations across activity, mating, exploration, aggression, and antipredator contexts in the European house cricket, Acheta domesticus," *Behav. Ecol. Sociobiol.*, vol. 64, no. 2010, pp. 703–715, 2010.

[161] R. D. . Alexander, "Aggressiveness , Territoriality , and Sexual Behavior in Field Crickets (Orthoptera : Gryllidae)," vol. 17, no. 2, pp. 130–223, 1961.

[162] B. M. & J. G. William D. Brown, Adam T. Smith, "Aggressive contests in house crickets : size , motivation and the information content of aggressive songs," vol. 72, pp. 225–233, 2006.

[163] A. Osyczka and S. Kundu, "A modified distance method for multicriteria optimization, using genetic algorithms," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 871–882, 1996.

[164] T. Ray and K. M. Liew, "A swarm metaphor for multiobjective design optimization," *Eng. Optim.*, vol. 34, no. 2, pp. 141–153, 2002.

[165] A. R. Yıldız, N. Öztürk, N. Kaya, and F. Öztürk, "Hybrid multi-objective shape design optimization using Taguchi's method and genetic algorithm," *Struct. Multidiscip. Optim.*, vol. 34, no. 4, pp. 317–332, 2007.

[166] A. R. Yildiz, "An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry," *J. Mater. Process. Technol.*, vol. 209, no. 6, pp. 2773–2780, 2009.

[167] X. S. Yang, M. Karamanoglu, and X. He, “Multi-objective flower algorithm for optimization,” in *Procedia Computer Science*, 2013, vol. 18, pp. 861–868.

[168] G. Reynoso-Meza, X. Blasco, J. Sanchis, and J. M. Herrero, “Comparison of design concepts in multi-criteria decision-making using level diagrams,” *Inf. Sci. (Ny).*, vol. 221, pp. 124–141, 2013.

[169] P. Sabarinath, M. R. Thansekhar, and R. Saravanan, “Multiobjective optimization method based on adaptive parameter harmony search algorithm,” *J. Appl. Math.*, vol. 2015, 2015.

[170] M. Sarkar, “Multi-Objective Welded Beam Design Optimization using T-Norm and T-Co-norm based Intuitionistic Fuzzy Optimization Technique,” vol. 12, no. 3, pp. 549–575, 2017.

[171] S. Mirjalili, P. Jangir, and S. Saremi, “Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems,” *Appl. Intell.*, vol. 46, no. 1, pp. 79–95, 2017.

[172] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[173] T. C. Service, “A No Free Lunch theorem for multi-objective optimization,” *Inf. Process. Lett.*, vol. 110, no. 21, pp. 917–923, 2010.

[174] A. Dolbear, “The cricket as a thermometer,” *Am. Nat.*, pp. 970–971, 1897.

[175] K. Deb, “Optimal Design of a Welded Beam via Genetic Algorithms,” *AIAA J.*, vol. 29, no. 11, pp. 2013–2015, 1991.

[176] S. He, E. Prempain, and Q. H. Wu, “An improved particle swarm optimizer for mechanical design optimization problems,” *Eng. Optim.*, vol. 36, no. 5, pp. 585–605, 2004.

[177] N. OTSU, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Trans. Syst. Man. Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[178] Z. Wang, A. C. Bovik, H. R. Sheikh, S. Member, E. P. Simoncelli, and S. Member, “Image Quality Assessment : From Error Visibility to Structural Similarity,” *2 IEEE Trans. IMAGE Process.*, vol. 13, no. 4, pp. 1–14, 2004.

[179] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image*

Process., vol. 13, no. 4, pp. 600–612, 2004.

[180] S. Garc??a, D. Molina, M. Lozano, and F. Herrera, “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: A case study on the CEC’2005 Special Session on Real Parameter Optimization,” *J. Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[181] W. Gong, Z. Cai, and L. Zhu, “An efficient multiobjective differential evolution algorithm for engineering design,” *Struct. Multidiscip. Optim.*, vol. 38, no. 2, pp. 137–157, 2009.

[182] J. Andersson and J. Andersson, “A survey of multiobjective optimization in engineering design,” *Optimization*, p. 34, 2000.

[183] R. T. Marler and J. S. Arora, “Survey of multi-objective optimization methods for engineering,” *Structural and Multidisciplinary Optimization*, vol. 26, no. 6. pp. 369–395, 2004.

[184] B. Akay, “A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding,” *Appl. Soft Comput. J.*, vol. 13, no. 6, pp. 3066–3091, 2013.

[185] M. Caramia and P. Dell’Olmo, “Multi-objective Optimization,” *Multi-objective Manag. Freight Logist.*, pp. 11–36, 2008.

LIST OF PUBLICATION

International Journal Publications

1. Jonti Deuri and S. Siva Sathya., "A Novel Cricket Chirping Algorithm for Engineering Optimization Problem". *Adv. in Nat. Appl. Sci.,* 9(6): 397-402, 2015.

2. Jonti Deuri and S. Siva Sathya "Impact of Parameter Tuning on the Cricket Chirping Algorithm." *International Journal of Intelligent Systems and Applications* 9, no. 9 (2017): 58.

3. Jonti Deuri and S. Siva Sathya "Cricket Chirping Algorithm: an efficient meta-heuristic for numerical function optimization", Int. J. Computational Science and Engineering, ***Inderscience***, (in Press),

Book Chapter Publication

4. Sathya, S. Siva, and Jonti Deuri. "Multilevel Thresholding for Image Segmentation Using Cricket Chirping Algorithm." *Bio-Inspired Computing for Image and Video Processing* (2018).

International Conference Publication

5. Jonti Deuri, and S. Siva Sathya. "Cricket chirping algorithm for multi-objective engineering design optimization." *Information Communication and Embedded Systems (ICICES), 2017 International Conference on*. IEEE, 2017.

Communicated

6. Jonti Deuri and S. Siva Sathya, " Cricket Chirping Algorithm for Multi-objective Optimization'', Applied Computing and Informatics, *Elsevier*

7. Jonti Deuri and S. Siva Sathya, "*A Survey on recent Swarm Based Multi-objective optimization Techniques*"